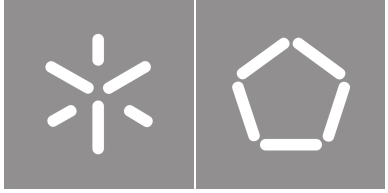
Universidade do Minho
Escola de Engenharia

Luís Pedro Barbosa Ferreira

Temporal Object Detection for Autonomous Driving

November, 2022



Universidade do Minho
Escola de Engenharia

Luís Pedro Barbosa Ferreira

Temporal Object Detection for Autonomous Driving

Master Thesis
Integrated Master's in Informatics Engineering

Work developed under the supervision of:
Paulo Novais (UMinho)
Filipe Gonçalves (Bosch)

November, 2022

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.8.0) [30].

Acknowledgements

I wish to express my gratitude to everyone who supported and helped me during this lengthy journey, in particular to:

Professor Paulo Novais, for allowing me to work alongside him on this project.

Filipe Gonçalves, for supervising and always being ready to offer assistance and advice in order to overcome any obstacles that may have hampered the development of this project.

Carolina Pinto, for her document reviews bringing the information quality up a notch.

Bosch and all team colleagues, for providing a safe and healthy working environment and an unforgettable experience.

André Figueiredo, for being a high-level partner who was always willing to lend a hand.

Cristiana Cunha, for joining me on this long journey, having the patience to deal in the most stressful moments, and for always encouraging me to move forward.

My parents and sister, for always being willing to help and alleviate any concerns that might have a negative impact on the development of this dissertation.

Finally, to all of my family and friends who were always there to provide emotional support when it was needed.

This work is supported by European Structural and Investment Fund in the FEDER component through the Operational Competitiveness and Internalisation Programme (COMPETE 2020) [Project n°047264; Funding Reference: POCI-01-0247-FEDER-047264]

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the Universidade do Minho.

_____, _____

(Place) (Date)

_____

(Luís Pedro Barbosa Ferreira)

“Simply shine your light on the road ahead, and you are helping others to see their way out of darkness. ” (Katrina Mayer)

Resumo

Deteção Temporal de Objetos para Condução Autónoma

A Inteligência Artificial tem sido cada vez mais utilizada nas nossas tarefas diárias, melhorando a qualidade de vida e a segurança das pessoas. A condução autónoma é um dos campos onde esta inteligência está a ser cada vez mais utilizada. Este processo envolve a transformação de veículos normalmente conduzidos por humanos em veículos autônomos com capacidade de agir sem intervenção humana.

A condução autónoma pode usar dados de vários sensores, como câmaras, RADAR e/ou LiDAR, para percecionar automaticamente o ambiente ao redor e tomar decisões inteligentes. Para compreender o ambiente percecionado, um módulo de deteção de objetos deve ser desenvolvido e interconectado para uma implementação confiável de um veículo autónomo. Essa tarefa é a base da condução autónomo, pois permite que o veículo reconheça objetos que possam estar presentes na área envolvente, tais como pedestres ou outros veículos. As deteções geralmente são feitas usando algoritmos de Aprendizagem Profunda que utilizam redes neuronais convolucionais e grandes quantidades de dados de treino e validação.

Apesar de vários esforços para acelerar o desenvolvimento de veículos autónomos, essa tarefa tem se mostrado complexa. Quase todas as técnicas desenvolvidas para resolver este problema foram implementadas realizando deteções em cada frame separadamente. Essa estratégia, no entanto, não aborda questões como oclusão, ruído do sinal, ou outros tipos de falta de informação dos objetos. Como resultado, uma tecnologia consistente temporalmente que emprega as informações de um objeto num contexto de múltiplos frames, ou seja, em uma sequência de frames, pode aumentar a qualidade da deteção.

No caso desta dissertação, a perceção autônoma será realizada usando a deteção de objetos temporal com base em dados coletados de sensores LiDAR. Esses sensores são capazes de obter dados de alta fidelidade que permitem uma representação 3D de alta qualidade do cenário, conhecidas como nuvens de pontos.

Palavras-chave: Condução Autónoma, Deteção de Objetos, Inteligência Artificial, Aprendizagem Profunda, Perceção Máquina, LiDAR, Nuvens de Pontos, Multi-frame

Abstract

Temporal Object Detection for Autonomous Driving

Artificial Intelligence has been increasingly used in our daily tasks, improving people's quality of life and safety. Autonomous driving is one of the fields where this intelligence is increasingly being used. This process entails the transformation of vehicles normally driven by humans into autonomous vehicles with the ability to act without human intervention.

Autonomous driving may use data from various sensors, such as cameras, RADAR, and/or LiDAR, to automatically perceive the surrounding environment and make intelligent decisions. To comprehend the perceived environment, an object detection module must be developed and interconnected for a reliable autonomous vehicle implementation. This task is the foundation of autonomous driving since it allows the vehicle to recognise objects who may be present in the area, such as pedestrians or other vehicles. The detections are usually made using Deep Learning algorithms that utilise convolutional neural networks and massive amounts of training data.

Despite several efforts to accelerate the development of autonomous vehicles, this task has proven to be complex. Almost all developed techniques to solve this problem have been implemented by performing detections on each frame separately. This strategy, however, does not address issues like occlusion or other sorts of object information deficiency. As a result, a temporal consistent technology that employs an object's information in a multi-frame context, that is, in a sequence of frames, can increase detection quality.

In the instance of this dissertation, autonomous perception will be accomplished using temporal object detection based on data collected from LiDAR sensors. These sensors are capable of obtaining high-fidelity data that allows for a high-quality 3D representation of the scenario, known as point clouds.

Keywords: Autonomous Driving, Object Detection, Artificial Intelligence, Deep Learning, Machine Perception, LiDAR, Point Clouds, Multi-frame

Contents

List of Figures xi

List of Tables xiii

Acronyms xvi

1 Introduction 1

1.1 Motivation & Challenges . . . 1
1.2 Scope . . . 2
1.2.1 Autonomous Driving . . . 2
1.2.2 Artificial Intelligence . . . 4
1.2.3 Machine Learning . . . 5
1.2.4 Deep Learning . . . 7
1.2.5 Machine Perception . . . 8
1.2.6 Object Detection . . . 10
1.2.7 Temporal 3D Object Detection . . . 11
1.2.8 Recurrent Neural Network Architectures . . . 12
1.3 Document Structure . . . 15

2 State of the Art 17

2.1 An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds . . . 17
2.1.1 Pipeline . . . 18
2.1.2 Results . . . 19
2.2 3D-MAN: 3D Multi-frame Attention Network for Object Detection . . . 20
2.2.1 Pipeline . . . 20
2.2.2 Results . . . 21
2.3 Tracklet Proposal Network for Multi-Object Tracking on Point Clouds . . . 23
2.3.1 Pipeline . . . 23
2.3.2 Results . . . 25
2.4 Sparse Fuse Dense: Towards High Quality 3D Detection With Depth Completion . . . 27

2.4.1 Pipeline . . . 28
2.4.2 Results . . . 29
2.5 SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Clouds . . . 31
2.5.1 Pipeline . . . 32
2.5.2 Results . . . 33
2.6 SECOND: Sparsely Embedded Convolutional Detection . . . 34
2.6.1 Pipeline . . . 35
2.6.2 Results . . . 37
2.7 PointPillars: Fast Encoders for Object Detection from Point Clouds . . . 38
2.7.1 Pipeline . . . 39
2.7.2 Results . . . 40
2.8 3D Object Detection For Autonomous Driving Using Temporal LiDAR Data . . . 41
2.8.1 Pipeline . . . 41
2.8.2 Results . . . 43
2.9 Discussion and Conclusions . . . 44

3 Objectives **47**
3.1 Schedule Plan . . . 47
3.2 Research Contributions . . . 48

4 Methodology **50**
4.1 Datasets . . . 50
4.1.1 KITTI . . . 50
4.1.2 *Waymo* . . . 52
4.1.3 NuScenes . . . 53
4.1.4 Summary . . . 54
4.2 Data Representation . . . 55
4.3 Pipeline Structure . . . 57
4.4 Evaluation Metrics . . . 58

5 Temporal 3D Object Detection **61**
5.1 Hardware & Software . . . 61
5.2 The Dataset . . . 62
5.2.1 Sensors Data . . . 62
5.2.2 Classes . . . 63
5.2.3 Data Structure . . . 65
5.3 PointPillars . . . 66
5.3.1 Data Pre-processing . . . 66

5.4 Temporal PointPillars . . . 67
5.4.1 Pipeline . . . 68
5.4.2 Data Pre-processing . . . 68
5.4.3 Memory Bank . . . 69
5.4.4 Temporal Module . . . 70
5.5 Benchmark and Discussion . . . 70
5.5.1 SOTA PP versus Our PP . . . 70
5.5.2 PP-REC versus Our Temporal PP . . . 71
5.5.3 RNN versus LSTM . . . 72
5.5.4 PP versus LSTM . . . 74
5.5.5 Model Learning Analysis . . . 75
5.6 Visualisation and Demonstration . . . 78

6 Conclusion and Future Work **82**
6.1 Contributions . . . 83
6.2 Risks and Limitations . . . 83
6.3 Next Steps . . . 84

Bibliography **86**

List of Figures

1 AI, ML and DL association, extracted from [8]. 7
2 Deep Neural Network architecture, acquired from [3]. 8
3 Example of LiDAR Point Cloud (with bounding boxes), produced by [16]. 9
4 OD application example, extracted from [37]. 10
5 KITTI [13] dataset image sequence with human annotated bounding boxes. 11
6 Standard convolutional RNN cell. 13
7 ConvLSTM cell. 14
8 ConvGRU cell. 15

9 LSTM pipeline, presented in [20]. 18
10 2D NMS example, extracted from [41]. 19
11 3D-MAN pipeline, proposed in [58]. 21
12 PC-TCNN based pipeline, proposed in [55]. 24
13 A long-time 340 meters tracking example with an heavily occluded object (cyan bounding box with ID 2), produced by [55]. 26
14 SFD pipeline, proposed in [56]. 28
15 Illustration of SynAugment, presented in [56]. 29
16 Comparing SFD and Voxel-RCNN [10] detections in two scenes (each square). In each square, left columns represent SFD detections, and the others represents Voxel-RCNN detections, illustrated by [56]. 30
17 Comparison results with SOTA methods on the 3D KITTI [13] test set for car detection, extracted from [56]. 30
18 Inference time and AP between SOTA methods, on 3D and BEV benchmark. “Our” refers to the SE-SSD model. Presented in [60]. 31
19 SE-SSD pipeline, proposed in [60]. 32
20 Shape-aware data augmentation, presented in [60]. 33
21 SECOND pipeline, produced by [57]. 35
22 PP pipeline, presented in [24]. 39
23 PP variants speed vs mAP against other methods on KITTI [13] validation set, produced by [24]. 41

24 PP-REC pipeline, presented in [32]. 42
25 PP-REC and PP data preparation, produced by [32]. 42

26 KITTI information illustration example, extracted from [14]. 51
27 *Waymo* information illustration example, extracted from [45]. 52
28 NuScenes information illustration example, extracted from [6]. 53
29 Raw point cloud visualisation, presented in [47]. 55
30 BEV point cloud representation. 56
31 Example of a voxelized point cloud in a semantic scene, adapted from [1]. 56
32 RV point cloud representation, adapted from [4]. 57
33 OD pipeline example. 57
34 2D NMS example, extracted from [41]. 59
35 IoU visualisation, extracted from [21]. 60
36 Bounding box centre distance visualisation. 60

37 NuScenes vehicle's sensors, extracted from [6]. 63
38 Classes distribution on train and validation set, extracted from [6]. 63
39 Density map for bounding box annotations in which the radial axis is the distance in meters from the ego-vehicle and the polar axis is the direction angle with respect to the ego-vehicle. More bounding box annotations in a region lead to a darker bin. Figure presented in [6]. 65
40 Processing sample from configuration file. 66
41 PP frames concatenation. 67
42 Temporal PP pipeline. 68
43 Temporal frames grouping processing. 68
44 Temporal frames grouping processing with sliding window. 69
45 Losses progression. Green and blue colours refers to training loss and grey and red colours to validation loss. 76
46 Localisation loss progression for Car class. Orange colour refers to training loss and blue colour to validation loss. 76
47 RPN accuracy progression. Blue colour refers to training accuracy and Orange colour to validation accuracy. 77
48 AP progression with 0.5 and 4 distance thresholds for Car class. 77
49 Visualiser with different views. Green bounding boxes refer to ground-truth and red bounding boxes refer to LSTM PP predictions. Score refers to the model confidence on the prediction. 78
50 Comparative views for point cloud 38 from validation set, with ground-truth and predictions. 79
51 Scene annotation differences from the same sequence. 80
52 Similar classes ground-truth and prediction comparison. 81

List of Tables

1 LSTM frames number benchmark comparison on *Waymo* [45] open validation set (for class vehicle). Values produced by [20]. 19

2 Results on 3D OD on *Waymo* [45] open dataset validation set (class "vehicle" on level 1 difficulty). Models are single-frame, unless mentioned otherwise. Values extracted from [20]. 20

3 Vehicle velocity breakdowns through different number of frames using PointPillars [24]. Values extracted from [58]. 22

4 Vehicle velocity breakdowns through different number of frames using 3D-MAN. Adapted from [58]. 22

5 Validation AP comparison among different multi-frame fusion approaches on class "vehicle". Adapted from [58]. 22

6 3D Average Precision (AP) results on *Waymo* [45] open dataset validation set for class "vehicle". *Methods utilise multi-frame point clouds for detection. Acquired from [58]. . . . 23

7 Results on the KITTI [13] validation set using different number of frames per temporal sequence, adapted from [55]. 25

8 Car tracking results on the test set of the KITTI [13] tracking benchmark, extracted from [55]. 27

9 Comparisons between different implemented techniques on the 3D KITTI [13] validation split for car detection, produced by [60]. 34

10 SE-SSD comparisons with SOTA on the 3D and BEV KITTI [13] benchmarks, extracted from [60]. 34

11 SECOND results on 3D KITTI [13] benchmark, presented in [57]. 37

12 SECOND and small variant results on "car" class on KITTI [13] 3D validation set, produced by [57]. 37

13 SECOND and small variant results on "car" class on KITTI [13] BEV validation set, presented in [57]. 38

14 PP results on 3D KITTI [13] benchmark, presented in [24]. 40

15 PP-REC results against PP [24], values obtained in [32]. AP@all is averaged AP across the four nuScenes [6] matching thresholds (0.5, 1, 2 and 4). All PP-REC(-S) models use groups of 3 frames for uniformity. 43

16 PP-REC and PP [24] speeds, values extracted from [32]. 43
17 Results between LSTM [20] and 3D-MAN [58] methods on the *Waymo* [45] open dataset. * refers to multi-frame methodologies. 44
18 SOTA results on the KITTI [13] 3D benchmark for car detection. Values are extracted from [13]. 45
19 SOTA results on the nuScenes [6] validation set. Values are extracted from respective papers [24] [32]. 45
20 Gantt Diagram. 48
21 Dataset comparison. “k” represents thousands. 54
22 Evaluated classes information per frame. “Mean” and “Max” represent the mean and maximum number of objects per frame, respectively. 64
23 Results between original PP [24] and our PP proof of concept. * represent approaches with self-declared AP values. 70
24 Results between PP-REC [32], our replication and our Temporal PP approaches. IT means inference time. * represent approaches with self-declared AP values. All methodologies were trained and evaluated with 3 sweeps per point cloud. 71
25 Results between our (Standard) RNN PP, and our LSTM PP with different values for sweeps per point clouds. IT means inference time. 73
26 Results between original PP [24], and our LSTM PP with different values for sweeps per point clouds. IT means inference time. 74

Acronyms

AD	Autonomous Driving
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
AV	Autonomous Vehicle
BEV	Bird Eye View
CNN	Convolutional Neural Network
CPU	Central Processing Unit
dist	distance
DL	Deep Learning
DNN	Deep Neural Network
FCN	Fully Connected Network
FoV	Field of View
FPS	Frames Per Seconds
GPS	Global Positioning System
GPU	Graphics Processing Unit
GRU	Gated Recurrent Unit
Hz	Hertz
IMU	Inertial Measurement Unit
IoU	Intersection of Union
LiDAR	Light Detection and Ranging

LSTM Long Short-Term Memory

mAP mean Average Precision
ML Machine Learning
MOTA Multiple Object Tracking Accuracy

NMS Non-Maximum Suppression
NN Neural Network

OD Object Detection

PP PointPillars
PP-REC PointPillars Recursive

RADAR Radio Detection and Ranging
RAM Random Access Memory
ReLU Rectified Linear Unit
RGB Red Green Blue
RNN Recurrent Neural Network
RoI Region of Interest
RPN Region Proposal Network
RV Range View

SAE Society of Automotive Engineers
SE-SSD Self-Ensembling Single-Stage Object Detector
SECOND Sparsely Embedded Convolutional Detection
SFD Sparse Fuse Dense
SOTA State-of-the-Art
SSD Single-Shot Detector

TPU Tensor Processing Unit

VFE Voxel Feature Enconding

Chapter 1

Introduction

This chapter presents an introduction to this dissertation, exposing the motivations and challenges, as well as the analysis of its scope and context of this work. The topics covered are Autonomous Driving (AD), Artificial Intelligence (AI), Machine Learning (ML), Deep Learning (DL), machine perception, Object Detection (OD) and temporal architectures. It will also be referred how the document is structured.

1.1 Motivation & Challenges

The world is growing at a fast rate not only in population terms, but also technologically. This development needs to be accompanied by evidence that proves an healthy evolution. This evidence must corroborate with all the quality and security claims made by enterprises that develop these products. This process helps people to trust all these technological products that are showing up more and more often.

The subject of AD is one example which has been closely followed by people, however with a lot of scepticism on their part. People won't feel safe on an AD if they don't trust the quality and safety of this technology. A lot of works [42, 36] that studies people's behaviour towards AD are being made in order to create reliable and trustworthy products.

Several brands, not only directly focused on the automotive industry, e.g. *Tesla*, *Kia-Hyundai*, *Ford* and *Audi*, but also other major companies, e.g., *Google*, *Apple* and *Huawei*, are actively trying to design new ways to bring a fast and secure adoption of AVs. An AV has to be able to localise itself in the road and needs to take into account all the other road users, e.g., vehicles, cyclists and pedestrians. Therefore, OD is one of the subjects that is being explored the most, due to its heavy importance in the safety of vehicle locomotion.

Having a personal self-driving vehicle will also bring a level of comfort which plenty of people would be glad to have. Based on this change, users view of AV will no longer be seen only as a means of transport,

but also a space for leisure and work, optimising their time during their rides, and do activities, such as reading, watching movies, playing games, or other things they need to do, for instance attend work video calls or other job related tasks.

This dissertation will explore current work on this scope, analysing current solutions and providing value to the topic of OD in an AD context.

1.2 Scope

This work will regard a variety of technological subjects, in which all of them are particularly decisive on the matter of AD. Combining them will provide an adequate environment for the development and implementation of a robust solution.

1.2.1 Autonomous Driving

AV is the main concept of this work, and is a topic that has been growing immensely in the last few years. However, despite this fact, we are still midway of what is possible to implement. AD has been divided by the SAE International into six different levels[1], from no driving automation at all, up until fully automated driving. These levels are periodically being revised, with the latest revision published in April 2021[2]:

- **Level 0 - No Automation**

  In this introductory level, drivers have complete control over the vehicle and must constantly supervise any rudimentary extra support features. The vehicle can have warning and momentary assistance features, e.g., automatic emergency braking, blind spot warning and lane departure warning.

- **Level 1 – Driver Assistance**

  As a level 1, safety and convenience features can take control over the vehicle for longer periods of time. Nonetheless, drivers are still the ones fully responsible for driving, hence they have to oversee those features.

  Steering or braking/accelerating features can be the provided support to the driver. Nevertheless, these features cannot be supplied at the same time. The two most used features are lane centring that keeps the vehicle centred in the lane or cruise control which controls the vehicle's speed and, in more advanced configurations, the distance to the vehicle in front.

[1] https://www.sae.org/blog/sae-j3016-update
[2] https://www.sae.org/standards/content/j3016_202104/

- **Level 2 – Partial Automation**

  Nowadays vehicles are frequently placed in the Level 2 type. This level is the bridge from assistant features to automated driving features. Multiple assistance systems can be combined with each other, in favour of the vehicle being able to perform individual driving manoeuvres independently. Once again, in spite of drivers being able to give the control to the vehicle during these operations, they still have responsibility over the car manoeuvre, consequently having to remain alert for the purpose of being able to intervene if something doesn't work as expected.

  Most of the example features are the same as level 1, i.e., lane centring and adaptive cruise control. However, at this level, these can be used at the same time. Another example is letting the vehicle park by itself.

- **Level 3 – Conditional Automation**

  Things start to get more interesting with level 3. Under limited conditions vehicles can self-drive and drivers may remove their hands from the wheel and even focus on other activities. Although, if all required conditions aren't met, the automation features won't be able to operate, consequently the driver will oblige if the vehicle requests to take control and drive.

  This level allows highway driving to be subjected to interesting changes. Driving in a straight line can be tedious to some drivers, and it is something straightforward to automate when compared to city traffic navigation. During this time, drivers may focus on other tasks which they find more relevant. The driver takes control again when it's time to exit the highway, or when the vehicle perceives some unforeseen situation, such as an animal crossing the road or road work.

  Due to level 3 being the next step for AD from our current state, various questions and challenges arise, e.g., while the vehicle is self-driving what kind of tasks could the driver perform? At what distance should the vehicle provide a proximity warning trigger to the highway exit? What should the vehicle do if the driver doesn't take control when he should? All these security issues must be addressed for the vehicle to be able to handle as many situations as possible.

- **Level 4 – High Automation**

  In this stage the driver won't be required to take control over the vehicle. It's expected that drivers will be able to perform longer duration tasks that completely divert the focus from driving, e.g, getting to the back seat or even sleeping. Similarly to the previous level, only in restricted circumstances will the vehicle be allowed to operate. At this level, taxis without a driver may start circulating in a restricted area and only in certain conditions.

  People who are driving enthusiasts and want to decide when to drive themselves, don't need to be concerned since they still can take back the vehicle control (e.g., pedals and steering wheel) whenever they desire. Nevertheless, pedals and steering wheel will become optional in some vehicles

and if they consider their further operation poses a risk to the occupant's safety, the vehicle will proceed to make a complete safe stop.

- **Level 5 – Full Automation**

  The last level of AD is level 5 where a fully AV can self-drive everywhere, despite the conditions. Pedals and steering wheel will cease to exist inside the vehicle, as well as the driver, which instead will become a passenger. With no driver required, the vehicle doesn't need someone inside with a driving license to operate.

