1 2 9 0

UNIVERSIDADE Ð
COIMBRA

Marta Sofia Martins

EVALUATION OF POINT CLOUD DATA AUGMENTATION FOR 3D-LIDAR OBJECT DETECTION FOR AUTONOMOUS DRIVING

VOLUME 1

Dissertação no âmbito do Mestrado em Engenharia Eletrotécnica e de Computadores no Ramo de Robótica, Controlo e Inteligência Artificial orientada pelo Professor Doutor Cristiano Premebida e apresentada à Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Setembro de 2023

UNIVERSIDADE D COIMBRA

Evaluation of Point Cloud Data Augmentation for 3D-LiDAR Object Detection in Autonomous Driving

Marta Sofia Martins

Coimbra, September 2023

1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

Evaluation of Point Cloud Data Augmentation for 3D-LiDAR Object Detection in Autonomous Driving

Supervisor:

Dr. Cristiano Premebida

Co-Supervisor:

MSc. Iago Gomess

Jury:

Prof. Dr. Urbano Nunes

Prof. Dr. Rui Cortesão

Prof. Dr. Cristiano Premebida

Dissertation submitted in partial fulfillment for the degree of Master of Science in Electrical and Computer Engineering.

Coimbra, September 2023

Acknowledgements

Firstly, I would like to thank my advisors, Professor Cristiano Premebida and Master Iago Gomes, for their support, guidance, and motivation throughout my thesis. Their collective experience and knowledge that they shared with me proved indispensable to the completion of this thesis.

Finally, I am deeply grateful for the unwavering support of my family and friends, who believed in me even when I doubted myself. My most sincere acknowledgment.

Resumo

Os veículos autónomos fizeram progressos significativos nos últimos anos para enfrentar a natureza imprevisível das condições de condução no mundo real. Os veículos autónomos de mais alto nível são sistemas complexos que, no limite, podem tomar decisões sem intervenção humana com base em muitos módulos avançados, como o sistema de perceção. Nesta perspetiva, esta tese concentra-se no componente/módulo de perceção da condução autónoma. A perceção é a capacidade de um sistema autónomo de recolher e extrair dados relevantes do ambiente. A perceção ambiental (também conhecida como consciência situacional) é uma compreensão contextual referente à localização dos obstáculos e qual a classe/categoria semântica que representam, por exemplo, utilizadores da estrada (*e.g.*, peões, ciclistas, veículos) e sinais/marcações. De modo a imitar a perceção humana de profundidade do ambiente, a tecnologia de deteção de objetos 3D é frequentemente usada. Esta tese, em particular, utiliza um modelo de aprendizagem profunda (DL) para detetar objetos 3D em cenários urbanos. A aprendizagem profunda (DL) e outras áreas pertencentes à inteligência artificial/aprendizagem computacional (AI/ML) são um elemento-chave para robôs autónomos reproduzirem, entenderem e adaptarem-se ao ambiente envolvente.

Esta tese analisa métodos de aumento de dados em nuvens de pontos, assim como os seus respetivos efeitos na estimativa da posição e orientação de objetos usando uma rede profunda recente baseada em LiDAR (PointPillars). Nesse sentido, a arquitetura PointPillars extrai recursos densos e robustos de nuvens de pontos de sensores LiDAR e, em seguida, uma rede de aprendizagem profunda 2D com uma rede de deteção de objetos SSD modificada é usada para estimar posições, orientações e previsões de classe dos objetos na nuvem de pontos. Adicionalmente, esta dissertação apresenta uma avaliação comparativa das diferentes técnicas de deteção de objetos 3D em nuvens de pontos. Os resultados mostram que as técnicas de *augmentation* globais nas nuvens de pontos têm um impacto significativo na estimativa da posição e orientação dos objetos.

Palavras-chave: deteção de objetos 3D; condução autónoma; aprendizagem profunda; aumento de dados

Abstract

Autonomous vehicles have made significant progress in recent years to tackle the unpredictable nature of real-world driving conditions. Highest-level self-driving vehicles are complex systems that, in the limit, can make decisions without human intervention based on many advanced modules such as the perception system. In this context, this thesis concentrates on the perception component/module of autonomous driving. Perception refers to an autonomous system's ability to collect information and extract significant data from the environment. Environmental perception (a.k.a. situational awareness) refers to a contextual understanding of where the obstacles (*e.g.*, pedestrians, cyclists, cars) and signs/markings are located and also predicting their semantic meaning. To mimic human depth perception of the environment, 3D object detection technology is often used. This thesis in particular utilizes a deep learning (DL) model to detect 3D objects in urban scenarios. DL and other fields related to artificial intelligence and machine learning (AI/ML) are becoming vital for autonomous robots to represent, understand, and eventually adapt to their surroundings.

This thesis surveys data augmentation methods in point clouds and their effects on estimating the position and orientation of objects using a recent LiDAR-based deep network (the PointPillars). In this regard, the PointPillars architecture extracts dense, robust features from LiDAR sensor point clouds, and then a 2D deep learning network with a modified SSD object detection network is used to estimate joint positions, orientations, and class predictions of the objects in the point cloud. Additionally, this dissertation presents a comparative evaluation of different techniques for 3D object detection on point clouds. The results show that global augmentation techniques on the point clouds have a significant impact on the estimation of the position and orientation of objects.

Keywords: 3D object detection; autonomous driving; deep learning; data augmentation.

"The answers are always inside the problem, not outside."

— Marshall McLuhan

Contents

List of Acronyms

AI	Artificial Intelligence
ALV	Autonomous Land Vehicle
AOS	Average Orientation Similarity
BEV	Bird's-Eye View
BN	Batch Normalization
CV	Computer Vision
CNN	Convolutional Neural Network
DA	Data Augmentation
DARPA	Defense Advanced Research Projects Agency
DL	Deep Learning
GT	Ground Truth
IoU	Intersection over Union
GM	General Motors
HED	Honda Research Institute 3D
HRL	Hughes Research Labs

KITTI	Karlsruhe Institute of Technology and Toyota Technological Institute
LiDAR	Light Detection and Ranging
LR	Learning Rate
ML	Machine Learning
RADAR	Radio Detection and Ranging
R-CNNs	Region-based Convolutional Neural Networks
ReLU	Rectified Linear Unit
RoI	Region of Interest
RPN	Region Proposal Network
RV	Range View
SA-SSD	Structure Aware Single Stage Detector
SSD	Single Shot Detector
VGG	Visual Geometry Group
V2V	Vehicle-to-Vehicle

List of Figures

List of Tables

1 Introduction

A self-driving car, belonging to the highest level of automation *i.e.*, level 5, is a complex system capable of moving and making decisions without direct human interaction [71]. It is composed of sensors, actuators, complex algorithms including machine learning (ML), and processors [100]. To have the ability to drive autonomously, such a system creates a representation of their surroundings based on the sensor readings. In terms of sensor functionality, radar sensors are typically used to examine the position of nearby vehicles. Video cameras identify traffic lights, read road signs, track other vehicles, and look for pedestrians. LiDAR sensors reflect pulses of light around the car's surroundings to measure distances, identify road edges, and detect and locate obstacles [71]. Ultrasonic sensors detect obstacles and calculate the ideal steering angle during parking [60].

As illustrated in Figure.1.1, the core competencies of an autonomous vehicle software system can be categorized into three categories, perception, planning, and control, with the interactions between these competencies and the vehicle's interactions with the environment. Also, Vehicle-to-Vehicle (V2V) communications can be leveraged to realize further improvements in perception and/or planning areas through vehicle cooperation [13].

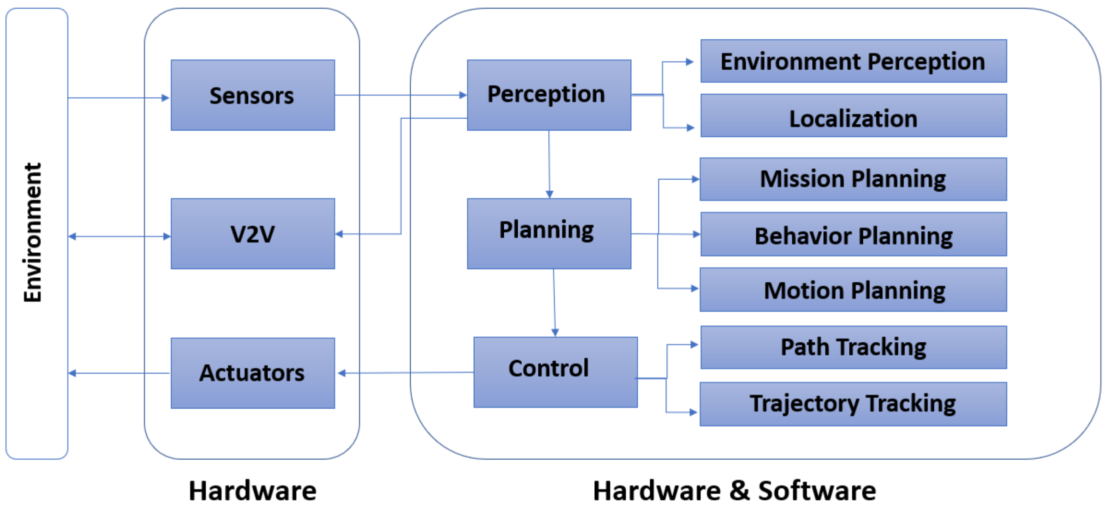


Figure 1.1: A typical autonomous vehicle system. Based on [12].

Given the aforementioned perspective, the main objective of this thesis is to approach the perception component of self-driving cars. To tackle sensory perception, this thesis primarily relies on the use of a deep learning model called PointPillar to detect 3D objects (*i.e.*, pedestrians, cars, and cyclists) in urban scenarios, based on the LiDAR data provided by the well-known KITTI dataset. Moreover, data augmentation techniques for a point cloud and their effect on the detector's performance were evaluated.

1.1 Motivation

On average, 1.35 million people die annually because of road accidents, about 3700 people a day (*i.e.*, one person every 24 seconds), in addition to 50 million who are injured. While the main cause of accidents is speeding or alcohol consumption, it is also known that the state of mind can influence driving, leading to a lack of attention and, consequently, a mishap. In conclusion, in about 90% of road accidents, the cause is related to human error [63][110]. With such a grim landscape, the need for transformative solutions becomes paramount.

In pursuit of safer roads, advanced technologies are explored to reduce the number of accidents. One of those technologies is 3D object detection, which holds the potential to revolutionize the capabilities of autonomous vehicles. Traditional 2D object detection systems have been a step forward, enabling rudimentary collision avoidance. However, to ensure the highest level of safety, autonomous vehicles need to perceive their surroundings in 3D. With 3D object detection integrated into autonomous vehicles, their sensors can accurately gauge objects' height, depth, and volume, making more informed decisions and significantly reducing the likelihood of collisions [4]. By embracing 3D object detection technology, self-driving cars can be equipped with the ability to mimic human depth perception, translating to safer and more efficient navigation on our roads.

Although 3D object detection enhances perception, the main point of autonomous driving lies in the artificial intelligence that “drives" these vehicles. This is where the origin of the data augmentation concept occurs. Data augmentation involves diversifying the training dataset by introducing variations to existing data [17]. In the context of self-driving cars, this approach is fundamental in creating more resilient AI models.

Data augmentation assists in addressing complex scenarios, such as detecting pedestrians in a crowd or identifying obstacles on poorly lit roads or different weather scenarios [29]. Therefore, data augmentation emerges as a pivotal factor in adequately equipping AI systems to contend with the inherent unpredictability of real-world driving conditions. By integrating these systems with a diverse dataset, autonomous vehicles can handle an array of scenarios with confidence, elevating overall safety and performance standards.

In summary, the reality of road accidents serves as a booster for innovative interventions. By integrating 3D object detection technology and embracing data augmentation techniques, it is possible to redefine road safety. The advancements improve the quality of life for elderly, visually impaired, and disabled individuals beyond the realm of statistics [71].

1.2 Context and scope

As mentioned before, artificial or sensory perception is a critical aspect of autonomous driving as the vehicle needs to understand its environment. Perception is the capacity of an autonomous system to collect information and extract important data from the environment. Environmental perception refers to a contextual understanding of where the obstacles are located, detecting road signs/markings, and categorizing data by their semantic meaning. Localization is the ability of the robot to determine its position related to the environment [13]. Environment perception is a function that allows autonomous vehicles to have fundamental information about the driving environment, including the free drivable area and surrounding obstacles' locations, velocities, and predictions of their future states [100]. The environment perception can be made by LiDARs, cameras, or a fusion between these two devices.

Among the several tasks for a robust environmental perception, obstacle detection is essential for autonomous vehicles to safely navigate. The standard approach for this task employs deep learning models based on Convolutional Neural Networks (CNN) [21] – such as *YOLO* [45], to extract features from images, and detect target objects based on the feature maps.

An important remark regarding 3D detection is the complexity of the annotation process of the data, which results in the small number of large-scale datasets available for training and testing models when compared to 2D detection datasets. In this sense, some works in the literature explored data augmentation techniques for 3D data to increase the number of samples and their representativeness [55] [49] [48] [95] [26] [84].

However, not every augmentation operation exerts a positive influence on model training and performance. On the contrary, when an operation is not applied correctly, it tends to only increase the memory usage, time, and cost of training and developing detection methods. Therefore, this thesis presents an experimental evaluation of 3D data augmentation operations for 3D obstacle detection on the KITTI dataset.

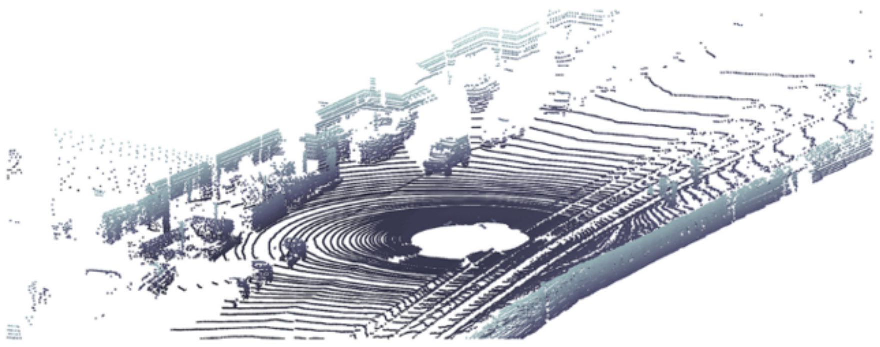
1.3 Aims and objectives

A perception system of a self-driving car uses, among other modules, object detection algorithms to create a list of 3D bounding boxes around objects of interest. To evaluate the 3D object detection, it was used the KITTI dataset [34]. KITTI is one of the most popular mobile robotics and autonomous driving datasets. It consists of hours of traffic scenarios recorded with different sensor modalities, including high-resolution RGB, grayscale stereo cameras, and a 3D laser scanner.

The KITTI dataset contains 15k frames and 12919 images RGB images (see Figure 1.2), as well as labels for eight different classes, such as 'Car'; 'Van'; 'Truck'; 'Pedestrian'; 'Person sitting'; 'Cyclist'; 'Tram'; 'Misc' or 'DontCare' [18][35]. In this dissertation, just the classes 'Car', 'Cyclist', and 'Pedestrian' are evaluated due to their representative aspect concerning the state-of-the-art and importance.

Figure 1.2: Some examples from the KITTI dataset [35].

Among the diverse sensor modalities involved within the extensive KITTI dataset, the focus of this project converged on using the precision and depth of its LiDAR point cloud. Figure 1.3 shows a sample LiDAR point cloud from the KITTI dataset.



((a)) LiDAR Point Cloud

((b)) Camera Image

Figure 1.3: A sample point cloud from the KITTI dataset alongside the corresponding camera image [101].

For the 3D LiDAR object detection, it was taken advantage of the PointPillars architecture, which extracts dense, robust features from LiDAR sensor point clouds and then uses a 2D deep learning network with a modified SSD object detection network to estimate joint 3D bounding boxes, orientations, and class predictions, such as pedestrians, bicycles, and cars. To determine if the detection is correct, standard detection metrics were calculated to evaluate the detector's performance.

Different techniques for data augmentation in the point cloud were explored to improve the detector's performance and add robustness to the model (Appendix C contains the experiment's preliminary results submitted to the **Sixth Iberian Robotics Conference**.) Data augmentation is a process that generates extended training data by applying various transformations, modifications, or manipulations to existing data, which allows machine learning to increase performance [17].

In short, the main objectives of this thesis are:

- Evaluate a LiDAR-based 3D obstacle detector for pedestrian, cyclist, and car classes. The evaluation's focus is the performance in locating the position of the 3D object and its orientation.

- Evaluate global and local data augmentation techniques on point clouds and their effects on estimating the position and orientation of objects.

The implementation part and respective coding have beenwritten in Python in conjunction with PyTorch environment, an open-source machine learning library used to develop and train neural network-based deep learning models.

1.4 Outline

The thesis consists of several chapters, which are listed below:

- **Chapter 2** provides a general approach to the context of autonomous driving. It covers the history, current state of the art, and previous works. Additionally, the chapter discusses the most relevant topics of autonomous cars for this work. These topics include the perception and 3D recognition of objects and sensor and dataset comparisons to determine the best options for this work.
- **Chapter 3** describes various 3D object detection methods based on LiDAR data. After the chapter, a concise summary of each method is provided.
- **Chapter 4** supplies additional details on the chosen method based on outlined in Chapter 3, including its structure and implementation.
- **Chapter 5** discusses data augmentation techniques that can improve model performance.
- **Chapter 6** explains the experimental part, including dataset manipulation, evaluation metrics implementation, and data augmentation technique selection.
- **Chapter 7** comprises a detailed presentation of all the results that have been obtained, along with comprehensive comments on each of them.
- **Chapter 8** is the final section of this thesis, which presents the conclusions of the work carried out and provides some suggestions for future research.

2 Related work and background

The company Houdina Radio Control has taken the first step towards autonomous driving. This company demonstrated a radio-controlled car called Linriccan Wonder [10]. The car was a 1926 Chandler that had to transmit antennae on its backward compartment, and it was controlled by another car that sent radio impulses while following it. The transmitting antennae captured the radio signals and sent them to the circuit-breakers, which controlled electric motors that directed the vehicle's movements. It was one of the most primitive forms of autonomous vehicles. Months after, a modified form of Linriccan Wonder called "Phantom Auto" was demonstrated by Achen Motors [10].

In 1939, GM sponsored Norman Bel Geddes's exhibit Futurama at the World's Fair, which portrayed embedded-circuit-powered electric cars. The circuits were embedded in the road and controlled by radio, like the previously seen attempts at driverless cars. So, RCA Labs showed a notably advanced model for autonomous vehicles [10].

In 1953, RCA Labs built a miniature car controlled by wires placed in a pattern on a laboratory floor. The engineers Leland Hancock and L. N. Ress took the idea of RCA Labs to a higher level. In 1958, they experimented with the system in actual highway installations of surroundings just outside the town of Lincoln, Neb. The car was guided by impulses sent from a detector circuits series hidden in the pavement. These detectors could determine the presence and the velocity of metallic vehicles on the surface [10].

Based on sophisticated models in 1959 and throughout the 1960s, General Motors presented Firebird, which was a series of trial cars that had an electronic guide system that could accelerate it over an automatic highway without the driver's participation [27] [15] [104]. This preceded Ohio State University's Communication and Control Systems Laboratory to launch a project in 1966 to develop cars that don't need drivers. These cars were triggered by electronic devices embedded in the road.

During the 1960s, United Kingdom's Transport and Road Research Laboratory tested a car named Citroen DS, which did not need a driver, that interacted with magnetic wires embedded in the road. Citroen DS traveled in a far more effective way than by human control [31].

In the 1980s, Ernst Dickmanns and his team at the Bundeswehr University Munich, in Germany, created a vision-guided Mercedes-Benz robotic van without any interaction from a driver. After the discovery of this project, there have been progressions in the field of autonomous driving technology. From 1987 to 1995, EUREKA performed the Prometheus Project on self-driving vehicles. Another responsible for the breakthrough in the field of autonomous cars was the DARPA agency of the U.S. Department of Defense [10].

In the United States, new technologies were developed by Carnegie Mellon University, the Environmental Research Institute of Michigan, the University of Maryland, Martin Marietta, and SRI International to be used in ALV projects. The AVL was the first project that created a road-following robotic vehicle using computer vision, LIDAR, and autonomous control [43] [51] [20]. Based on the AVL project, the HRL laboratory designed the first vehicle with an off-road map and self-driving navigation [10]. Throughout the times, there has been significant progress in autonomous vehicles.

In 1991, Ernst Dickmanns and Daimler-Benz of Bundeswehr University Munich developed the twin robots VaMP and Vita.2. These robots could drive on a heavy traffic road with little human intervention. Although they were semi-autonomous vehicles, they circulated on free lanes and performed lane changes with the autonomous overtaking of other cars [6].

In 1995, the Mercedes-Benz S-Class created by Dickmanns drove a 1.590 km journey from Munich to Copenhagen, where about 95% of the trip was in autonomous driving mode. The car used computer vision and integral memory's microprocessors intended with parallel processing to respond in real-time. In the same year, a semi-autonomous car from Carnegie Mellon University's Navlab with neural networks to control the steering wheel and human's control for throttle and brakes [105] [82] reached 98.2% autonomy on a 5.000 km journey. This project was called "no hands across America" or HOA.

In 1996, Alberto Broggi presented the ARGO project, which consisted of a modified Lancia Thema that followed lane marks on roads. The Lancia Thema contained two low-cost video cameras on board and stereoscopic vision algorithms to perceive the surrounding environment [14]. ARGO project realized a 1.900 km journey over six days, where 94% of the time was entirely in automatic mode.

In the 2000s, the Netherlands officialized the first autonomous public transport [98] Panatoya, 2003; Andréasson, 2001). Also, the US invested in autonomous vehicles for the military forces such as Demo I (US Army), Demo II (DARPA), and Demo III (US Army).

In 2021, Demo III proved the capacity of self-navigation on rough road terrains diverging obstacles like rocks and trees [10].

2.1 Perception for Self-Driving Cars

Self-driving cars are intended to improve driving safety and traffic efficiency. Consequently, an accurate environment perception system is crucial to gain information and control decisions. Perception is an essential aspect of Autonomous Driving as the vehicle needs to perceive its environment. To perceive its surroundings the system needs two stages. The first stage consists of scanning the road forward to detect changes in driving situations (traffic lights and signs, pedestrian crossing, and barriers, among others). The second stage is related to the perception of other vehicles [13]. To achieve object detection, cognition, and scene perception, self-driving cars need to perceive surroundings in a way at least like the way the human eye processes information [108]. This leads to cognitive AI systems that can be able to learn, relearn and act [42].

The viability of DL in autonomous driving is being recognized as some state-of-the-art results have been attained by Google cars and Uber cars in maps-based localization, which was trained to drive with little prior knowledge of the roads [11]. These vehicles take advantage of DL for path planning and obstacle avoidance to process camera-based information to solve complex CV problems [33]. Even though DL algorithms learn effective perception control from data, LiDAR costs and the expenses involved in manually annotating the maps limit the application of DL in self-driving cars [57].

DL and ML are classified into three categories; supervised, unsupervised, and reinforcement learning [64]. Supervised learning uses labeled datasets to train algorithms to classify data or predict results accurately. It adjusts its weights until the model has been fitted appropriately, which happens as part of the cross-validation process [38]. Unsupervised learning uses machine learning algorithms to examine and cluster unlabeled datasets. These algorithms find hidden patterns or data groupings without human intervention[38]. Reinforcement learning is a technique where a computer agent learns to perform a task due to repeated trial-and-error interactions with a dynamic environment [38].

The key steps to implementing these DL techniques consists of the following :

- Backpropagation: the primary method of learning. Compute the error and tries to minimize the error function in subsequent forward passes [38].

- Apply gradient descent to backpropagate the error function [38].
- Subtract a fraction of the gradient from the weight [38].
- Recalculate the weights responsible for making a correct or incorrect decision [38].
- The objective is to minimize the error function, by updating the weights, e.g., using minibatch or stochastic gradient descent [38].

Observation: More data and very large networks lead to too many parameters and increased training times [38].

2.2 What is 3D Object Detection?

Definition. 3D object detection seeks to predict the attributes of 3D objects in driving scenarios from sensory inputs. In most cases, a 3D object can be represented as a 3D cuboid (as represented in Figure 2.1), called a 3D bounding box, that includes the object inside.

The 3D bounding box is defined by [76]:

- 3D Center Coordinate: $T = [x_c, y_c, z_c]'$
- Dimensions: $D = [dx, dy, dz]$ (length, width, and height)
- Heading Angle: θ (the yaw angle, of a cuboid on the ground plane)
- Class (denotes the category of a 3D object)

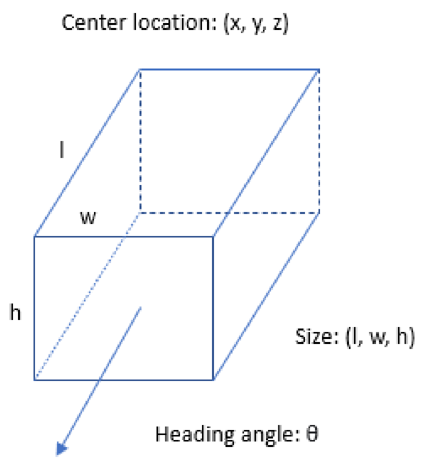


Figure 2.1: 3D Object Annotations [68].

In summary, the 3D bounding box can be represented:

$$B = [x_c, y_c, z_c, d_x, d_y, d_z, \theta, class] \tag{2.1}$$

2.3 Sensors

A fundamental requirement for a perception system is the choice of sensors that best adapt to the environmental characteristics with which the robot will be confronted. In the development of this system can be used ultrasonic sensors, cameras, radar, and LiDAR. Even though the numerous differences that these sensors have themselves, they are studied and analyzed according to a similar methodology. They are exteroceptive sensors, which means all the information is acquired from the environment. They are also active sensors since they emit an excitation signal to the outside space, waiting afterward for the occurrence of detection of objects that could be present in the field of view. In this way, the robot manages to have a perception of the surroundings.

2.3.1 Ultrasonic Sensor

Ultrasonic sensors use sound waves above the 20 kHz range [77] to detect and measure the distance of the surrounding objects by producing and monitoring an ultrasonic echo (as illustrated in Figure 2.2), similar to how bats use echolocation to maneuver without colliding with obstacles. Depending on the sensor and object properties, the effective range in air is between a few centimeters up to several meters [77]. In the automotive space, ultrasonic sensors are prevalent for ADAS (Advanced Driver-Assistant Systems) applications. Ultrasonic sensors are used in robotics applications that require reliable presence, proximity, or position sensing. Figure 2.3 presents an example of ultrasound sensors used in cars.

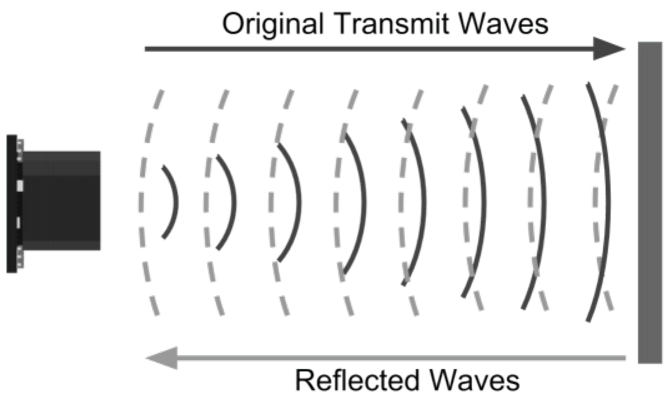


Figure 2.2: Ultrasonic Sensor Principle [72].

Figure 2.3: Ultrasonic car sensors from Bosch [74].

2.3.2 RADAR

RADAR is a sensor that uses radio waves to calculate distance, velocity, and angle. A RADAR system contains a transmitter producing electromagnetic waves in the radio or microwave domain, a transmitting antenna, a receiving antenna, and a receiver and processor to determine the properties of the objects. The transmitter's radio waves reflect off the objects and return to the receiver, giving information about the objects' locations and speeds. RADAR operates in extreme weather conditions and over long distances. But it may falsely identify objects. This sensor can detect pedestrians waiting at a pedestrian crossing [3], as shown in Figure 2.4.

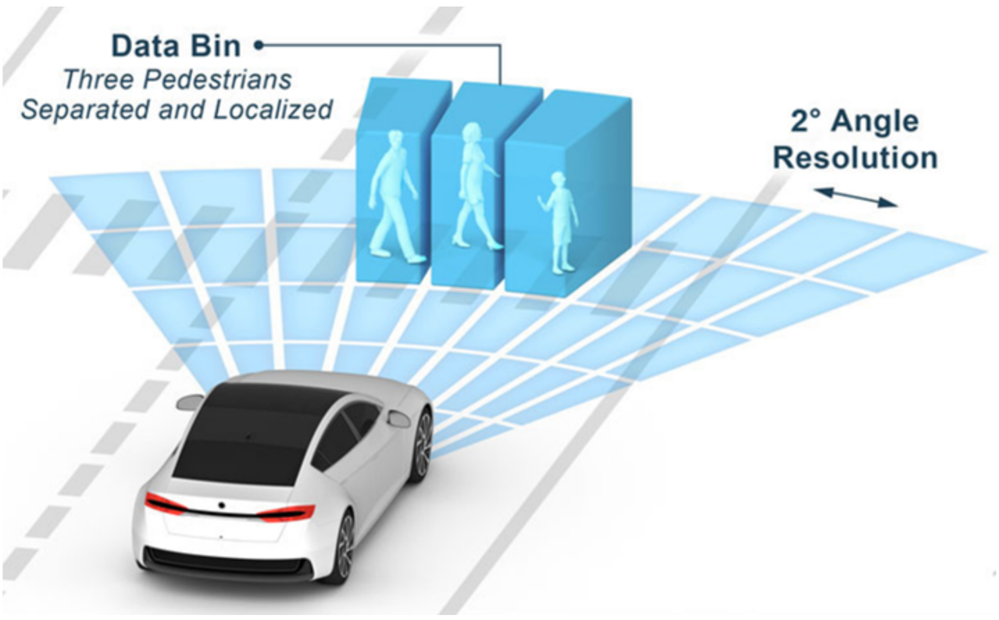


Figure 2.4: Radar for Road Cars [44].

2.3.3 LiDAR

LiDAR is a sensor for determining distance by targeting an object or a surface with a laser and measuring the time for the reflected light to return to the receiver. Figure [?] illustrates this principle. LiDAR sensors operate without visible light, which improves the environmental perception for night driving or low light conditions. In the case of adverse weather conditions, extra noise is added that changes the quality of the data, which affects the detection performance [37]. Many 3D object detectors are based on LiDAR sensors as sensors have a positive influence on the 3D object detection task. LiDAR is a powerful and efficient sensor, but it is expensive.

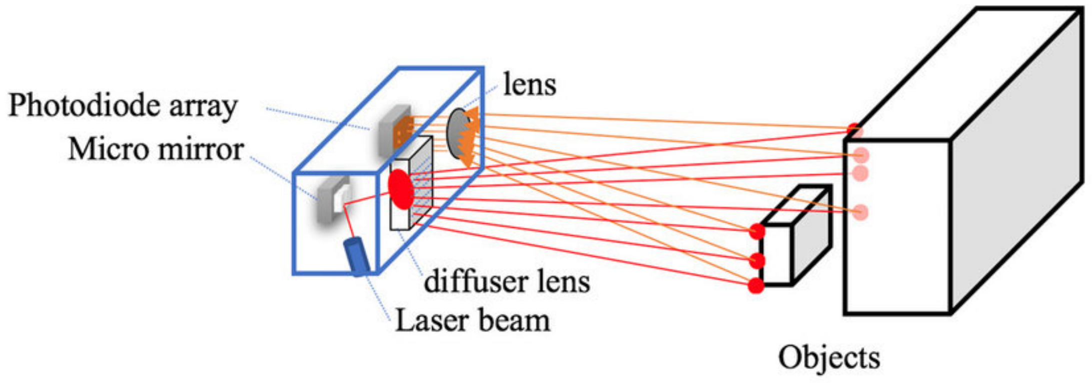


Figure 2.5: LiDAR Sensor Principle [90].

2.3.4 Cameras

The principle of working with a digital camera consists of a light incident on an object, and the reflected light is directed through an aperture present in the camera. These rays are focused with the help of a lens. A digital camera has sensors divided into red, green, and blue pixels. When light hits on these pixels, it is converted into energy. This energy will help to calculate the intensity of the pixel. The pixel intensity can determine the light and dark areas of the image, and this information is put together to create a digital image of the captured scene.

Generally, the most used cameras are depth cameras (RGBD) which have an additional parameter D from RGB cameras[65]. D stands for Depth. The depth sensor has a monochrome CMOS sensor and an infrared emitter which helps map the 3D image. It measures the distance of each point on the object by emitting infrared wavelengths and calculating the time after the object's reflection [65]. Figure 2.6 displays an image of a camera that is installed in a car.

Figure 2.6: Car cameras [93].

2.3.5 Sensors Comparison

The LiDAR sensor is becoming a key element in self-driving cars. This long-range sensor with 360º scanning and high resolution provides 5D information and good performance under distinct light conditions (night or day) due to its light sources. Unlike cameras, LiDAR is not blinded when pointed in light or dark directions. LiDAR has a very high performance[36]. LiDAR technology provides more depth and detail than other solutions, such as radar and cameras. And also, unlike ultrasonic sensors that are used to detect nearby objects, lidar has long-range scanning. Due to the ranging accuracy superiority of lidar, the provided physical information is highly trustworthy [56]. However, the LiDAR sensor has some disadvantages, such as being an expensive technology compared to other solutions, and it is affected by some specific atmospheric weather conditions, such as the case of fog and smoke presence.

Since the goal of this project is 3D object detection, it's necessary to choose a sensor with good depth information, details, and performance. In this case, the LiDAR sensor is the best solution for this problem.

2.4 Datasets

Multiple driving datasets have been created to provide multi-modal sensory data and

3D observations for 3D object detection. A few of the most popular datasets are introduced below, and a summary of them can be seen in table 2.1.

2.4.1 KITTI

KITTI was a pioneering work that offers a standard data collection and annotation model: providing a vehicle with cameras and LiDAR sensors, driving it on roads for data collection, and annotating 3D objects from the collected data [36]. KITTI includes real-world images such as rural, urban, and highways. One of these images can have up 15 cars and 30 pedestrians [53]. The datasets are classified as Road, City, Residential, Campus, and Person. For 3D object detection, labels are subdivided into car, truck, pedestrian, cyclist, etc. The KITTI dataset is the most used computer vision algorithm evaluation world's dataset for autonomous driving scenes [53].

2.4.2 ApolloScape

ApolloScape was an evolving research project that provided innovation across autonomous driving aspects, such as perception, navigation, and control. The apolloScape dataset is Baidu's dataset for autonomous driving [62]. The dataset consists of 80k lidar point clouds and high-quality labels. It is collected in Beijing under various lighting conditions and traffic situations [53]. The apolloScape dataset contains 100K street view frames with complex traffic flows mixed with vehicles, cyclists, and pedestrians [53].

2.4.3 H3D

The H3D dataset was Honda's presented, it is a point cloud dataset for autonomous driving scenes [53]. The H3D dataset includes 104 hours of human driving data, which generates 150 GB of data (1280 $\times$ 720 resolution, 30fps) that involves GPS, images, lidar, car navigation, and driving performance. This dataset used three cameras, a 360-degree lidar dataset, a car dynamics analyzer, and a car controller area network (CAN) [62]. One thing that the H3D, ApolloScape, and KITTI datasets are in common is that all collected data with the cooperation of cameras, lidar, and GPS [62].

2.4.4 Waymo

Waymo is an autonomous driving car company owned by Google Inc [53]. The Waymo

Open dataset is an open-source multimodal sensor dataset for self-driving. Obtained from Waymo autonomous driving vehicles, the data covers an extensive range of driving scenarios and environments. Those scenarios can contain obstacles such as pedestrians, automobiles, and buildings [53] [62].

The Waymo dataset includes 1,000 distinct clips, each captured for 20 seconds, which is equivalent to 200,000 frames per sensor [62]. The Waymo dataset, as well as the Apollo dataset, both contain data collected on a rainy day besides a sunny day. Also, they both added lidar sensors with the cooperation of cameras to collect data, which would make data more accurate [62].

2.4.5 nuScenes

Developed by Motional, nuScenes was the first dataset to have fully autonomous sensors for vehicles. The set of sensors contained six cameras, five radars, and one LiDAR, all with a 360º field of view [16]. Recorded in Boston and Singapore, nuScenes comprises 1000 scenes capturing a diverse range of traffic situations, driving maneuvers, and unexpected behaviors, each with 20s duration. Every scene is fully annotated with 3D bounding boxes for 23 classes and eight attributes [16]. Compared with the KITTI dataset, nuScenes has seven times more annotations and a hundred times more images [16].

2.4.6 Lyft Level 5

Lyft Level 5 is well recognized as the KITTI dataset [53], and Lyft is also a large-scale dataset that uses the same data format as the nuScenes [47]. This dataset presents data collected by three raw LIDAR and seven camera inputs. These sensors are inserted into autonomous vehicles within a defined geographic area to collect raw data from sensors in other cars, pedestrians, traffic lights, etc [47]. The Level 5 dataset includes over 55,000 human-labeled 3D annotated frames, underlying HD spatial semantic maps, and surface maps [53].

2.4.7 PandaSet

PandaSet was the first open-source AV dataset accessible for commercial and academic usage [80]. This dataset combines one 360° mechanical spinning LiDAR (Panda64), one forward-facing long-range LiDAR with image-like resolution (PandaGT), and six cameras [114]. The data collected by the sensors was noted with a combination of cuboid and scale

3D sensor fusion segmentation [80]. PandaSet contains more than 100 scenes, each scene with a duration of 8 seconds, and provides 28 label types for object classification and 37 label types for semantic segmentation [114].

2.5 Summary

This master thesis used the KITTI dataset. As seen in Table 2.1, compared to other datasets, KITTI has fewer frames, which makes it simpler to manipulate and faster to run. KITTI is also one of the most used datasets in the context of autonomous vehicles, with well-established benchmarking for various perception tasks, including 3D detection. Several 3D detection techniques based on lidars were evaluated on KITTI and are present in its benchmarking, which presents detection performance and processing time (frame rate). Thus, KITTI stands out as the best option for this experiment.

Datasets	Year	Classes	Scenes	Frames	Sensors
KITTI	2012	8	22	15k	Camera + LiDAR
ApolloScape	2019	35	-	100k	Camera + LiDAR
H3D	2019	8	160	27k	Camera + LiDAR
Waymo	2020	4	1k	200k	Camera + LiDAR
nuScenes	2020	23	1k	40k	Camera + LiDAR
Lyft Level 5	2019	9	366	46k	Camera + LiDAR
PandaSet	2020	28	100	60k	Camera + LiDAR

Table 2.1: Dataset summary table.

3 3D LiDAR Object Detection

3.1 Methods

In this section, the 3D object detection methods centered on LiDAR data i.e., point clouds or range images, will be introduced. Contrary to images, a point cloud is a 3D representation that contains sparse and irregular pixels and requires a specific model for feature extraction. On the other hand, a range image is a dense and compact representation where range pixels have 3D information [69]. The LiDAR-based 3D object detection models are based on distinct data representations, including point-based, grid-based, point-voxel-based, and range-based methods.

3.1.1 Point-based 3D object detection

Point-based 3D object detection, represented in Figure 3.1, uses deep learning techniques on point clouds to leverage the potentialities of DL, namely CNN-architectures. Points are the raw output data from the capture sensors in mostly 3D geometry acquisition systems [92]. Using points instead of geometry allows several processing and display simplifications. Yet, it is hard to generate point-based models, especially when a system is developed with simple, low-cost components.

Point clouds pass over a point-based backbone network, where the points are sampled, and point cloud operators learn the features. Based on downsampled points and features, it will be predicted 3D bounding boxes [69]. Point cloud sampling and feature learning are two components of point-based 3D object detection.

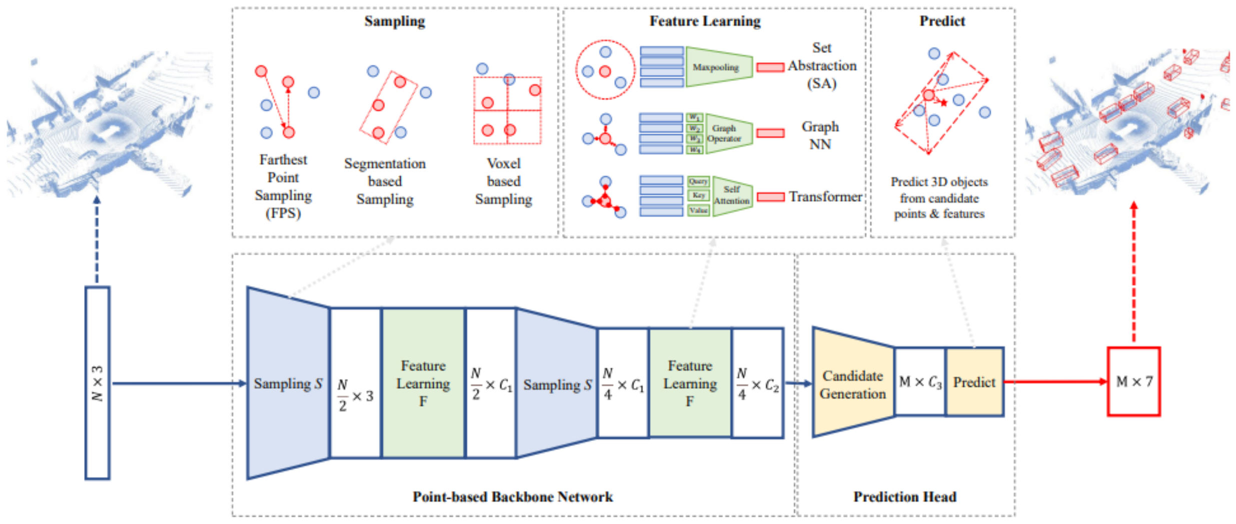


Figure 3.1: Point-based 3D object detection method [69].

Point Cloud Sampling

Furthest Point Sampling (FPS) in PointNet++ [84] has been implemented in detectors based on points, where the faraway points are chosen in sequence from the original point set. PointRCNN [97] takes advantage of the FPS technique to gradually downsample input point clouds and create 3D bounding box proposals.

Point Cloud Feature Learning

In PointNet [84], point-wise multilayer perceptron (MLP) blocks are sequentially applied to the raw point cloud to generate high-dimensional feature vectors based on points. The point features are aggregated through the max pooling operation to obtain the new feature vector [111].

In previous works, Wu et al. [112] present a convolution network, which through MLP networks and kernel density estimation, it learns the convolution kernel. Newly, Ma et al. [66] introduce PointMLP which is a deep network based on MLP to process the point cloud. The network is similar to the PointNet++ network, with additional residual connections and geometric affine modules.

3.1.2 Grid-based 3D object detection

3D object detectors based on grids convert point clouds into discrete grid representations, such as voxels, pillars, and bird's-eye view (BEV) feature maps. After, they use 2D convolutional neural networks or 3D sparse neural networks to extract grid features and then detect 3D objects from the BEV grid cells [69]. Figure 3.2 illustrates the method of detecting 3D objects using a grid-based system.

Grid-based detectors have two main components: grid-based representations and grid-based neural networks.

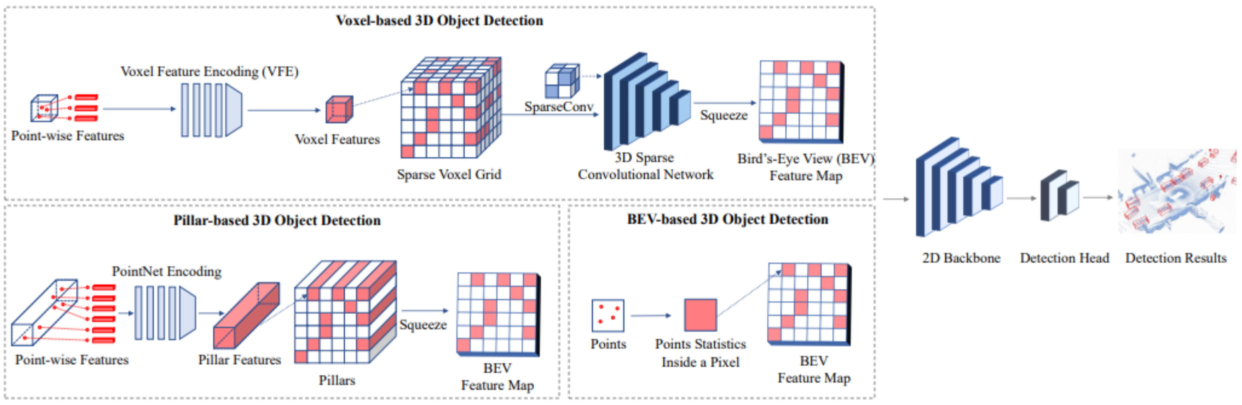

Voxel-based 3D Object Detection
Voxel Feature Encoding (VFE)
Point-wise Features
Voxel Features
Sparse Voxel Grid
SparseConv
3D Sparse Convolutional Network
Squeeze
Bird's-Eye View (BEV) Feature Map
Pillar-based 3D Object Detection
PointNet Encoding
Point-wise Features
Pillar Features
Pillars
Squeeze
BEV Feature Map
BEV-based 3D Object Detection
Points
Points Statistics Inside a Pixel
BEV Feature Map
2D Backbone
Detection Head
Detection Results

Figure 3.2: Grid-based 3D object detection method [69].

Grid-based representations

As mentioned before, the three types of grid representations are voxels, pillars, and BEV feature maps. Voxels represent a value on a grid in three-dimensional space. Voxelization converts point clouds into voxels. Since point clouds contain sparsely and irregularly distributed pixels, some voxel cells are empty [69]. Only non-empty voxels are used for feature extraction. Pillars are a kind of special voxels that have an unlimited voxel size in the vertical direction. Through a PointNet [84] network, points can be converted into pillar features. And finally, a bird's-eye view is a projection of the points into a planar top-down point-of-view. VoxelNet [123] is an example of a network that uses sparse voxel grids. This network proposes a new voxel feature encoding (VFE) layer to obtain features of voxel cell points. PointPillars [49] is a previous work that introduces the pillar representation.

There are two approaches to improving voxel representations: multi-view voxels and multi-scale voxels. Multi-view voxels propose a dynamic voxelization and fusion from different views, e.g., bird's-eyes and perspective views [121]. In multi-scale voxels, different voxels scales [116] or reconfigurable voxels [109] are used.

The bird's-eye view, BEV, feature map is a dense two-dimensional representation, where each pixel corresponds to a particular area and encodes the points information in this area [69]. Voxels and pillars can be lifted from the BEV feature maps, and a 3D point inside the voxel/pillar is projected into the bird's-eye view [54]. Another way to obtain BEV feature maps, it's from raw point clouds by summarizing points statistics inside the pixel area [69]. Binary occupancy [1] and local point clouds height and density [2] are the most commonly-used statists.

Grid-based neural networks

2D convolutional neural networks for BEV feature maps and pillars and 3D neural networks for voxels are the two main types of grid-based networks. When applying 2D convolutional neural networks to the BEV feature map, 3D objects can be detected from the bird's-eye view [69]. 3D sparse neural networks are based on sparse convolutions and submanifold convolutions [39]. Sparse convolutional operators are highly efficient and capable to generates a real-time inference speed [69]. The SECOND [115] uses two sparse operators that form a sparse convolution network to extract 3D voxel features. This network has been implemented in multiple previous works and turn the most used backbone network in voxel-based detectors.