  This level provides several innovative solutions for transportation, converting taxis to fully AVs with more efficient routes and more affordable to the consumer. This will also help people who can't drive due to their age, such as children and elders. It has been studied and concluded that human error is the main reason for road accidents. For instance, in [48] it was determined that human error was the sole cause in 57% of all accidents and was a contributing factor in over 90%. Therefore, a robust implementation of self-driving would greatly increase road safety, and decrease the number of minor and fatal accidents.

1.2.2 Artificial Intelligence

The creation of the first digital computer in the 1940s has revealed the potential of this programmable systems, that started of being able to compute proofs for mathematical theorems or to play logical games (e.g., chess), with considerable proficiency. Within a decade this potential started to grow into a finer concept.

AI was defined and redefined a couple of times since it's birth in [49], published in 1950 by Alan Turing the "father of computer science".

The definition accepted by most people was described in [31], mentioning that AI "is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence, but AI does not have to confine itself to methods that are biologically observable". Unlike other definitions which limit AI capabilities to what is possible to achieve with a human mind, this definition defends it isn't restricted only to the intelligence observed in humans, considering its superior potential.

Even after more than 70 years, despite the computers' growing performance and resources, these continue to be inferior to humans in various domains and assignments which require everyday experience and knowledge. Nevertheless, in certain specific tasks, some systems were able to achieve results similar or even better than human professionals and experts, making nowadays AI exceptional in particular applications such as computer search engines, voice or handwriting recognition and medical diagnosis.

AI is essential in the environment of developing a fully AV. The vehicle must be able to navigate through never experienced situations, relying on information learnt from previous similar events.

There are two types of AI, being normally referred as weak AI and strong AI:

- **Weak AI**

  Also referred as narrow AI, weak AI is the method developed having in mind and limited to a specific task or narrow area. Almost every AI in our surroundings is driven by this type of AI. Some of the implementations are *Apple's Siri*, *Amazon*'s suggestion engine, *Google* news feed and AV.

- **Strong AI**

  Strong AI is still in a theoretical form, and is divided in general AI and superintelligence. Systems with this type of intelligence would be capable of matching human intelligence. It would have a form of consciousness in order to solve problems, learn and plan for the future. On the other hand, superintelligence would surpass human intelligence, unlocking various complex tasks and with faster execution times.

  Despite being theoretical, strong AI is currently being researched, and some scientific articles and books are available discussing it, as is the example of the book "The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World"[11] by Pedro Domingos.

1.2.3 Machine Learning

ML is a sub-field of AI and focuses on the development of machines capable of utilising algorithms and provided data to learn gradually and improving its accuracy, similarly to humans.

The learning process starts with data acquisition, in the form of direct experience or external data. The machine will search for patterns in the given data, trying to learn which ones are key features. This way, the system will attempt to correlate information acquired in order to perform the best action on similar examples in the future.

ML can be divided into four different learning methods:

- **Supervised learning**

  Supervised learning algorithms train using labelled datasets with the objective of accurately predict similar events in the future. The learning algorithm starts by analysing the training dataset and creates an inferred function. The model is then fed with the input data, and, comparing with the correct output, regulates its layers weights until it fits accordingly. Underfitting and overfitting are avoided through processes of data augmentation, cross validation and other regularisation techniques.

  Some supervised learning implementations help solve a variety of real-world problems, e.g., problems related with image classification or image segmentation.

  The algorithms used by supervised learning include Neural Networks (NNs), naïve bayes, linear regression, logistic regression, random forest, support vector machine, and others.

- **Unsupervised learning**

  Contrary to supervised, unsupervised learning analyses unclassified and unlabelled datasets. The algorithm explores patterns in the training data, trying to associate and cluster similarities without the intervention of a human.

  Being able to link similar and distinct patterns between data makes this approach a great solution for customer segmentation, content recommendation, data dimension reduction and image and voice analysis.

  The algorithms used in unsupervised learning can come in a form of NNs, probabilistic clustering methods and anomaly detection methods.

- **Semi-supervised learning**

  Semi-supervised learning lies between supervised and unsupervised learning. Normally, systems which use this kind of learning are able to improve its accuracy effectively.

  Typically, this method is used when the dataset has a great size, and only a small portion is labelled, due to the lack of specialists or financial resources. The small labelled dataset is fed into the algorithm to guide classification and feature extraction from the larger unlabelled dataset.

- **Reinforcement learning**

  Reinforcement learning algorithms learn while interacting with the environment via trial and error actions. Despite being similar to supervising learning, this method, instead of receiving labelled data as input, it gets a reward dependent on its current state and action performed by an agent on a specific environment.

  A simple requirement of this method is the implementation of a reward function in order to provide feedback to the agent which will use it to learn the best action to take. The objective of this algorithm is to maximise its reward, leading to perform several successful actions consistently and consecutively.

ML has been a key factor in the data science field growth, making people's life easier. The complex problem of imitating human intelligence has an enormous potential in improving the population's lifestyle, being able to perform tedious tasks which people prefer not to do. Consequently, people are still afraid of its impact on their jobs.

Other problems are related to people's privacy, bias, discrimination and accountability. People's privacy becomes endangered due to this kind of data being delicate and very important for the algorithm training and validation, e.g., in products recommendation. Bias and discrimination are ethical problems on account of the possible bad intentions of corporation in using data to manipulate people's minds and actions. Accountability is related with the lack of legislation to regulate ML driven algorithms, where in the case of miss behaviour, someone needs to take responsibility, even when it's a physical machine, for-instance a fully AV.

1.2.4 Deep Learning

DL is a sub-field of ML, distinguished by the algorithms and the type and amount of data used during learning. Therefore, AI, ML and DL are deeply connected, both DL and ML being subsets of AI as seen in Figure 1.

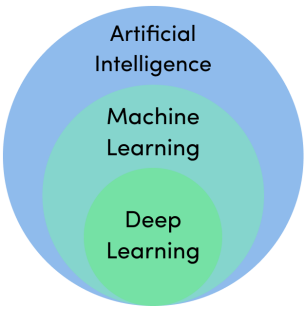


Figure 1: AI, ML and DL association, extracted from [8].

DL and ML are often used interchangeably, however there are nuances between the two. One of them is the way models in each method learns. While ML is more human dependent, needing AI specialists regulating feature importance in the data, in DL the process of feature extraction is done by the NN autonomously, which mitigates the complexity of data preparation and enables the use of bigger datasets, allowing scalability to the method. It also potentiates the use of crucial features possibly missed by the experts. As a disadvantage, DL requires the use of high computational resources for high-level processing.

DL consists in the application of Artificial Neural Networks (ANNs) methodologies, normally simplified to NNs, which are inspired by the architecture of the human brain system. NNs deal with information through one or more hidden layers, having a sub-category named Deep Neural Networks or DNNs, which need to have at least two hidden layers.

As seen in Figure 2 the architecture of a DNN contains multiple layers of nodes connected with other layers nodes. The input and output layers are the only two visible layers and the one's in between are called hidden layers. The network has an input layer which consumes the data to be processed, while the output layer is where the network provides its result or prediction. The hidden layers extract and process the features or information from the data, learning which of them are relevant and which are not.

As previously stated, a NN is comprised of layers, which in turn are composed of nodes. Each node may have multiple links from previous nodes, where each link has its own weight, which represents link importance, and how much of the signal which is passing through the link will reach the nodes. Normally, link weights referring to a node are called node weights. Node weights are essential to the NN learning process: The input data goes through the weighted nodes, and when the network provides the final output, the node weights are adjusted based on how close or far the prediction was from the expected output. This weight adjustment process is referred to as backpropagation, and it can be customised using a loss function (e.g., categorical crossentropy and mean squared error, to calculate the difference between the GT and predicted values from the network) and an optimiser (e.g., Adam [22] and SGD, responsible to

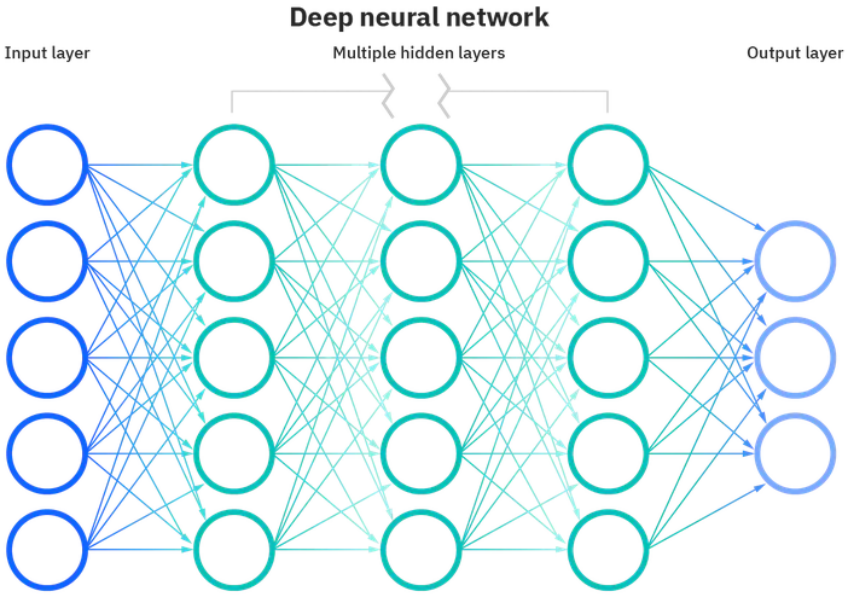


Figure 2: Deep Neural Network architecture, acquired from [3].

adapt the weights of the network per epoch), which are the ones responsible for determining how wrong or right the prediction was.

NNs can have more advanced architectures which can be employed in numerous types of applications:

- **Convolutional Neural Networks (CNNs)** can use their convolutional layers to discover patterns and features in a frame, allowing their use in computer vision, image classification and image segmentation applications. In an image context, every added convolutional hidden layer will increase the complexity of the feature extraction. Initially, it might be able to detect edges, but as we progress through the layers more complex shapes will appear.

- **Recurrent Neural Networks (RNNs)** may contain cycle connections between nodes, enabling the output of certain nodes to be the input to the same nodes. RNNs are normally applied on time series and sequential information, which makes them a great contender in natural language, speech recognition and video applications.

For an adequate training of a DL model, an extensive dataset is required, with thousands and thousands of data, but also a great computational power, in a form of number and quality of GPUs or TPUs.

1.2.5 Machine Perception

Humans consider that intelligence is a combination of conclusions, based on logical and structured decisions on perceived patterns. Machine-based intelligence comes in a combination of two factors as well. Typical computer tasks mimic the logical intelligence of human brains, whilst DL based AI impersonates human perception via interpretation of features in data.

Thereby, perception is the ability of scanning an environment using sensory organs. These can be biological in the human side, or artificial if we are considering machines. Computers take advantage of attached hardware to perceive the environment. Initially only a keyboard or mouse could be used as the input mechanism, but technological evolution, both in software and hardware, led to computers being able to deploy more advanced sensors to execute perceptually sophisticated tasks.

While humans present 5 senses (i.e., hearing, smell, taste, touch and vision), machines are able to implement complex sensors that translate in greater accuracy and wider range of senses, including not only computer vision, machine hearing, machine touch and machine smelling, but also, and not exclusively to, information related to temperature, pressure, humidity, air pollution and fire sensors.

Nowadays, the proficiency of machine perception already allows systems to use optical sensors to detect and identify objects, navigate through buildings doing a series of simple tasks and have vehicles drive at safe speeds on highways and urban roads.

In AD, perception is of crucial importance. The vehicle needs to perceive, in real-time, its surroundings through onboard sensors, such as Global Positioning System (GPS), cameras (RGB), Radio Detection and Ranging (RADAR) and Light Detection and Ranging (LiDAR).

The perception technology explored in this work will be LiDAR. This method uses light, more specifically laser pulses, to measure distances from the sensor to objects, and respective reflective intensity. It is normally used by airborne systems to generate precise information about distant objects. Being more accurate to measure distances than other methods (ultrasonic, infrared, RADAR, etc), LiDAR can be more reliable, and consequently, more secure.

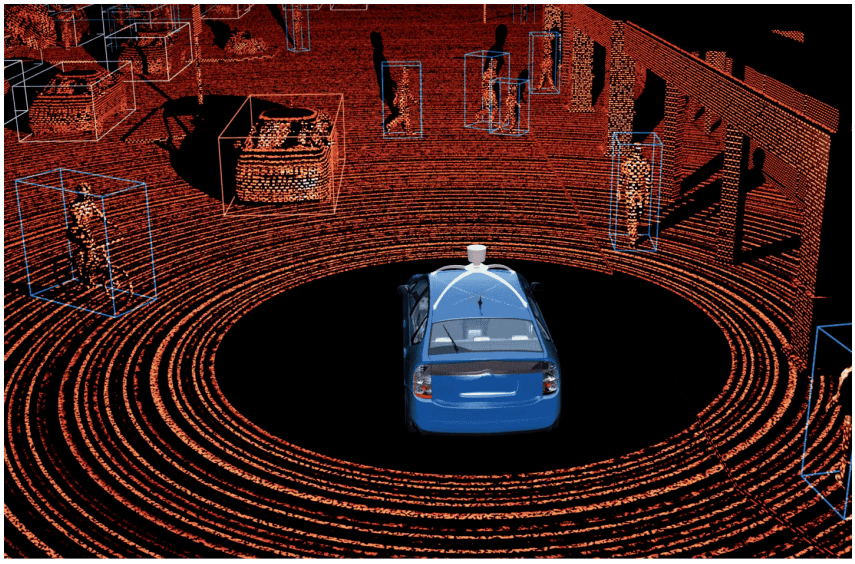


Figure 3: Example of LiDAR Point Cloud (with bounding boxes), produced by [16].

Road vehicles can also employ this technology, as demonstrated in Figure 3. This way, cars are able to use this form of perception to be aware of their surroundings, including other vehicles on the road or people passing the crosswalk. Since LiDAR point clouds are raw information captured by the sensors, it is necessary to teach models to interpret and distinguish objects in that data.

1.2.6 Object Detection

OD is typically a computer vision and image processing technique that tries to identify and locate objects accurately in a given image or video. Detections are composed of bounding boxes around the objects which locate where it is in the scene, followed by a label that identifies the object class. As seen in Figure 4, the objects detected can be, e.g., vehicles, people or gadgets.

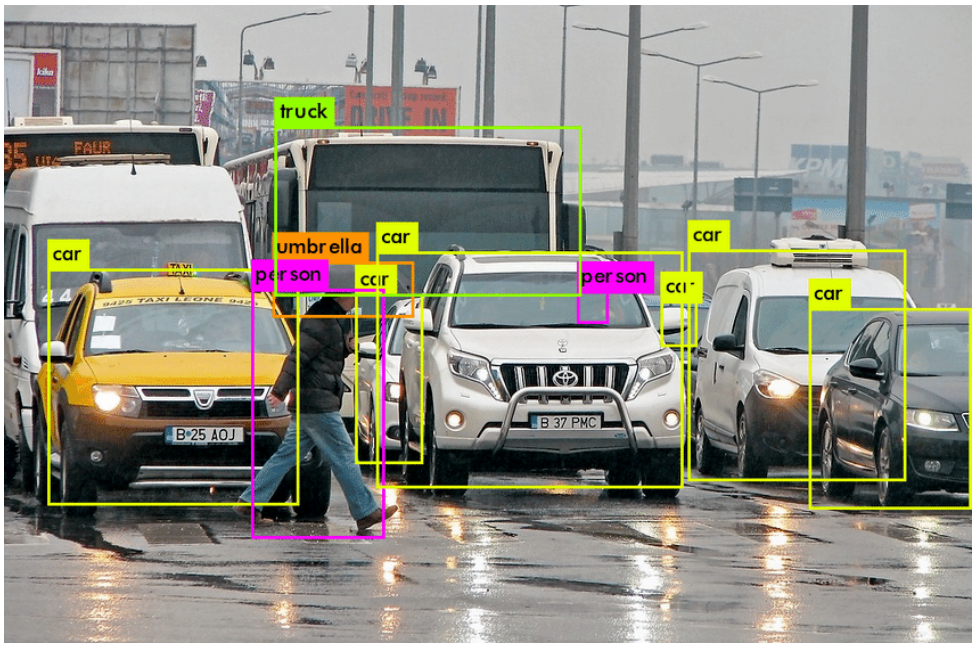


Figure 4: OD application example, extracted from [37].

Each object class has a set of important features which help in identifying the class. A dog, for example, normally has four legs, two ears and a tail. Some of these aspects can be shared with other classes, like with a cat, but two classes never have the same exact number and type of features.

OD is often mistaken with image recognition. While OD has to locate and identify various objects in an image, image recognition is a simpler task, as, normally, only one class can be found in the image, and its job is to give the correct class label to the image.

As mentioned previously, some tasks can use ML and DL methodologies, and OD is one of these techniques. DL methodologies are computationally intensive and typically produce superior results when applied to more complex problems than traditional ML methodologies. Due to the complexity of OD, a large majority of DL-focused SOTA papers, associated with a variety of OD-related fields, have achieved the best results.

OD technique is used in various fields and tasks, e.g., image annotation, image segmentation, entity counting, face detection, activity recognition, video surveillance, ball tracking in sport matches, and, even, more complex problems like self-driving cars.

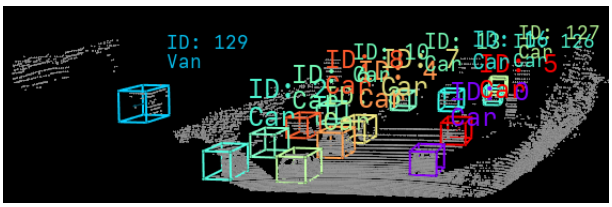
1.2.7 Temporal 3D Object Detection

The concept of temporal OD is an expansion to the traditional single-frame OD. Single-frame OD handles each frame separately to adapt the weights of the network, i.e., focusing on the information extracted of a single frame. Temporal OD takes OD a step further, transforming a single-frame approach into a sequential one, providing the means to use information across a sequence of frames to detect objects in the last frame, i.e., temporal context.

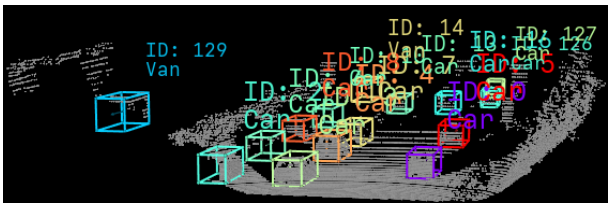
Objects in a scene tend to maintain their attributes in certain values, e.g., a vehicle should always have the similar length, width and height. A temporal network will use memory modules to take advantage of features processed previously and obtain cumulative information about an object by shaping the values or enhancing the certainty of already known attributes. This approach is also particularly relevant to tackle other problems, such as object occlusion or range limitations.

There is no consensus regarding whether certain pre-processing techniques, e.g., concatenating information from multiple frames into a single one, maintain the implemented approach as single-frame or convert it into multi-frame. In this work, due to the fact that frame concatenation loses all temporal characteristics, the implemented methods will still be considered single-frame methods that use multi-frame pre-processing. On the other hand, techniques that use multi-frame information temporally will be referred to as temporal.

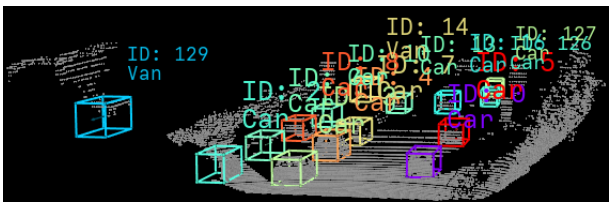
In the same way that temporal OD is an expansion to the traditional single-frame OD, 3D OD is an extension of classic 2D OD. Normally, visual information is captured in a 2 dimensional cartesian system or, in other words, 2D frames. However, information can also come in 3D frames also. A good example of 3D data, is the information captured by LiDAR sensors, named point clouds.



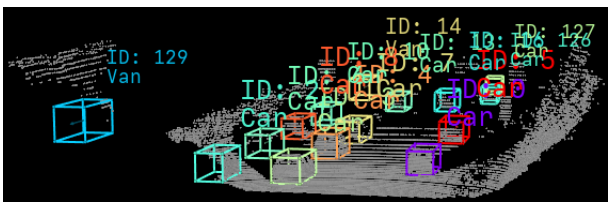
(a) Frame 1.



(b) Frame 2.



(c) Frame 3.



(d) Frame 4.

Figure 5: KITTI [13] dataset image sequence with human annotated bounding boxes.

The images in Figure 5 represent a sequence of four 3D frames from the KITTI [13] dataset. The first frame is on the top left and the last on the bottom right. A van with the ID 129 can be observed moving

through the road, with its number of cloud points decreasing due to occlusion, where in the last one, very few points are representing it.

It's extremely difficult for a single-frame model to predict accurately the centre and dimensions of that entity on that last frame, however, as the object was present in previous frames, information from those frames will be useful for a good prediction on the part of a temporal model.

Temporal architectures may take the following forms:

- **RNNs** were inspired by biological neurons and have been around since at least 1986, with the three most prevalent forms being Standard RNN, LSTM and GRU. These architectures are further discussed in section 1.2.8.

- **Transformers** are still in an embryonic stage, having started in 2017 with the publication of the paper [51], which focused on natural language processing. This architecture is entirely based on attention mechanisms, eliminating convolutions and recurrence, enabling parallel data processing. Very recently, this architecture has been adapted to an image context [12] in 2020 and even video [2] in 2021.

1.2.8 Recurrent Neural Network Architectures

RNN concepts have existed for a very long time. In 1943, a neurophysiological paper [33] discussed nervous activities, neural events and relations between them, describing how they can be represented as nets and, in more complex cases, calling them nets with circles. However, the first publicly accessible mention of recurrent networks was made in 1986 by [40], which only debated feed-forward networks. However, they mention that "Minsky and Papert point out, there is, for every recurrent network, a feed-forward network with identical behavior", indicating that RNNs are introduced in 1969 by [34], but this book isn't freely available.

RNNs are designed to process sequential data, including text, audio, and video forms, among others. In this study, sequential data will be represented as videos composed of LiDAR frames. Thus, RNN are able to use information from prior frames to get the current result.

An RNN architecture is composed of one or more cells attached sequentially. More cells cause an increase in complexity and decrease in processing speed.

Since their initial conception, RNNs have undergone a number of mutations; these modifications were made in an attempt to solve problems that previous architectures had. In this section, the three most popular RNN implementations will be presented and their goals will be discussed.