3.1.3 Point-voxel based 3D object detection

Point-voxel-based methods apply a hybrid structure that uses voxels and points as representations [58]. Generally, those methods are distributed into two categories: the single-stage and two-stage detection frameworks. Figure 3.3 provides an illustration of the structure for both methods.

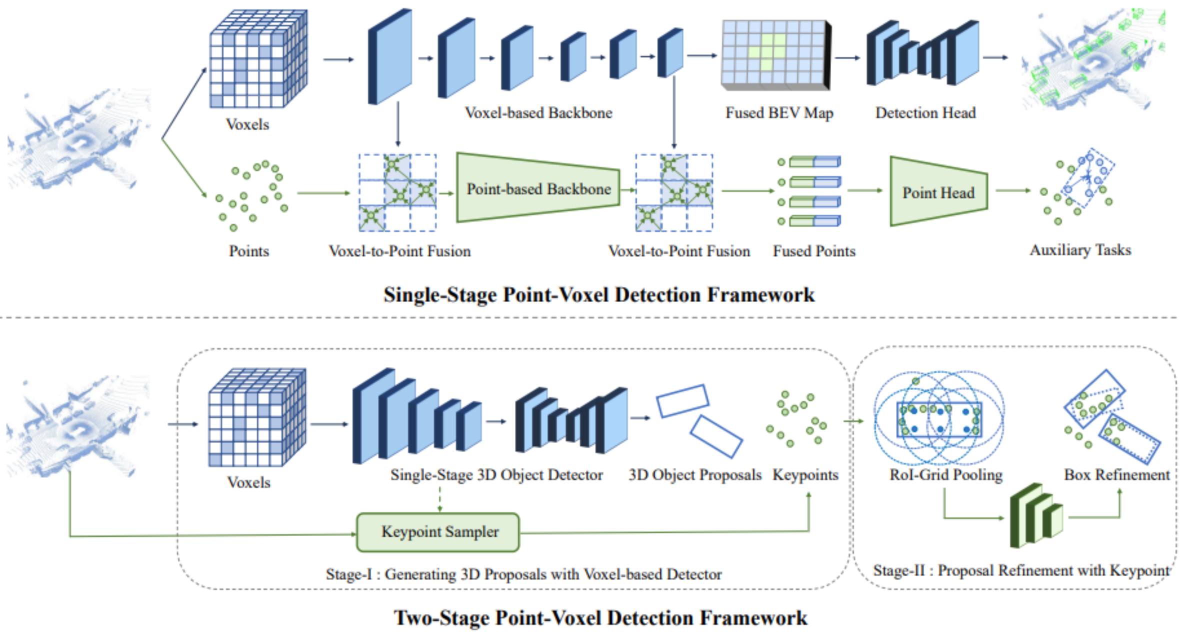


Figure 3.3: Point-voxel based 3D object detection method [69].

Single-stage point-voxel detection frameworks

As voxel-based methods have high computational efficiency and point-based methods include more details, single-stage point-voxel-based methods naturally benefit from both two representations [58]. SA-SSD [41] is an example of a single-stage network. This network uses voxel and point representations, respectively, in the backbone and the auxiliary branch. In the branch, the 3D SCNN extracts feature that is posteriorly converted into points in particular layers, and data is processed across point-based networks. The branch predictions calculate loss and optimize the branch and the backbone. After training, the auxiliary branch is no longer necessary and can be removed. SA-SSD [41] benefits from two types of representations.

Two-stage point-voxel detection frameworks

Generally, voxel representations are used in the first stage to provide a set of 3D object proposals, and point representations are applied to get refined details in the next stage [58]. Fast Point RCNN [22] makes use of the two-stage approach. In the first stage, voxels generate proposals with convolution methods at the level. And in the second stage, the points within the proposals are deemed, and the first-stage predictions are refined with fused features.

PV-RCNN [96] was one of the first works of a two-stage method. PV-RCNN uses SECOND [115] as the voxel-based network in the first stage and the RoI-grid pooling operator as the point-based representation in the second stage. Some works try to improve the performance in the second stage by adding new modules and operators, such as RefinerNet [22], channel-wise Transformer in CT3D [94], and RoI-grid attention in Pyramid R-CNN [67].

3.1.4 Range-based 3D object detection

A range image is a dense and compact 2D representation where each pixel expresses the distance between a known reference frame and a perceptible point in the scene [102]. So, a range image reproduces an estimation of the 3D structure of a scene. Range images can also be mentioned as depth images, depth maps, XYZ maps, and 2.5 D images [8]. To address detection problems, range-based methods project new models and operators adapted for range images (see Figure 3.4) and select appropriate views for detection [69].

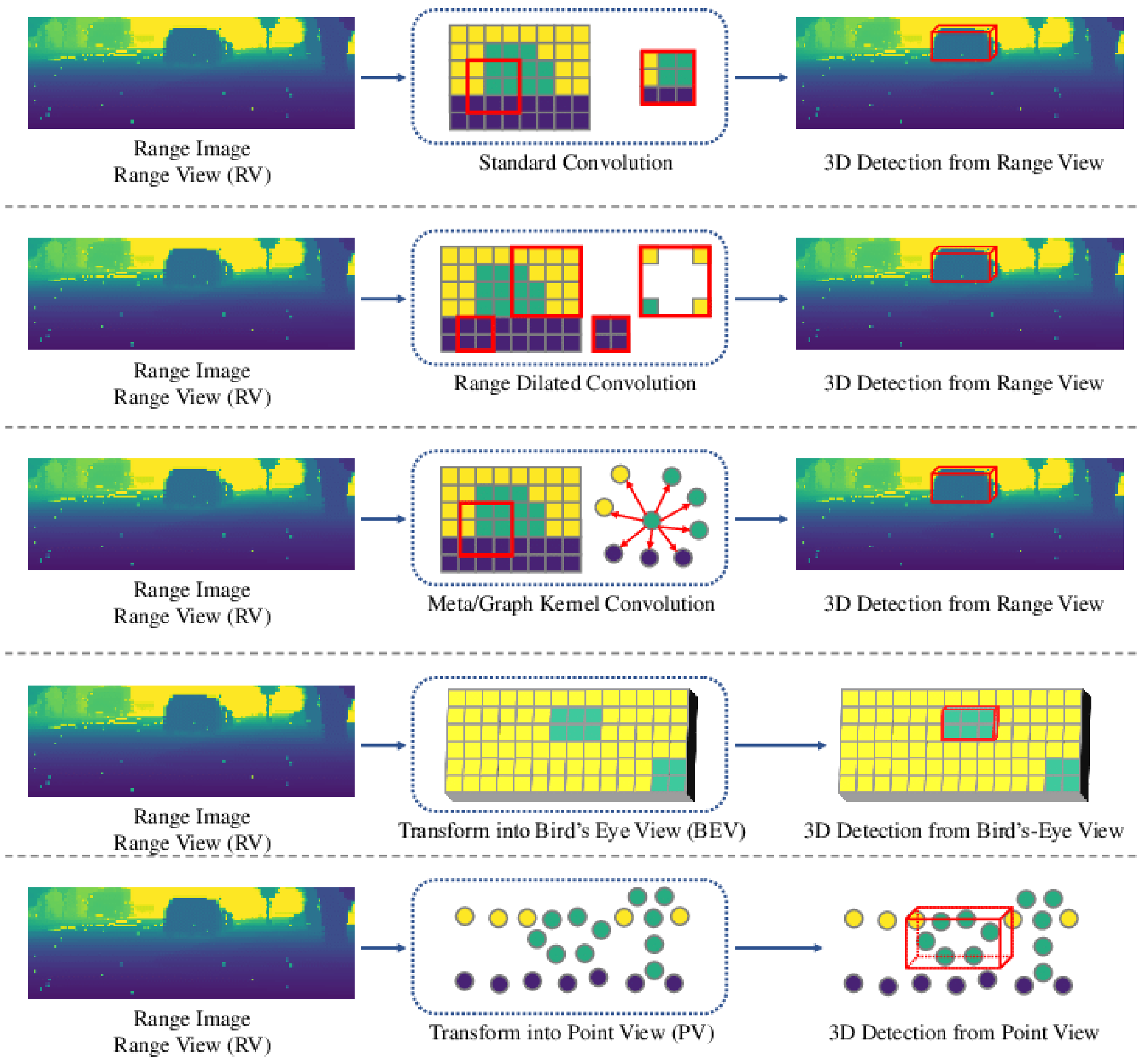

Range Image
Range View (RV)
Standard Convolution
3D Detection from Range View
Range Image
Range View (RV)
Range Dilated Convolution
3D Detection from Range View
Range Image
Range View (RV)
Meta/Graph Kernel Convolution
3D Detection from Range View
Range Image
Range View (RV)
Transform into Bird's Eye View (BEV)
3D Detection from Bird's-Eye View
Range Image
Range View (RV)
Transform into Point View (PV)
3D Detection from Point View

Figure 3.4: Range-based 3D object detection method [69].

Range-based detection models

Singe range images are 2D representations, 3D object detectors based on the range can share the models with 2D object detectors to handle range images. In LaserNet [73], multi-scale features and 3D object detection from range images are achieved with a deep layer aggregation network (DLA-Net).

Range-based operators

Since range image pixels have 3D distance information, the typical convolutional operator in a 2D network architecture isn't the best choice for range-based detection because the pixels in a sliding window can be distant from each in a 3D space [69]. Previous works used new range-based operators for features extracting from range pixels, including range-dilated convolutions [9], graph operators [19], and meta-kernel convolutions [32].

Views for range-based detection

Range images are taken from the range view (RV), a point cloud spherical projection. It has been a solution in based-range methods for 3D object detection from the range view [69]. However, range view detections suffer from spherical projection issues, such as occlusion and scale variation [69]. To improve that, many methods leverage other views for 3D object predictions, like the cylindrical view (CYV) [85] and a combination of the range view, bird's-eye view (BEV), and point view [59].

3.2 Summary

As seen, there are different methods for 3D object detection, focusing on LiDAR data, particularly point clouds and range images. The Pillar method is one of the main approaches discussed previously. In summary:

- **Point-Based 3D Object Detection:**
  - Point-based methods utilize deep learning on point clouds for successful object detection.
  - Raw points obtained from LiDAR sensors are processed through a point-based backbone network.
  - Point cloud sampling (e.g., Furthest Point Sampling) and feature learning (using PointNet, PointMLP, among others) are crucial steps.
- **Grid-Based 3D Object Detection:**
  - Grid-based methods convert point clouds into discrete grid representations, such as voxels, pillars, and bird's-eye view (BEV) feature maps.
  - Grid-based detectors consist of grid-based representations (voxels, pillars, BEV) and grid-based neural networks (2D CNNs for BEV and pillars, 3D CNNs for voxels).
- **Point-Voxel Based 3D Object Detection:**

  - Point-voxel-based methods combine voxel and point representations for detection.
  - Single-stage frameworks (e.g., SA-SSD) and two-stage frameworks (e.g., Fast Point RCNN, PV-RCNN) benefit from both representations.

- **Range-Based 3D Object Detection:**
  - Range images, dense 2D representations of distance, are used for 3D object detection.
  - Range-based models (e.g., LaserNet) and operators (range-dilated convolutions, graph operators) are tailored for range images.
  - Different views (range view, cylindrical view, combination of views) are leveraged to improve detection accuracy.

The Pillar method involves point-based and grid-based approaches, combining raw point processing and grid representations to achieve accurate 3D object detection from LiDAR data.

4 PointPillars Network

Lang et al. [49] have proposed PointPillars, a novel encoder that uses PointNets to learn a representation of point clouds organized in vertical columns (pillars). According to them, PointPillars, compared to other methods, achieved superior results in all classes and difficulty strata except for the easy subset for the car class. It also outperforms fusion-based methods on cars and cyclists. PointPillars also has a significant improvement in inference runtime.

Among the learned encoders VoxelNet is marginally stronger than PointPillars. However, since the VoxelNet encoder has a slower magnitude and more parameters than PointPillars, it's not a fair comparison, since when compared to similar inference times, PointPillars offers a better mean average precision. Lang et al. [49] also demonstrate that PointPillars dominates all existing methods on the KITTI challenge by offering higher detection performance at a faster speed. Their results suggest that PointPillars is a relevant architecture for 3D object detection from LiDAR.

PointPillars. A method for 3D object detection that allows end-to-end learning with only 2D convolutional layers. PointPillars utilizes an encoder that learns features on pillars (vertical columns) of the point cloud to predict 3D-oriented boxes for objects [49]. End-to-end learning is a deep learning process where the model learns all the steps between the initial input phase and the final output result. In this technique, all the parts are simultaneously trained instead of sequentially.

PointPillars accepts dense, robust features from point clouds from a LiDAR sensor as input, estimates oriented 3D bounding boxes, and class predictions, such as pedestrians, bicycles, and cars. Point Pillars has three steps (Fig. 4.1): **(1)** a feature encoder, where point clouds are converted to a pseudo-image; **2)** a backbone (2D CNN) to process the pseudo-image into a high-level representation; and **(3)** a detection head (SSD) that detects the objects and create 3D bounding boxes around it.

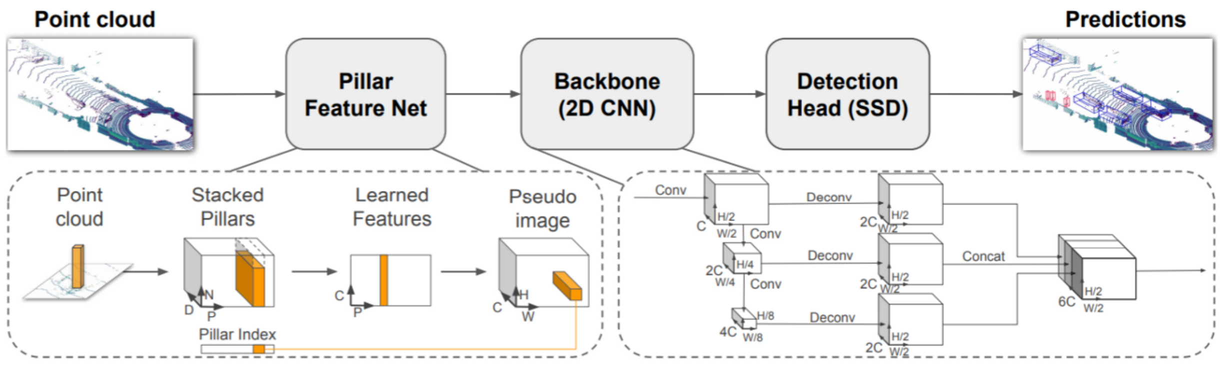


Figure 4.1: PointPillars Architecture [49].

4.1 Feature Encoder

First, the point cloud is divided into grids in the x-y coordinates, creating pillars set. Each point in the cloud, which is a 4-dimensional vector $(x, y, z, reflectance)$, is converted to a 9-dimensional vector containing:

$$D = [x, y, z, r, X_c, Y_c, Z_c, X_p, Y_p] \tag{4.1}$$

where $[X_c, Y_c, Z_c]$ is the distance from the geometric center point of the pillar to the point cloud point, r is the reflectance, and $[X_p, Y_p]$ is the distance of the point from the center of the pillar in the x-y coordinate system.

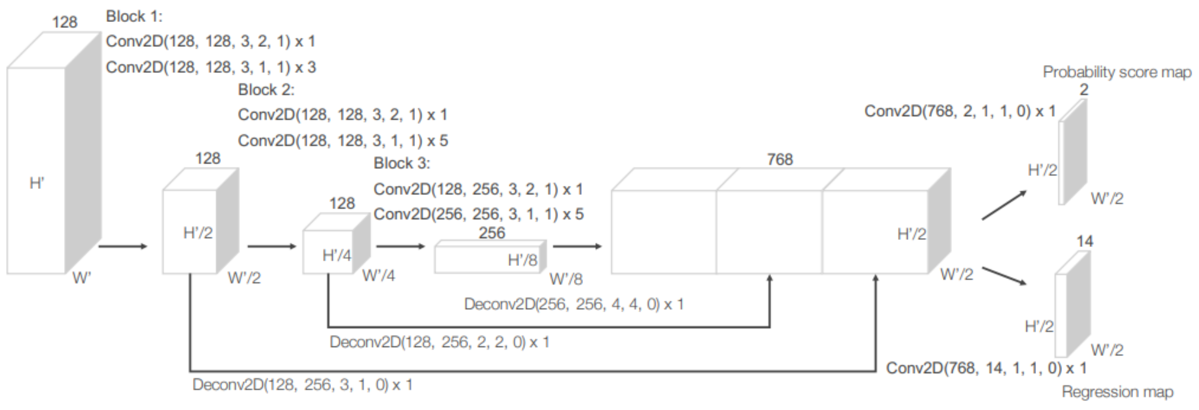
The pillar sets will be mostly empty due to the sparsity of the point cloud, and the non-empty pillars will contain few points in them. This sparsity is exploited by imposing a limit both on the number of non-empty pillars per sample (P) and on the number of points per pillar (N) to create a dense tensor of size (D, P, N). If a sample or pillar holds too much data to fit in this tensor, the data is randomly sampled. Oppositely, if a sample/pillar has too little data to populate the tensor, zero padding is applied [49].

Next, it uses the PointNet network to extract features from the pillars. PointNet applies to each point, a linear layer followed by BatchNorm and ReLU to generate high-level features, which in this case is of dimension (C, P, N). This is followed by a max pool operation which converts this (C, P, N) dimensional tensor to a (C, P) dimensional tensor.

4.2 Backbone (2D CNN)

The backbone constitutes sequential 2D convolutional layers to learn features from the input. The input is the feature map generated from the feature encoder. Region Proposal

Network (RPN) [124] is an example of a backbone.



128
Block 1:
Conv2D(128, 128, 3, 2, 1) x 1
Conv2D(128, 128, 3, 1, 1) x 3
Block 2:
Conv2D(128, 128, 3, 2, 1) x 1
Conv2D(128, 128, 3, 1, 1) x 5
Block 3:
Conv2D(128, 256, 3, 2, 1) x 1
Conv2D(256, 256, 3, 1, 1) x 5
H'
W'
128
H'/2
W'/2
128
H'/4
W'/4
256
H'/8
W'/8
768
H'/2
W'/2
Deconv2D(256, 256, 4, 4, 0) x 1
Deconv2D(128, 256, 2, 2, 0) x 1
Deconv2D(128, 256, 3, 1, 0) x 1
Probability score map
2
Conv2D(768, 2, 1, 1, 0) x 1
H'/2
W'/2
14
H'/2
W'/2
Conv2D(768, 14, 1, 1, 0) x 1
Regression map

Figure 4.2: Region Proposal Network (RPN) architecture [122].

This backbone network (Fig.4.2) has three blocks of fully convolution layers. The first layer of each block downsamples the feature map by half via convolution with a stride size of 2, followed by a sequence of convolutions of stride 1. After each convolution layer, BN and ReLU operations are applied. Then, upsample the output of each block to a fixed size and concatenate them to build the high-resolution feature map.

4.2.1 RPN

A region proposal network (RPN) is a highly optimized algorithm for object detection effectiveness. RPN (Fig.4.3) generates the proposal to the objects. To create "proposals" for the area where the object is located, a small network is slid on a convolutional feature map, which is the last convolutional layer output [87].

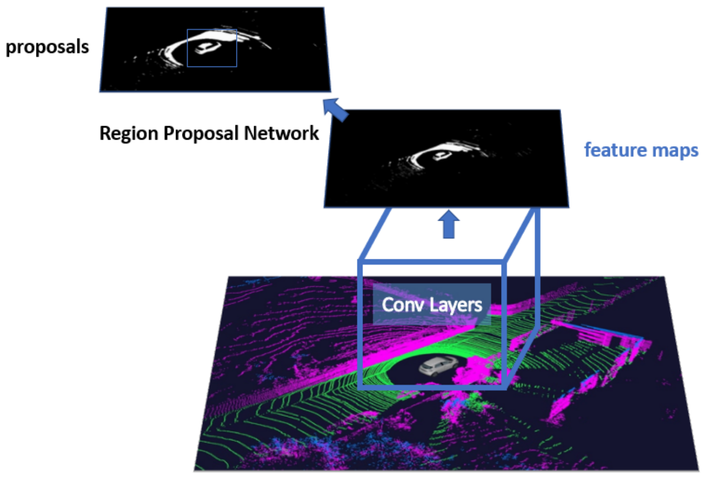


Figure 4.3: Region Proposal Network (RPN). Based on [30].

RPN contains a classifier and a regressor. The classifier calculates the proposal probability of getting a target object, and the regression regresses the proposal coordinates [91].

4.2.2 Anchors

RPN uses anchor boxes (Fig.4.4) to compute the bounding box regression value and class label. An anchor is a central point at the sliding window in question [87].

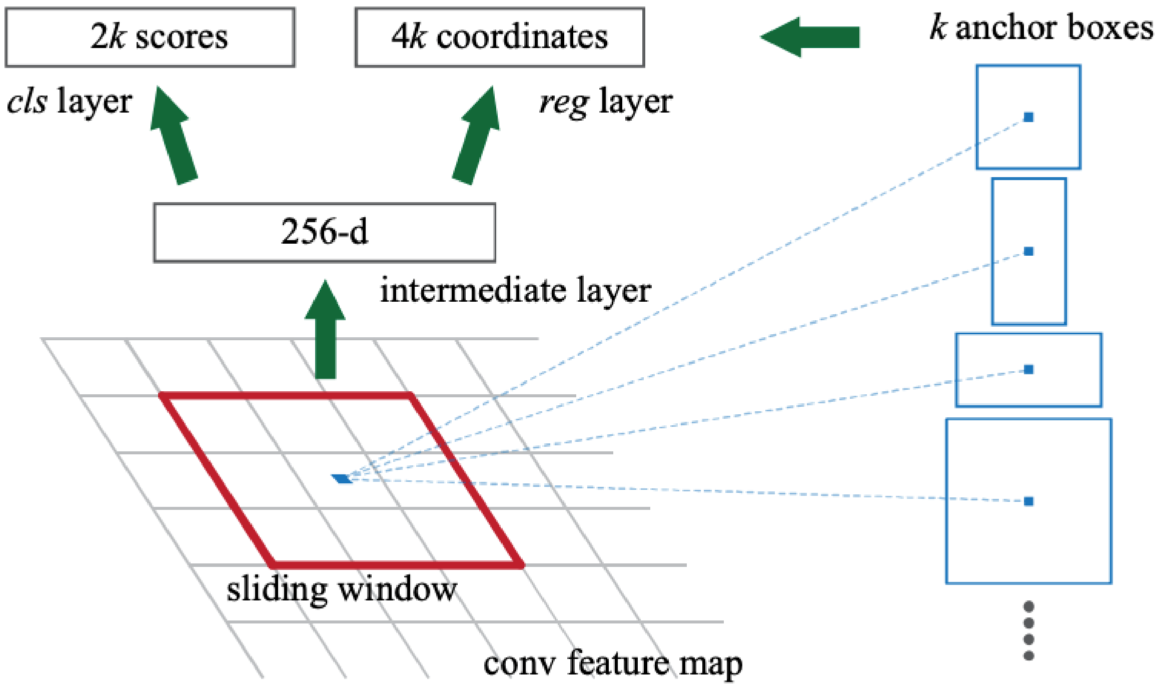


Figure 4.4: Anchor boxes generation [87].

By default, anchors use 3 scales and 3 aspect ratios [87]. Scale, which is the size of an image, and aspect ratio are two important parameters for every image. The aspect ratio can be determined using the following formula:

$$ratio = W/H \tag{4.2}$$

where W is the width of the image and H is the height of the image.

So, a total of $k = 9$ anchors at each pixel. For the entire image, the total of anchors is WHk. This algorithm presents robustness against translations, so it is a translational invariant algorithm [87].

Classify anchor boxes

For the classification of each anchor box, Intersection over Union (IoU) distance metric it's used. IoU is applied to define the extent of anchor box overlap with the target object, which can be also called a ground truth object [87]. The greater the overlap region, the greater the IoU.