1.2.8.1 Standard Recurrent Neural Network

As illustrated in Figure 6, the most basic form of an RNN cell consists of only a convolutional layer and an activation function.

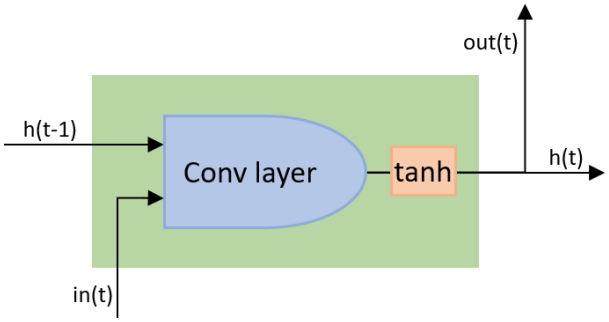


Figure 6: Standard convolutional RNN cell.

In the example, the "tahn" activation function is used, however, other activation functions, such as "relu" or "sigmoid", may also be used.

The processing frame (in(t)) and hidden state from previously processed frames (h(t-1)) are inputs to the cell. Both are passed through the convolutional layer, with the output going through the activation function to become the cell output (out(t)) and the hidden state for subsequent frames (h(t)). After calculating convolutional layer weights during training, these values will be shared across all sequential frames throughout inference.

This conventional RNN architecture has an issue with the phenomenons of vanishing and exploding gradient. For the back-propagation process, RNNs are expanded into deep feed-forward networks, in which a new layer is formed for each time step in the input sequence. The gradient is computed at each time step, and it is utilised to update network's weights.

The vanishing gradient problem originates from large sequences. If the influence of the preceding frame on the present iteration is minimal, then so will be the gradient value. This causes the gradients to diminish exponentially as we back-propagate. Smaller gradients have negligible effect on the weight update process. Stopping the updating of weights will prevent the network from learning.

The exploding gradient, on the other hand, occurs when differences in training sequence data are so large that weights change by a massive margin, never converging to any value, and thereby preventing the model from training as well.

Two additional popular RNN variants, GRU and LSTM, were created in an effort to overcome these limitations.

1.2.8.2 Long Short-Term Memory

Hochreiter and Schmidhuber proposed LSTM in 1995 in a technical review, with the main publication [19] being published in 1997. It was the first RNN architecture to address the vanishing and exploding gradients problem.

Figure 7 encapsulates the most popular LSTM cell. Not often included in other RNN variants, the LSTM adds a cell state, which is used to store the main information over time, being modified by the various gates: the input gate, the forget gate and the output gate. Gates use a sigmoid activation to produce values

between 0 and 1, to act as a filter when multiplying with the information. Together with the candidate cell, these gates play a crucial role in determining which portion of the information passing through the cell is significant for a more accurate prediction:

- **Forget gate** (f(t)) determines how much of the preceding t cell state (c(t-1)) is relevant to the result. This gate is responsible for mitigating the problem of vanishing and exploding gradients by choosing which previous information to retain and which to discard.
- **Input gate** (i(t)) controls which and how much of the input or current data should be utilised to predict the output.
- **Candidate cell** ($\hat{c}$(t)) is the information obtained from processing the current input (in(t)) data and previous hidden state (h(t-1)), which will be filtered by the input gate and added to the processed cell state.
- **Output gate** (o(t)) identifies determines which contents of the determined current cell state should be taken into account for generating the output (out(t)) and next hidden state (h(t)).

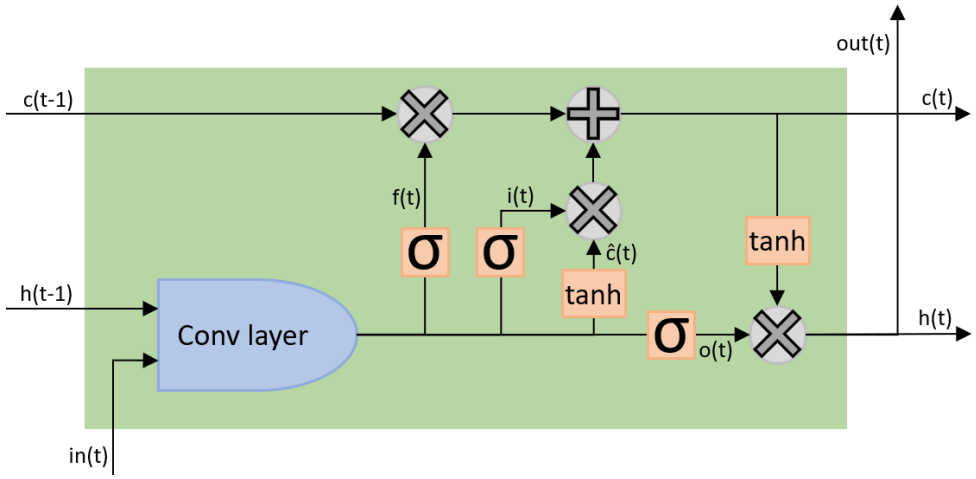


Figure 7: ConvLSTM cell.

The gates and candidate cell could have been produced from one convolution layer each, however employing only one layer with a number of parameters equal to the sum of the other's parameters is more efficient and achieves identical results.

Although the LSTM design has experienced several changes since its origin, leading in other similar variants, the one presented here is the most prevalent.

1.2.8.3 Gated Recurrent Unit

GRU is highly inspired by the younger sibling of LSTM, and was proposed by Chung et al. [7] in 2014. GRU attempts to overcome the same problem of vanishing and exploding gradients as LSTM. In comparison to LSTM, GRU has a simpler design with fewer parameters and operations, which enables it to achieve identical results more efficiently.

As illustrated by Figure 8, GRU is composed of a candidate cell ($\hat{h}$(t)) and two gates: the update gate and the reset gate. Not only does it have one fewer gate than LSTM, but the hidden state also behaves as the major information flow, similarly to the cell state.

- **Update gate** (z(t)) specifies which and how much of previous information must be relayed to future steps. It is analogous to the output gate of the LSTM architecture.
- **Reset gate** (r(t)) decides the amount of prior information to forget. It resembles the combination of the input gate and forget gate in the LSTM architecture.
- **Candidate cell** ($\hat{h}$(t)) is the information obtained from processing the current input (in(t)) data and hidden state after reset gate (r(t)) multiplication, which will be filtered by the update gate (z(t)) and added to the processed hidden state, which is used as output (out(t)).

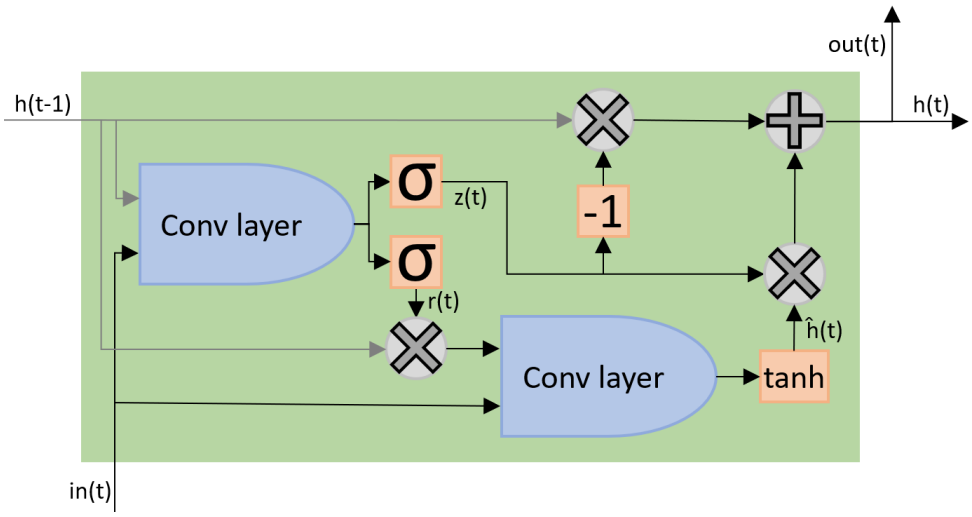


Figure 8: ConvGRU cell.

GRU has also different variants, however the one presented is the most popular. Since one of the convolutions requires the output of another, they cannot be combined into a single convolution layer, resulting in a decreased speed efficiency. Depending on the data used, it may be faster or slower than the LSTM architecture.

1.3 Document Structure

Next chapters will further explore the theme of temporal 3D OD using LiDAR, having each one a different contribution for its understanding.

Chapter 2 will present some relevant SOTA algorithms, studying their pipelines and key techniques. At the end of the chapter, pros and cons of the analysed SOTAs will be stated, and confronted with each others to decide which of them are predominant.

Chapter 3 will refer the objectives of this dissertation as well as how the problem will be addressed and what will be done to reach the respective solutions.

Chapter 4 will introduce some important temporal OD information, such as datasets and their benchmarks, data representations, standard deep learning architectures, proposed pipeline structure, evaluation metrics and other relevant concepts.

Chapter 5 will demonstrate how the developed work was implemented, mentioning the hardware and software used, introducing the dataset and baseline in greater detail, highlighting how the temporal implementation was structured, and concluding with a benchmark and visualisation of the results obtained for the various implementations.

Lastly, Chapter 6 will summarise all the conclusions gathered from previous chapters and from this work as a whole, mentioning its contributions to the field of temporal OD, risks and limitation when developing this work and which will be the next steps to take.

Chapter 2

State of the Art

SOTA scientific publications refer to the highest level of knowledge or general development achieved in a technique or scientific field at a particular time.

In this chapter, relevant SOTA documentation in the area of OD in AD will be presented and analysed. Although the main field of this dissertation is temporal OD, not only temporal focused deep learning architectures were analysed, but also single-frame, tracking, 2D and 3D methodologies, LiDAR sensor information, camera sensor data and fusion (LiDAR and camera) focused. All metrics used in the models results are explained in chapter 4.

The study of other domains will provide further knowledge regarding major techniques that can be adapted into a temporal context, considering that temporal models are still behind single-frame models in the main OD benchmarks.

2.1 An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds

As one of the pioneers in temporal 3D OD in AD using LiDAR, on July 24th 2020, Huang published a paper [20] providing a solution for 3D OD, through the introduction of an Long Short-Term Memory (LSTM) network. One of the first facts that this paper mentions is the lack of sequential data in public datasets, e.g. the KITTI [13] 3D OD challenge, which analyses one 3D point cloud based on LiDAR independently. However, despite some temporal datasets being released (e.g., nuScenes [6] and *Waymo* [45]), single-frame object detectors were still used to process those sequences.

2.1.1 Pipeline

Their method implements an U-Net style sparse 3D convolution backbone which receives, as input, a sequence of LiDAR point clouds. The framework structure is presented in Figure 9, complemented with a detailed version of the LSTM block.

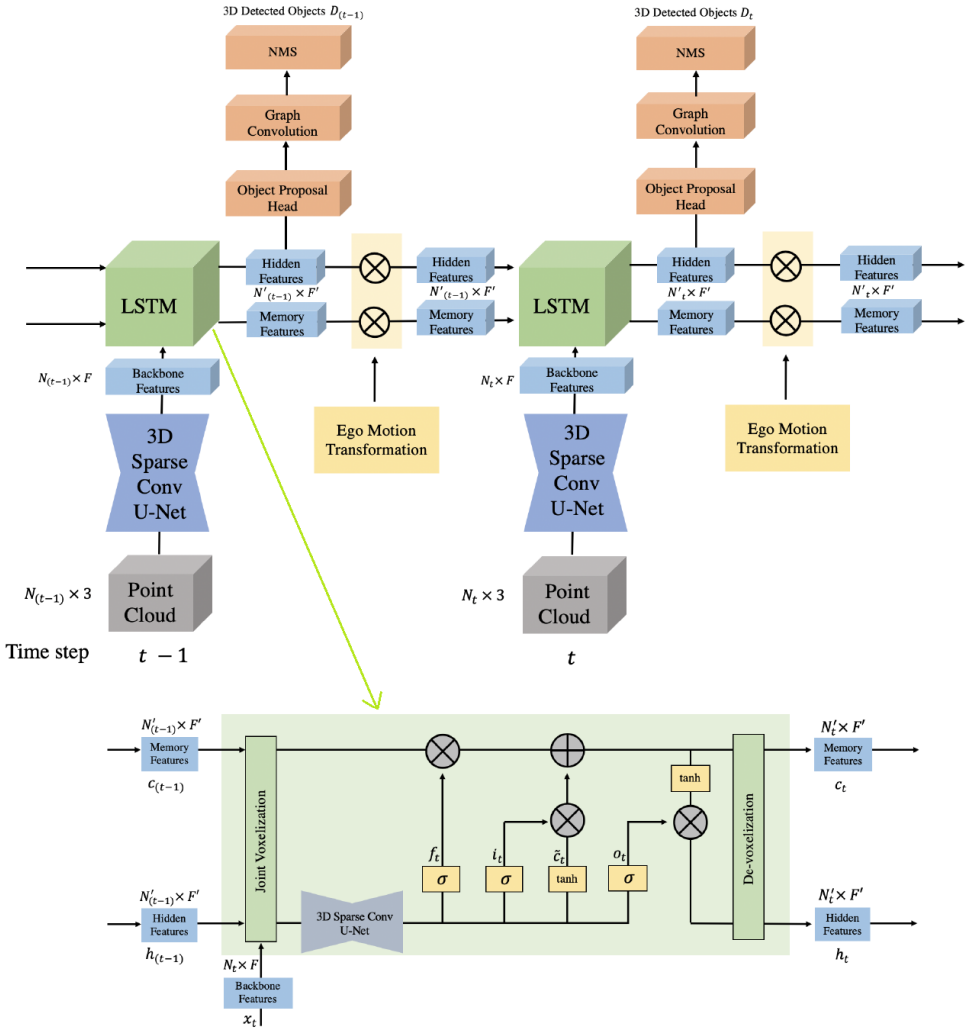


Figure 9: LSTM pipeline, presented in [20].

At each time step, a 3D sparse voxel convolution U-Net extracts point features from the input point cloud, becoming classified as backbone features. The extracted features together with the memory and hidden features from previous frames are fed to the 3D sparse convolutional LSTM, which processes them and returns the hidden and memory features that will be provided to the next frame.

An OD head is applied to the hidden features to produce 3D object proposals, having centre, size, rotation and confidence per detection in each frame. The proposals go through a graph convolution stage and, finally, a Non-Maximum Suppression (NMS) is applied, removing repeated detections by measuring the overlap between bounding boxes as a way to select the one with the highest confidence score for a given object, similar to the 2D RGB image example illustrated in Figure 10.

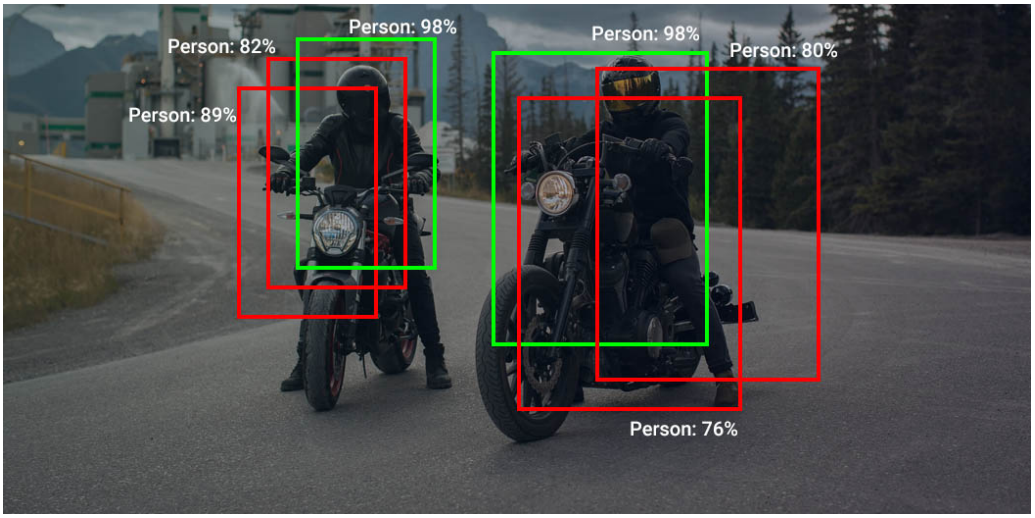


Figure 10: 2D NMS example, extracted from [41].

One interesting operation this method implements is joint voxelization: following ego-motion[1] compensation, this operation deals with other objects motion. It voxelizes the previous three point clouds, producing three voxel grids which are then aggregated in feature dimension.

2.1.2 Results

Table 1 demonstrates the mean Average Precision with 0.7 IoU threshold (further explained in Section 4.4) after training and validating the proposed architecture with *Waymo* [45] dataset, on the class "vehicle". Specifically in this benchmark, it was explored the performance of the learning model when trained and validated using different numbers of points cloud on the input sequence. As we can see in the table results, showing the results tested on *Waymo* [45] open validation set on the class "vehicle". Using two frames instead of one, increases the mAP@0.7IoU (explained in Chapter 4) by 1%, and when doubling the value again to four frames it achieves an increase of 3.9%. At the end, it is possible to notice a major increase, from the initial value using 1 frame to 4 frames, from 58.7% to 63.6% which is quite significant.

	mAP@0.7IoU (Vehicle)
1 frame	58.7
2 frame	59.7
4 frame	**63.6**

Table 1: LSTM frames number benchmark comparison on *Waymo* [45] open validation set (for class vehicle). Values produced by [20].

A comparison between different models can be observed in Table 2. The LSTM model had the best results on the *Waymo* [45] open dataset validation set, even when comparing with the single-frame model

[1] motion of the vehicle capturing the information

MVF [62], which was in first place in the benchmark. The LSTM approach reached a value of mAP@0.7IoU on the class "vehicle" of 63.6%, while MVF [62] presented 62.9%, and also beats a multi-frame concatenation baseline model, which reached 62.4%.

Model	mAP@0.7IoU (Vehicle)
StarNet[35]	53.7
PointPillars[24]	57.2
MVF[62]	62.9
U-Net	56.1
U-Net + Kalman Filter[54]	56.8
Concatenation (4 frames)	62.4
LSTM (4 frames)	**63.6**

Table 2: Results on 3D OD on *Waymo* [45] open dataset validation set (class "vehicle" on level 1 difficulty). Models are single-frame, unless mentioned otherwise. Values extracted from [20].

2.2 3D-MAN: 3D Multi-frame Attention Network for Object Detection

3D-MAN [58] presents a temporal ODLiDAR only solution to *Waymo* [45] open dataset, achieving SOTA results when compared to either single or temporal methods at the time.

This paper also discusses the lack of temporal datasets with well-calibrated multi-frame data, with the *Waymo* [45] and nuScenes [6] datasets being the only ones to provide large-scale 3D sequence data.

2.2.1 Pipeline

The pipeline of 3D-MAN is represented in Figure 11, and can be divided in 3 sections:

- **Fast single-frame detector**

  Firstly, bounding boxes are proposed using a novel fast single-frame detector. Normally, NMS is used to remove overlapping predictions to the same object. NMS finds the highest scoring prediction and suppresses every overlapping prediction, repeating this operation until all predictions are dealt with. As all the predictions in the frame need to be compared, this process doesn't scale well with the increase of predictions. To mitigate this problem, this method implements a MaxPoolNMS.

  MaxPoolNMS is a variant of NMS, where before the process of NMS, it executes max pooling to find local score peaks. Local peaks are then kept as predictions and other locations are ignored. This NMS variant operation use low computation resources and has highly parallelism potential.

Another interesting technique used in this section is called hungarian matching, which tries to force the model to predict only one bounding box per object. From a set of predictions and a set of ground-truths, pair-wise scores are calculated, obtaining a single match for each pair maximising the overall match score.

- **Memory bank**

  Afterwards, a memory bank stores the predicted bounding boxes and their respective feature maps that comes from the fast single-frame detector. Through this bank, it helps to increase the model recall by preventing missed objects[2] by the detector in one frame, from not being processed.

- **Multi-view alignment and aggregation**

  Every time a new frame arrives to the memory bank, the newly received data and all previous stored data goes through a multi-view alignment and aggregation process, where attention networks are used to extract and aggregate their temporal features. This results in combining features from multiple views across time in the scene.

  A cross-view loss is used to encourage the model to extract features with sufficient information about the objects, with the help of classification and regression heads.

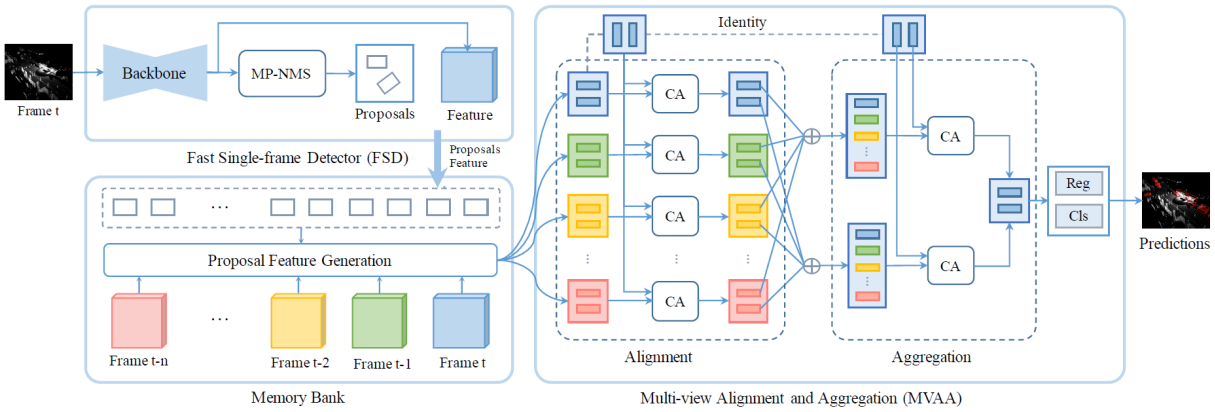


Figure 11: 3D-MAN pipeline, proposed in [58].

2.2.2 Results

As seen in Table 3 and Table 4, this paper compares results using multiple number of sequential frames to predict bounding boxes from different object velocities.

In Table 3, it tests PointPillars [24], a multi-frame concatenation model. The use of 8 frames has benefits in slow-moving targets, and lower results than the use of 4 frames in medium and fast moving targets. Despite inference time not being mentioned, it is expected that an increase of frames will deteriorate processing speed.

[2] due to those objects being partial or fully occluded.

Model	Stationary AP (%)	Slow AP (%)	Medium AP (%)	Fast AP (%)
1 frame	60.01	66.64	65.02	71.90
4 frame	62.40	67.39	**66.68**	**77.99**
8 frame	**63.71**	**67.98**	66.29	72.30

Table 3: Vehicle velocity breakdowns through different number of frames using PointPillars [24]. Values extracted from [58].

Table 4 shows 3D-MAN architecture results.. The use of 4 frames has a significant increase in AP in every object speed when compared with PointPillars [24]. With 16 frames it provides even better results, but once again, inference time are not mentioned.

Model	Stationary AP (%)	Slow AP (%)	Medium AP (%)	Fast AP (%)
4 frame	69.5	68.6	67.1	78.3
16 frame	**73.2**	**70.4**	**68.9**	**79.2**

Table 4: Vehicle velocity breakdowns through different number of frames using 3D-MAN. Adapted from [58].

A comparison between different multi-frame fusion approaches is present in Table 5. The lowest results come from the baseline, followed by the concatenation technique with some significant improvements in performance. Nevertheless, the usage of the multi-view alignment and aggregation (MVAA) module brings best results when comparing with all the other techniques, increasing 2% when comparing with concatenation.

	Baseline	Concatenation	MVAA
AP (%)	68.2	70.5	**72.5**

Table 5: Validation AP comparison among different multi-frame fusion approaches on class “vehicle”. Adapted from [58].

Comparison results against other main algorithms were also included for the *Waymo* [45] open dataset, on the class “vehicle”, presented in Table 6.

Table 6 presents two levels (LEVEL_1 and LEVEL_2), which represent the levels of object detection difficulty on the *Waymo* [45] open dataset. Level 1 identifies 3D bounding box labels with at least 5 LiDAR points, and Level 2 includes every bounding box label with 1 or more LiDAR points.

At the time, 3D-MAN outperformed all the single-frame and t models on the “vehicle” class, presenting an increase of 4% regarding precision on Level 1 and over 2% on Level 2.

Difficulty	Method	3D AP (IoU=0.7)			
		Overall	0-30m	30-50m	50m-Inf
LEVEL_1	StarNet	55.11	80.48	48.61	27.74
	PointPillars	63.27	84.90	59.18	35.79
	MVF	62.93	86.30	60.02	36.02
	AFDet	63.69	87.38	62.19	29.27
	RCD	68.95	87.22	66.53	44.53
	PV-RCNN	70.30	91.92	69.21	42.17
	3D-MAN	69.03	87.99	66.55	43.15
	PointPillars*	65.41	85.58	61.51	39.51
	ConvLSTM*	63.6	-	-	-
	3D-MAN*	**74.53**	**92.19**	**72.77**	**51.66**
LEVEL_2	StarNet	48.69	79.67	43.57	20.53
	PointPillars	55.18	83.61	53.01	26.73
	PV-RCNN	65.36	91.58	65.13	36.46
	3D-MAN	60.16	87.10	59.27	32.69
	PointPillars*	57.28	84.31	55.41	29.71
	3D-MAN*	**67.61**	**92.00**	**67.20**	**41.38**

Table 6: 3D Average Precision (AP) results on *Waymo* [45] open dataset validation set for class "vehicle". *Methods utilise multi-frame point clouds for detection. Acquired from [58].

2.3 Tracklet Proposal Network for Multi-Object Tracking on Point Clouds

This paper presents the first tracklets proposal network, named PC-TCNN, which reached first place on the KITTI [13] tracking benchmark, at the time. It does not focus on temporal OD, but on object tracking, and uses not only LiDAR, but also IMU/GPS data. Object tracking goes one step further than OD by storing objects in memory and associating them with IDs in order to track their attributes, such as velocity. It mentions that tracking methods normally use a tracking-by-detection paradigm that links detected objects across a sequence of frames, having their performance limited by detection failures.

Similarly to other temporal SOTA solutions, this paper justifies the use of information from previous frames on the fact that an object shares consistent spatial-temporal features across a sequence of frames, i.e., in rigid objects (e.g., vehicles), the width, length and height are usually very similar in a sequence of frames, and even on non-rigid objects (e.g. pedestrians), the volume, size and geometry are consistent in consecutive frames.

2.3.1 Pipeline

A novel tracklet proposal CNN to deal with point clouds was developed for multi object tracking, named PC-TCNN, having the following pipeline (Figure 12):

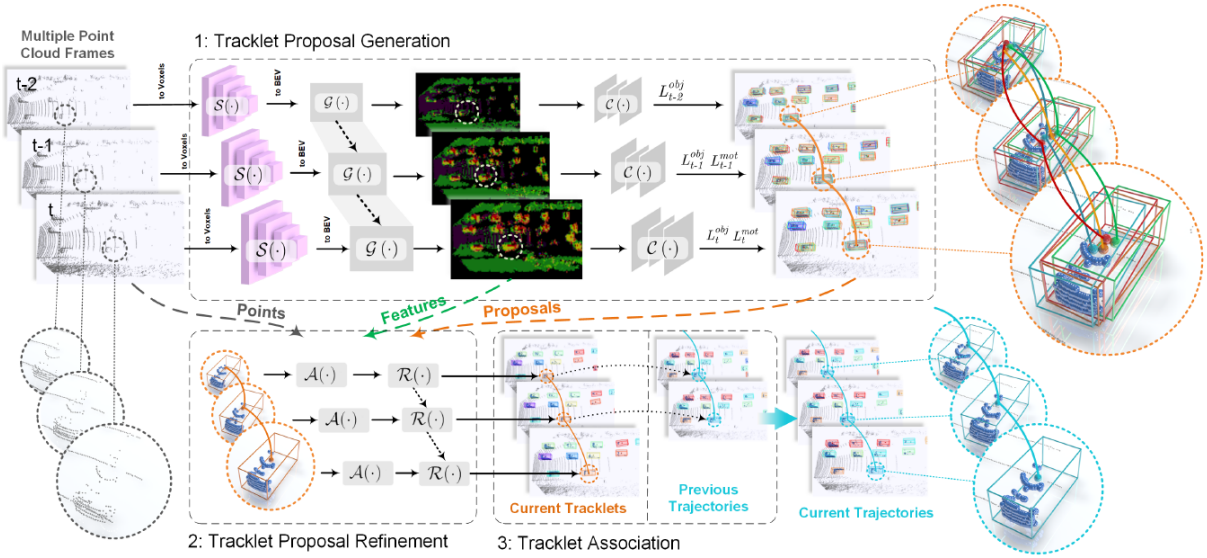


Figure 12: PC-TCNN based pipeline, proposed in [55].

As depicted in Figure 12, the pipeline has three main segments, each one being responsible for a specific task:

- **Tracklet proposal generation**

  This segment's role is to search point clouds to locate spatial-temporal regions of interest, denoted as tracklet candidates or proposals.

  With each arrival of a new frame, this module uses the last frames along with the new one as input, and transforms the point clouds into multiple Bird Eye View (or BEV, further explained in Section 4.2) feature maps. Lastly, a set of tracklet candidates are generated with a 3D object proposal generation and motion regression on the feature maps.

  Similarly to ConvLSTM [20], this method uses 3D sparse convolution to efficiently convert a sequence of point clouds into 3D feature volumes. After processing the frames to BEV feature maps, this method uses a ConvGRU [46] for temporal features modulation, as this model proved to be quite effective in shaping temporal features in 2D data.

  A tracklet proposal head processes the sequence of BEV feature maps and at each pixel, it's employed an object proposal head to calculate the object confidence and residuals, and a motion head to determine a cross frame offset of the objects motion.

- **Tracklet proposal refinement**

  A tracklet proposal refinement module is implemented to improve the quality of detected objects. A major obstacle in the refinement stage is the ability to efficiently extract features from an area of interest. Previous methods used techniques aimed at single-frame, however, they are very difficult to adapt to point clouds sequences.

A tracklet feature aggregation method was created to obtain features from regions of interest in a sequence of point clouds. It applies a context-aware point cloud pooling [43] to analyse points of each tracklet prediction.

A refinement head method will apply during training a confidence loss formulated as:

$$min(1; max(0; 2 * IoU - 0.5)$$

With Intersection of Union (IoU, further explained in Section 4.4) as the only variable. This formula is used to minimise the cross-entropy loss between confidence predictions and ground-truth objects. During inference, NMS is used on the last frame to acquire refined tracklets with more accurate object sizes, motions and correspondences.

- **Tracklet association**

  The tracking of objects is initialised with an empty trajectory set. Subsequently, with each new frame, previous trajectories of refined tracklets are associated using a greedy matching algorithm based on 3D IoU. Successful tracklet associations will have their trajectory updated, while unsuccessful ones will have their trajectories initialised. Missing objects can be recovered if they emerge across the next sequence of frames.

In terms of data augmentation, to avoid overfitting, it was applied random flipping along the X axis, global scaling and random global rotations around the Z axis. One interesting data augmentation process is the use of ground-truths tracklets from other scenes, however the paper does not disclose any details about this practice.

2.3.2 Results

The results presented in this paper are focused on object tracking. To simplify the visualisation, some tables include values not relevant to OD, so these were removed.

Frames	2D AP@t (%)	3D AP@t (%)	ID Switches	Speed (ms)
1 frame	91.25	91.16	48	**75**
2 frame	92.78	93.12	18	158
3 frame	93.59	94.18	7	240
4 frame	**94.03**	94.48	**2**	312
5 frame	93.51	**94.49**	3	388

Table 7: Results on the KITTI [13] validation set using different number of frames per temporal sequence, adapted from [55].

Table 7 studies various results dependent on the number of frames per sequence being processed. Note that there are 2D and 3D precision values, given that the study was carried out in the KITTI [13] validation set. AP@t refers to the average precision (IoU=0.5) of tracklet detections in the last frame. ID

switches regard to not detecting an object during tracking, resulting in changing their ID with a new one. Speed refers to the inference speed of the model. The overall best result is achieved when using 4 frames. It only loses to 5 frames when comparing 3D AP@t, but the value differs by only 0.01%, not being worth the 25% increase in inference time. The only amount of sequential frames capable of real-time inference (100ms) is the use of only 1 frame per sequence. Even with pipeline modifications and the removal of some tracking aspects, the use of 4 frames could hardly be viable for real-time temporal 3D OD.

Despite ID Switches being a multi-object tracking metric, it is quite helpful in showing how well the algorithm can detect and follow an object without losing it and changing its ID. Consequently, the less the number of ID switches the better. As such, the use of 4 frames was also the best approach in terms of ID switches, presenting only 2.

Temporal algorithms present advantages when dealing with occluded objects, and Figure 13 shows an example of an object being tracked for 340 meters and being occluded by another object. In frame (b) and (c) it's possible to observe the cyan 3D bounding box, corresponding to the object with ID 2, being fully occluded by the object with ID 101 (with the orange 3D bounding box). Despite the occlusion, the algorithm was able to always keep track of the occluded object, not switching its ID.

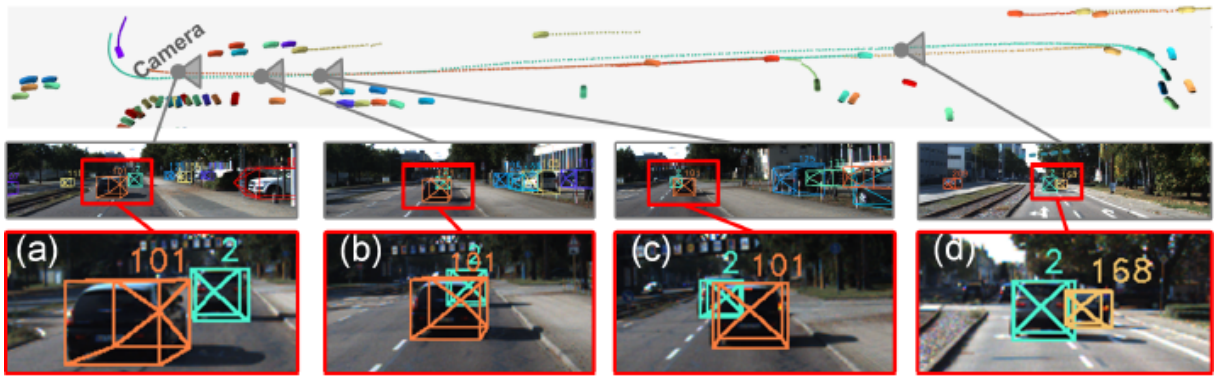


Figure 13: A long-time 340 meters tracking example with an heavily occluded object (cyan bounding box with ID 2), produced by [55].

Comparisons made by the paper do not use any metrics directly relevant to the detection of objects, but can be approximated by the Multiple Object Tracking Accuracy (MOTA) results.

$$MOTA = 1 - \frac{|FN| + |FP| + |IDS|}{|GTDet|}$$

MOTA uses detection ground-truth (GTDet) as the base of its formula, and also utilising ID switches (IDS) in its calculation, resulting in a measurement of the overall accuracy of the detector and tracker.

As shown in Table 8, PC-TCNN has a significant 8% increase in MOTA value over the second best 3D point clouds tracking algorithm, indicating that it is a robust object detection and tracking implementation.

Method	Input	MOTA	MOTP	Recall	Precision	MT	ML	IDS	FRAG
mono3DT	2D	84.52%	85.64%	88.81%	97.95%	73.38%	2.77%	377	847
TuSimple	2D	86.62%	83.97%	90.50%	97.99%	72.46%	6.77%	293	501
CenterTrack	2D	89.44%	85.05%	93.20%	97.73%	82.31%	**2.31%**	116	334
ODESA	2D	90.03%	84.32%	92.62%	**98.77%**	82.62%	**2.31%**	90	501
GNN3DMOT	2D+3D	82.24%	84.05%	-	-	64.92%	6.00%	142	416
aUToTrack	2D+3D	82.25%	80.52%	89.36%	97.03%	56.77%	7.38%	1025	1402
mmMOT	2D+3D	84.77%	85.21%	88.81%	97.93%	73.23%	2.77%	284	753
JRMOT	2D+3D	85.70%	85.48%	89.51%	97.81%	71.85%	4.00%	98	372
PointTrackNet	3D	68.24%	76.57%	83.56%	88.10%	60.62%	12.31%	111	725
ComplexerYOLO	3D	75.70%	78.46%	85.32%	95.18%	58.00%	5.08%	1186	2092
AB3DMOT	3D	83.84%	85.24%	88.32%	96.98%	66.92%	11.38%	**9**	224
PC-TCNN	3D	**91.75%**	**86.17%**	**96.08%**	96.45%	**87.54%**	2.92%	26	**118**

Table 8: Car tracking results on the test set of the KITTI [13] tracking benchmark, extracted from [55].

2.4 Sparse Fuse Dense: Towards High Quality 3D Detection With Depth Completion

Sparse Fuse Dense (SFD) [56] is a single-frame method which ranked first on the KITTI [13] 3D OD benchmark, and presents an innovative multi-modal framework. Multi-modal refers to the usage of multiple types of data, in this case, meaning the usage of not only LiDAR data, but also colour images (RGB).

This paper mentions the problem of current LiDAR-only 3D OD methods suffering from the sparsity of point clouds data. Point cloud sparsity can lead to detection confusion, as it lacks geometric and semantic information. Some multi-modal methods attenuate this problem using different data representations. However, there are limitations when fusing sparse point clouds and other types of data, translating into a suboptimal performance.

Even with a larger amount of data and annotations, current multi-modal methods fall behind LiDAR-only methods. This can be explained by three main reasons:

- **Dimension gap**: There is an intrinsic dimension gap between images and point clouds, resulting in difficulties in merging 2D images and 3D point clouds information directly, ignoring links between 2D pixels and 3D points, leading to a drop in performance.

- **Information loss**: Methods which mitigate the dimension gap by establishing correspondences between sparse point clouds and images lead to image features being sparse, resulting in an increased loss of information.

- **Data augmentation**: Due to large differences between 2D images and 3D point clouds, they need to be operated differently. This complicates data augmentation transformations, e.g., rotations, crops, scaling and reshapes, which are essential to greatly increase the dataset size and generalisation ability of models.

SFD employs sparse point clouds derived from LiDAR and creates pseudo point clouds from depth completion. This technique estimates a dense depth image from given RGB images with the corresponding sparse depth measurements sparse reconstruction, in this case, obtained via LiDAR data.

Pseudo point clouds came to solve the three aforementioned problems related to multi-modal methods, since they behave the same way raw LiDAR point clouds do, and contain all the RGB image information.

A method which they called SynAugment was implemented to deal with their multi-modal data. Multi-modal data augmentation for sparse and pseudo point clouds combined, can use the same approaches as LiDAR-only data, contrary to what happens with other types of multi-modal methods.

2.4.1 Pipeline

The SFD pipeline consists in three main parts: the LiDAR stream, pseudo stream and sparse dense fusion head. Their structure, components and the interactions they have with each other are represented in Figure 14.

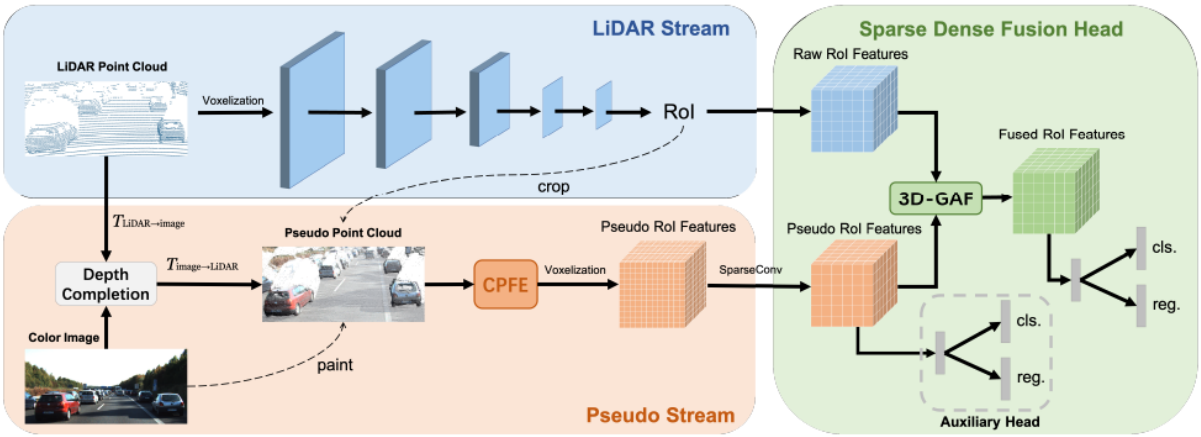


Figure 14: SFD pipeline, proposed in [56].

Each part of the pipeline has its specific role, which can be described by the following aspects:

- **LiDAR stream**

  Raw LiDAR point clouds are fed into the LiDAR Stream, which serves as an Region Proposal Network (RPN) passing the raw point clouds by a process of voxelization, in order to generate 3D Region of Interest (RoI). Then, raw point clouds and pseudo clouds (created by the pseudo stream by a process of depth completion) are cropped using the generated RoIs.

- **Pseudo stream**

  Raw point clouds are also given to the pseudo stream, along with RGB images, to produce pseudo point clouds through a process called depth completion. Pseudo clouds are painted with the respective RGB image values, creating colourful pseudo clouds. The RoI cropped pseudo cloud goes

through a colour point feature extractor to extract its information. Lastly, the pseudo point cloud gets voxelized, and its features are extracted when applying a 3D sparse convolution, serving as the only output of the segment.

- **Sparse dense fusion head**

  RoI features from raw point clouds (LiDAR stream output) and RoI features from pseudo point clouds (pseudo stream output) are fed to the Sparse Dense Fusion Head. These features are fused by a 3D-GAF module and used to predict bounding boxes and classes confidences. An auxiliary head to regularise the model is also employed that can be detached after training.

Data augmentation has a crucial role on a model's performance, so they developed a synchronised augmentation, which they called SynAugment, to augment both raw and pseudo point clouds in the same way. The transformations implemented regard data sampling (GT-sampling), data random flip (RandomFlip), rotation (RandomRotate) and scaling (RandomScale), as depicted in Figure 15.

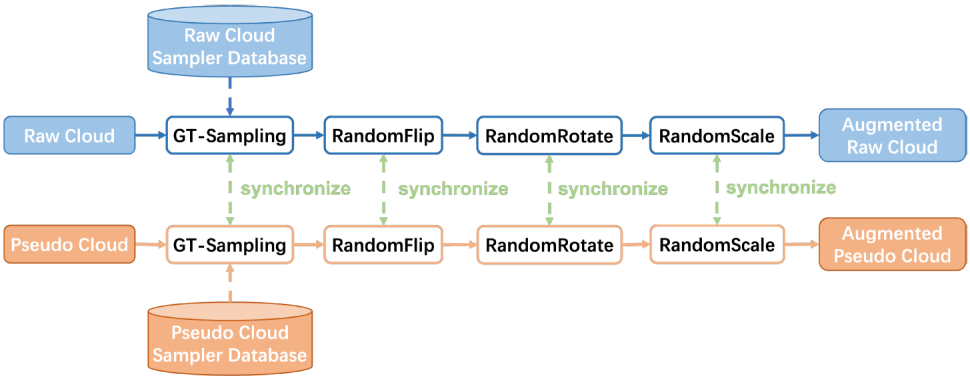


Figure 15: Illustration of SynAugment, presented in [56].

2.4.2 Results

Different tests and comparisons were done to demonstrate developed SOTA techniques results and to uncover what they bring to the 3D OD domain.

The first comparison comes in a form of various images (Figure 16), representing detections generated by 2 methods, SFD and Voxel-RCNN.

Figure 16 illustrates two different scenarios. In each square, the left column represents SFD's information, and the right column constitutes Voxel-RCNN's [10] data. SFD works with LiDAR and RGB information, while Voxel-RCNN [10] only handles LiDAR data, so SFD images contain coloured point cloud images against normal point clouds in the Voxel-RCNN [10].

Both scenes show situations where Voxel-RCNN [10] had false positive detections, that is, vehicle proposals where, in reality, it is not possible to find any. In the scene on the left, Voxel-RCNN [10] predicted

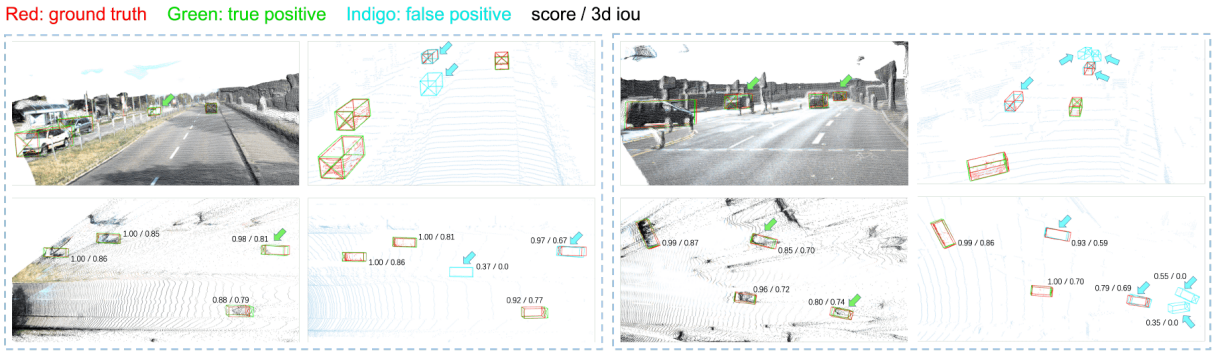


Figure 16: Comparing SFD and Voxel-RCNN [10] detections in two scenes (each square). In each square, left columns represent SFD detections, and the others represents Voxel-RCNN detections, illustrated by [56].

incorrectly a vehicle located relatively near to the sensor, and on the scene on the right two vehicles far from the sensor.