The bounding boxes are not directly predicted by the network, although the network predicts the probabilities and refinements that correspond to the anchor boxes [70]. The network returns a unique set of several anchor box predictions for a different object size. The final feature map contains object detections for every single class.

Loss Function

The loss function of the RPN is the sum of classification (cls) and regression (reg) loss. It can be represented as follows [87]:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum L_{cls}(p_i, p_i^*) + \frac{\lambda}{N_{reg}} \sum p_i^* L_{reg}(t_i, t_i^*) \tag{4.3}$$

where i is the index of an anchor; p_i is the probability of anchor i being an object or not; p_i^* is the ground-truth label, which is 1 if the anchor is positive, and 0 if the anchor is negative; t_i is a vector of 4 parameterized coordinates of the predicted bounding box; and t_i^* is the ground-truth box associated with a positive anchor. The classification loss, L_{cls}, represents log loss over two classes: object or not an object.

To obtain the regression loss can be used the following formula [87]:

$$L_r eg(t_i, t_i^*) = R(t_i - t_i^*) \tag{4.4}$$

where R is the robust loss function. The variables N_{cls} and N_{reg} are the normalizations, and λ is 10 by default and scales the classifier and regressor on the same level.

4.3 Detection Head (SSD)

Single Shot Detector (SSD) setup to perform 3D object detection. The objective of the SSD network (Fig.4.5) is to generate 2D bounding boxes on the features generated from the backbone layer of the Point Pillars network. Several important reasons for choosing SSD as a one-shot bounding box detection algorithm are: fast inference and great accuracy. The single shot detector (SSD) uses features from the network VGG16 [88].

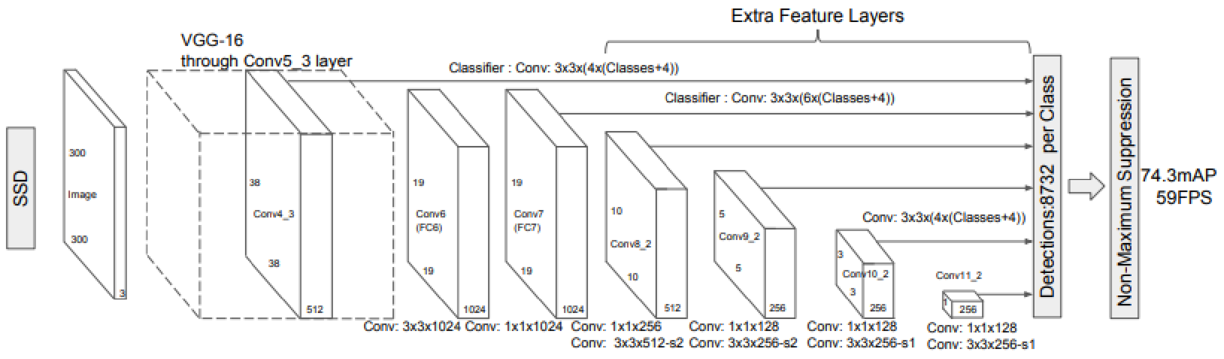


Figure 4.5: Single Shot Detector Architecture [61].

4.3.1 VGG16 Model

The VGG model stands for Visual Geometry Group and it's from Oxford; it's a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The "deep" refers to the number of layers, for example, VGG-16 has 16 convolutional layers. In the structure of the VGG16 neural network, as shown in Figure 4.6, all convolution layers have a 3×3 convolution core size, 1 step size, and there is a maximum of five pooling layers, which all are 2×2 and the step size is 2. There are three full connection layers, where the third layer symbolizes the label categories. The final layer is a softmax layer. All hidden layers are followed by a ReLU nonlinear activation function [103].

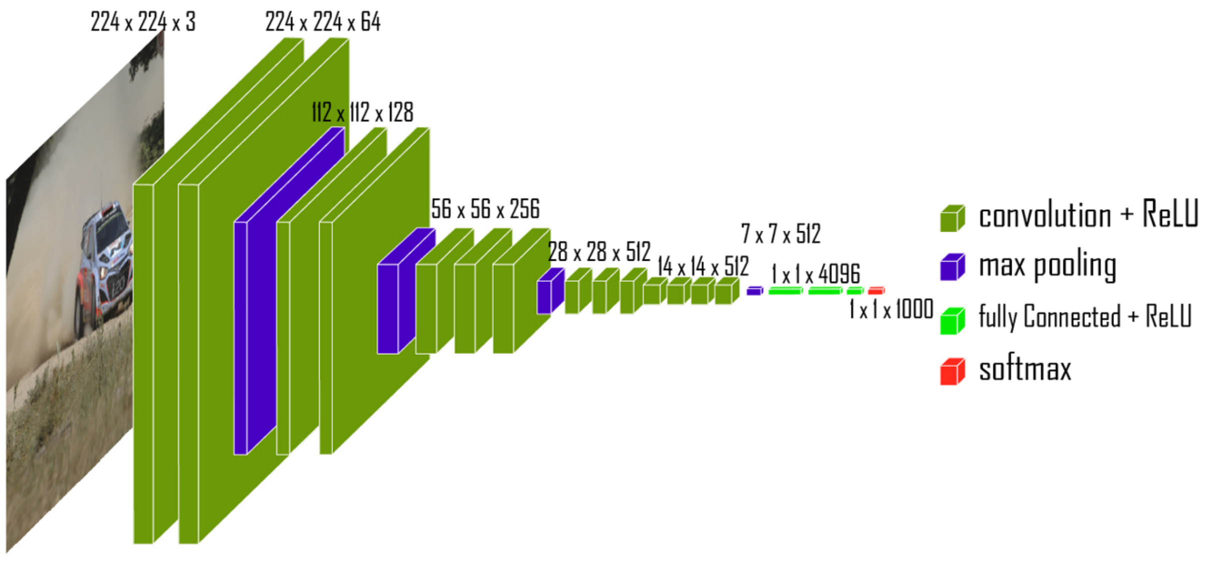


Figure 4.6: VGG16 Model [81].

4.3.2 VGG 16 Transfer Learning Model

Transfer learning is a machine learning mechanism that takes a pre-trained model on a large dataset and transfers its knowledge to a smaller dataset, which can be called the target field so that the target field can reach a better learning effect [25]. VGG16 is a deep large convolutional neural network that extracts and refines images in depth. Firstly, the convolutional layers extract the image features, whilst the network layers that are deeper handle specific tasks [117].

In the migration learning process, pre-trained models can be applied, retaining the lower layers' weights and the higher layers, and adjusting the relevant parameters. Since the source domain data have different distributions, the model is adjusted using the dataset of the target domain by removing the original upper layer, inserting a new output layer, and adding up a softmax function to classify the new problem [117]. A process of migrating the model, by transferring the weights trained in the source field and adjusting the model using the target field, simplifies the model training process in a new field problem.

5 Data Augmentation

5.1 Data augmentation for 2D images

In deep neural networks, training data has a crucial role in learning to perform tasks. However, compared to the real-world complexity, training data has a limited quantity, so to enlarge the training set and maximize the knowledge of a network, data augmentation (DA) is required. A conventional DA method is a global augmentation that learns transformations invariance in image recognition chores, such as random cropping, random scaling, random erasing, color jittering, etc [55].

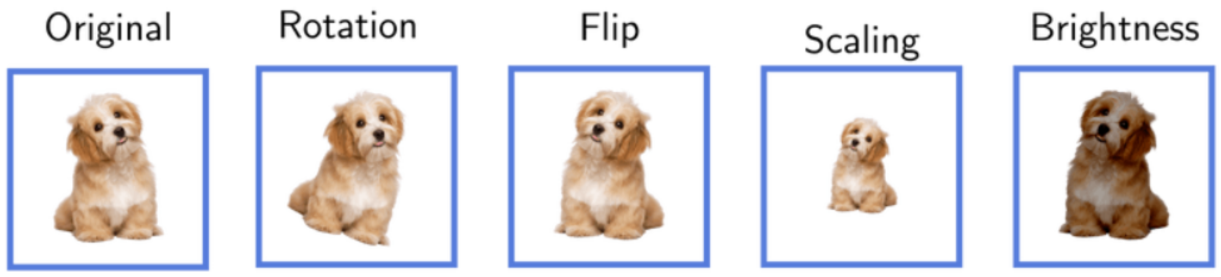


Figure 5.1: Some examples of data augmentation for 2D images [5].

Another approach of DA is local augmentation which generates new training data performing diverse mix operations. MixUp is an example of local data augmentation [113]. MixUp [120] generates new training data using convex combinations of the input pixels/feature and the output labels. Cutmix [120] swaps removed regions with a patch from other images instead of mixing the whole images. Cutblur [119] (Fig.5.2) cuts a low-resolution patch and swaps it with the corresponding high-resolution image region and vice versa, which is practically the same as making the image partially sparse. This method allows the model to understand "how" and "where" when super-resolves the image [26].

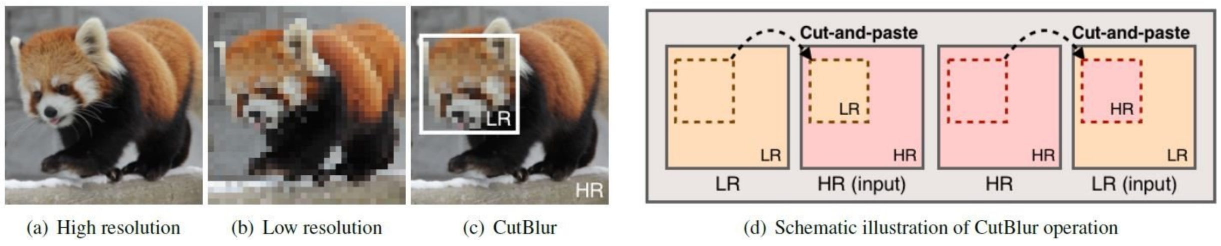


Figure 5.2: Example of Cutblur on 2D images [118].

In addition to global and local DA approaches, random transformation is a well-known DA technique for 2D data [120]. As an alternative to random transformations of training data, previous works tried to generate augmented samples with image combination [52], generative adversarial network (GAN) [99], Bayesian optimization [106], and image interpolation [24] in the latent space from the original data. However, these methods generate different data comparing the original data, resulting in untrustworthy samples. Interpolation-based approaches [52] can involve pixel-wise interpolation for images. For example, Smart Augmentation [52] merges samples from the same class to generate augmented data. Due to disorderly properties and irregular structure, these methods cannot be applied to point clouds [95].

An optimal combination of predefined transformation functions is another approach to augment the training samples [55]. AutoAugment [28] suggests a reinforcement learning strategy to generate symbolic transformation, and then learn DA policies from the data. However, AutoAugment is not computationally practical for large-scale problems. Soon after, Fast AutoAugment finds a more efficient search strategy. This method explores advanced hyper-parameter optimization methods to strike better transformations for augmentation. Although it sees the best transformations, it is restricted to discovering a fixed augmentation strategy for all training samples. Contrary to these methods, PointAugment [55], instead of discovering a fixed augmentation strategy for all the training samples, generates the transformation functions based on the individual training sample properties and the network capability during the training process.

5.2 Data augmentation for 3D point clouds

Since 3D object recognition datasets, which include KITTI datasets, have a limited amount of samples, increasing the size of the data is one of the ways to reduce overfitting and improve performance. [26]. As with 2D computer vision tasks, one direct approach is to embrace a global augmentation such as translation, random flipping, shifting, scaling, jittering, and rotation, which can be directly incorporated for expanding 3D objects [55]. Oversampling was also used to solve the foreground-background class imbalance problem by increasing the number of objects within the point cloud so that the difference between the number of points belonging to an object and the background in the point cloud is minor after applying this technique.[26].

Previous works such as PointNet and PointNet++ [84] use DA techniques of random rotation about the up-axis scaling, random rotation with perturbations, random shifting, and random jittering of points on the input object sample. RS-CNN and DensePoint followed identical DA strategies with little variations [95]. Regardless of these DA augmentation techniques (such as: random rotation, random shifting, random jittering.) with effectiveness on the models, these data augmentation methods do not fully utilize the point cloud richer information when compared to the counterparts for 2D images [26].

Several studies investigated augmenting local structures of point clouds [113]. PointAugment [55] is an auto-augmentation network for shape-wise transformation and pointwise displacement based on the individual training sample's properties and the network capability under the training process. PatchAugment [95] explores data augmentation in regional areas. PointWOLF [48] applies weighted transformations in local neighborhoods to increase the 3D objects' diversity. However, the works mentioned earlier aren't appropriate for scene-level point clouds because they focus on object-level augmentation.

Other studies investigated the mixing idea for augmenting point clouds. PointMixUp [23], for example, interpolates 3D objects to generate new samples for training. PointCutMix [50] supersedes point object subsets with that of other objects to improve training data. However, both works focus only on object-level augmentation. Several studies also explore scene-level mix. For example, GT-Aug cuts instances and attaches them to other LiDAR scans for the object detection task. PolarMix [113] can accomplish both object-level and scene-level augmentation. PolarMix designs are aligned with LiDAR-specific data properties such as partial visibility and density variation, thus guaranteeing high fidelity and effectiveness of the augmented point clouds. PolarMix is generic and applicable to object detection tasks.

Patch-based DA methods for 2D have improved performance [95]. Part-aware [26] firstly extends 2D image patches to 3D partitions and after extends 3D partitions to 3D point clouds. Part-aware (Fig.5.3) applies five distinct DA types to different partitions, adds robustness to the network, and improves performance.

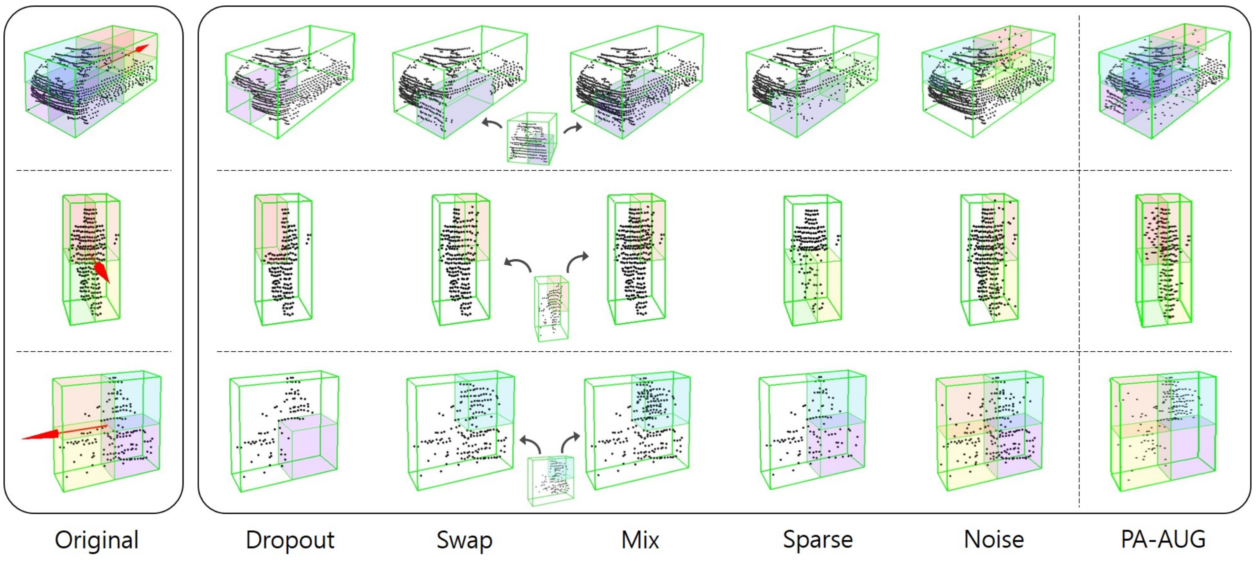


Figure 5.3: Part-Aware Data Augmentation for 3D Object Detection in Point Cloud [26].

5.3 Global and Local Data Augmentation

Data augmentation techniques can be divided into two classes, global and local data augmentation [40]. Global data augmentation refers to operations that apply transformations to the entire point cloud or scene. In turn, local augmentation focuses on specific regions of the point cloud. Thus, global operations aim to improve the robustness and generalization of 3D object detectors without introducing local bias. This strategy also enhances model generalization and is more effective for real-world applications, such as 3D object detection for autonomous vehicles on urban roads. However, it is important to ensure that the operation preserves the spatial and geometric relationship between the objects and the foreground/background of the scene. In addition, the operations should also preserve the objects' shapes, since they are important for classification.

This work studied global and local augmentation techniques for object detection in urban roads, using the KITTI dataset. Among the global augmentation techniques, operations for geometric and spatial transformation, noise-based, and sampling-based operations were selected. These techniques are described as follows (and for more details, Appendix 6 contains the hyper-parameter values used in each data augmentation technique).

5.3.1 Geometric and Spatial Transformations

The geometric transformations consist of rigid transformation (*i.e.*, rotation, translation, and scaling) in the entire point cloud. They help the model to become invariant to the orientation, translation, and scaling (*i.e.*, affect the size of objects) of the scene. Some operations are: rigid transformations (translation + rotation); similarity transformations (translation + rotation + isotropic scaling); affine transformations (translation + rotation + arbitrary scaling + shearing); and random flips [83].

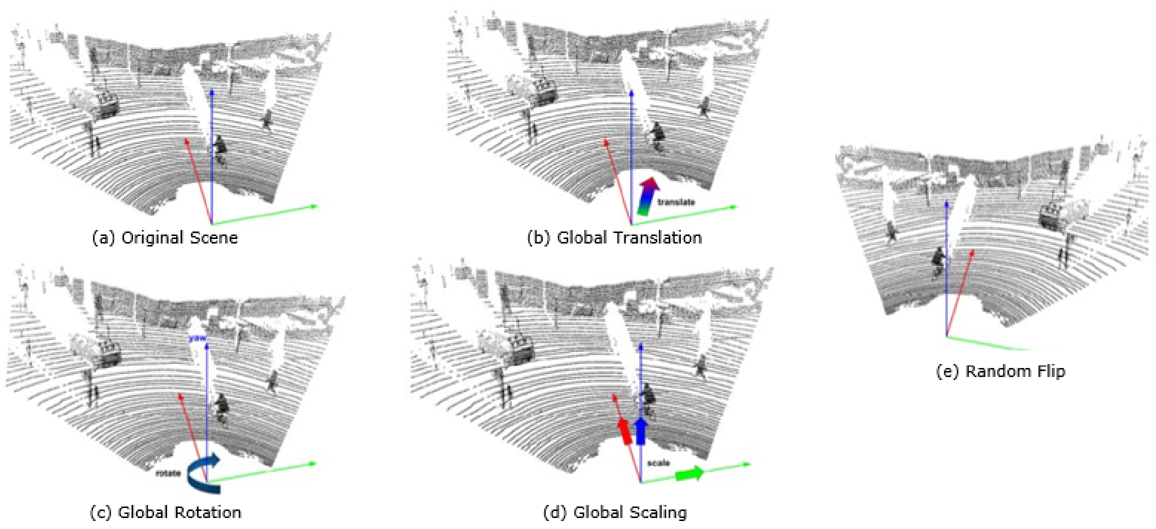


Figure 5.4: Geometric and Spatial transformations on point cloud [40].

5.3.2 Noise-based Operation

These operations involve introducing random or controlled variations to the original data to create new samples with minor perturbations, with the aim of increasing data diversity and including noise that may actually be present in the data acquisition process. Some operations are: jitter; Gaussian noise; and shuffling.

- **Jittered Points**
  - Jitter adds uniform noise to point coordinates that randomizes the point locations by slightly altering their values [86]. This method is helpful when precisely aligned data can be a problem.
- **Gaussian Noise**
  - Adding noise to the input data allows more data gained for the deep neural network to train on and improves the robustness and generalization of CNNs [75].
- **Shuffling**

  - One way to improve invariance to permutations is by training the model with shuffling of points [7].

5.3.3 Sampling-based Operation

Sampling-based operations involve manipulating a dataset by selecting certain elements or generating new ones based on specific criteria. These operations are typically used to manage the data size, quality, or representation. Sampling-based operations can be categorized into oversampling, subsampling, upsampling, and downsampling. While a diverse array of sampling-based operations exists, this experiment only evaluates the effectiveness of the upsampling operation.

- **Upsampling**
  - Upsampling is a technique used to increase the spatial resolution of the input feature map. One of the main advantages is that upsampling increases the resolution of the output, which can help neural networks localize features more accurately.

6 Developed Work and Experiments

This chapter describes the design of the experiments, the techniques, and the evaluation metrics used. Figure 6.1 shows a general flowchart for 3D object detection using point clouds with data augmentation. The first step is the data acquisition and representation, which consists of obtaining the point cloud from a LiDAR sensor and then converting it to a data representation (*e.g.*, point cloud, range image, grid, or voxels). Each sample has a list of annotations associated with it. Afterward, the samples are divided into train, validation, and test sets. The training set undergoes data augmentation operations to either increase the number of samples, attenuate data imbalance or improve representativeness. Finally, the data feeds a deep learning model that learns how to detect and classify objects using the training and validation sets. This final model is evaluated using the testing dataset.

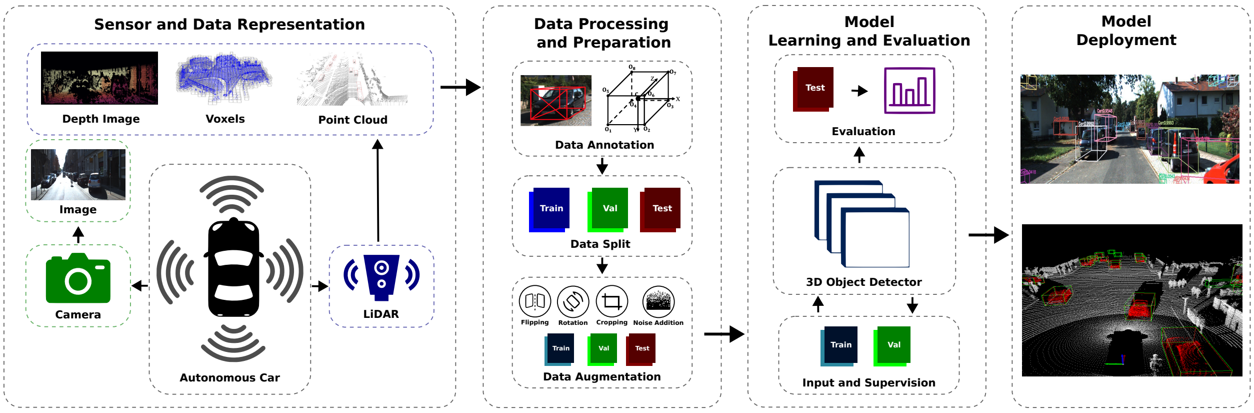


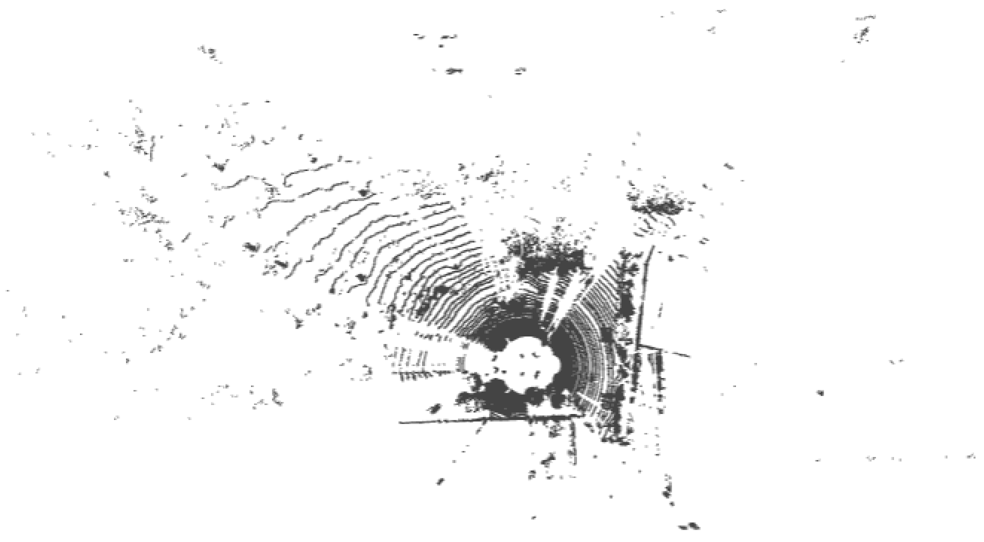
Figure 6.1: Flowchart for Evaluation of Point Cloud Data Augmentation for 3D-LiDAR Object Detection. Based on [68].

The following subchapters will provide a more detailed explanation of each step in the flowchart to clarify the developed work.

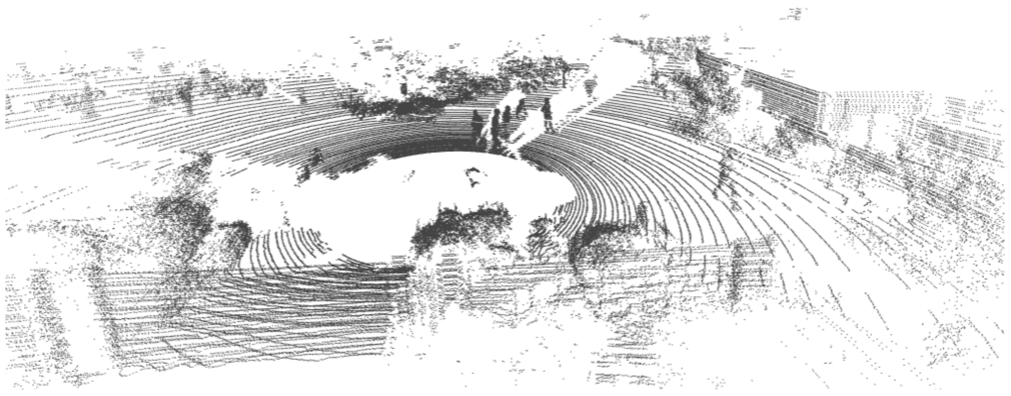
6.1 KITTI Dataset

The KITTI dataset used in this work has 2D images, 3D point clouds, and label files.

Figure.6.2 is an example of the LiDAR data from the KITTI dataset. The images show two different views from the same point cloud for a better understanding. In the first image (figure.6.2(a)) is very hard to perceive something, but in the second image (figure.6.2(b)), some persons can be seen in the street. In this work, both view perspectives were used.



((a)) BEV



((b)) Front of View

Figure 6.2: Velodyne point clouds.

All the rows in the KITTI dataset's label files are composed by type of object (car, pedestrian, cyclist, etc), truncated float (where 0 means truncated, and 1 otherwise), occlusion state (0=fully visible, 1=partly occluded, 2=largely occluded, 3=unknown), alpha observation angle, 2D bounding box parameters (center, length, width), 3D object dimensions (height, width, length), 3D object location [x, y, z] (in camera coordinates), and yaw angle, respectively.

Since the bounding box information provided in the ground truth (GT) label is in the camera coordinate system, external parameters are crucial to converting it to the LiDAR coordinate (Fig.6.3) system during training.

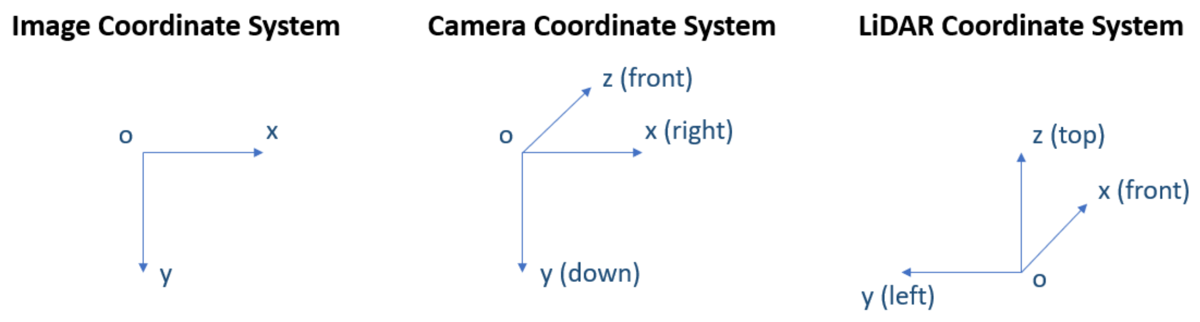


Figure 6.3: KITTI Coordinate System.

6.2 Data Augmentation

The hyperparameters that control the data augmentation techniques used in the code-base are presented below.

Key data augmentation hyperparameters include:

```
global_rot_parameter=[-0.78539816, 0.78539816],
global_scale_parameter = [0.95, 1.05],
global_translation_parameter = [0, 0, 0],
random_flip_ratio=0.5,
jitter_sigma=0.01,
jitter_clip=0.05,
gaussian_noise=dict(
    num_try=100,
    translation_std=[0.25, 0.25, 0.25],
    rot_range=[-0.15707963267, 0.15707963267]
    ),
upsampling_num_samples=4,
local_rot_range=[-0.78539816, 0.78539816],
local_std_range=[0.1, 0.25] # local gaussian noise
```

For more details about the meaning of the hyperparameters, see Annex B.

6.3 Model Learning

6.3.1 Defining Initial Configurations

Since LiDARs have a wide range, limits of the x, y, and z dimensions are established to focus on a smaller region. These limits define the region of interest (RoI) in which it will be predicting the bounding boxes. The NMS method will be applied to the generated bounding boxes. The NMS method walks through all classes and, for each class, looks for possible overlaps (IoU — Intersection over Union) between all bounding boxes. If the $IoU > thr$ between two boxes of the same class, the algorithm determines that the two boxes refer to the same object and throws away the box with the lower confidence score. Based on $nms\_thr = 0.01$, NMS filters out overlapping boxes. To avoid too many remaining bounding boxes, $max\_num = 50$ is defined. Other configurations also defined include the maximum number of points, the maximum number of voxels, etc. These configurations allow transferring the point cloud from 3D coordinates to Pillar coordinates in the Point Pillars detection pipeline.

Configurations:

```
#Voxelization
nclasses=3
voxel_size=[0.16, 0.16, 4]
point_cloud_range=[0, -39.68, -3, 69.12, 39.68, 1]
max_num_points=32
max_voxels=(16000, 40000)

#Anchors
ranges = [[0, -39.68, -0.6, 69.12, 39.68, -0.6],
          [0, -39.68, -0.6, 69.12, 39.68, -0.6],
          [0, -39.68, -1.78, 69.12, 39.68, -1.78]]
sizes = [[0.6, 0.8, 1.73], [0.6, 1.76, 1.73], [1.6, 3.9, 1.56]]
rotations=[0, 1.57]

#Training
assigners = [
    {'pos_iou_thr': 0.5, 'neg_iou_thr': 0.35, 'min_iou_thr': 0.35},
    {'pos_iou_thr': 0.5, 'neg_iou_thr': 0.35, 'min_iou_thr': 0.35},
    {'pos_iou_thr': 0.6, 'neg_iou_thr': 0.45, 'min_iou_thr': 0.45},
]

#Validation and Test
nms_pre = 100
nms_thr = 0.01
score_thr = 0.1
max_num = 50
```

During the model training, the AdamW optimizer is used to modify the typical implementation of weight decay in Adam by decoupling weight decay from the gradient update. AdamW improves training loss (to see more details about loss functions, see Annex A), and the generalized models are much better than the trained models with Adam.

The learning rate (LR) schedule optimizer is also used during the model training to adjust the LR based on the number of epochs during the model training. The model was trained for 160 epochs with an LR initial value of 0.00025. To determine the optimal LR, optimizer, and number of epochs, an iterative trial and error process was conducted until the best hyperparameters were identified for the model.

6.3.2 Building the Point Pillars Network

Feature Encoder

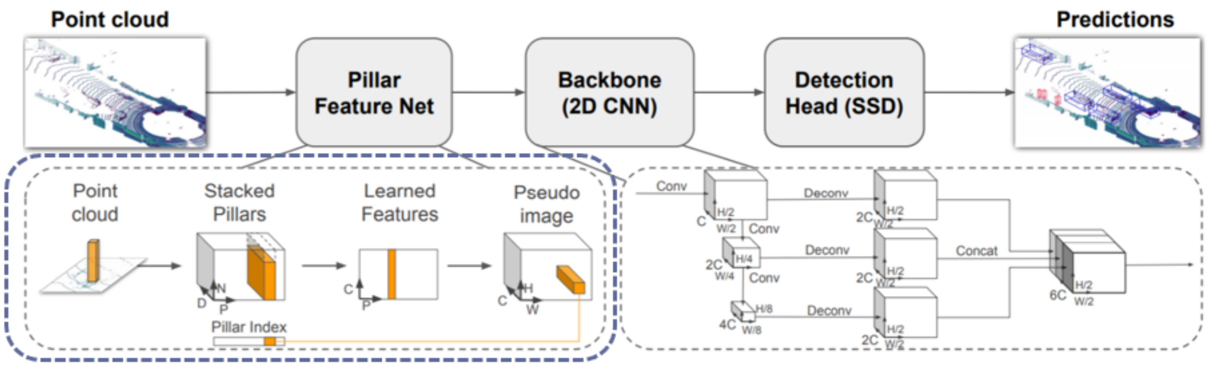


Figure 6.4: The first step of PointPillars Architecture: **Pillar Feature Net**.

The feature encoder (Figure 6.4) can be divided into two steps: pillar layer and pillar encoder.

- **Pillar Layer:** Based on predefined voxel size, the point cloud with N points is divided into $(432, 496, 1)$ pillars. The pillars can take the value $P = 16000$ in the case of the training set or $P = 40000$ in the test set. Each pillar chooses $M = 32$ points and, if $M < 32$ points, a padding operation with null points (0) is applied. In the pillar layer, the data shape changes from $(N, 4)$ to $(P, M, 4)$, and it records the position in coordinates of the pillars in the $(432, 496)$ map and the number of practical points in each pillar.

- **Pillar Encoder:** Firstly, the pillar encoder performs de-mean encoding on the points in each pillar, transforming $(P, M, 4)$ into $(P, M, 3)$. After decentralizing the valid points in each pillar, this is $(P, M, 2)$, the original $(P, M, 4)$ is combined with the results of de-mean encoding and de-center encoding and gets $(P, M, 9)$ vector. To the obtained vector will be applied convolution kernel pooling: $(P, M, 9) -> (P, M, 64) -> (P, 64)$. According to the coordinate position of the pillars on the map, the resource scatter of P pillars is placed on the $(432, 496)$ resource map, and the map of resources of $(64, 496, 432)$ is obtained and can be registered as (C, H, W).

Backbone

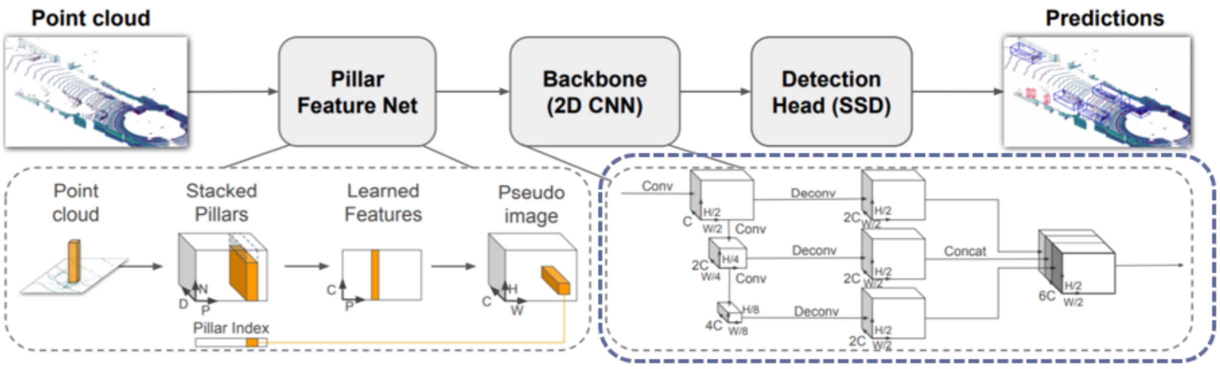


Figure 6.5: The second step of PointPillars Architecture: **Backbone 2D CNN**.

A backbone (Figure 6.5) is a simple layer that operates in 2D, which can be seen as a combination of $Conv2d + Bn + ReLU$. The input layer is the pillar features encoded in an x-y grid. The x-y grid is converted into different scales with features extracted from them, and in the end, all the features with distinct scales are concatenated into a single tensor (Figure 6.6). Since the input values were $(C, H, W) = (64, 496, 432)$ after the combination of $Conv2d + BN + ReLU$, the final data shape will be $(256, 62, 54)$.

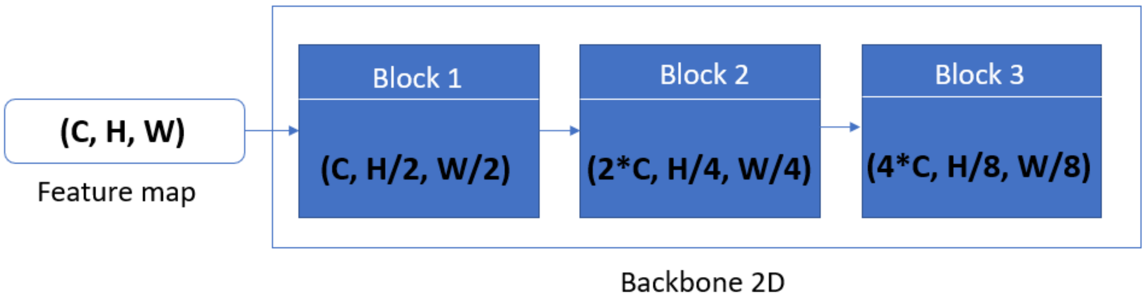


Figure 6.6: Backbone 2D CNN.

Detection Head

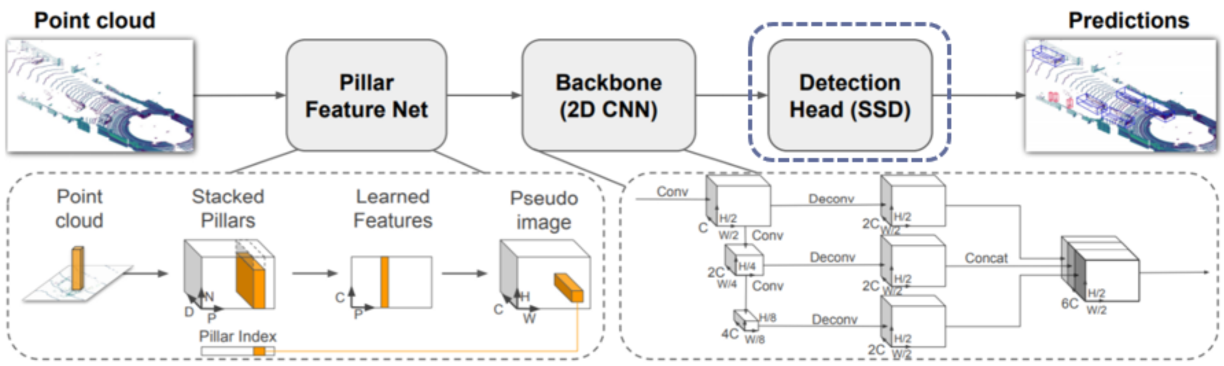


Figure 6.7: The third step of PointPillars Architecture: **Detection Head (SSD)**.

The SSD approach (Figure 6.7) is based on a feed-forward convolutional network that generates bounding boxes with a fixed size and scores for the case that object class instances existed in those boxes, followed by a non-maximum suppression process for the final detections [61]. The first network layers (as seen in the image 4.5 of the chapter 4.3) are based on a typical architecture used in high-quality image classification that will be called the base network. Then an auxiliary structure, specifically a decoder network, is added to the network to generate the detections (Figure 6.8).

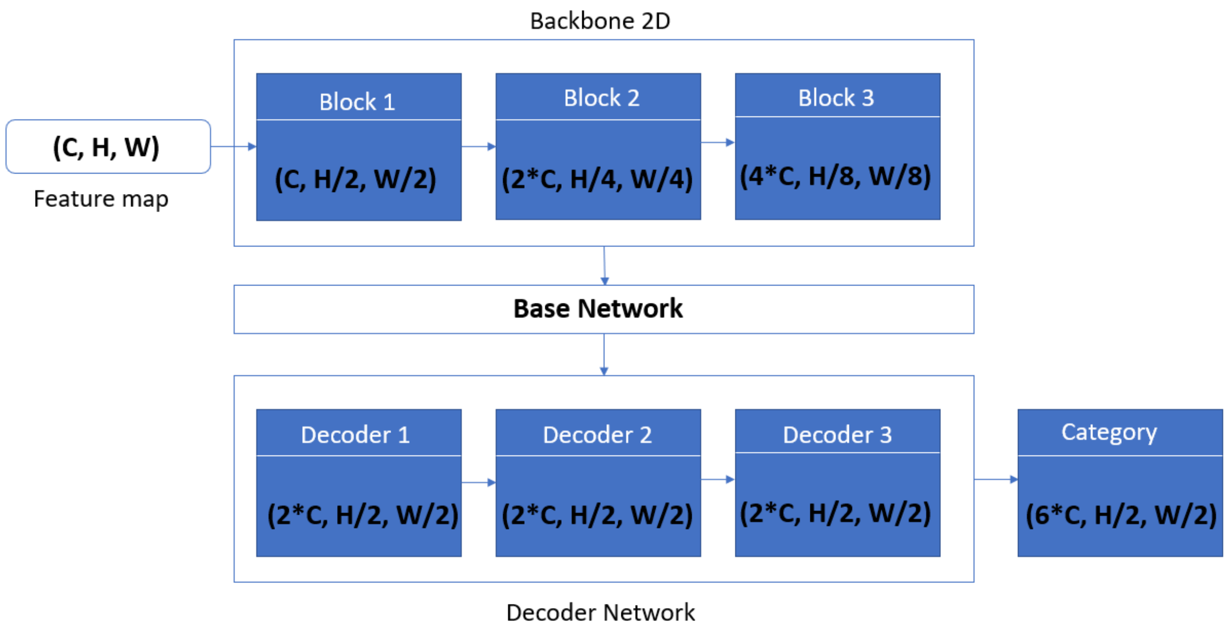


Figure 6.8: Detection Head insert on Backbone 2D CNN.

The structure of the base network in Figure 6.8 is represented in Figure 6.9.

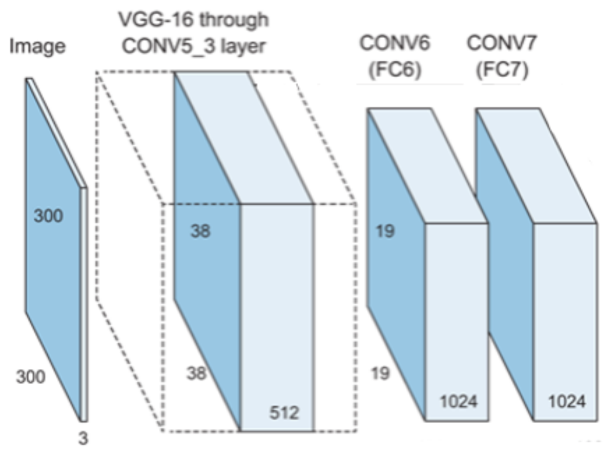


Figure 6.9: Base Network Structure.

PointPillars has three anchors of different sizes, and each anchor size has two angles, so there are six anchors in total. The network is trained in 3 categories: Pedestrian, Cyclist, and Car.

- **Classifier:** $(6 * C, H/2, W/2) - > (6 * 3, H/2, W/2)$, i.e. $(18, 248, 216)$
- **Box Regressor:** $(6 * C, H/2, W/2) - > (6 * 7, H/2, W/2)$, i.e. $(42, 248, 216)$
- **Orientation Classifier:** $(6 * C, H/2, W/2) - > (6 * 2, H/2, W/2)$, i.e $(12, 248, 216)$

So the data shape changes from $(6{*}C, H/2, W/2) -> [(6{*}3, H/2, W/2), (6{*}7, H/2, W/2), (6{*}2, H/2, W/2)]$.

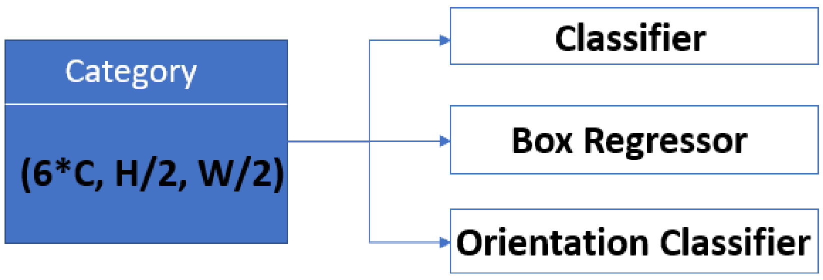


Figure 6.10: Final Detection.

6.3.3 GT value generation

The three branches of the detection head (classifier, box regressor, and orientation classifier) predict the category based on the anchor, the category of the bounding box (relative to the offset and size ratio of the anchor), and the category of the rotation angle. So how the ground truth (GT) value corresponding to each anchor during the training is obtained?

Firstly the anchors are generated. For three different categories, the anchor contains a total of 3 sizes: $[0.6, 0.8, 1.73]$, $[0.6, 1.76, 1.73]$, and $[1.6, 3.9, 1.56]$, and a rotation arc: 0 and $\pi/2$. To get the anchor tensor, 3*2 anchors are placed on each position on the feature map with the size $(H/2, W/2)$, generating an anchor tensor with shape $(H/2, W/2, 3, 2, 7) = (248, 216, 3, 2, 7)$. Then correspondence between anchors and GT bounding boxes is established. Here, taking the anchor with a size of $[0.6, 0.8, 1.73]$ as an example, first, the IoU (see Appendix B) with n GT bounding boxes and $248 * 216 * 2$ anchors is calculated, and after the anchors are successively divided into positive and negative anchors.

- **Positive anchor: (1)** If the maximum IoU between the anchor and all GT bounding boxes is greater than pos_iou_thr (0.5), then this anchor is positive, and is responsible for the GT bounding box with the maximum IoU; **(2)** For each GT bounding box, select the anchor with the greatest IoU, if its IoU is greater than min_iou_thr (0.35), then this anchor is positive, and is responsible for the GT bounding box.

- **Negative anchor:** If the maximum IoU between the anchor and all GT bounding boxes is less than neg_iou_thr (0.35), then the anchor is a negative anchor.

Some anchors do not belong either to the group of positive or neither negative anchors, which are anchors whose maximum IoU with GT bounding boxes is between 0.35-0.5.

After the correspondence between the anchors and the GT bounding boxes, it is known which anchors are positive and negative. It is also known which GT bounding box the positive anchor corresponds to. Finally, the GT value output by the detection head will be obtained in three different categories: category classifier, box regression, and orientation classifier.

- **Category classifier:** Positive and negative anchors take part in the supervision of category classification. The output here is the result of 3 sigmoids, that is, the anchor is the confidence of the 0, 1, and 2 classes. The GT value of the negative anchor is (0, 0, 0). The GT value of the positive anchor is (0, 0, 1) or (0, 1, 0) or (1, 0, 0).