Comparisons were made between various models in different modalities, LiDAR-only and LiDAR together with RGB images (Fusion).

Method	Modality	AP_{3D}		
		Easy	Mod	Hard
SECOND	LiDAR	83.34	72.55	65.82
PointPillars	LiDAR	82.58	74.31	68.99
PointRCNN	LiDAR	86.96	75.64	70.70
Part-A^2	LiDAR	87.81	78.49	73.51
SA-SSD	LiDAR	88.75	79.79	74.16
STD	LiDAR	87.95	79.71	75.09
CIA-SSD	LiDAR	89.59	80.28	72.87
PV-RCNN	LiDAR	90.25	81.43	76.82
Voxel-RCNN	LiDAR	90.90	81.62	77.06
CT3D	LiDAR	87.83	81.77	77.16
SE-SSD	LiDAR	**91.49**	82.54	77.15
MV3D	LiDAR+RGB	74.97	63.63	54.00
ContFuse	LiDAR+RGB	83.68	68.78	61.67
F-PointNet	LiDAR+RGB	82.19	69.79	60.59
AVOD	LiDAR+RGB	83.07	71.76	65.73
PI-RCNN	LiDAR+RGB	84.37	74.82	70.03
UberATG-MMF	LiDAR+RGB	88.40	77.43	70.22
EPNet	LiDAR+RGB	89.81	79.28	74.59
3D-CVF	LiDAR+RGB	89.20	80.05	73.11
CLOCs PVCas	LiDAR+RGB	88.94	80.67	77.15
SFD	LiDAR+RGB	90.83	**83.96**	**77.47**

Figure 17: Comparison results with SOTA methods on the 3D KITTI [13] test set for car detection, extracted from [56].

As shown in Figure 17, SFD presents the best results when compared to other fusion models in every difficulty and overall. However, the LiDAR-only model SE-SSD [60] has the advantage in the easy AP difficulty. These values placed SFD first on the main category (moderate) on the KITTI [13] 3D OD benchmark.

2.5 SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Clouds

Self-Ensembling Single-Stage Object Detector (SE-SSD) [60] is a single-frame LiDAR-only method, which brought very accurate and efficient 3D detections to the KITTI 3D OD benchmark.

Using only LiDAR data is considered a robust approach as point clouds are less susceptible to variables like weather or illumination settings than some other types of data, e.g., camera sensor data.

SE-SSD reached incredible results without sacrificing on inference time. On the KITTI [13] 3D OD benchmark it managed an honourable first place, and achieved second place on the BEV leaderboard, both with a very high inference speed (around 30ms).

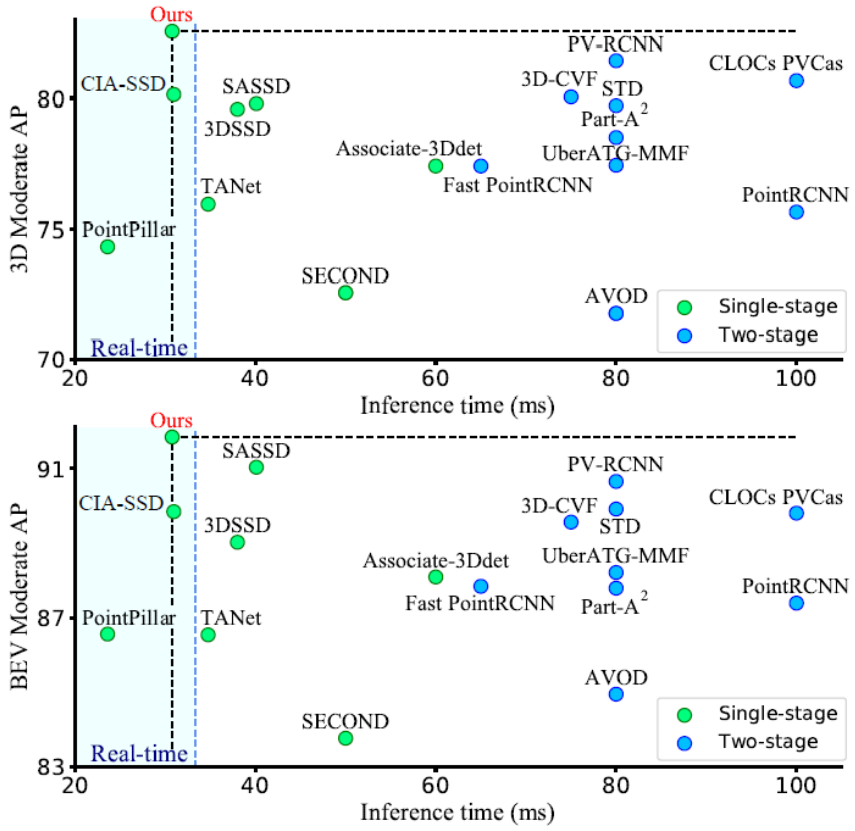


Figure 18: Inference time and AP between SOTA methods, on 3D and BEV benchmark. "Our" refers to the SE-SSD model. Presented in [60].

This method is a single-stage or single-shot detector (SSD), which means it generates bounding boxes and respective confidences from features extracted from the input data. There are also two-stage methods that require one extra step to refine the predictions made in the first one, with the help of region-proposal-aligned features.

Typically, two-stage detectors can get better predictions provided by the refinement from the last stage. Nevertheless, SSDs take advantage of their simpler network structures to have faster inference times.

SSD have been explored with the goal of getting fine predictions without the need of an extra step. As demonstrated in Figure 18, single-stage models have in fact lower inference times and start to have similar precisions as two-stage methods, where SE-SSD managed to surpass every single and two stage algorithms, getting a low inference time and high precision predictions.

2.5.1 Pipeline

The pipeline of the SE-SSD is supported by two SSDs. It is usually a design with fast speeds and satisfactory performance, as presented in Figure 18. As illustrated in Figure 19, one of the SSDs will be considered the teacher learning model (left) and the other will be considered the student learning model (right). The two SSDs have the same architecture and are initialised with pre-trained weights before they start training simultaneously. The teacher starts by receiving raw point clouds in order to predict somewhat precise bounding boxes along with respective confidences.

SSDs use a similar structure to [59] in order to encode point clouds efficiently. They include a sparse convolution network (SPConvNet), a BEV convolution network (BEVConvNet), and a multi-task brain (MT-Head). The point clouds are voxelized in order to get the mean 3D coordinates and point density per voxel as the starting feature, followed by the SPConvNet to the extraction of features. The sparse 3D feature along z is then processed into dense 2D features for extraction with BEVConvNet. Finally, the MTHead is utilised to compute the classification and regression on bounding boxes.

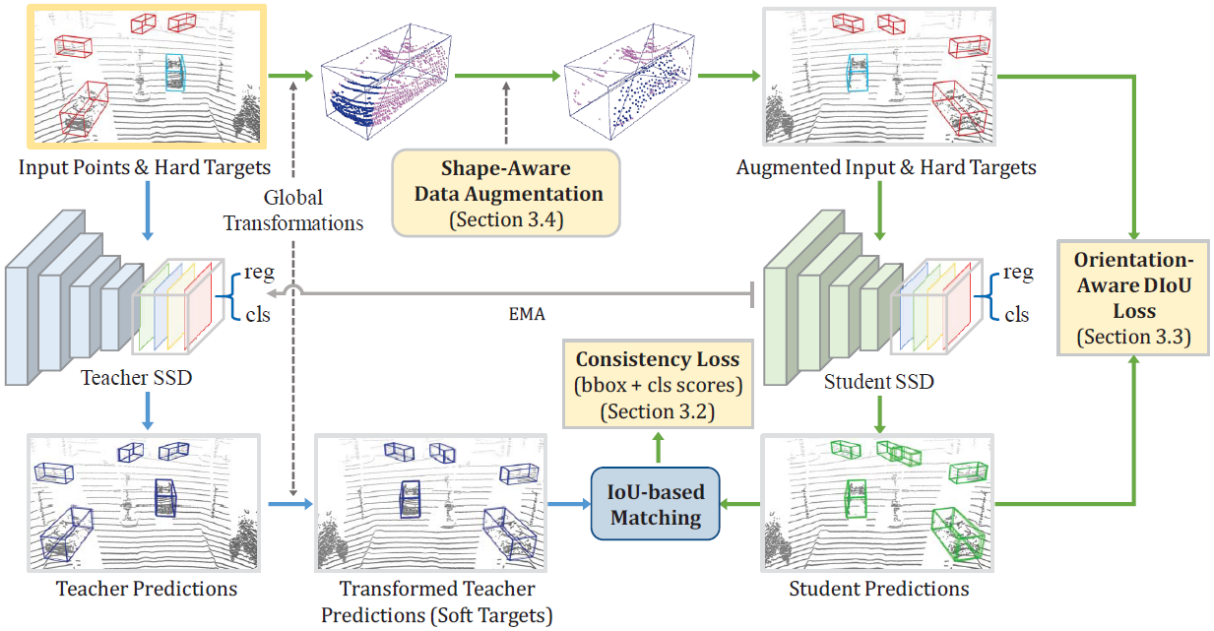


Figure 19: SE-SSD pipeline, proposed in [60].

On one of the paths (corresponding to the blue arrows in Figure 19), a set of global transformations are applied to the newly generated bounding boxes. These soft targets, combined with an implemented consistency loss, are used as supervision to the student SSD.

On the other path (corresponding to the green arrows in Figure 19), the raw input point clouds suffer the same global transformations as the previous path, where shape-aware data augmentation is also applied. The processed data is, subsequently, fed as input to the student SSD, to train, together with the consistency loss, to orient its predictions with teacher's generated soft targets. Using an exponential moving target strategy, the teacher gets its parameters updated based on the student parameters. Thus, this method can significantly improve the detectors' predictions, without spending extra computational power during inference to achieve it.

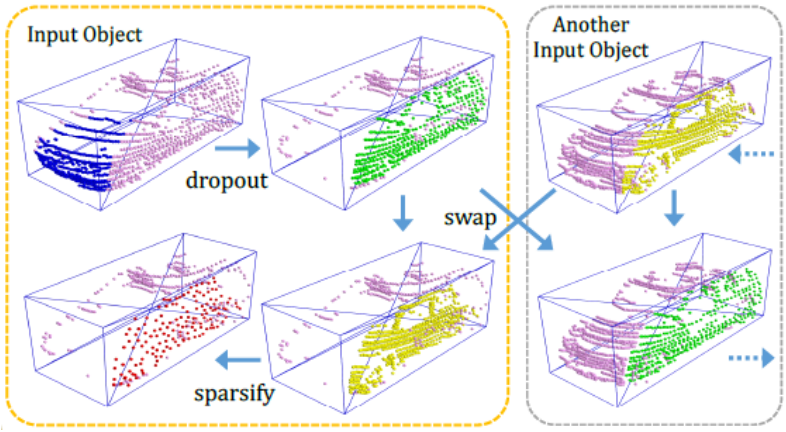


Figure 20: Shape-aware data augmentation, presented in [60].

Figure 20 illustrates various transformations that shape-aware data augmentation performs to the objects. Objects get their points divided into six subsets, each one corresponding to each face of the bounding box. The implemented operations are dropout, sparsify, and swapping bounding boxes faces with other objects.

An additional interesting technique is the orientation-aware distance-IoU loss, which was implemented with long distances predictions and occlusions of objects in mind. These situations make it difficult to acquire enough data from point clouds to correctly predict the dimensions and orientations of the bounding boxes. Finally, the orientation-aware distance-IoU loss constrains disparities in the orientation and alignment to the centre between ground-truth and predicted bounding boxes, in order to reduce the centre misalignment.

2.5.2 Results

There were three main techniques implemented by this paper [60], so a comparison between the use of these techniques is available in Table 9.

It is taken into consideration the precision in three levels of difficulty on the 3D KITTI [13] validation split for car detection, where Cons Loss, ODIoU and SA-DA refer to the implemented consistency loss, orientation-aware distance-IoU loss and shape-aware data augmentation, respectively. The KITTI [13] 3D OD benchmark (explained in Section 4.1) is well disputed, so every minor increase in predictions accuracy will make a difference. When comparing the "moderate" difficulty (main difficulty on the benchmark), the

Cons loss	ODIoU	SA-DA	Easy	Moderate	Hard
-	-	-	92.58	83.22	80.15
-	-	✓	93.02	83.70	80.68
-	✓	-	93.07	83.85	80.78
✓	-	-	93.13	84.15	81.17
✓	✓	-	93.17	85.81	83.01
✓	✓	✓	**93.19**	**86.12**	**83.31**

Table 9: Comparisons between different implemented techniques on the 3D KITTI [13] validation split for car detection, produced by [60].

use of these techniques brings a major increase of 2.9 percentile points, from 83.22% to 86.12%, which made the implementation of every module crucial in establishing and maintaining top performance on the KITTI [13] benchmarks, with incredibly fast speeds.

A comparison between all the other SOTAs methodologies was also made, in order to show how well SE-SSD performs. Table 10 compares all previous published methods, and confirms SE-SSD outperforms every single one of them in all difficulty levels, by an average of 2.8% in 3D detection and by an average of 0.8% in BEV detection[3]. Furthermore, it also ranked second in terms of inference time.

	Method	Reference	Modality	*3D*				*BEV*				Time (ms)
				Easy	Mod	Hard	mAP	Easy	Mod	Hard	mAP	
Two-stage	MV3D	CVPR 2017	LiDAR+RGB	74.97	63.63	54.00	64.2	86.62	78.93	69.80	78.45	360
	F-PointNet	CVPR 2018	LiDAR+RGB	82.19	69.79	60.59	70.86	91.17	84.67	74.77	83.54	170
	AVOD	IROS 2018	LiDAR+RGB	83.07	71.76	65.73	73.52	89.75	84.95	78.32	84.34	100
	PointRCNN	CVPR 2019	LiDAR	86.96	75.64	70.70	77.77	92.13	87.39	82.72	87.41	100
	F-ConvNet	IROS 2019	LiDAR+RGB	87.36	76.39	66.69	76.81	91.51	85.84	76.11	84.49	470*
	3D IoU Loss	3DV 2019	LiDAR	86.16	76.50	71.39	78.02	91.36	86.22	81.20	86.26	80*
	Fast PointRCNN	ICCV 2019	LiDAR	85.29	77.40	70.24	77.64	90.87	87.84	80.52	86.41	65
	UberATG-MMF	CVPR 2019	LiDAR+RGB	88.40	77.43	70.22	78.68	93.67	88.21	81.99	87.96	80
	Part-A^2	TPAMI 2020	LiDAR	87.81	78.49	73.51	79.94	91.70	87.79	84.61	88.03	80
	STD	ICCV 2019	LiDAR	87.95	79.71	75.09	80.92	94.74	89.19	86.42	90.12	80
	3D-CVF	ECCV 2020	LiDAR+RGB	89.20	80.05	73.11	80.79	93.52	89.56	82.45	88.51	75
	CLOCs PVCas	IROS 2020	LiDAR+RGB	88.94	80.67	77.15	82.25	93.05	89.80	86.57	89.81	100*
	PV-RCNN	CVPR 2020	LiDAR	90.25	81.43	76.82	82.83	94.98	90.65	86.14	90.59	80*
	De-PV-RCNN	ECCVW 2020	LiDAR	88.25	81.46	76.96	82.22	92.42	90.13	85.93	89.49	80*
One-stage	VoxelNet	CVPR 2018	LiDAR	77.82	64.17	57.51	66.5	87.95	78.39	71.29	79.21	220
	ContFuse	ECCV 2018	LiDAR+RGB	83.68	68.78	61.67	71.38	94.07	85.35	75.88	85.1	60
	SECOND	Sensors 2018	LiDAR	83.34	72.55	65.82	73.9	89.39	83.77	78.59	83.92	50
	PointPillars	CVPR 2019	LiDAR	82.58	74.31	68.99	75.29	90.07	86.56	82.81	86.48	**23.6**
	TANet	AAAI 2020	LiDAR	84.39	75.94	68.82	76.38	91.58	86.54	81.19	86.44	34.75
	Associate-3Ddet	CVPR 2020	LiDAR	85.99	77.40	70.53	77.97	91.40	88.09	82.96	87.48	60
	HotSpotNet	ECCV 2020	LiDAR	87.60	78.31	73.34	79.75	94.06	88.09	83.24	88.46	40*
	Point-GNN	CVPR 2020	LiDAR	88.33	79.47	72.29	80.03	93.11	89.17	83.90	88.73	643
	3DSSD	CVPR 2020	LiDAR	88.36	79.57	74.55	80.83	92.66	89.02	85.86	89.18	38
	SA-SSD	CVPR 2020	LiDAR	88.75	79.79	74.16	80.90	95.03	91.03	85.96	90.67	40.1
	CIA-SSD	AAAI 2021	LiDAR	89.59	80.28	72.87	80.91	93.74	89.84	82.39	88.66	30.76
	SE-SSD	-	LiDAR	**91.49**	**82.54**	**77.15**	**83.73**	**95.68**	**91.84**	**86.72**	**91.41**	30.56

Table 10: SE-SSD comparisons with SOTA on the 3D and BEV KITTI [13] benchmarks, extracted from [60].

2.6 SECOND: Sparsely Embedded Convolutional Detection

Sparsely Embedded Convolutional Detection (SECOND) [57] is a single-frame approach which tries to solve slow inference times and low orientation estimation performance from previous works. They

[3]The first model in the BEV benchmark wasn't a published work, and has been removed since then.

explore an improved sparse convolution method for better speeds on training and inference. To improve the orientation estimation performance they also introduce a new angle loss regression. In addition, they implemented new data augmentation techniques for a faster model convergence and better performance.

SECOND is based on an approach called VoxelNet [61]. VoxelNet [61] uses a single-stage end-to-end network to execute raw point cloud feature extraction and voxel-based feature extraction. Firstly, the cloud points are grouped into voxels. Then, linear models process each voxel individually converting them into dense 3D tensors to be fed into a RPN. At the time, this was the SOTA approach, even though its speed being too slow to run in real-time. SECOND addresses these challenges by taking advantage of further rich 3D data available in the point cloud. Spatially sparse convolutional networks are used to extract z-axis information before the point cloud being downsampled to something similar to 2D image information. A GPU-based rule generation algorithm for sparse convolution was also implemented which was able to increase speed by a factor-of-4 during training and a factor-of-3 during inference on the KITTI [13] OD dataset. Additionally, a smaller model was also implemented that achieved even better speeds at a small cost in performance, reaching an inference time of 25ms on a GTX 1080 Ti GPU.

At the time, SECOND was able to achieve SOTA results on the KITTI [13] 3D object detection benchmark, while having an inference time of 50ms for the larger model and 25ms for the smaller one.

2.6.1 Pipeline

The SECOND pipeline closely resembles the VoxelNet [61] pipeline, and subsists of various components as illustrated in Figure 21.

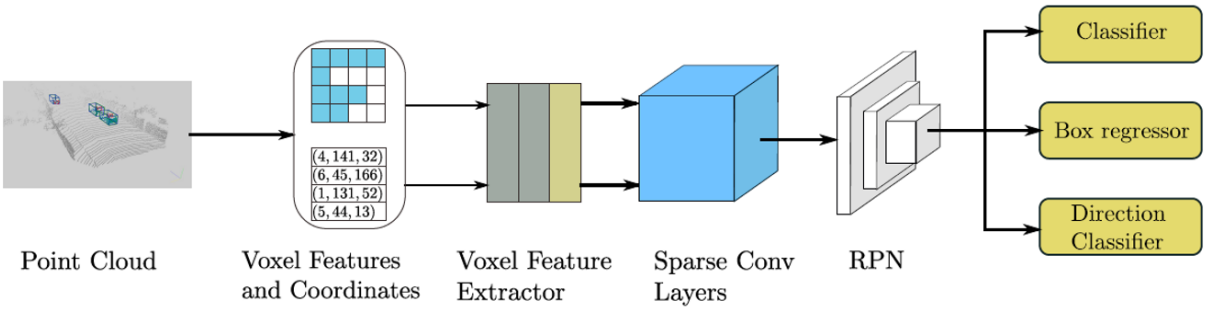


Figure 21: SECOND pipeline, produced by [57].

After receiving a raw point cloud as input, the model starts by converting it to voxel features and coordinates, and, then, employs two voxel feature encoding layers, a linear layer and ends with a sparse CNN. Lastly, detections are generated by a RPN. Each phase can be described as follow:

- **Point Cloud Pre-processing**

  From the point clouds, the respective voxel representation is obtained, as employed by [61]. The points from the cloud are iterated, assigning them to an existing voxel or creating a new one. This iterative process adds voxels into a hash table and ends when the number of voxels reaches a previously stipulated limit. Voxels dimensions (x, y and z) are defined in advance.

- **Voxelwise Feature Extractor**

  A Voxel Feature Enconding (VFE) layer is used to extract voxelwise features from the generated voxels. The VFE layer employs a Fully Connected Network (FCN) comprised of a linear layer, a batch normalisation (BatchNorm) layer, and a Rectified Linear Unit (ReLU) layer to extract pointwise information from all the points in a single voxel. In order to extract the locally aggregated features for each voxel, elementwise max pooling is used. The collected features are then tiled and combined with the pointwise features in the final step. The voxelwise feature extractor is made up of an FCN layer, along with a number of VFE layers. All together, a single FCN layer and multiple VFE layers constitute the voxelwise feature extractor.

- **Sparse Convolutional Middle Extractor**

  The sparse convolutional middle extractor purpose is to extract the spatial features present in the voxel features. Spatially sparse convolutions were first introduced in [17], which provides computational advantages for detection using LiDAR information. The submanifold sparse convolution [18] came as an alternative to the normal sparse convolution. It confines the active state of an output location to that of the matching input location alone. By doing this, it prevents too many active locations generated, which could slow down the speed of following convolution layers due to the high number of active points.