- **Bounding box regression:** Positive anchors take part in the supervision of the bounding box regression. For the anchor $(x^a, y^a, z^a, w^a, l^a, h^a, \theta^a)$ and its corresponding GT bounding box $(x^{gt}, y^{gt}, z^{gt}, w^{gt}, l^{gt}, h^{gt}, \theta^{gt})$, the position offset and the size ratio of the GT bounding box and the predicted anchor by the model:

  $\Delta x = \frac{x^{gt}-x^a}{d^a}$, $\Delta y = \frac{y^{gt}-y^a}{d^a}$, $\Delta z = \frac{z^{gt}-z^a}{d^a}$, where $d^a = \sqrt{((w^a)^2 + (l^a)^2)}$

  $\Delta w = log\frac{w^{gt}}{w^a}$, $\Delta l = log\frac{l^{gt}}{l^a}$, $\Delta h = log\frac{h^{gt}}{h^a}$

  $\Delta\theta = sin(\theta^{gt} - \theta^a)$

- **Orientation classifier:** Positive anchors take part in the supervision of the angle classification. If the angle of the GT bounding box corresponds to the anchor is in $[0, \pi]$, then the label is 0; if the angle of the GT bounding box corresponds to the anchor is in $[\pi, 2\pi]$, then the label is 1. The orientation classification is important because if the orientation of two cars differs by 180°, for example, then $sin(\theta^{gt} - \theta^a) = sin((\theta^{gt} + \pi) - \theta^a)$, that is, the regressive $\Delta\theta$ is the same since the orientation of the car cannot be distinguished.

6.4 Evaluation metrics

To evaluate the experimental results, classification and regression quantitative metrics were employed. The classification metrics are estimated based on a confusion matrix with true positive (TP), false negative (FN), false positive (FP), and true negative (TN).

- **Recall**

$$Recall = \frac{TP}{TP + FN} \tag{6.1}$$

- **Precision**

$$Precision = \frac{TP}{TP + FP} \tag{6.2}$$

- **F1 score**

$$F1\ score = \frac{2 \times precision \times recall}{precision + recall} \tag{6.3}$$

- **Mean average precision** (mAP)

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k \tag{6.4}$$

  where n is the number of classes and AP_k is the average precision of class k. Average precision (AP) is one way of calculating the area under the PR (precision-recall) curve and can be defined as:

$$AP_k = \frac{TP(k)}{TP(k) + FP(k)} \tag{6.5}$$

The mean average precision (mAP) is calculated for the 2D bounding box, AOS (average orientation similarity), BEV (bird's eye view), and 3D bounding box detections. The mAP of each detection method was calculated for three different degrees of difficulty (0 - easy, 1 - medium, or 2 - difficult). The difficulty is defined according to the height of the 2D bounding box, the degree of occlusion, and the degree of truncation.

AOS (average orientation similarity) is used to evaluate orientation, usually considered together with a 2D bounding box because a 2D bounding box is based on an axis-aligned bounding box during the evaluation. Briefly, AOS measures the similarity between the predicted rotation and the ground truth rotation angles:

$$AOS = \frac{1 + cos(\alpha^{gt} - \alpha^{p})}{2} \tag{6.6}$$

6.4.1 Steps for AP Calculation

- **1.** Calculate the 3D IoU, which is used to determine whether a determinate bounding box matches the GT box ($IoU > 0.7$).

- **2.** According to the category and the difficulty, GT bounding boxes and detected bounding boxes are selected.

- **3.** Determine the score threshold corresponding to the point pair (P_i, R_i) in the PR curve.
  - Create an empty score collection, S;
  - For each GT bounding box, select the largest detected bounding box (d_b) that has not been matched and its $IoU > 0.7$ as a matching box. The predicted score is added to the collection S; if no conditions are met, there is no need to d_b. The predicted score is added to the collection S middle;
  - Collect pairs S. Intermediate scores are ranked from high to low;
  - Calculate the number of GT bounding boxes according to S. Calculate the recall at $|S|/|GT\_bbox|$. The scores create a threshold set S^*.
- **4.** Calculate the PR curve and AP.
  - Calculate TP (all matching detected bounding boxes), FN (all unmatched GT bounding boxes), and FP (all unmatched $predicted\_scores \geq s_i^{\mathbf{(1)}}$);
  - Calculate:
    - Precision
    - Recall
  - The PR curve and AP calculation are based on the point pair (P_i, R_i).

(1) s_i: each threshold set of S^* its composed by s_i.

6.4.2 Evaluation of the Test set: illustration and metrics

Figures 6.11 and 6.13 show some qualitative results (as 3D detection) by illustrating the 3D-boxes on a given frame of the test set belonging to KITTI dataset image.

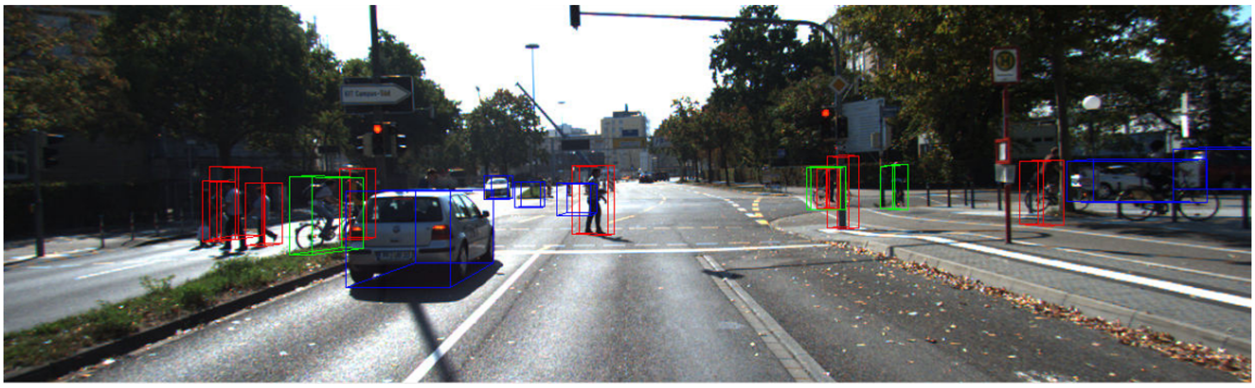


Figure 6.11: 3D Object Detection from 2D image.

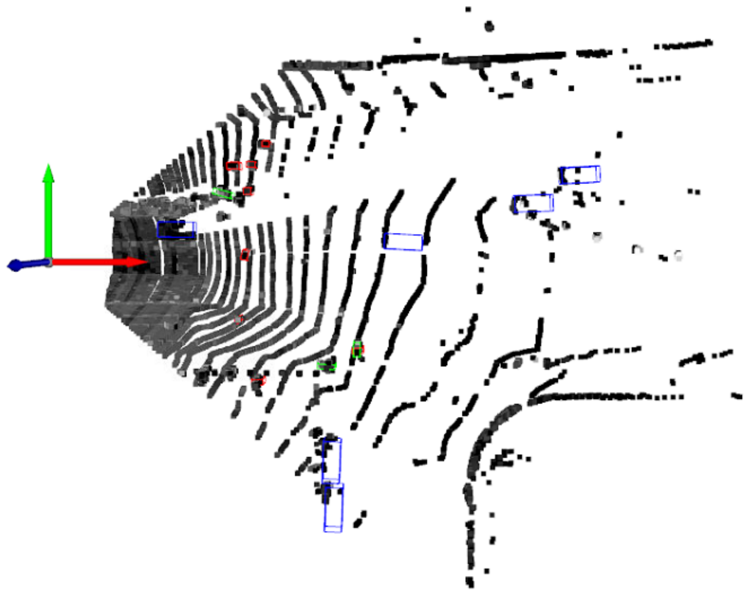


Figure 6.12: 3D Object Detection from Point Cloud.

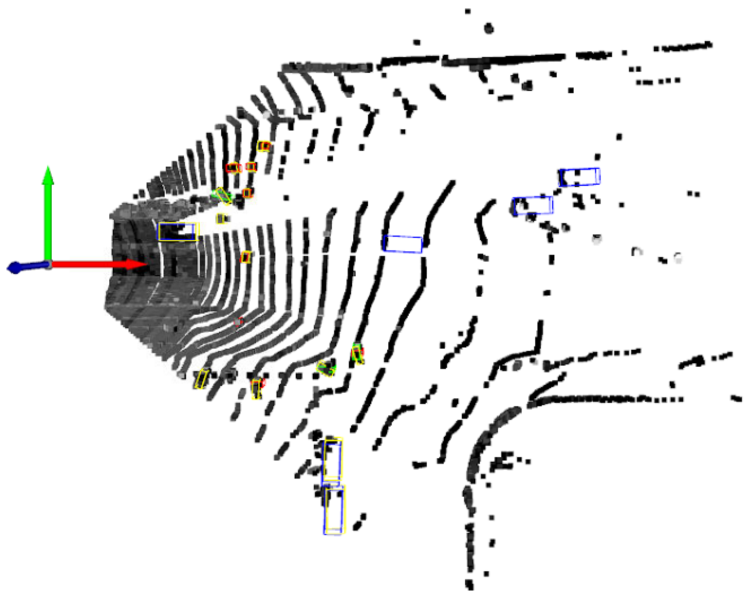


Figure 6.13: 3D Object Detection from Point Cloud (with GT).

The respective legend to guide the understanding of the testing results are:

- **Red:** Pedestrian
- **Green:** Cyclist
- **Blue:** Car
- **Yellow:** GT (Ground Truth)

7 Analysis and discussion of results

7.1 Results

Table 7.5 shows the average precision (AP) of the different classes and the difficulty levels regarding the detection methods, while Table 7.2 presents the mean average precision (mAP) of the different difficulty levels per class. To evaluate the model, classification performance-measures such as F1 score, recall, and precision, were calculated (see A.1 from annex A).

Analyzing Table 7.5 and Table 7.2, it is observed that the 'Car' class obtained the best performance among all classes since it represents the majority of the annotated labels, and also the size and rectangular shape of the 3D bounding belonging to this class help the prediction of its orientation when compared to the remaining classes shape (*i.e.*, 'Pedestrian' and 'Cyclist').

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	46.6385	42.6914	40.5610	58.1782	43.5697	40.0704	86.1092	77.9969	77.2292
AOS	27.7677	26.5258	25.3001	54.3573	39.0934	35.9742	85.3043	76.4905	74.8277
BBOX_BEV	37.6112	36.6923	33.5501	52.1940	37.6584	35.9049	85.6656	77.6653	76.6165
BBOX_3D	26.9381	25.1430	24.1276	44.2416	30.3132	29.0027	66.5733	60.2184	54.6960

Table 7.1: AP values of different detection methods on the KITTI dataset.

Difficulty	**Easy**	**Moderate**	**Hard**
BBOX_2D	63.6420	54.7527	52.6206
AOS	55.8097	47.3699	45.3673
BBOX_BEV	58.4902	50.6714	48.6905
BBOX_3D	45.9170	38.5582	35.9421

Table 7.2: mAP values of different detection methods on the KITTI dataset.

7.1.1 Global Operations

In general, the obtained results have a small percent precision, therefore some data augmentation techniques were applied to the training data of the baseline model to increase the precision of the model. The first data augmentation techniques that were applied are geometric transformations, such as random rotation, scaling, translation, and flip on the points clouds, and the obtained individual results, respectively, are shown in the following Tables:

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	65.8472	62.1493	58.3800	80.4138	69.4442	67.2450	90.4396	88.5991	85.8627
AOS	46.5390	43.2790	40.1873	78.7066	61.2886	59.0642	90.0831	87.5234	84.4588
BBOX_BEV	56.6091	50.6296	47.1768	71.7167	58.6304	54.8938	89.4101	86.5404	79.5422
BBOX_3D	49.5107	44.2302	41.2375	68.0679	53.5679	50.8584	82.8154	74.0379	68.1278

Table 7.3: AP values of different detection methods on the KITTI dataset with random rotation data augmentation technique.

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	61.5757	58.6031	54.6101	79.0814	67.1805	63.1162	89.7583	85.3930	79.6553
AOS	34.7239	32.9851	30.6550	76.9454	61.7475	58.2367	89.5584	84.5648	78.4260
BBOX_BEV	57.6708	51.7335	47.7150	73.5661	57.4302	53.8111	88.6441	84.1217	78.4111
BBOX_3D	48.4136	44.0084	41.2220	69.4400	52.1917	49.3618	75.0672	66.0250	64.5190

Table 7.4: AP values of different detection methods on the KITTI dataset with scaling data augmentation technique.

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	53.6988	49.8462	47.8971	70.4430	50.3021	46.1745	86.6287	78.0526	76.9926
AOS	32.4891	30.7464	29.8796	67.2182	46.4454	42.8748	86.0683	76.7673	75.0623
BBOX_BEV	48.3293	44.7255	41.0463	65.5104	44.8617	42.5579	84.9580	77.1724	76.0925
BBOX_3D	39.0264	36.4320	33.4319	65.5104	44.8617	42.5779	84.9580	77.1724	76.0925

Table 7.5: AP values of different detection methods on the KITTI dataset with random translation data augmentation technique.

Class	Pedestrian			Cyclist			Car		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	57.3324	53.6357	51.9135	73.6033	58.1698	54.0455	88.4448	84.3588	78.7506
AOS	35.1320	33.2585	32.4012	72.1542	55.1395	51.3180	88.0094	83.2988	77.1851
BBOX_BEV	53.83350	48.1993	45.0751	71.5840	50.5661	47.6089	86.9917	83.1604	77.6882
BBOX_3D	45.6012	41.4136	37.8224	68.5832	46.9737	43.3134	70.9394	62.7611	60.7148

Table 7.6: AP values of different detection methods on the KITTI dataset with random flip data augmentation technique.

As discussed earlier, the main goal of applying geometric transformations is to enhance the model's ability to handle rotation, translation, scaling, and flip invariance. These transformations modify the data samples, making them more representative of these variations.

In general, the random rotation transformation resulted in more significant improvements in orientation and pose estimation for all classes compared to other geometric transformations. When compared solely to random flit (another operation that affects object orientation), random rotations yielded notable differences in performance. Specifically, the AOS metric for the Hard set showed improvements of approximately 7.79, 7.75, and 7.27 for Pedestrian, Cyclist, and Car, respectively. These results suggest that, between the two operations, random rotation alone can significantly enhance the classifier's robustness concerning object orientation. This simplifies the process by eliminating the need for two distinct methods.

Furthermore, concerning pose estimation, random rotation demonstrated superior overall performance. However, when evaluating the $BBOX\_3D$ metric, scaling outperformed random rotation by a small margin of approximately 1.38. Additionally, in the Car class for the easy, moderate, and hard subsets, random translation surpassed random rotation. This implies that a combination of random rotation and translation can offer better contributions to pose and orientation estimation. Nevertheless, if only one option is feasible, relying on random rotation should suffice for achieving satisfactory overall performance.

After the geometric transformations application on the point cloud, noise-based operations, such as jitter, Gaussian noise, and shuffling were explored and can be seen in the tables below.

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	53.3772	50.6876	48.4943	65.3464	48.3602	45.1187	87.8342	78.2490	76.9634
AOS	31.7557	30.4270	29.3447	63.3383	43.9017	40.9737	87.5738	77.3371	75.4795
BBOX_BEV	46.4298	44.3443	41.3281	61.5708	44.6678	43.2509	86.2903	77.7343	76.4139
BBOX_3D	37.6627	36.1600	33.1984	58.4343	41.5782	37.8097	66.8201	58.1882	53.9783

Table 7.7: AP values of different detection methods on the KITTI dataset with jitter points data augmentation technique.

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	61.2492	58.0822	54.4647	84.3089	69.6610	67.1053	89.4946	87.5695	85.0064
AOS	34.9442	33.4838	31.7364	82.3835	63.5745	61.2440	89.4026	86.9786	84.1780
BBOX_BEV	54.2564	49.7521	45.2796	80.1874	59.1168	56.4294	88.8394	85.7651	79.1107
BBOX_3D	48.8127	43.8644	39.9126	76.0929	55.2278	51.9355	79.8890	71.0105	66.2267

Table 7.8: AP values of different detection methods on the KITTI dataset with Gaussian noise data augmentation technique.

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	52.4766	49.0509	46.7121	71.0058	54.7001	52.1896	87.0185	78.1385	76.9038
AOS	31.3664	29.6963	28.7487	68.1336	50.3405	47.9886	86.4446	76.9225	75.1122
BBOX_BEV	47.0222	44.4001	40.6238	67.0049	47.9679	45.8521	86.2825	77.6774	76.2880
BBOX_3D	39.2020	36.3416	33.3256	64.3584	44.0106	41.8283	68.3598	58.6953	54.5362

Table 7.9: AP values of different detection methods on the KITTI dataset with shuffled points data augmentation technique.

Furthermore, the noise-based techniques also contributed to the detector's performance enhancement, but their impact was lower than that of geometric transformations. In general, the Gaussian noise had a more positive influence on orientation and pose estimation. When it comes to 2D and 3D pose estimation, Gaussian noise consistently outperformed all other operations, such as jittering points and shuffling, across all subsets—easy, moderate, and hard. Similarly, Gaussian noise demonstrated superior performance in orientation estimation, as evidenced by the analysis of the AOS metric. However, it's worth noting that when specifically considering this metric, random rotation achieved slightly better results.

These findings suggest that Gaussian noise excels in terms of overall performance compared to other noise-based operations, particularly in the context of pose estimation. On the other hand, for orientation estimation, the geometric operation of random rotation still maintains an edge over Gaussian noise in terms of performance.

Class	Pedestrian			Cyclist			Car		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	55.7855	51.9292	50.0641	70.5508	52.7233	50.4653	87.6876	78.3487	77.3512
AOS	34.0496	31.7363	30.9056	67.1535	48.8495	46.2775	86.9784	76.9475	75.1824
BBOX_BEV	49.0861	46.3124	42.6059	66.9836	46.5759	43.8467	85.6268	77.3618	76.2507
BBOX_3D	40.6971	38.1660	35.5330	64.6337	43.2629	41.8263	664.3221	57.8460	53.2556

Table 7.10: AP values of different detection methods on the KITTI dataset with upsampling data augmentation technique.

In conclusion, all data augmentation techniques increased the performance of the model when compared to the baseline model. The effects of the data imbalance were also attenuated. Moreover, random rotation, random translation, and Gaussian noise emerged as the primary contributors to pose and orientation estimation.

7.1.2 Local Operations

Once global augmentation techniques have been implemented, local ones are introduced, starting with local random rotations.

Class	Pedestrian			Cyclist			Car		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	31.3154	26.4510	23.6553	37.5467	33.5032	31.9275	82.7686	63.7786	58.8584
AOS	18.7430	16.9266	15.6738	33.2203	20.0641	19.2504	80.9542	61.7335	56.4501
BBOX_BEV	36.3634	30.5639	26.5352	30.6835	29.1837	27.5405	75.4780	57.9100	54.5686
BBOX_3D	23.0426	18.8785	16.8463	24.3210	22.1568	21.1199	55.0470	43.1295	38.8464

Table 7.11: AP values of different detection methods on the KITTI dataset with local random rotation data augmentation techniques.

Based on a comparison of the results obtained with the baseline, Table 7.1, it is evident that there has been a decline. Therefore, it can be concluded that the random local rotation use is not adequate for this case. The next step involved the implementation of the local Gaussian noise technique.

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	46.0877	43.5223	41.0731	61.5304	47.6809	44.0202	87.3664	78.3642	76.8851
AOS	23.9701	22.7201	21.4170	60.4238	45.0272	41.6812	86.8633	77.4982	75.4747
BBOX_BEV	34.3515	32.3396	30.7956	56.7320	39.6605	37.9864	83.6090	77.0285	74.3497
BBOX_3D	24.0265	22.1576	21.4579	52.3665	35.1093	34.2189	65.1113	57.3223	53.5171

Table 7.12: AP values of different detection methods on the KITTI dataset with local gaussian noise data augmentation techniques.

The local Gaussian noise implementation resulted in slightly better results than the previous technique but obtained a considerable decrease in accuracy compared to the baseline (Table 7.1). After thorough analysis, it has been concluded that neither of the local techniques studied is appropriate for this model.

7.1.3 Mixed Data Augmentation Results

Class	**Pedestrian**			**Cyclist**			**Car**		
Difficulty	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**	**Easy**	**Moderate**	**Hard**
BBOX_2D	67.5380	64.1617	59.8471	86.7113	72.3143	68.9202	90.7051	89.4844	87.7153
AOS	49.4916	47.0509	44.11555	85.4363	69.2709	65.7936	90.5950	89.0021	86.9100
BBOX_BEV	60.0715	55.7617	51.6344	83.8891	66.6532	62.9511	89.9997	87.5861	85.6329
BBOX_3D	54.0405	49.5208	45.2106	82.0563	65.0002	61.1422	86.2241	76.4731	74.0131

Table 7.13: AP values of different detection methods on the KITTI dataset with mixed data augmentation techniques.

Difficulty	**Easy**	**Moderate**	**Hard**
BBOX_2D	81.6515	75.3201	72.1609
AOS	75.1743	68.4413	65.6064
BBOX_BEV	77.9867	68.4413	65.6064
BBOX_3D	74.1070	63.6674	60.1220

Table 7.14: mAP values of different detection methods on the KITTI dataset with mixed data augmentation techniques.

After applying individual data augmentation techniques, they are combined to optimize performance. In summary, the choice of augmentation techniques had varying impacts on different classes and tasks, underlining the importance of carefully selecting appropriate methods to achieve optimal performance for specific scenarios. Although the technique of shuffled points improved some classes, it was observed that combining it with other approaches decreased the overall performance. Consequently, this technique was not included in the different data augmentation techniques combination.

8 Conclusion

Data augmentation is a process that generates extended training data by applying various transformations, modifications, or manipulations to the existing data, which allows machine learning to increase performance in situations where the training data is limited or low quality. This thesis presents an understanding survey of data augmentation methods in 3D-LiDAR object detection for autonomous driving. The survey covers different point cloud manipulation techniques and their evaluation.

Deep learning models are computationally expensive and require correctly labeled data. To improve a model, more data is needed to identify more features. Since data augmentation can generate additional data from the existing data, it not only improves the model accuracy, as mentioned before but also saves time and money when collecting more real data is difficult. Data augmentation offers benefits such as:

- Reduces the cost of data acquisition and data labeling;
- Improves model generalization;
- Improves model accuracy in prediction as more data is used to train the model;
- Data overfitting reduction;
- Deals with an imbalance in the dataset by augmenting the samples from the minority class.

Although data augmentation is a robust process, it's crucial to use it cautiously. This process may not be justified from the computational cost perspective since it can increase the computational costs and the addiction memory requirements and consequential impose restrictions on the volume of data generated. While data augmentation prevents the model from overfitting, some augmentation combinations can conduct underfitting. This slows training, leading to a massive strain on resources, such as available processing time and

GPU quotas. Furthermore, the model can't learn all the presented information to give accurate predictions, which leads to high prediction errors.

Future research on expanding the scope of techniques studied with more options for global and local operations and calibration-related data augmentation techniques can enhance the accuracy and robustness of LiDAR-based perception systems [78] [79] [107]. Investigating the performance of object detection models under various calibration settings and promptly developing effective methods for adapting to out-of-distribution or cross-sensor calibrations can be promising. To effectively manage calibration changes during operation, it is crucial to incorporate synthetic calibration variations into the data augmentation pipeline for LiDAR-based perception system technology advancements.

Including calibration as a data augmentation technique enhances the model's ability to generalize and adapt to varying LiDAR sensor configurations in real-world scenarios.

9 Bibliography

[1] Hamed H. Aghdam, Elnaz J. Heravi, Selameab S. Demilew, and Robert Laganiere. Rad: Realtime and accurate 3d object detection on embedded systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2875–2883, June 2021.

[2] Waleed Ali, Sherif Abdelkarim, Mohamed Zahran, Mahmoud Zidan, and Ahmad El Sallab. Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud, Aug 2018.

[3] Neeraj Kumar Arzoo Miglani. Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges., Sep 2019.

[4] Abhishek Balasubramaniam and Sudeep Pasricha. Object detection in autonomous vehicles: Status and open challenges, 2022.

[5] Saulo Barreto. Data augmentation, Mar 2023.

[6] Reinhold Behringer and N. Muller. Autonomous road vehicle guidance from autobahnen to narrow curves. *Robotics and Automation, IEEE Transactions on*, 14:810 – 815, 11 1998.

[7] Frederik Benzing, Simon Schug, Robert Meier, Johannes von Oswald, Yassir Akram, Nicolas Zucchet, Laurence Aitchison, and Angelika Steger. Random initialisations performing above chance and how to find them, 2022.

[8] Paul J. Besl. *Surfaces in range image understanding*. Springer-Verlag, 1988.

[9] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. *arXiv preprint arXiv:2005.09927*, 2020.

[10] Keshav Bimbraw. Autonomous cars: Past, present and future - a review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings*, 1:191–198, 01 2015.

[11] M. Birdsall. Google and ite: The road ahead for self-driving cars: Semantic scholar, Jan 1970.

[12] Thomas Braud, Jordan Ivanchev, Corvin Deboeser, Alois Knoll, David Eckhoff, and Alberto Vincentelli. Avdm: A hierarchical command-and-control system architecture for cooperative autonomous vehicles in highways scenario using microscopic simulations. *Autonomous Agents and Multi-Agent Systems*, 35, 04 2021.

[13] Deboeser C. et al Braud T., Ivanchev J. Avdm: A hierarchical command-and-control system architecture for cooperative autonomous vehicles in highways scenario using microscopic simulations. autonomous agents and multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 35, 2021.

[14] Alberto Broggi, Massimo Bertozzi, and Alessandra Fascioli. Architectural issues on vision-based automatic vehicle guidance: The experience of the argo project. *Real-Time Imaging*, 6:313–324, 08 2000.

[15] Michael Burgan. *The Pontiac Firebird*. Capstone Books, 1999.

[16] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[17] Chengtai Cao, Fan Zhou, Yurou Dai, and Jianping Wang. A survey of mix-based data augmentation: Taxonomy, methods, applications, and explainability, 2022.

[18] Minh Cao and Ramin Ramezani. Data generation using simulation technology to improve perception mechanism of autonomous vehicles, 06 2022.

[19] Yuning Chai, Pei Sun, Jiquan Ngiam, Weiyue Wang, Benjamin Caine, Vijay Vasudevan, Xiao Zhang, and Dragomir Anguelov. To the point: Efficient 3d object detection in the range image with graph convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2021.

[20] Sharat Chandran, Larry S. Davis, Daniel DeMenthon, Sven J. Dickenson, Suresh Gajulapalli, Shie-rei Huang, Todd R. Kushner, Jacqueline LeMoigne, Suni Puri, Tharakesh Siddalingaiah, and et al. *An overview of vision-based navigation for autonomous land vehicles 1986*. U.S. Army Engineer Topographic Laboratories, 1987.

[21] Long Chen, Shaobo Lin, Xiankai Lu, Dongpu Cao, Hangbin Wu, Chi Guo, Chun Liu, and Fei-Yue Wang. Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[22] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn, Aug 2019.

[23] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees G. M. Snoek. Pointmixup: Augmentation for point clouds. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 330–345, Cham, 2020. Springer International Publishing.

[24] Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. Multi-level semantic feature augmentation for one-shot learning. *IEEE Transactions on Image Processing*, 28(9):4594–4605, 2019.

[25] Shohei Chiba and Hisayuki Sasaoka. Basic study for transfer learning for autonomous driving in car race of model car. In *2021 6th International Conference on Business and Industrial Research (ICBIR)*, pages 138–141, 2021.

[26] Jaeseok Choi, Yeji Song, and Nojun Kwak. Part-aware data augmentation for 3d object detection in point cloud. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3391–3397, 2021.

[27] Marc Cranswick. *Pontiac firebird - the auto-biography*. Veloce Publishing, 2017.

[28] Ekin D. Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019.

[29] Sebastian Cygert and Andrzej Czyzewski. Toward robust pedestrian detection with data augmentation. *IEEE Access*, PP:1–1, 07 2020.

[30] Lixuan Du, Rongyu Zhang, and Xiaotian Wang. Overview of two-stage object detection algorithms. *Journal of Physics: Conference Series*, 1544:012033, 05 2020.

[31] Cardew K H F. *The automatic steering of vehicles. an experimental system fitted to a DS 19 Citroen Car.* 1970.

[32] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2918–2927, 2021.

[33] Reimer B. Victor T. Fridman L., Lee J. 'owl' and 'lizard': Patterns of head pose and eye pose in driver gaze classification. *IET Computer Vision*, 10:308–314, 2016.

[34] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[35] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[36] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012.

[37] Ioannis Pratikakis Georgios ZamanakosaLazarosTsochatzidis, Angelos Amanatiadis. A comprehensive survey of lidar-based 3d object detection methods with deep learning for autonomous driving., Jul 2021.

[38] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[39] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. pages 9224–9232, 06 2018.

[40] Martin Hahner, Dengxin Dai, Alexander Liniger, and Luc Van Gool. Quantifying data augmentation for lidar based 3d object detection. *arXiv preprint arXiv:2004.01643*, 2020.

[41] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the*

IEEE/CVF conference on computer vision and pattern recognition, pages 11873–11882, 2020.

[42] Daniël D. Heikoop, Joost C.F. de Winter, Bart van Arem, and Neville A. Stanton. Effects of platooning on signal-detection performance, workload, and stress: A driving simulator study, Nov 2016.

[43] Thomas Henderson. *Traditional and non-traditional robotic sensors*. Springer, 1990.

[44] Chris Jacobs. From adas to driver replacement-is actual radar performance good enough? | analog devices.

[45] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, and Bo Ma. A review of yolo algorithm developments. *Procedia Computer Science*, 199:1066–1073, 2022.

[46] Kaggle. Lyft 3d object detection for autonomous vehicles.

[47] Tahir Emre Kalayci, Gabriela Ozegovic, Bor Bricelj, Marko Lah, and Alexander Stocker. Object detection in driving datasets using a high-performance computing platform: A benchmark study. *IEEE Access*, 10:61666–61677, 2022.

[48] Sihyeon Kim, Sanghyeok Lee, Dasol Hwang, Jaewon Lee, Seong Jae Hwang, and Hyunwoo J. Kim. Point cloud augmentation with weighted local transformations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 548–557, October 2021.

[49] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12689–12697, 2019.

[50] Dogyoon Lee, Jaeha Lee, Junhyeop Lee, Hyeongmin Lee, Minhyeok Lee, Sungmin Woo, and Sangyoun Lee. Regularization strategy for point cloud via rigidly mixed sample. In *Proceedings - 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2021*, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 15895–15904, United States, 2021. IEEE Computer Society.

[51] Robert D. Leighty. *DARPA ALV (Autonomous Land Vehicle) summary*. Defense Technical Information Center, 1986.

[52] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017.

[53] Du Y. Zhu M. Zhou S. amp; Zhang L. Li, Z. A survey of 3d object detection algorithms for intelligent vehicles development - artificial life and robotics. *SpringerLink*, 2021.

[54] Hongyang Li, Chonghao Sima, Jifeng Dai, Wenhai Wang, Lewei Lu, Huijie Wang, Enze Xie, Zhiqi Li, Hanming Deng, Hao Tian, and et al. Delving into the devils of bird's-eye-view perception: A review, evaluation and recipe, Sep 2022.

[55] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Pointaugment: An auto-augmentation framework for point cloud classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6377–6386, 2020.

[56] You Li and Javier Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, 2020.

[57] Yuming Li, Jue Wang, Tengfei Xing, Tianlu Liu, Chengjun Li, and Kuifeng Su. Tad16k: An enhanced benchmark for autonomous driving. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 2344–2348, 2017.

[58] Zirui Li. Lidar-based 3d object detection for autonomous driving. In *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, pages 507–512, 2022.

[59] Zhidong Liang, Ming Zhang, Zehan Zhang, Xian Zhao, and Shiliang Pu. Rangercnn: Towards fast and accurate 3d object detection with range image representation, Mar 2021.

[60] Bing Shun Lim, Sye Loong Keoh, and Vrizlynn L. L. Thing. Autonomous vehicle ultrasonic sensor vulnerability and impact assessment. In *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*, pages 231–236, 2018.

[61] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing.

[62] Weiyu Liu, Qian Dong, Pengqi Wang, Guang Yang, Lingzhong Meng, You Song, Yuan Shi, and Yunzhi Xue. A survey on autonomous driving datasets. In *2021 8th International Conference on Dependable Systems and Their Applications (DSA)*, pages 399–407, 2021.

[63] Lusa. Portugal regista 6880 mortos em acidentes rodoviários na última década, Nov 2020.

[64] Antonio M. López, Gabriel Villalonga, Laura Sellart, Germán Ros, David Vázquez, Jiaolong Xu, Javier Marín, and Azadeh Mozafari. Training my car to see using virtual worlds, Aug 2017.

[65] Apoorva M, Nikhil Muralidhar Shanbhogue, Samarth Shreepad Hegde, Yogesh P Rao, and Chaitanya L. Rgb camera based object detection and object co-ordinate extraction. In *2022 IEEE 7th International conference for Convergence in Technology (I2CT)*, pages 1–5, 2022.

[66] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework, Nov 2022.

[67] Jiageng Mao, Minzhe Niu, Haoyue Bai, Xiaodan Liang, Hang Xu, and Chunjing Xu. Pyramid r-cnn: Towards better performance and adaptability for 3d object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2703–2712, 2021.

[68] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A review and new outlooks. 06 2022.

[69] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A review and new outlooks. *arXiv preprint arXiv:2206.09474*, 2022.

[70] MathWorks. Anchor boxes for object detection.

[71] Markus Maurer, J. Gerdes, Barbara Lenz, and Hermann Winner. *Autonomous Driving. Technical, Legal and Social Aspects*. 05 2016.

[72] Meenakshy. Ultrasonic sensor hc-sr04 with arduino, Nov.

[73] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving, Mar 2019.

[74] Bosch Mobility. Ultrasonic sensor.

[75] Mohammad Momeny, Ali Asghar Neshat, Mohammad Arafat Hussain, Solmaz Kia, Mahmoud Marhamati, Ahmad Jahanbakhshi, and Ghassan Hamarneh. Learning-to-augment strategy using noisy and denoised data: Improving generalizability of deep cnn for the detection of covid-19 in x-ray images. *Computers in Biology and Medicine*, 136:104704, 2021.

[76] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Košecká. 3d bounding box estimation using deep learning and geometry. 12 2016.

[77] Akeem Whitehead Mubina Toa. Ultrasonic sensing basics (rev. d), Sep 2019.

[78] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip H. S. Torr, and Puneet K. Dokania. Calibrating deep neural networks using focal loss, 2020.

[79] Jeremy Nixon, Mike Dusenberry, Ghassen Jerfel, Timothy Nguyen, Jeremiah Liu, Linchuan Zhang, and Dustin Tran. Measuring calibration in deep learning, 2020.

[80] PandaSet. Pandaset - open-source dataset for self-driving cars.

[81] Pawangfg. Vgg-16: Cnn model, Jan 2023.

[82] Dean A. Pomerleau. *Knowledge-Based Training of Artificial Neural Networks for Autonomous Robot Driving*, pages 19–43. Springer US, Boston, MA, 1993.

[83] PyPI. Gryds: a python package for geometric transformations of images.

[84] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[85] Meytal Rapoport-Lavie and Dan Raviv. It's all around you: Range-guided cylindrical network for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 2992–3001, October 2021.

[86] Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. *ArXiv*, abs/2202.03377, 2022.

[87] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, Jan 2016.

[88] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[89] Adrian Rosebrock. Intersection over union (iou) for object detection, May 2023.

[90] Francisca Rosique, Pedro Navarro Lorente, Carlos Fernandez, and Antonio Padilla. A systematic review of perception system and simulators for autonomous vehicles research. *Sensors*, 19:648, 02 2019.

[91] Vidhya Sagar, Sailesh Jain, and VIDHYASAGAR BS. Yield estimation using faster r-cnn. 05 2018.

[92] M. Sainz, R. Pajarola, A. Mercade, and A. Susin. A simple approach for point-based object capturing and rendering. *IEEE Computer Graphics and Applications*, 24(4):24–33, 2004.

[93] SEAT. Rear view camera - car terms.

[94] Hualian Shenga, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2723–2732, 2021.

[95] 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)*, pages 2118–2127, 2021.

[96] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10526–10535, 2020.

[97] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[98] Steven E. Shladover S. *Lane Assist Systems for Bus Rapid Transit. Volume I: Technology Assessment.* " Publication RTA 65A0160, US Department of Transportation, 2007.

[99] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2242–2251, 2017.

[100] Russell SJ and Norvig P. *Artificial Intelligence: A Modern Approach. 2nd ed.* Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, 2003.

[101] Lis K. Gorgon M. Stanisz, J. Implementation of the pointpillars network for 3d object detection in reprogrammable heterogeneous devices using finn. *J Sign Process Syst 94*, 06 2022.

[102] Shanmugalingam Suganthan, Sonya Coleman, and Bryan Scotney. Range image feature extraction with varying degrees of data irregularity. In *International Machine Vision and Image Processing Conference (IMVIP 2007)*, pages 33–40, 2007.

[103] Jiahui Tao, Yuehan Gu, JiaZheng Sun, Yuxuan Bie, and Hui Wang. Research on vgg16 convolutional neural network feature classification algorithm based on transfer learning. In *2021 2nd China International SAR Symposium (CISS)*, pages 1–3, 2021.

[104] David W. Temple and Dennis Adler. *GM's Motorama: The glamorous show cars of a cultural phenomenon.* MBI Pub., 2006.

[105] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer. Toward autonomous driving: the cmu navlab. i. perception. *IEEE Expert*, 6(4):31–42, 1991.

[106] Pham T. Carneiro G. Palmer L. amp; Reid I. Tran, T. A bayesian data augmentation approach for learning deep models, October 2017.

[107] Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas Schön. Evaluating model calibration in classification. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International*

Conference on Artificial Intelligence and Statistics, volume 89 of *Proceedings of Machine Learning Research*, pages 3459–3467. PMLR, 16–18 Apr 2019.

[108] Tobias Vogelpohl, Matthias Kühn, Thomas Hummel, Tina Gehlert, and Mark Vollrath. Transitioning to manual driving requires additional time after automation deactivation, Apr 2018.

[109] Tai Wang, Xinge Zhu, and Dahua Lin. Reconfigurable voxels: A new representation for lidar-based point clouds, Oct 2020.

[110] Li K. Li J. Wang J., Huang H. Towards the unified principles for level 5 autonomous vehicles. *Engineering*, 7:1313–1325, 2021.

[111] Kevin Wijaya, Dong-Hee Paek, and Seung-Hyun Kong. Advanced feature learning on point clouds using multi-resolution features and learnable pooling, 05 2022.

[112] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9613–9622, 2019.

[113] Huang J. Guan D. Cui K. Lu S. amp; Shao L. Xiao, A. Polarmix: A general data augmentation technique for lidar point clouds, July 2022.

[114] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, Yunlong Wang, and Diange Yang. Pandaset: Advanced sensor suite dataset for autonomous driving. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 3095–3101, 2021.

[115] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18:3337, 10 2018.

[116] Maosheng Ye, Shuangjie Xu, and Tongyi Cao. Hvnet: Hybrid voxel network for lidar based 3d object detection. pages 1628–1637, 06 2020.

[117] Weiguo Yi, Siwei Ma, Heng Zhang, and Bin Ma. Classification and improvement of multi label image based on vgg16 network. In *2022 3rd International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, pages 243–246, 2022.

[118] Jaejun Yoo, Namhyuk Ahn, and Kyung-Ah Sohn. Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy, 2020.

[119] Jaejun Yoo, Namhyuk Ahn, and Kyung-Ah Sohn. Rethinking data augmentation for image super-resolution: A comprehensive analysis and a new strategy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[120] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[121] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3d object detection in lidar point clouds, Oct 2019.

[122] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. pages 4490–4499, 06 2018.

[123] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[124] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018.

Annex A

A.1 Loss Functions

- **Category classification loss:** Focal Loss

  $FL(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t)$, where $\alpha = 0.25$, $\gamma = 2.0$

$$\alpha_t = \begin{cases} \alpha, & y = 1 \\ 1-\alpha, & y = 0 \end{cases}$$

$$p_t = \begin{cases} p, & y = 1 \\ 1-p, & y = 0 \end{cases}$$

- **Return loss:** Smooth L1 Loss

$$L(x_n, y_n) = \begin{cases} 0.5(x_n - y_n)^2/\beta, & if\ |x_n - y_n| < \beta \\ |x_n - y_n| - 0.5\beta, & otherwise \end{cases}, where\ \beta = \frac{1}{9}$$

- **Classification loss:** Cross Entropy Loss

  $L(y, \hat{y}) = -\sum_{n}^{i=1} \hat{y}_i log(y_i)$

- **Total loss:** Total loss = 1.0*class classification loss + 2.0*regression loss + 2.0*toward classification loss

Annex B

B.1 Intersection over Union

Intersection over Union is an evaluation metric that measures the accuracy of an object detector. IoU is often used to evaluate the performance of convolution neural network detectors. Any algorithm that generates predicted bounding boxes can be evaluated using IoU. For the application of the IoU algorithm to evaluate an object detector, its needs:

- The GT bounding boxes
- The predicted bounding boxes from the used model

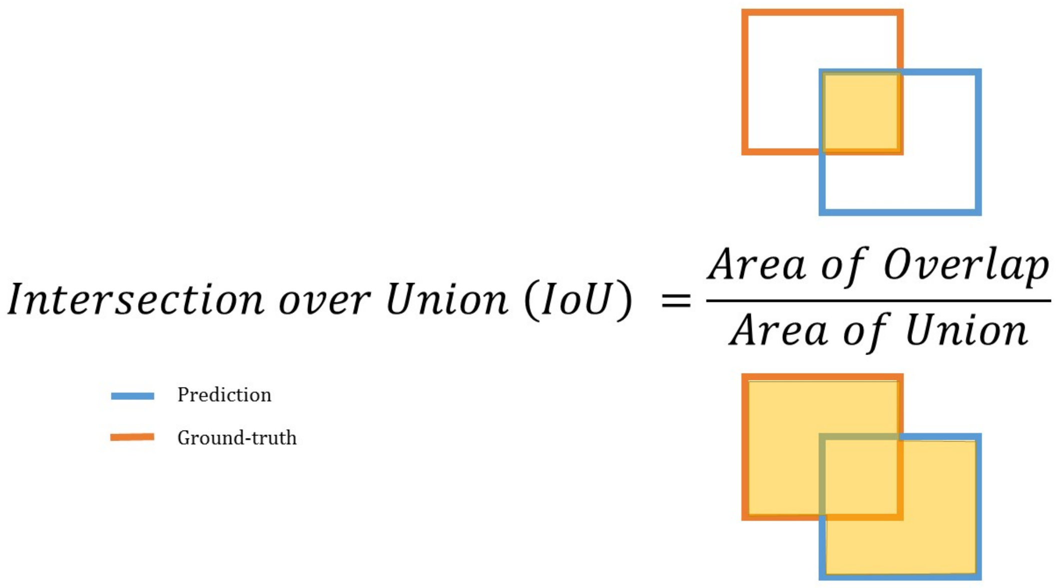

$Intersection\ over\ Union\ (IoU) = \frac{Area\ of\ Overlap}{Area\ of\ Union}$
Prediction
Ground-truth

Figure B.1: Intersection over Union [46]

In the figure above, Figure B.1, it can be seen how the IoU calculation is done.

Due to the diversity of parameters present in the model, a total match between the predicted and ground-truth bounding boxes is unrealistic. Therefore, an evaluation metric that rewards predicted bounding boxes for heavily overlapping with the ground truth needs to be defined:

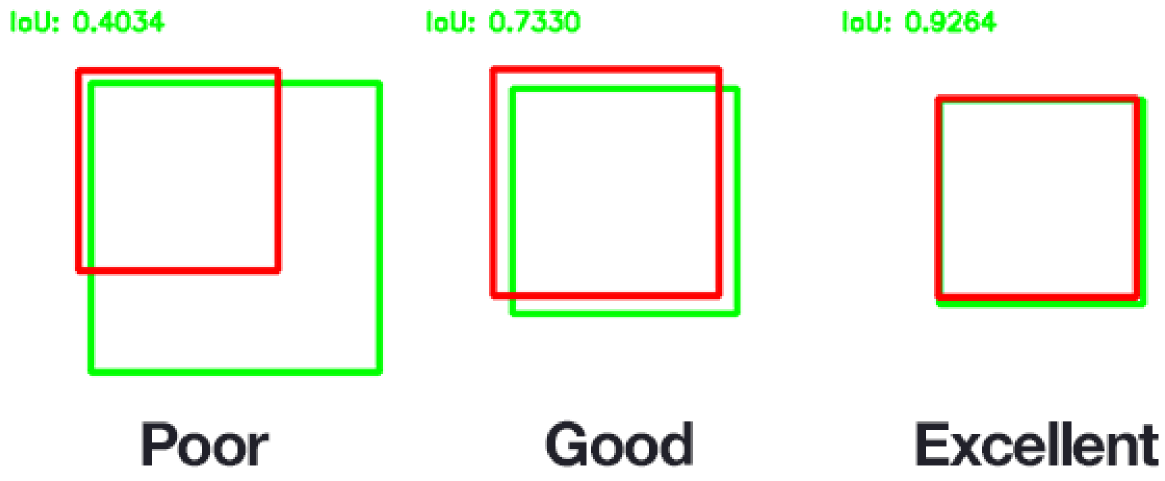


Figure B.2: Intersection over Union - poor, good and excellent score[89]

Appendix A

A.1 Evaluation Metrics

Table A.1 shows the evaluation metrics of the model without data augmentation techniques.

	Evaluation Metrics		
Methods	**F1 score**	**Recall**	**Precision**
BBOX_2D	51.7978	41.5207	93.3830
BBOX_BEV	51.6526	41.5287	92.9913
BBOX_3D	42.5625	33.7913	82.8324

Table A.1: Evaluation metrics of different detection methods applied to the KITTI dataset.

Overall it was obtained:

- **F1 score:** 48.6710
- **Recall:** 38.9469
- **Precisions:** 89.7356

In turn, Table A.2 shows the evaluation metrics of the model with mixed data augmentation techniques.

	Evaluation Metrics		
Methods	**F1 score**	**Recall**	**Precision**
$BBOX\_2D$	56.6407	46.3314	96.8260
$BBOX\_BEV$	55.2690	45.0781	95.7270
$BBOX\_3D$	49.8629	39.9867	92.0419

Table A.2: Evaluation metrics of different detection methods applied to the KITTI dataset with mixed data augmentation techniques.

Overall it was obtained:

- **F1 score:** 53.9242
- **Recall:** 43.7987
- **Precisions:** 94.8650

Appendix B

B.1 Hyper-parameters of Data Augmentation Techniques

Below is an explanation of the specific functions of these hyperparameters to provide further clarification.

- **Global/Local Rotation Parameter:** Controlling random rotation within the chosen radians range.
- **Global Scale Parameter:** Managing the scaling of input data within the chosen range.
- **Global Translation Parameter:** This parameter governs global translations applied to the data.
- **Random Flip Ratio:** This parameter determines the likelihood of horizontal flipping during data augmentation.
- **Jitter Sigma:** Jitter sigma influences the amount of noise added to the data, contributing to the model's ability to handle noisy input.
- **Jitter Clip:** Jitter clip constrains the maximum amount of jitter applied to the data, ensuring controlled noise levels.
- **Global Gaussian Noise:** This technique encompasses hyperparameters associated with the addition of Gaussian noise to the data, including the number of trials (num_try), translation standard deviations ($translation\_std$), and rotation range (rot_range) parameters.

- **Local Gaussian Noise:** The standard deviation of the normal distribution is sampled from a uniform distribution with a range of $[min, max] = [std_range[0], std_range[1]]$
- **Upsampling Num Samples:** This parameter regulates the number of samples generated during upsampling operations.