- **Region Proposal Network**

  When this method was developed, numerous detection frameworks had started to use RPNs [15]. SECOND builds an RPN using a single shot multibox detector [28] architecture. The feature maps from the sparse convolutional middle extractor are the input to the RPN. This RPN architecture consists of three distinct phases. Each stage begins with a downsampled convolutional layer, followed by other convolutional layers. Each convolutional layer is followed by BatchNorm and ReLU layers. The output of each step is then upsampled to a feature map of the same size, and these feature maps are concatenated into a single feature map. Three last 1 X 1 convolutions are performed to predict the class, regression offsets, and direction.

Some interesting data augmentation techniques are also used:

- Sampling/concatenating ground-truth objects from other scenes to the one being processed, to maximise the existence of object bounding boxes in a scene for a model to learn;
- Adding noise to objects in the scene;
- Rotating the whole point cloud;
- Scaling the whole point cloud.

2.6.2 Results

SECOND not only outperformed the VoxelNet [61] model on which it was based, but also achieved SOTA results on the KITTI [13] 3D benchmark (explained in Section 4.1). This solution achieved the best result to date on the moderate difficulty level of the "car" class, which are the main level and class on the KITTI [13] benchmark, as shown in Table 11. Despite F-PointNet [38] getting better results for classes "pedestrian" and "cyclist" than, its 2D detector was fine-tuned using ImageNet [9] weights, while SECOND uses only LiDAR data and is trained from scratch.

Method	Time (s)	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D	0.36	71.09	62.35	55.12	N/A	N/A	N/A	N/A	N/A	N/A
MV3D (LiDAR)	0.24	66.77	52.73	51.31	N/A	N/A	N/A	N/A	N/A	N/A
F-PointNet	0.17	81.20	70.39	62.19	**51.21**	**44.89**	**40.23**	**71.96**	**56.77**	**50.39**
AVOD	0.08	73.59	65.78	58.38	38.28	31.51	26.98	60.11	44.90	38.80
AVOD-FPN	0.1	81.94	71.88	**66.38**	46.35	39.00	36.58	59.97	46.12	42.36
VoxelNet (LiDAR)	0.23	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
SECOND	**0.05**	**83.13**	**73.66**	66.20	51.07	42.56	37.29	70.51	53.85	46.90

Table 11: SECOND results on 3D KITTI [13] benchmark, presented in [57].

Since the computational complexity of a 3D CNN grows cubically with the voxel resolution, a smaller SECOND was also developed. Tables 12 and 13 show a new evaluation on the KITTI [13] 3D and BEV (explained in Section 4.2) validation set, respectively, demonstrating this smaller model maintaining better results than the others and with a faster speed by a factor of two. The smaller model with an average decline of about 0.4 percent when comparing with the large one, the results for BEV did not exhibit a substantial decline.

Method	Time (s)	Easy	Moderate	Hard
MV3D	0.36	71.29	62.68	56.56
F-PointNet	0.17	83.76	70.92	63.65
AVOD-FPN	0.1	84.41	74.44	68.65
VoxelNet	0.23	81.97	65.46	62.85
SECOND	0.05	**87.43**	**76.48**	**69.10**
SECOND (small)	**0.025**	85.50	75.04	68.78

Table 12: SECOND and small variant results on "car" class on KITTI [13] 3D validation set, produced by [57].

Method	Time (s)	Easy	Moderate	Hard
MV3D	0.36	86.55	78.10	76.67
F-PointNet	0.17	88.16	84.02	76.44
VoxelNet	0.23	89.60	84.81	78.57
SECOND	0.05	**89.96**	**87.07**	**79.66**
SECOND (small)	**0.025**	89.79	86.20	79.55

Table 13: SECOND and small variant results on "car" class on KITTI [13] BEV validation set, presented in [57].

2.7 PointPillars: Fast Encoders for Object Detection from Point Clouds

PointPillars (PP) is a encoder which utilises PointNets [39] to extract information from point clouds divided in vertical columns (pillars). PP performs significantly better than earlier encoders in terms of both speed and accuracy. Despite simply using LiDAR, the whole detection pipeline greatly exceeds the SOTA on the 3D and BEV KITTI [13] benchmarks (explained in Section 4.1), even among fusion approaches (LiDAR and image). A runtime improvement of a factor of 3 when comparing to the fastest method so far (SECOND [57]) is attained, processing 62 frames per second. A faster version was also developed which ties SOTA results, running at 106 FPS on a 1080ti GPU.

The VoxelNet [61], which was based on the PointNet architecture created by Qi et al. [39], served as an inspiration for PP. VoxelNet [61] divides the space into voxels, applies a PointNet [39] to each voxel, then applies a 3D convolutional intermediate layer to consolidate the vertical axis, followed by the application of a 2D convolutional detection architecture. VoxelNet [61] performs well, but the inference time, at 4.4 fps or 227ms on a 1080ti GPU, is too slow for real-time deployment. Later, SECOND [57], introduced in 2.6, accelerated VoxelNet's [61] inference performance, however the 3D convolutions continued to be a bottleneck.

PP enables end-to-end learning using only 2D convolutional layers. To predict 3D oriented boxes for objects, PP uses a new encoder that learns features on the pillars of the point cloud. This strategy has a number of benefits. First, PP can take advantage of the whole information provided by the point cloud by learning features rather than depending on fixed encoders. Additionally, manual voxel height tuning is not necessary, since pillars are used instead of voxels. Last but not least, pillars are very effective since all critical operations can be expressed as 2D convolutions, which are very effective to compute on a GPU.

According to Lin et al.'s [26] study, a single stage method outperforms two-stage methods in terms of accuracy and runtime when using their proposed focal loss function. Therefore, PP employs a single stage detector.

2.7.1 Pipeline

As illustrated in Figure 22, the PP network is built on a identical VoxelNet [61] design similarly used by SECOND described in Section 2.6. Both PP and SECOND [57] use an SSD for the detection head (or RPN) slot. The main difference comes from SECOND [57] transforming and processing the input point clouds to voxels while PP uses pillars, enabling the application of 2D convolutional layers instead of 3D. It should be noted that a pillar is a voxel with unbounded spatial extension in the z direction, therefore a hyper parameter is not required to adjust the binning in the z dimension. Although LiDAR point clouds are employed, PP may use other point clouds, such as radar or RGB-D, by modifying the point's decorations.

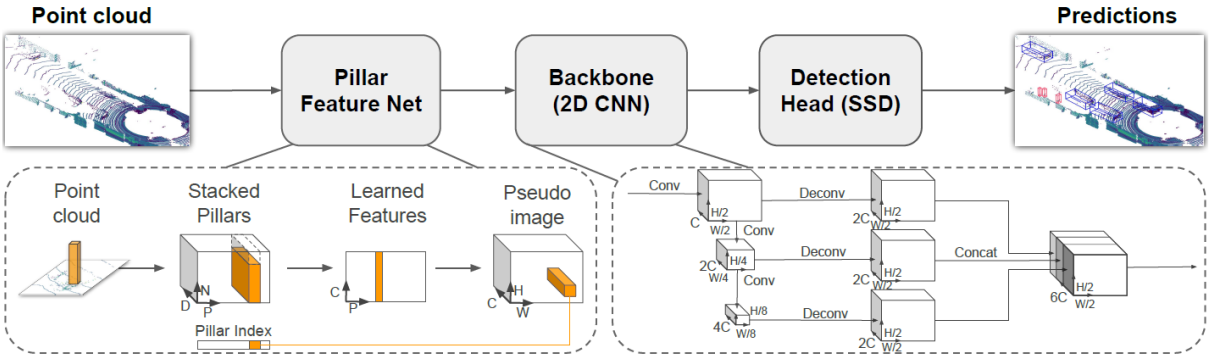


Figure 22: PP pipeline, presented in [24].

The Pillar Feature Network, a Backbone and an SSD Detection Head are the core components of the PP network, described as follows:

- **Pillar Feature Network** The point cloud must be processed into a pseudo-image by the Pillar Feature Network before being routed to the backbone. The point cloud is first adjusted into an x-y grid with uniform spacing, resulting in a collection of pillars. Due to the sparsity of the point cloud, most of the pillars will be empty, and the pillars that aren't empty will often have few points in them. Some LiDAR points are excluded when a pillar contains too much data to fit in this tensor. On the other hand, zero padding is used when a pillar has insufficient data to populate the tensor.

  Then, a simplified version of PointNet [39] is employed, applying a linear layer for each point, accompanied by Batch Normalisation and ReLU layers. Convolutions with kernel size of 1x1 can be used as the linear layer leading to a very fast computation time. After being encoded, the features are returned to the original pillar positions to produce a pseudo-image, whose height and width represent pillars distributed in the canvas. As mentioned before, by using pillars instead of voxels, slow 3D convolutions from VoxelNet's [61] Convolutional Middle Layer can be avoided.

- **Backbone**

  The backbone applies 2D convolutions with the goal of generating features from the created 2D pseudo-images. This backbone is similar to the one found in VoxelNet [61] architecture, and consists of two sub-networks: one top-down network that generates features with progressively lower

spatial resolution, and another network that conducts upsampling and concatenation on the top-down features. A set of blocks define the top-down backbone. A block is composed of 3x3 2D convolutional layers, each followed by batch normalisation and ReLU layers. The final features from each top-down block are combined through upsampling and concatenation at different stride values. Resulting upsampled features go, once again, through batch normalisation and ReLU layers. The final output features are a concatenation of all features from different strides.

- **Detection Head**

  The detection head predicts 3D bounding boxes using the information (features) from the backbone. An SSD is used to achieve this task. Ground-truth and predicted bounding boxes are matched using 2D IoU. Instead of incorporating the height and elevation of the bounding box into the 2D IoU matching, the height and elevation became additional regression targets.

In terms of data augmentations, they apply similar techniques as SECOND [57]:

- Sampling ground-truth objects from other scenes to the one being processed, adding that selecting 15, 0 and 8 ground-truths for cars, pedestrians and cyclists, respectively, achieved the best results;
- Object oriented augmentation: bounding box rotation and translation;
- Scene oriented augmentation: flipping (x and/or y axis), rotating, scaling and translating the whole point cloud.

2.7.2 Results

PP was able to achieve superior results compared to all previous implementations, not only in terms of overall mAP values, but also in regard to inference time or speed. These results can be analysed in Table 14, where 3D KITTI [13] benchmark (explained in Section 4.1) is presented.

Method	Modality	Speed (Hz)	mAP	Car			Pedestrian			Cyclist		
			Mod.	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
MV3D	Lidar & Img.	2.8	N/A	71.09	62.35	55.12	N/A	N/A	N/A	N/A	N/A	N/A
Cont-Fuse	Lidar & Img.	16.7	N/A	82.54	66.22	64.04	N/A	N/A	N/A	N/A	N/A	N/A
Roarnet	Lidar & Img.	10	N/A	**83.71**	73.04	59.16	N/A	N/A	N/A	N/A	N/A	N/A
AVOD-FPN	Lidar & Img.	10	55.62	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
F-PointNet	Lidar & Img.	5.9	57.35	81.20	70.39	62.19	51.21	**44.89**	40.23	71.96	56.77	50.39
VoxelNet	Lidar	4.4	49.05	77.47	65.11	57.73	39.48	33.69	31.5	61.22	48.36	44.37
SECOND	Lidar	20	56.69	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointPillars	Lidar	**62**	**59.20**	79.05	**74.99**	**68.30**	**52.08**	43.53	**41.49**	**75.78**	**59.07**	**52.92**

Table 14: PP results on 3D KITTI [13] benchmark, presented in [24].

The results are quite respectable, achieving the best value on the main class (car) and level (moderate) of the KITTI [13] benchmark, with a 1.33% improvement over the second-best result, SECOND. When comparing the mAP (explained in Section 4.4) to all SOTA methods, PP achieved a respectable 1.85%

improved performance over Roarnet [44], which obtained the second-best mAP value, using LiDAR and image information.

Even though the PP performance on the main class and level and on the mAP improved, the inference speed increase was the most remarkable. Based on the values presented in Table 14, PP obtains inference times that are more than 10 times greater than Roarnet [44] and 3 times greater than SECOND.

Figure 23 compares inference speeds and mAPs for different PP variants against other relevant SOTAs, using the KITTI [13] validation set. PP variants are composed of different pillar grid sizes created in the Pillar Feature Net. Decent mAP values are maintained, even for high inference times, making PP a prime candidate for a temporal adaptation.

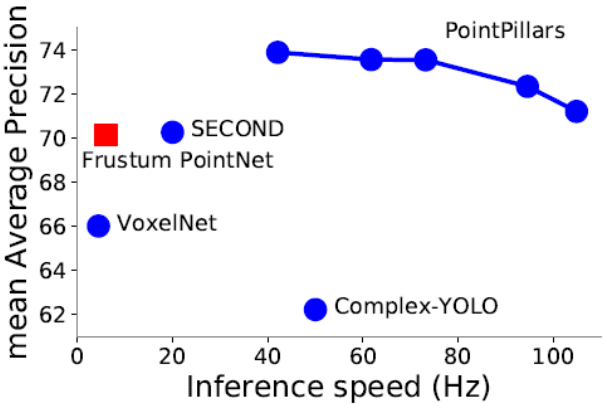


Figure 23: PP variants speed vs mAP against other methods on KITTI [13] validation set, produced by [24].

2.8 3D Object Detection For Autonomous Driving Using Temporal LiDAR Data

Recurrent PointPillars [32] or PP-REC isn't a SOTA method, although it serves as the initial temporal adaptation to the PP architecture. McCrae and Zakhor [32] justify the decision to adapt PP in its ability of learning features rather than depending on fixed encoders, therefore enabling it to utilise the complete information contained on the point cloud. When acting on pillars as opposed to voxels, it is not necessary to manually adjust the vertical's bin size. Most crucially, pillars are very efficient since all critical operations can be processed as 2D convolutions, which are incredibly efficient to run on a GPU. PP does not require manual adjustment to use multiple point cloud configurations, which is an added benefit of learning features. For instance, it may include several LiDAR frames or radar point clouds with relative ease.

2.8.1 Pipeline

PP-REC pipeline is highly based on the PP architecture, as displayed in Figure 24.

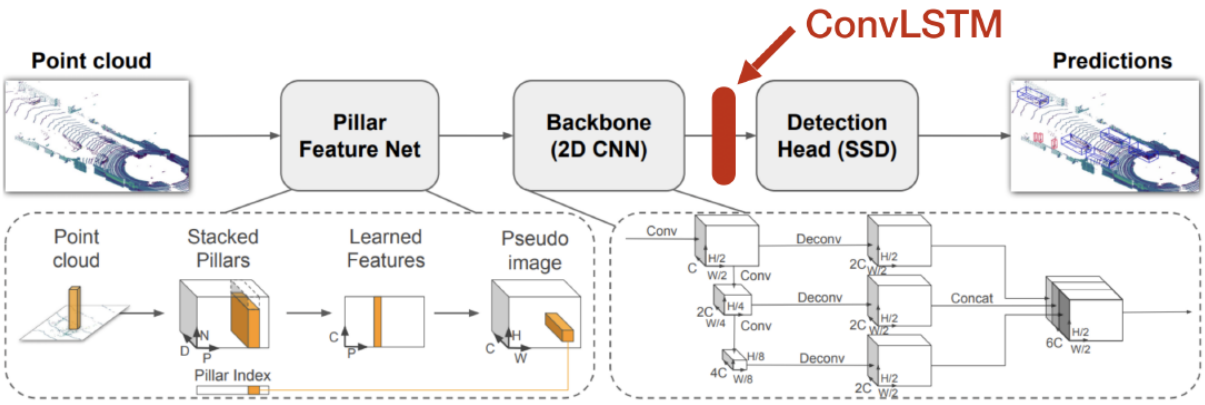


Figure 24: PP-REC pipeline, presented in [32].

One temporal module has been added, where a convolutional LSTM (ConvLSTM) was used, similar to that presented in Section 2.1. The ConvLSTM is placed between the output of the backbone output and the detection head's input. This allows the network to transmit spatial information contained in each feature-rich point cloud pseudo-image across time. Thereby, ConvLSTM transform the spatial information from various frames into temporal information. Instead of needing a dense point cloud made of several LiDAR frames as input, PP-REC may accept a point cloud with fewer frames and instead utilise its past information from memory. This results in processing fewer data at the feature extraction stage decreasing complexity and less blurring or artifacting related to object motion.

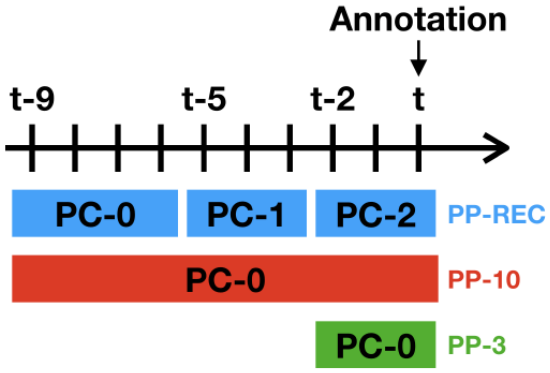


Figure 25: PP-REC and PP data preparation, produced by [32].

PP-REC separates the ten latest LiDAR frames into 3 point clouds. As seen in Figure 25, these point clouds are processed sequentially, using the previous two point clouds (PC-0 and PC-1) to generate recurrent memory, and the latest point cloud (PC-2) to provide detection results. PP-REC-S, an experimental streaming method that uses a sliding window to combine LiDAR frames one frame at a time, was also develop.

2.8.2 Results

Since PP-REC is an adaptation of the original PP, where a temporal module was added, the authors compared their implementation against the original PP to evaluate its added value, as can be analysed in Table 15.

Method	AP@all (%)			
	Coarse		Fine	
	Car	Pedestrian	Car	Pedestrian
PP (20 frames)	74.81	52.32	**75.44**	59.44
PP (10 frames)	69.43	44.27	74.68	**60.26**
PP (3 frames)	65.79	30.51	71.64	50.82
PP-REC (20 frames)	70.26	**53.01**	N/A	N/A
PP-REC (10 frames)	67.04	52.46	67.97	56.87
PP-REC-S (7 frames)	**75.79**	49.41	N/A	N/A

Table 15: PP-REC results against PP [24], values obtained in [32]. AP@all is averaged AP across the four nuScenes [6] matching thresholds (0.5, 1, 2 and 4). All PP-REC(-S) models use groups of 3 frames for uniformity.

The original paper refers to the averaged AP across the four nuScenes [6] matching thresholds has mAP, this was changed to AP@all, since mAP normally refers to the average values between classes.

Table 15 evaluates in two different pillars sizes, "coarse" pillars have a base dimension of 0.3125m x 0.3125m, while "fine" pillars have a base dimension of 0.25m x 0.25m. Therefore, coarse pillars have less resolution than fine ones. The recurrent hidden dimensions is 64 for the coarse model and 128 for the fine model. Both pillar's base dimension and hidden dimension have an impact on models accuracy and speed, as can be examined in Table 16. As could be expected, in coarse mode, all the models are able to get faster speeds than with finer pillars. PP using 3 frames or 10 frames has few impact on the model inference speed, since they get concatenated and processed as one point cloud. When comparing the original PP and the temporal adaptation (PP-REC), it can be concluded that the latest is slower by almost a factor of four.

Method	Speed (FPS)	
	Coarse	Fine
PP (10 frames)	116.08	88.97
PP (3 frames)	115.49	71.64
PP-REC	32.98	22.38

Table 16: PP-REC and PP [24] speeds, values extracted from [32].

As can be seen in Table 15, the approach with a sliding window, PP-REC-S, is able to achieve better AP@all results than every other presented method, in the car class with less resolution pillars (coarse), even when compared with PP using 20 frames and fine pillars. In the pedestrian class, despite PP-REC

(20 frames) being able to get the best AP@all value with coarse pillars, the original PP (10 frames) is capable of achieving an additional 7.25%.

In conclusion, the use of the PP-REC-S can be justified by the improved results in the car class. Even with an abrupt drop in the speed of the algorithm, it remains real-time, as its 33FPS is higher than the 20FPS value of nuScenes [6], which captures point clouds faster than other datasets, such as the 10fps of KITTI [13] and *Waymo* [45]. Note that tests were performed on a very powerful Titan RTX GPU, whereas tests are typically conducted on much slower hardware, as in the case of PP, which used a GTX 1080ti GPU.

2.9 Discussion and Conclusions

On this chapter, several SOTA methods are presented and compared with some difficulties, due to the different computational platforms applied, different sensor characteristics, DL methodologies, training and evaluation datasets and benchmarks, and evaluation metrics under different scenarios (e.g., detection and tracking).

Models like LSTM [20] or 3D-MAN [58] are the only temporal detectors using *Waymo* [45] open dataset to train and benchmark their models. PC-TCNN [55] also explores temporal information, however, it uses IMU/GPS data and is focused on tracking on the KITTI [13] tracking dataset, since KITTI [13] 3D and BEV OD benchmarks have no temporal data. SFD and SE-SSD [60] are both single-frame approaches and make use of the KITTI [13] 3D and BEV OD benchmarks. Nevertheless, SFD [56] uses RGB images in addition to the use of LiDAR point clouds information. SECOND and PP are also single-frame implementation on KITTI [13] 3D and BEV OD benchmarks, however their code was adapted for nuScenes, which already has temporal data. PP-REC isn't a SOTA method, nevertheless it will be taken into account when comparing nuScenes [6] results.

Table 17 compares the LSTM [20] and 3D-MAN [58] approaches in terms of results on the *Waymo* [45] open dataset. 3D-MAN [58] is inspired on LSTM [20], and demonstrated a major increase in performance. Nevertheless, LSTM [20] refers that its single-frame feature extractor runs at 19ms on a Titan V GPU, whereas 3D-MAN [58] doesn't even mention inference times.

Difficulty	Method	3D AP@0.7IoU (%)
LEVEL_1	LSTM*	63.60
	3D-MAN	69.03
	3D-MAN*	**74.53**
LEVEL_2	3D-MAN	60.16
	3D-MAN*	**67.61**

Table 17: Results between LSTM [20] and 3D-MAN [58] methods on the *Waymo* [45] open dataset. * refers to multi-frame methodologies.

PC-TCNN [55] can't really be compared with the other models, since it's focused on another problem,

i.e., tracking. Still, it outperformed all prior submitted methods on the KITTI [13] tracking benchmark. In addition, it must be taken into account that the evaluation is 2D and not 3D.