Appendix C

Scientific Paper - ROBOT2023: Sixth Iberian Robotics Conference

Evaluation of Point Cloud Data Augmentation for 3D-LiDAR Object Detection in Autonomous Driving

Marta Martins[1], Iago P. Gomes[2], Denis F. Wolf[2], and Cristiano Premebida[1]

[1] Institute of Systems and Robotics, University of Coimbra, Portugal.
martasmartins10@gmail.com, cpremebida@deec.uc.pt
[2] Institute of Mathematics and Computer Science, University of São Paulo, Brazil.
iagogomes@usp.br, denis@icmc.usp.br

Abstract. Obstacle detection is an essential component for autonomous vehicles to navigate safely. To overcome some drawbacks of 2D object detection, many recent methods have been proposed to cope with 3D detection using LiDAR sensors. This sensor provides depth and more representative spatial and geometric information to the models, allowing the estimation of 3D bounding boxes with orientation around the objects of interest. However, the complexity of 3D annotation, in supervised object detection, imposes significant challenges for this task. To address the annotation problem, some studies have explored data augmentation techniques for 3D point clouds, but, not all augmentation methods yield positive impacts on model performance. Therefore, this paper presents an in-depth evaluation of global data augmentation techniques, specifically focusing on geometric transformation and noise-based methods for 3D object detection. The results reported in this paper, achieved on the KITTI dataset, showed a relevant difference in some geometric operations, and the importance of noise-based methods.

Keywords: 3D object detection, autonomous driving, deep learning, data augmentation.

1 Introduction

A self-driving car is a complex system capable of making decisions without human interaction [19], composed of sensors, actuators, complex algorithms, machine learning, and processors [22]. To have the ability to drive autonomously, this system creates a representation of their surroundings based on the different sensor readings. Typically, radar sensors examine the position of nearby vehicles, cameras can be used to identify traffic lights and road signs, track other vehicles, and look for vulnerable road users (*e.g.*, pedestrians and cyclists), while LiDAR sensors reflect pulses of light around the car's surroundings to measure distances, identify road edges, and detect and locate obstacles [19]. Ultrasonic sensors, on the other hand, can be used to detect obstacles and calculate the ideal steering angle during parking [16].

Among the several tasks for a robust environmental perception, obstacle detection is essential for autonomous vehicles to safely navigate. The standard approach for this task employs deep learning models based on Convolutional Neural Networks (CNN) [4] – such as *YOLO* [11], to extract features from images, and detect target objects based on the feature maps. However, to use these detections, an autonomous system needs to find a correspondence between the 2D bounding boxes in the image to a 3D coordinate system (*i.e.*, usually a global coordinate system used for path planning and obstacle avoidance). This transformation requires a calibrated sensor setup, which might not have a straightforward implementation [27, 1].

Therefore, to overcome the drawbacks of 2D object detection, several methods were proposed using different sensors, such as LiDAR, RADAR, and Stereo Cameras [1]. These methods provide depth and more representative spatial and geometric information to the models, allowing them to estimate 3D bounding boxes with orientation around objects of interest. However, the shortcomings of some of these approaches are the poor or complete lack of semantic information, useful for differentiating the object classes. Besides that, some techniques showed competitive results for detection and classification in standards benchmarking, such as the *PointPillars* network [13] on the KITTI dataset [8] using only LiDAR's point clouds.

Another important remark regarding 3D detection is the complexity of the annotation process of the data, which results in the small number of large-scale datasets available for training and testing models when compared to 2D detection datasets. In this sense, some works in the literature explored data augmentation techniques for 3D data to increase the number of samples and their representativeness [14, 13, 12, 24, 6, 21].

However, not every augmentation operation exerts a positive influence on model training and performance. On the contrary, when an operation is not applied correctly, it tends to only increase the memory usage, time, and cost of training and developing detection methods. Therefore, this paper presents an experimental evaluation of 3D data augmentation operations for 3D obstacles detection on the KITTI dataset.

The remainder of this paper is organized as follows: Section 2 presents the related works; Section 3 describes the experiment setup, network model, and data augmentation techniques; Section 4 presents and discusses the results; and, Section 5 concludes the paper with some remarks and future work.

2 Related Work

This section discusses recent related works on the 3D object detection methods centered on LiDAR data, i.e., point clouds or range images. Contrary to images, a point cloud is a 3D representation that contains sparse and irregular pixels and requires a specific model for feature extraction. On the other hand, a range image is a dense and compact representation where range pixels have 3D information [17]. The LiDAR-based 3D object detection models are based on distinct data representations, including point-based, grid-based, point-voxel-based, and range-based methods.

2.1 3D Obstacle Detection using LiDAR's Point Cloud

Point-based 3D object detection uses deep learning techniques on point clouds to estimate 3D bounding boxes. Points are the raw output data from the capture sensors in mostly 3D geometry acquisition systems [23]. This data structure allows several simplification and processing techniques. The point-based backbone network extracts features after sampling from a point cloud to reduce the processing time [17]. Afterward, a decoder network estimates the 3D bounding boxes using the extracted feature vectors. PointRCNN [25] uses the Furthest Point Sampling (FPS) technique to gradually downsample input point clouds and create 3D bounding box proposals. Shi and Rajkumar [26] proposed the Point-GNN, which uses a graph neural network to extract features from a point cloud and detect objects.

3D object detectors based on grids convert point clouds into discrete grid representations, such as voxels, pillars, and Bird's-eye View (BEV) structures. After, they use 2D convolutional neural networks or 3D sparse neural networks to extract grid features and then detect 3D objects from the grid cells [17]. Therefore, they have two main components: grid-based representation and neural networks. VoxelNet [29] is an example of a network that uses sparse voxel grids. This network proposes a new voxel feature encoding layer to obtain features of voxel cell points. PointPillars [13] is a network that introduces the pillar representation, which is a special kind of voxel that has an unlimited size in the vertical direction. The SECOND [28] uses two sparse operators that form a sparse convolution network to extract 3D voxel features. This network has been implemented in multiple works and became one of the most used backbone networks in voxel-based detectors.

Point-voxel-based methods apply a hybrid structure that uses voxels and points as representations [15]. Generally, those methods are distributed into two categories: the single-stage and two-stage detection frameworks. SA-SSD [10] is an example of a single-stage network. This network uses voxel and point representations, respectively, in the backbone and the auxiliary branch. In the branch, the 3D SCNN extracts features that are posteriorly converted into points and processed by point-based networks. Chen et al. [5] proposed the Fast Point R-CNN, a two-stage method, which relies on feature fusion and attention mechanism to combine voxel and points features.

Finally, range-based methods use range images (*e.g.*, depth images) built from point clouds, stereo vision, or specific sensors, to represent the spatial and geometrical information from the environment. These methods employ specialized operations to extract features and estimate the 3D bounding boxes, such as Range Dilated Convolution [2], graph operators [3], and meta-kernel convolutions [7].

2.2 3D Data Augmentation Methods

In deep neural networks, training data has a crucial role in learning to perform tasks. However, compared to the real-world complexity, training data has a limited quantity, so to enlarge the training set and maximize the knowledge of a network, data augmentation (DA) is required. A conventional DA method is a global augmentation that learns transformations invariance in image recognition tasks, such as random cropping, random scaling, random erasing, color jittering, etc. [14]. Another approach of DA is local augmentation which generates new training data performing diverse mix operations.

Since 3D object recognition datasets, which include KITTI datasets, have a limited number of samples, increasing the size of the data is one of the ways to reduce overfitting and improve performance [6]. As with 2D computer vision tasks, one direct approach is to embrace a global augmentation such as translation, random flipping, shifting, scaling, jittering, and rotation, which can be directly incorporated for expanding 3D objects [14]. Oversampling was also used to solve the foreground-background class imbalance problem [6], by introducing new objects into a sampled point cloud.

Previous works such as PointNet and PointNet++ [21] use DA techniques of random rotation about the up-axis scaling, random rotation with perturbations, random shifting, and random jittering of points on the input sample. RS-CNN and DensePoint followed identical DA strategies with little variations [24]. Several studies investigated augmenting local structures of point clouds [12, 24]. However, the works aren't appropriate for scene-level point clouds because they focus on object-level augmentation.

Therefore, this paper investigates 3D data augmentation techniques and their effects on 3D obstacle detection using the KITTI dataset. We focused on global augmentation techniques and present a discussion regarding their impact on the detector performance, evaluated by standard detection metrics. The PointPillars [13] was used as the baseline model in all experiments, because of its balance between performance and inference time. In addition, this architecture uses only LiDAR point clouds, which is the subject of study in this work.

3 3D Obstacle Detection and Data Augmentation

This section describes the design of the experiments, the techniques, and the evaluation metrics used. Figure 1 shows a general flowchart for 3D object detection using point clouds with data augmentation. The first step is the data acquisition and data representation, which consists of obtaining the point cloud from a LiDAR sensor and, then, converting it to a data representation (*e.g.*, point cloud, range image, grid, or voxels). Each sample has a list of annotations associated with it. Afterward, the samples are divided into train, validation, and test sets. The training set undergoes data augmentation operations to either increase the number of samples, attenuate data imbalance or improve representativeness. Finally, the data feed a deep learning model that learns how to detect

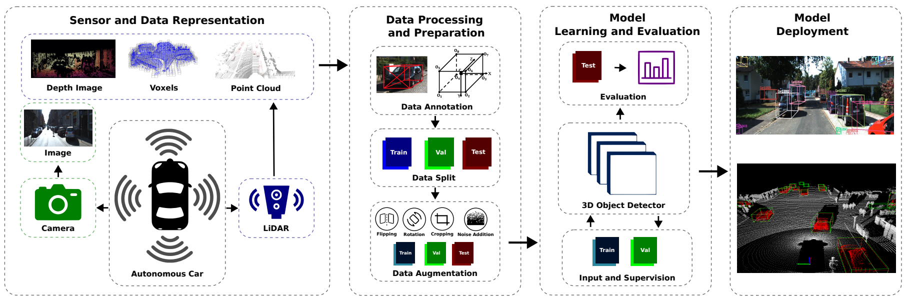


Fig. 1. General Flowchart for 3D Object Detection. Based on [18].

and classify objects, using the training and validation sets. This final model is evaluated using the testing dataset.

3.1 Problem Definition

Definition. 3D object detection seeks to predict the attributes of 3D objects in driving scenarios from sensory inputs. In most cases, a 3D object can be represented as a 3D cuboid, called a 3D bounding box, that includes the object inside.
The 3D bounding box is defined by [20]:

- **3D Center Coordinate:** $T = [x_c, y_c, z_c]'$
- **Dimensions:** $D = [l, w, h]$ (length, width, and height)
- **Heading Angle:** θ (the yaw angle, of a cuboid on the ground plane)
- **Class:** denotes the category of a 3D object

In summary, the 3D bounding box can be represented:

$$B = [x_c, y_c, z_c, l, w, h, \theta, class] \tag{1}$$

3.2 PointPillars

Lang et al. [13], proposed PointPillars, using an encoder with PointNets to learn a representation of point clouds organized in vertical columns (pillars). According to them, the main advantages of the model were the balance between performance and inference time. They demonstrated these advantages in the KITTI benchmark by offering higher detection performance at a faster speed. Their results suggest that PointPillars is a relevant architecture for 3D object detection using only LiDAR's point clouds.

PointPillars. A method for 3D object detection that allows end-to-end learning with only 2D convolutional layers. It uses an encoder that learns features on pillars (vertical columns) of the point cloud to predict 3D-oriented boxes for objects [13]. The network has three steps: **(1)** a feature encoder, where point

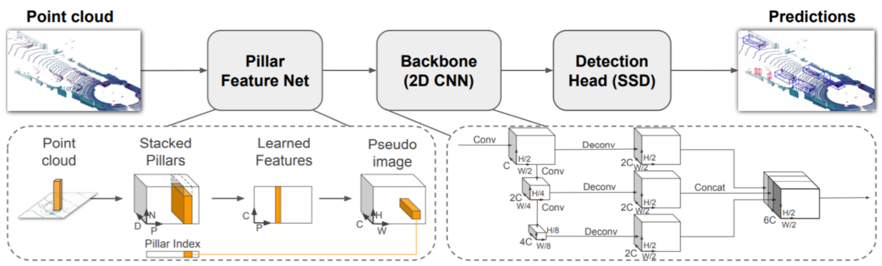


Fig. 2. PointPillars Architecture [13].

clouds are converted to a pseudo-image; **2)** a backbone (2D CNN) to process the pseudo-image into a high-level representation; and **(3)** a detection head (SSD - Single Shot Detector) that detects the objects and create 3D bounding boxes around it. Figure 2 shows the network flowchart presented in Lang et al. [13].

3.3 Data augmentation for 3D point clouds

Data augmentation techniques can be divided into two classes, global and local data augmentation [9]. Global data augmentation refers to operations that apply transformations to the entire point cloud or scene. In turn, local augmentation focuses on specific regions of the point cloud. Thus, global operations aim to improve the robustness and generalization of 3D object detectors without introducing local bias. This strategy also enhances model generalization and is more effective for real-world applications, such as 3D object detection for autonomous vehicles on urban roads. However, it is important to ensure that the operation preserves the spatial and geometric relationship between the objects and the foreground/background of the scene. In addition, the operations should also preserve the objects' shapes, since they are important for classification.

This work studied global augmentation techniques for object detection in urban roads, using the KITTI dataset. Among the global augmentation techniques, we selected operations for geometric and spatial transformation and noise-based operations. These techniques are described as follows.

• **Geometric and Spatial Transformation**: The geometric transformations consist of rigid transformation (*i.e.*, rotation, translation, and scaling) in the entire point cloud. They help the model to become invariant to the orientation, translation, and scaling (*i.e.*, affect the size of objects) of the scene. Some operations are: rigid transformations (translation + rotation); similarity transformations (translation + rotation + isotropic scaling); affine transformations (translation + rotation + arbitrary scaling + shearing); and random flips.

• **Noise-based Operation**: These operations involve introducing random or controlled variations to the original data to create new samples with minor perturbations, with the aim of increasing data diversity and including noise that may actually be present in the data acquisition process. Some operations are:

jittered points; Gaussian noise; shuffling; and, upsampling or downsampling of the points.

3.4 Evaluation Metrics

To assess the impact of various global data augmentation techniques on the KITTI dataset, we employed the Average Precision (AP) metric, focusing on the 3D bounding box evaluation for three specific classes: Pedestrian, Cyclist, and Car. Additionally, we considered three difficulty levels: easy, moderate, and hard. This metric estimates the area under the precision-recall curve, and provides a comprehensive assessment of the model's performance in terms of its ability to make precise and reliable predictions for each class.

Moreover, we also evaluated the performance of the orientation estimation, using the Average Orientation Similarity (AOS) metric. It measures the similarity between the predicted rotation and the ground truth rotation angles using the following equation:

$$AOS = \frac{1 + cos(\alpha^{GT} - \tilde{\alpha})}{2} \tag{2}$$

where α^{GT} is the ground-truth orientation, and $\tilde{\alpha}$ is the predicted orientation.

4 Results

Table 1 presents the results of the **Baseline** model, which is the PointPillars network trained on the standard KITTI dataset without any modifications. Additionally, the table includes results for the same model trained with various augmentation operations applied to the training data.

The **Baseline** results show an average performance for 3D object detection on the KITTI dataset, highlighting the effect of data imbalance on individual performance for each class. The *Car* class has the best performance among all classes, since it represents the majority of the annotated labels. The size and rectangular shape of the 3D bounding belonging to this class also help the prediction of its orientation, when compared to the shape of the remaining classes (*i.e.*, *Pedestrian* and *Cyclist).*

All data augmentation techniques improved the performance of the model when compared to the **Baseline**. The effects of the data imbalance were also attenuated. However, there are some important remarks regarding the operations for geometric and spatial transformation and noise-based operations.

As discussed in earlier sections, one of the main goals of applying geometric transformations is to enhance the model's ability to handle rotation, translation, and scaling invariance. These transformations are employed to modify the data samples, making them more representative of these variations. The **Random Flip** alone resulted in more significant improvements in orientation and pose estimation for all classes compared to using *Random Rotation, Scaling, and Translation* together.

Table 1. AP values of different detection methods on the KITTI dataset

		Pedestrian			**Cyclist**			**Car**		
Operation	**Metric**	*Easy*	*Moderate*	*Hard*	*Easy*	*Moderate*	*Hard*	*Easy*	*Moderate*	*Hard*
Baseline	*BBOX 3D*	26.9381	25.1430	24.1276	44.2416	30.3132	29.0027	66.5733	60.2184	54.6960
	AOS	27.7677	26.5258	25.3001	54.3573	39.0934	35.9742	85.3043	76.4905	74.8277
Random Rotation, Scaling, and Translation	*BBOX 3D*	50.1598	44.2435	39.9561	74.4308	58.2412	54.9572	83.9472	74.6200	68.0987
	AOS	50.2951	47.4574	44.0398	83.5650	64.3258	61.4730	90.3268	87.8260	84.7793
Random Flip	*BBOX 3D*	54.6959	48.3043	44.9938	80.9197	62.3310	57.8159	85.1716	76.6253	73.6535
	AOS	**53.7285**	**49.9011**	**46.5327**	85.7929	69.5090	67.5332	90.4288	88.6429	85.3264
Jittered Points	*BBOX 3D*	54.0822	47.4751	44.8138	78.6046	60.7153	57.2212	**86.5941**	**76.9589**	**73.9640**
	AOS	53.2396	49.4660	46.1822	85.3422	70.4192	67.4924	90.6147	88.7847	85.7357
Gaussian Noise	*BBOX 3D*	**55.6522**	**50.4675**	**45.9789**	81.1417	62.9864	60.0440	85.8992	76.3915	73.6888
	AOS	43.1251	40.6006	38.0768	85.0478	70.0361	66.7641	**90.6932**	88.8141	85.6916
Shuffled Points	*BBOX 3D*	52.3244	46.9996	43.1217	79.8587	62.6771	60.2295	85.2163	76.4071	73.5351
	AOS	46.4865	43.1040	40.4199	**86.6693**	**72.0034**	**68.0523**	90.4482	88.8746	86.6124
Random Upsampling	*BBOX 3D*	54.0405	49.5208	45.2106	**82.0563**	**65.0002**	**61.4122**	86.2241	76.4731	74.0131
	AOS	49.4916	47.0509	44.1155	85.4363	69.2709	65.7936	90.5950	**89.0021**	**86.9100**

Furthermore, the noise-based techniques also contributed to enhancing the detector's performance, but their impact varied for each class. The **Jittered Points** approach improved the detection performance of the *Car* class, while for the *Pedestrian* class, the **Gaussian Noise** had a more positive influence on pose estimation. On the other hand, the **Random Upsampling** technique proved beneficial for the *Cyclist* class.

Additionally, when considering orientation estimation, the effects of different methods were class-specific. The **Jittered Points** technique led to improved orientation estimation for the *Pedestrian* class, while the **Shuffled Points** and **Random Upsampling** methods enhanced orientation estimation for the *Cyclist* and *Car* classes, respectively. However, there was an exception for the *Car* class in the *Easy* set, where the **Gaussian Noise** method was the most effective.

In summary, the choice of augmentation techniques had varying impacts on different classes and tasks, underlining the importance of carefully selecting appropriate methods to achieve optimal performance for specific scenarios. For the geometric transformation, **Random Flip** alone achieved relevant performance. However, for noise-based methods, a combination of different techniques proved essential, as each method had distinct effects on the detector's performance for each class.

5 Conclusion

Data augmentation is a process that generates extended training data by applying various transformations, modifications, or manipulations to the existing data, which allows machine learning to increase performance in situations where the training data is limited or low quality. The focus of this paper was to present an in-depth study on global data augmentation methods in the context of 3D-LiDAR object detection for autonomous driving. The survey covers different point cloud manipulation techniques and their evaluation.

Finally, the proposed study has room for improvement, such as: including a more extensive range of global data augmentation operations to achieve a comprehensive analysis of their impact; adding local augmentation operations to the study and comparing their differences and impact on detector performance; investigating the effects of combining global and local augmentation techniques; and, study the combination of undersampling techniques and data augmentation as an alternative to reduce the effects of data imbalance together.

Acknowledgments

This paper has been partially supported by the project GreenBotics ref. PTDC/EEI-ROB/2459/2021 funded by FCT, Portugal. We thank the Coordination for the Improvement of Higher Education Personnel - Brazil (CAPES) for the financial support under grant 88887.500344/2020-0, and the São Paulo Research Foundation (FAPESP) for the financial support under grants 2019/27301-7 and 2022/04473-0.

References

1. Arnold, E., Al-Jarrah, O.Y., Dianati, M., Fallah, S., Oxtoby, D., Mouzakitis, A.: A survey on 3d object detection methods for autonomous driving applications. IEEE Transactions on Intelligent Transportation Systems (2019)
2. Bewley, A., Sun, P., Mensink, T., Anguelov, D., Sminchisescu, C.: Range conditioned dilated convolutions for scale invariant 3d object detection. arXiv preprint arXiv:2005.09927 (2020)
3. Chai, Y., Sun, P., Ngiam, J., Wang, W., Caine, B., Vasudevan, V., Zhang, X., Anguelov, D.: To the point: Efficient 3d object detection in the range image with graph convolution kernels. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 16,000–16,009 (2021)
4. Chen, L., Lin, S., Lu, X., Cao, D., Wu, H., Guo, C., Liu, C., Wang, F.Y.: Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems (2021)
5. Chen, Y., Liu, S., Shen, X., Jia, J.: Fast point r-cnn. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 9775–9784 (2019)
6. Choi, J., Song, Y., Kwak, N.: Part-aware data augmentation for 3d object detection in point cloud. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2021)
7. Fan, L., Xiong, X., Wang, F., Wang, N., Zhang, Z.: Rangedet: In defense of range view for lidar-based 3d object detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 2918–2927 (2021)
8. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: The kitti dataset. The International Journal of Robotics Research (2013)
9. Hahner, M., Dai, D., Liniger, A., Van Gool, L.: Quantifying data augmentation for lidar based 3d object detection. arXiv preprint arXiv:2004.01643 (2020)
10. He, C., Zeng, H., Huang, J., Hua, X.S., Zhang, L.: Structure aware single-stage 3d object detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2020)
11. Jiang, P., Ergu, D., Liu, F., Cai, Y., Ma, B.: A review of yolo algorithm developments. Procedia Computer Science **199**, 1066–1073 (2022)

12. Kim, S., Lee, S., Hwang, D., Lee, J., Hwang, S.J., Kim, H.J.: Point cloud augmentation with weighted local transformations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
13. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (2019)
14. Li, R., Li, X., Heng, P.A., Fu, C.W.: Pointaugment: An auto-augmentation framework for point cloud classification. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
15. Li, Z.: Lidar-based 3d object detection for autonomous driving. In: 2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML) (2022)
16. Lim, B.S., Keoh, S.L., Thing, V.L.L.: Autonomous vehicle ultrasonic sensor vulnerability and impact assessment. In: 2018 IEEE 4th World Forum on Internet of Things (WF-IoT) (2018)
17. Mao, J., Shi, S., Wang, X., Li, H.: 3d object detection for autonomous driving: A review and new outlooks. arXiv preprint arXiv:2206.09474 (2022)
18. Mao, J., Shi, S., Wang, X., Li, H.: 3d object detection for autonomous driving: A review and new outlooks. arXiv preprint arXiv:2206.09474 (2022)
19. Maurer, M., Gerdes, J., Lenz, B., Winner, H.: Autonomous Driving. Technical, Legal and Social Aspects (2016)
20. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (2017)
21. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems (2017)
22. Russell, S.J.: Artificial intelligence a modern approach. Pearson Education, Inc. (2010)
23. Sainz, M., Pajarola, R., Mercade, A., Susin, A.: A simple approach for point-based object capturing and rendering. IEEE Computer Graphics and Applications (2004). DOI 10.1109/MCG.2004.1
24. Sheshappanavar, S.V., Singh, V.V., Kambhamettu, C.: Patchaugment: Local neighborhood augmentation in point cloud classification. In: 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW) (2021)
25. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 770–779 (2019)
26. Shi, W., Rajkumar, R.: Point-gnn: Graph neural network for 3d object detection in a point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 1711–1719 (2020)
27. Wang, Y., Mao, Q., Zhu, H., Deng, J., Zhang, Y., Ji, J., Li, H., Zhang, Y.: Multi-modal 3d object detection in autonomous driving: a survey. International Journal of Computer Vision pp. 1–31 (2023)
28. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors (2018)
29. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4490–4499 (2018)