Method	Modality	3D AP@0.7IoU (%)			Time (ms)
		Easy	Moderate	Hard	
SECOND	LiDAR	83.13	73.66	66.20	50
PP	LiDAR	82.58	74.31	68.99	**23.6**
SE-SSD	LiDAR	**91.49**	82.54	77.15	30
SFD	LiDAR+RGB	90.83	**83.96**	**77.47**	100

Table 18: SOTA results on the KITTI [13] 3D benchmark for car detection. Values are extracted from [13].

SFD [56], SE-SSD [60], SECOND [57] and PP [24] can be compared, as presented in Table 18. SFD [56] prediction results are slightly better than SE-SSD [60], nevertheless, SFD [56] needs additional information (i.e., RGB images) to achieve these results.

Apart from that, both are single-frame models, which makes the inference time factor relevant to qualify the model for a conversion to a temporal context. All things considered, between the two methodologies, SE-SSD [60] is the best contender for the transition into a temporal object detector.

In terms of performance, SECOND [57] and PP [24] fall a bit behind the other two, but they have other advantages, such as already being adapted to a sequential dataset (nuScenes [6]), and PP [24] presents an inference speed edge that can be beneficial for maintaining temporal adaptation in real-time.

As the PP-REC-S model in Table 19 was not submitted to the nuScenes benchmark, this table shows the values obtained in the nuScenes validation set. The PP-REC time values were normalised in consideration of the nuScenes [6] benchmark's speed values. PP-REC authors conducted tests on a very powerful Titan RTX GPU, whereas tests are typically conducted on much slower hardware, as is the case with the nuScenes [6] benchmark.

Method	Class	AP@0.5	AP@1.0	AP@2.0	AP@4.0	AP@all	Time (FPS)
PP	Car	61.91	76.75	80.94	82.53	75.53	**61.2**
	Pedestrian	55.5	58.74	61.27	63.93	**59.86**	
PP-REC-S	Car	N/A	N/A	N/A	N/A	**75.79**	~16
	Pedestrian	N/A	N/A	N/A	N/A	49.41	

Table 19: SOTA results on the nuScenes [6] validation set. Values are extracted from respective papers [24] [32].

Only the PP-REC-S variation was selected for comparison, as it is the only one that achieves superior results in a particular class (car) when compared to the original PP.

PP-REC-S, with the times normalised by the benchmark, can no longer process frames faster than the nuScenes [6] sensor captures them (20FPS), although 16FPS is a solid value and is enough for datasets like KITTI [13] and *Waymo* [45] that output at FPS. Even so, its inference time should be something to improve, since other tasks, such as object tracking and motion prediction, are performed after OD task, and each module must be as quick as possible for an autonomous vehicle's process to be feasible.

In conclusion, PP [24] and SE-SSD [60] have the biggest potential compared to all other methods, due to being LiDAR-only, and having the lowest inference times of 23.6ms and 30ms, respectively, enabling their conversion into a multi-frame context.

SE-SSD [60] was implemented for the KITTI [13] 3D object detection, however this dataset does not have temporal data, i.e., its data isn't composed of temporal sequences of frames, but extracted frames from the data recordings with no connection to each other. As adapting this method to a new dataset would be laborious and validating the results obtained in said dataset would be a complex and demanding task, the PP [24] has the advantage. Bearing in mind that PP-REC is a first temporal adaptation of PP [24], it can serve as an inspiration and a point of comparison.

All presented models have interesting techniques that can be useful in the development of a SOTA real-time LiDAR-only temporal 3D object detection model, which is the objective of this work.

Chapter 3

Objectives

The goal of this dissertation is to research and develop an OD approach and analyse its feasibility in the context of AD, taking into account:

1. R&D of multiple SOTA approaches to detect objects in the context of AVs;
2. Benchmark the performance and computational cost of the proposed solutions;
3. Identification of its main limitations.

Essentially, it is proposed a temporal (multi-frame) end-to-end 3D OD DL algorithm, which uses LiDAR information, and capable of detecting objects, considering their location, size, rotation, class, among other parameters in a sequence of frames. This process should lead to accurate estimations of objects' bounding boxes with high confidence scores.

Public datasets with LiDAR data will be explored, in order to train and benchmark the implemented algorithm with other SOTA methods.

3.1 Schedule Plan

For a healthier evolution of this work, a schedule plan was elaborated. The schedule process was divided into two phases, one for the pre-dissertation (introductory & SOTA documentation analysis) and another for the dissertation itself (algorithm implementation & benchmark of results). Each phase is composed as follows:

First phase – Pre-dissertation

1. Literature review of methodologies that fall under the same topic of research;

2. Definition of problem restrictions;

3. SOTA research and plan elaboration.

Second phase – Dissertation

1. First proof of concept of 3D OD pipeline;

2. Prepare the data to be used for training and testing accordingly;

3. Implement, train and optimise temporal neural networks and its hyper-parameters;

4. Comparison of the advantages/drawbacks of each method when applied to AD;

5. Evaluation of the obtained results through benchmarking.

A Gantt diagram was also created, as presented in Table 20 for a better visualisation of how the activities are allocated in the following months of the development of this work:

	2021			2022								
	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
Literature review												
Definition of problem restrictions												
SOTA research and plan elaboration												
First proof of concept of 3D OD pipeline												
Data preparation												
Implement, train and optimise models												
Comparison of each method												
Model evaluation												
Dissertation writing												

Table 20: Gantt Diagram.

3.2 Research Contributions

One very important aspect of the development of this dissertation is to add value to the present state of OD in the context of AD. The main research contributions of this research work include:

- Analysis of SOTA 3D OD implementations in the context of AD, including single-frame and temporal, which use different datasets and various types of data (e.g., LiDAR, RGB, GPS and IMU).

- Research about publicly available SOTA datasets and benchmarks ([13, 45, 6]), comparing technical information for the context of temporal OD.

- Develop an end-to-end temporal 3D OD implementation, capable of using different types of temporal modules (e.g., LSTM [20] and GRU [7]).

- Refine the developed implementation, modifying important implementation aspects (e.g., data preparation, temporal module and data augmentation), through pipeline refining, architecture fine-tuning and hyper-parameterisation.
- Benchmark the developed implementation with other single-frame and temporal SOTA methodologies, producing an in-depth evaluation analysis considering their performance and inference times.
- Visualisation and demonstration of the acquired results, comparing a single-frame baseline and a temporal approach visually.
- Comparison of the advantages, drawbacks and restrictions of each benchmarked methodology when applied to AD;

Chapter 4

Methodology

This chapter will introduce some fundamental object detection tools and concepts. They will help to understand some aspects of OD. The three most used datasets in the context of OD in AD will be analysed in detail, putting together a comparison summary between them at the end. Evaluation metrics and notations mentioned in this dissertation will also be explained. Finally, it will be carried out with a brief description of a possible pipeline implementation.

4.1 Datasets

Choosing a good dataset is as important as implementing a good algorithm. Different datasets have different specifications, which can greatly influence how the algorithm learns and consequently the results it can achieve. Thus, the three datasets that are mentioned in this dissertation, which are the three main ones for OD, will be exposed and compared.

All presented datasets have one or more associated benchmarks, however these may use less information and classes for model evaluation than what the respective dataset offers.

4.1.1 KITTI

KITTI [13] datasets (illustrated in Figure [13]) have been around since 2012 and is composed of many variants, however the main dataset will be the first introduced, as it is the most explored and evaluated in the domain of OD.

Four high resolution video cameras, a Velodyne laser scanner and a localisation system were used to produce this dataset. This made it possible to create benchmarks for tasks in stereo, optical flow, visual odometry/SLAM and 3D OD, however only the latest is relevant for this work. The Velodyne laser scanner is the sensor responsible to record the 3D point cloud based LiDAR data, the only type of information

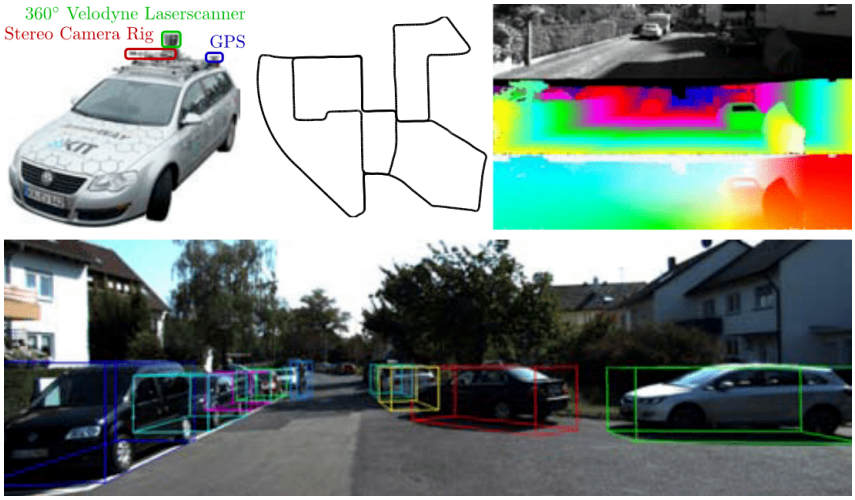


Figure 26: KITTI information illustration example, extracted from [14].

which can be used in this work. Nevertheless, models in these benchmarks usually use the information coming from more sensors, having a slight advantage in terms of the amount of information used.

Despite KITTI [14] presented sequential raw data, 3D and BEV OD datasets and benchmarks are single-frame, i.e., they use dispersed frames from a scene, not linked with each other in a temporal manner. This characteristic makes it difficult to train and evaluate a temporal end-to-end learning model.

The two KITTI [13] OD benchmarks mentioned in this work to validate results are the following:

- **KITTI 3D OD Benchmark**

  As the name suggests, KITTI [13] 3D OD benchmark is the most important KITTI [13] benchmark, as it is focused on the detection of 3D objects. However, as an OD benchmark it uses single-frame evaluation as mentioned previously.

- **KITTI BEV OD Benchmark**

  The KITTI [13] BEV benchmark, despite providing 3D data for OD, the evaluation is done by projecting the detections onto a 2D plane. Once again, being a KITTI [13] OD benchmark it uses single-frame evaluation.

Other popular KITTI variant is the KITTI [13] tracking dataset, which already has sequential annotated data, but as the name implies, this information is based on tracking features rather than OD. KITTI [13] tracking dataset is similar to the OD dataset, however its data and annotated information differs slightly, making it difficult to adapt a model between the two datasets.

- **KITTI Tracking (2D) Benchmark**

  Despite being a tracking benchmark, before keeping object on track it is necessary to detect them. Unlike OD benchmarks, this benchmark already contains sequential data, enabling the evaluation

of temporal models. However, only a 2D evaluation is performed, and the evaluation metrics used are very tracking focused.

This tracking benchmark could be used as the official evaluation tool, using the metrics only associated with the detection of the object, ignoring the others.

All KITTI [13] benchmarks have three levels of difficulty oriented to the occlusion of the objects:

- **Easy**: Fully visible objects with a minimum bounding box height of 40 pixels;
- **Moderate**: Partially occluded objects with a minimum bounding box height of 25 pixels;
- **Hard**: Difficult to see objects with a minimum bounding box height of 25 pixels.

A new dataset, named KITTI-360 [25], was recently publish in September 2021. This dataset being a recent iteration, got improved with new sensors and 360° view, more annotated classes, and most important of all, with 3D sequential evaluation benchmarks.

However, there are still no relevant models in the OD domain that implements this dataset nor use its benchmarks. Thus, this dataset is not expected to be taken into account for the development of this work.

4.1.2 *Waymo*

Waymo [45] dataset (illustrated in Figure 27) comes from a technology development company for self-driving cars, part of *Alphabet Inc.*, the conglomerate that *Google* owns.

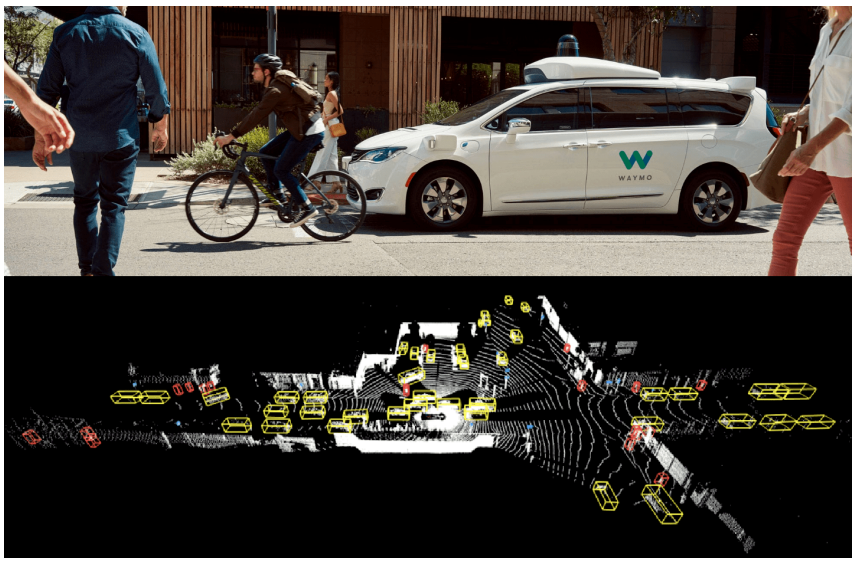


Figure 27: *Waymo* information illustration example, extracted from [45].

Due to the use of one mid-range and four short-range LiDAR sensor, *Waymo* [45] is a high-quality dataset. It also has synchronised data from five cameras which potentiates the use of multi-modal algorithms.

Not that many models are created around this dataset, making it difficult to compare one implementation to other already-developed methodologies.

Similarly to KITTI [13] OD benchmarks, *Waymo* [45] also has difficulty levels to deal with objects suffering from occlusion or lack of information:

- **Level 1** - easier level which identifies 3D labels with at least 5 LiDAR point;
- **Level 2** - most difficult level that includes every label with 1 or more LiDAR points.

4.1.3 NuScenes

Created by *Motional*, the nuScenes [6] dataset (illustrated in Figure 28) is one of the largest datasets, due to the use of a various sensor information (i.e. one LiDAR, six RGB cameras, five radars, IMU and GPS).

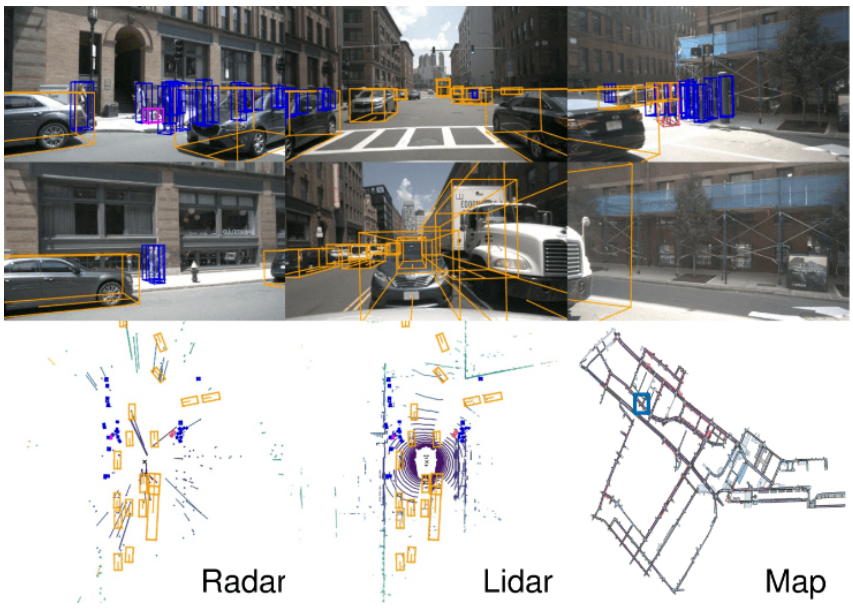


Figure 28: NuScenes information illustration example, extracted from [6].

The nuScenes [6] dataset captures data at 20 Frames Per Seconds (FPS), however only has 2 annotated FPS, instead of the typical 10 FPS of other datasets, nevertheless models can implement specific techniques to utilise all of the 20 frames. For each annotated frame, there are nine unannotated frames, therefore, models typically combine the points of all 10 frames into a single one, using the annotation of the first one as ground-truth.

Temporal models that compared results between nuScenes [6] and their own dataset got very different results. MVFusenet [23] achieved 67.8% AP@0.7IoU on nuScenes [6], and 78.4% AP@0.7IoU on an internal dataset, both in the "vehicle" class. Therefore, it is possible that implementing a temporal algorithm first on nuScenes and then adapting it to a more temporal oriented dataset can improve results.

4.1.4 Summary

To facilitate the comparison and visualisation of the datasets, all relevant LiDAR information was prepared and can be analysed in Table 21.

	KITTI Tracking	NuScenes	Waymo
Frames per Second	10	**20**	10
Annotated Frames per Second	**10**	2	**10**
LiDAR Frames	19k	**400k**	230k
Annotated LiDAR Frames	8k	40k	**230k**
Frames per Sequence	~**900**	400	200
Annotated Frames per Sequence	~**380**	40	200
Total Sequences	50	1000	**1200**
Annotated Sequences	21	1000	**1200**
Average LiDAR Points per Frame	120k	34k	**177k**
LiDAR Sensors	1	1	**5**
Classes Evaluated	3	**10**	4
Evaluation	2D	**3D**	**3D**
Annotation Coverage	Front Camera FoV	**360º**	**360º**

Table 21: Dataset comparison. "k" represents thousands.

KITTI [13] 3D/BEV datasets are not listed on Table 21 as they lack temporal related information, e.g., FPS and Total Sequences, and, therefore, cannot be used to develop a temporal model.

KITTI [13] Tracking is a weak candidate since it is not only unfocused on OD, but also presents "poor" values compared to the rest of the datasets. It has the highest number of frames per sequence (900), which is only possible due to its low number of sequences (50) when compared to 1000 and 1200 for nuScenes [6] and *Waymo* [45], respectively, both with scenes of 20 seconds each. *Waymo* [45] is composed of 1000 scenes for training and validation, and 150 scenes for testing, while nuScenes [6] consists of 850 scenes for training and validation and 150 for testing.

As mentioned before, FPS is an important factor for the training of a temporal model. Despite nuScenes [6] only having 1 annotated point cloud for each 10 frames, this can be used to develop a temporal model. The same can be said for the total number of LiDAR frames, which despite only having 40 thousand annotated frames, in total there are 400 thousand frames, being this a solid number and much higher than the other benchmarked datasets. Additionally, the number of sequences is high, maintaining a healthy level of environmental variation. The number of available evaluated classes is essential for validating the model's robustness against a wide variety of objects, and its value in nuScenes [6] is significantly greater than in other datasets. The final advantage listed in the table is the 360-degree annotation and 3D scene evaluation, which matches the values on *Waymo* [45].

Waymo [45] has the advantage on the most factors. The most impressive value is the 177 thousand average LiDAR points per frame, which corresponds to a higher amount of information per frame and scales well with the large number of total frames. *Waymo* [45] has the same values for annotated and

unannotated information, however, unannotated data can also be utilised through pre-processing techniques (e.g., concatenation) and since nuScenes [6] has the highest value of unannotated data, it has a slight edge on the number of total frames that can be used. Very few models use *Waymo* [45], so it was decided to not be considered for a first algorithm implementation.

All things considered, nuScenes [6] is the best candidate to implement a first algorithm iteration, as it not only has strong values for training and evaluating an implemented model, but also has single and temporal SOTA methods implemented on it.

4.2 Data Representation

LiDAR data can have various representations. Each different pre-processing technique influence the amount and quality of the information provided to the models. Typically, higher complexity of the input information leads to an increased number of computational resources. In order to analyse the different point cloud representations, a list of multiple pre-processing techniques is presented as followed:

- **Point Cloud**

  Raw point clouds are the native 3D information of the LiDAR sensor. Each point in a point cloud is represented by its coordinates (x, y, z), as well as its reflection intensity (ρ).

  The raw point cloud is the input for the 3D OD models. 3D detection methods need to work, atleast partially, with 3D data representations to create 3D bounding boxes. An example of a raw point cloud is illustrated by Figure 29.

Figure 29: Raw point cloud visualisation, presented in [47].

- **Bird Eye View (BEV)**

  BEV information is the projected view from the top of a point cloud into a 2D image, also called top-down view, as presented in Figure 30.

Figure 30: BEV point cloud representation.

This is a robust representation to factor object scale variation and occlusion. The "z" or height variable is transformed into a feature channel for the 2D convolutions. This projection into a 2D view has the advantage of allowing the use of 2D convolutions, which is much faster and with less latency than 3D convolutions.

- **3D Voxels**

  3D Voxels are an ordered 3D representation of raw point clouds, which divide the scene into various cubes, named voxels, as illustrated in Figure 31. Voxels have the same size throughout a scene, but the different selected sizes will change the performance of the model and processing times of each frame.

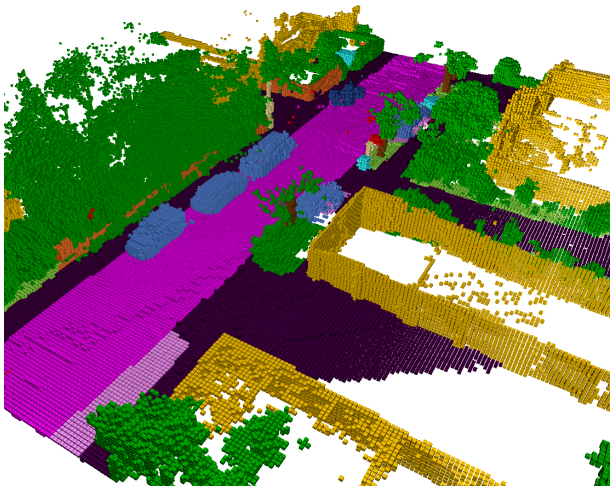


Figure 31: Example of a voxelized point cloud in a semantic scene, adapted from [1].

Voxels are a great way to model and visualise volumetric data, as is the case of the data produced in AD scenes. Point clouds are voxelized into 3D voxel grids to allow for 3D convolutions. However, 3D convolutions require more computational power and greater latency than 2D convolutions.

A variant of voxels with fixed height and that can't be stacked on top of each other are called pillars. Pillars, being height independent, can be treated similarly to the BEV, as they can be efficiently processed using 2D convolutions, maintaining a lower computational power and achieving lower inference times.

- **Range View (RV)**

  RV representation also projects the point cloud into a 2D image created from the defined-view perspective (e.g., front-view, side-view, etc.) of the sensor/vehicle. The points in the image are coloured differently depending on their distance to the sensor, as illustrated in Figure 32.

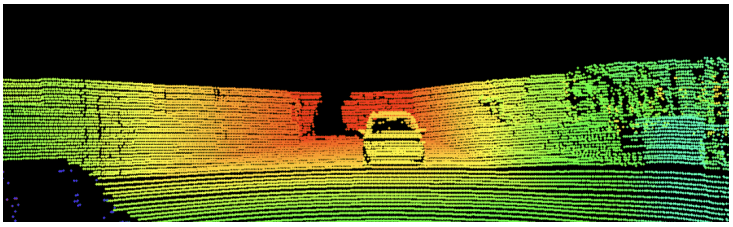


Figure 32: RV point cloud representation, adapted from [4].

This representation preserves the "z" value and, it is a straightforward transformation from sparse information to dense, allows a simple fusion of LiDAR and RGB camera information. Nevertheless, it has object scale variation and occlusion limitations. Additionally, the advantage of this RV is due to the application of a 2D projection, which allows 2D convolutions to be applied.

4.3 Pipeline Structure

It is possible to observe, in Figure 33, a general pipeline architecture for temporal OD.

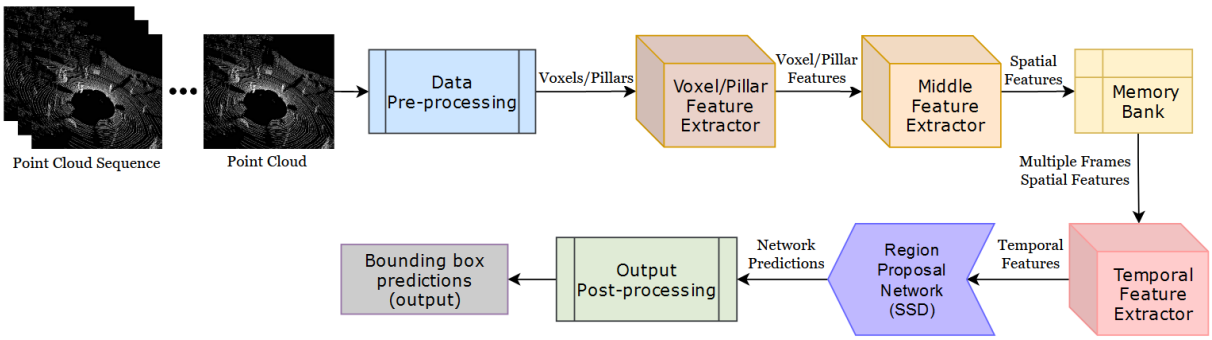


Figure 33: OD pipeline example.

For each frame fed to the pipeline, it is necessary to perform its pre-processing (blue box in Figure 33)). This module is responsible for transforming the raw point clouds into a different data representation(s), chosen for the feature extractor. 3D voxels, BEV and RV are examples of these representations and are explained in Chapter 4.2.

Despite not being directly part of the pipeline structure, it is also necessary to take into consideration the use of data augmentation techniques. Some of these operations can be applied to the scene (e.g., scale, rotation, flip, etc.) while others can be performed directly on the object in a similar way as SE-SSD's [60] shape-aware data augmentation (e.g., bounding box face dropout, swap and sparsify). Scene operations need to be applied taking into account the whole sequence, as it can harm the model performance if temporal information isn't congruent. Through these techniques, the main advantage is the use of generated synthetic data, which allows models to assimilate new patterns as a way to improve its performance.

A feature extractor (orange box in Figure 33) is required to extract all essential spatial features from the processed data. Features extraction can be composed of 2D and/or 3D convolutions.

A memory bank (yellow box in Figure 33) can be used to store the most diversified information, e.g., raw point clouds, extracted features and/or model states. The memory bank doesn't have a specific place in the pipeline, as it is dependent on what it is chosen to be stored. The one in Figure 33, similarly to the one in 3D-MAN [58], stores spatial features from previous frames to be used by the temporal feature extractor.

A temporal feature extractor (red box in Figure 33) has a similar objective to the regular feature extractor. It uses current frame extracted features and a predefined number of the most recently extracted spatial features from the memory bank to generate temporal features. This module will transform the entire single-frame process of extracting information into a temporal context.

A region proposal network (purple box in Figure 33) will process the spatial features to generate predictions. Every predicted bounding box should be represented by a centre, a size consisting of a length, a width and a height and a rotation or orientation. Additional, each bounding box may be associated with an id object, its corresponding class and confidence.

It is necessary to develop a module (green box in Figure 33) that analyses the predictions and performs useful operations, such as NMS, hungarian matching and/or filter predictions by confidence. It will be necessary to select wisely the application of these operations, as they can bring a heavy computational burden, e.g., avoiding the use of NMS (demonstrated in Figure 34) as it is a costly operation, which doesn't scale well with an increase of predictions. A solution could be forcing the model to predict bounding boxes without overlapping, through hungarian matching or similar techniques.

The final predictions, after all processing, are represented by the output box (the grey box in Figure 33).

4.4 Evaluation Metrics

The success of the similar model's behaviour can be measured with the use of evaluation metrics. In the context of OD, evaluation metrics provide a numerical value corresponding to how well the model detects all objects in a scene.

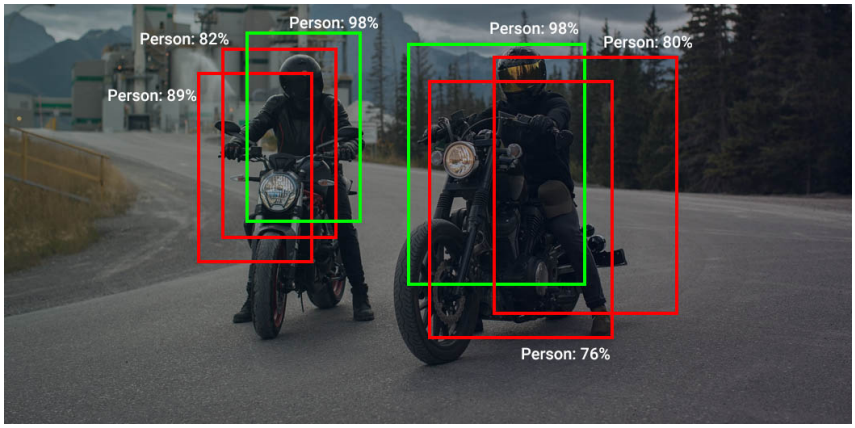


Figure 34: 2D NMS example, extracted from [41].

The most basic metrics, which are not just related to OD, and from which the more complex ones derive, are the following:

- **Precision**: measures how accurate predictions are, i.e., the percentage of predictions that corresponds to ground-truth bounding boxes.
- **Recall**: measures the fraction of ground-truth bounding boxes which are correctly predicted.

Some of the OD metrics make use of the more basic ones to create metrics more suitable for the context of OD. The main metrics are the following:

- **Average Precision (AP)**: is a way to summarise the precision-recall curve into a single value. AP is only calculated individually for each class.
- **mean Average Precision (mAP)**: AP values averaged over multiple classes.
- **Intersection of Union (IoU)**: measures the overlap between two bounding boxes, as demonstrated in Figure 35. It's used as a threshold for accepting a prediction (e.g., for the threshold of 70% or 0.7IoU, if the IoU of the two bounding boxes reaches a value above 70%, the prediction will be accepted as a good prediction or true positive).

  KITTI [13] benchmarks uses the AP and mAP metrics with multiple values of IoU threshold, resulting in the expression $AP@IoU$ and $mAP@IoU$, respectively.

  Some contests also apply other variants to the evaluation process. For instance, the COCO [27] competition uses AP@[.50:.05:.95], meaning the IoU starts with 0.5 and ends with 0.95 with a step size of 0.05, which results in a AP value averaged over ten IoU levels.

- **distance (dist)**: in the context of OD, it concerns to the distance between the centre of two bounding boxes, as demonstrated in Figure 36.

  Similar to IoU, the metric AP uses the dist as a threshold, resulting in the terminology dist AP@X, where X denotes the threshold value, typically in meters. This metric is used by nuScenes [6] with

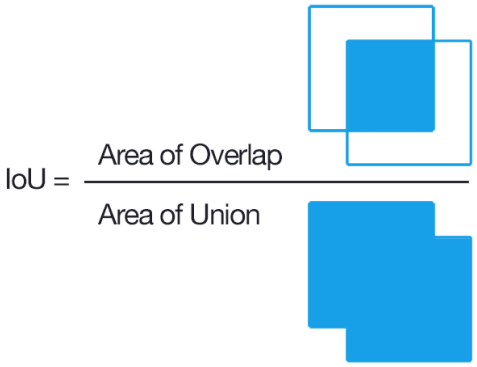


Figure 35: IoU visualisation, extracted from [21].

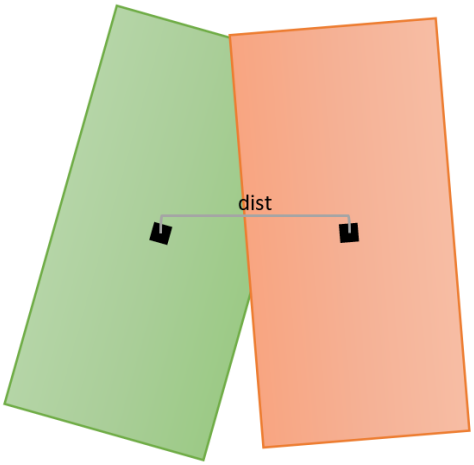


Figure 36: Bounding box centre distance visualisation.

thresholds of 0.5, 1, 2 and 4 meters. When mAP is mentioned in nuScenes [6], it refers to AP averaged across the thresholds and then averaged over all classes. The AP@all in this work refers to the AP averaged over all match thresholds (0.5, 1, 2 and 4).

This metric has limitations as it only compares the distance between the bonding box centres, ignoring other factors like size and orientation. This can result in two bounding boxes with very close centres but very different sizes and orientations, leading to a high AP when the prediction is not particularly accurate. In order to lessen these limitations, this metric is typically used in conjunction with some attributes like translation, scale, and orientation errors.

Chapter 5

Temporal 3D Object Detection

This chapter will detail our implementation of a temporal algorithm capable of utilising several frames of a scene and detect the most diverse objects. The hardware and software used to develop train and evaluate the methodologies will be presented. More technical information will be provided about the selected dataset, elaborating on specific features that are necessary to comprehend the implementation around that dataset. In addition, two particular methods will be described, one of which will be an existing SOTA algorithm (PP [24]) and the other will be a temporal variant of the algorithm inspired by both the original [24] and a first temporal iteration created in [32]. The pipeline, how the data is pre-processed and a slightly more comprehensive model architecture will be presented. In the end, a comparison will be made between the SOTA findings and the acquired results, assessing and comparing the acquired values. Ending with a brief demonstration of a training evolution case and visualisation of some dataset point clouds with both successful and unsuccessful results.

5.1 Hardware & Software

To elaborate the algorithms for this study, it was necessary to have adequate hardware for the development, training and testing of DL algorithms. The hardware specifications are was the following:

- **Hardware**:
  - CPU: *Intel* Xeon Gold 6248 (max 8 cores);
  - GPU: *NVIDIA* Tesla V100 32GB;
  - RAM: 6 GB.

In order to replicate the development environment for training and testing on multiple platforms, a Docker image was used having all the necessary dependencies and libraries installed:

- **Software**:
  - Docker 20.10.12;
  - Linux Ubuntu 20.04;
  - CUDA 11.3;
  - CuDNN 8.2;
  - NCCL 2.9.9;
  - Python 3.9.7:
    - PyTorch 1.10.0;
    - Torchvision 0.11.0;
    - NumPy 1.21.0;
    - Tensorboard 2.10.1;
    - SpConv 1.1;
    - Nuscenes-devkit 1.0.1.

5.2 The Dataset

The dataset chosen for the development of the temporal algorithm was nuScenes [6], both as it is a technically robust temporal dataset, composed of 1000 scenes of 20 seconds each, divided into 700 scenes for training, 150 for validation and 150 for testing, and as it was the dataset with temporal data implemented by the PP [24] method.

In this section, some additional technical information about nuScenes [6] dataset will be presented to aid comprehension of its value, delving deeper into components like the quality of sensors, available classes and organisation of their data.

5.2.1 Sensors Data

NuScenes [6] is a multimodal dataset, i.e., it has data from different type of sensors. Figure 37 illustrates the organisation the sensors used by nuScenes [6] vehicle.

Six cameras capturing images at 12 FPS with a cropped resolution of 1600x900 produce a 360° FoV when combined. Additionally, there are five RADARs capturing at a 13Hz frequency that together create a 360° FoV cloud with a range of up to 250 meters. IMU and GPS sensors are integrated to add positioning and localisation information to the dataset. A Velodyne sensor is responsible for recording LiDAR information, which, similarly to RADAR, it creates point clouds of the environment, however instead of radio waves as in RADAR, it uses beams of light to localise the objects. The beams of light have the advantage of carrying objects reflectance[1] information after hitting an object. NuScenes [6] uses a Velodyne sensor configuration which captures LiDAR information at 20Hz, which is higher than the 10Hz from KITTI [13] and *Waymo* [45], however with the disadvantage of having less points per point cloud.

[1] Reflectance is the ability of the object's surface to reflect radiant energy.

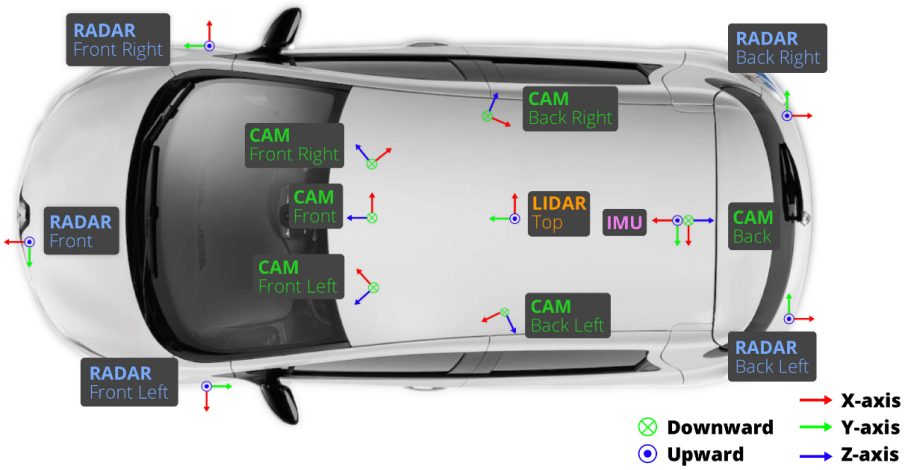


Figure 37: NuScenes vehicle's sensors, extracted from [6].

Despite the Velodyne sensor being able to collect data with a range up to 100 meters, only points below 70 meters distance are usable. Even with LiDAR having a lower capture range compared to RADAR, it has a much better point accuracy (with less than 2 centimetre error) and less sparsity, being the main causes of LiDAR methods showing better performance than RADAR in the literature.

Despite the abundance of temporal data from various sensors provided by this dataset, the goal of this study is for the developed model to detect objects using only LiDAR information.

5.2.2 Classes

The high number of annotated and evaluated classes is a major strength of the nuScenes [6] dataset. Figure 38 depicts the distribution of the 1166187 objects from 23 classes within the train and validation sets.

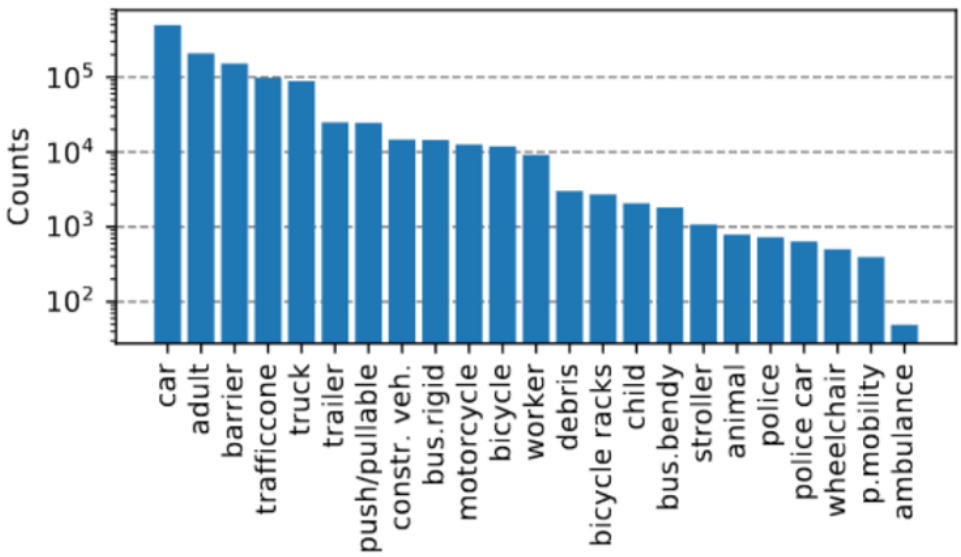


Figure 38: Classes distribution on train and validation set, extracted from [6].

After all 23 classes being grouped, it originates a total of 11 evaluation classes (Car, Pedestrian, Barrier, Traffic Cone, Truck, Trailer, Bus, Construction Vehicle, Motorcycle, Bicycle and Ignore), as illustrated in Table 22.

Class	Mean	Max	Frames w/ objects	Primary classes
Car	14,45	97	33434	Car
Pedestrian	6,45	89	27533	Adult, Child, Worker and Police
Barrier	4,45	105	11242	Barrier
Traffic Cone	2,87	56	14856	Traffic Cone
Truck	2,59	31	24396	Truck
Trailer	0,73	25	8661	Trailer
Bus	0,48	8	11268	Bus Rigid and Bus Bendy
Construction Vehicle	0,43	8	8428	Construction
Motorcycle	0,37	12	8058	Motorcycle
Bicycle	0,35	9	7841	Bicycle
Ignore	0,99	38	10561	Others
Total (all classes)	34,15	160	34140	

Table 22: Evaluated classes information per frame. “Mean” and “Max” represent the mean and maximum number of objects per frame, respectively.

Despite the fact that the developed method predicts for all eleven evaluation classes, only three will be the target classes, i.e., the Car, Pedestrian, and Motorcycle classes.

Typically, approaches only analyse the Var and Pedestrian classes because they have more samples for training and validation. However, assessing a class with fewer samples will help to understand how the model behaves in classes with less information, both in the training and validation data.

High-risk collisions between vehicles and pedestrians, which may result in pedestrian fatalities, make the Pedestrian the most important class. The second most important class is the Car, as it is the class with the most information and with which the ego-vehicle[2] shares the road most frequently. As a vehicle with a high potential for injury in a accident and the least amount of information, motorcycles are also relevant.

The 493322 samples of the Car class emerge in 33434 of the 40000 annotated frames from the dataset, with a maximum of 97 objects per frame and an average of 14.45 objects per frame. The Pedestrian shows up in 27533 frames with an average of 6.45 objects per frame and a maximum of 89 objects. The motorcycle class only appears in a total of 8058 frames, with a maximum of 12 objects and an average of 0.37 per frame.

The motorcycle class was selected not only due to having a small sample size of 12617 objects and a presence of 1.08%, but also because it is an extremely important class in an AD, sharing the road together with the ego-vehicle. The motorcycle class was chosen over the bicycle class due to the bicycle’s near-zero performance in the examined SOTA methods.

[2]Vehicle equipped with sensors and algorithms for AD.

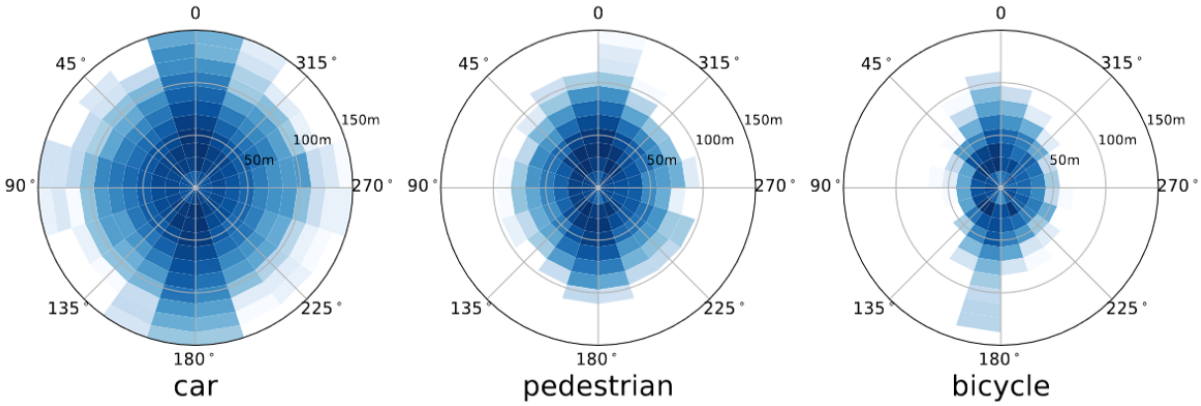


Figure 39: Density map for bounding box annotations in which the radial axis is the distance in meters from the ego-vehicle and the polar axis is the direction angle with respect to the ego-vehicle. More bounding box annotations in a region lead to a darker bin. Figure presented in [6].

Figure 39 displays three density maps for all sensors data bounding boxes annotations for the classes Car, Pedestrian and Bicycle. Object annotations tend to show up more frequently on the front and rear of the ego-vehicle, since most objects that travel on the road with the ego-vehicle are oriented parallel to it, and due to the slight focus bias of sensors on those directions. Most annotations are concentrated between 0 and 100 meters, which can be explained by the fact that most annotations are done on data from LiDAR sensors, which have a range of up to 100 meters. Given the small number of samples and the relatively compact nature of bicycles, their annotations are concentrated within a range of up to 50 meters, and it is reasonable to assume that the density map for motorcycles would look very similar, as they share many of the same annotation characteristics and related physical structure.

5.2.3 Data Structure

NuScenes [6] LiDAR, RADAR and camera data can be accessed and downloaded from the download tab in their website[3]. In order to handle the data relevant to LiDAR processing, a "ROOT" folder with the downloaded information must be organised as follows:

- Samples: Key point cloud frames;
- Sweeps: Point cloud frames without annotation;
- v1.0-trainval: metadata and annotations for key frames.

When all the data is structured in the folder, a python shell command, developed by the SECOND [57] creator, should be run to create a ground-truth database with the train and validation binary data, in a pickle format:

```
python create_data.py nuscenes_data_prep --data_path=ROOT --version=v1.0-trainval
```

By modifying the configuration file, adding the respective database file paths for training and validation, the methods are able to acquire the point cloud and ground-truth information.

[3] https://www.nuscenes.org/nuscenes#download

5.3 PointPillars

PP is the SOTA approach chosen as the baseline for this work implementation. It was chosen due to being LiDAR-only, and having the lowest inference time, making it the best choice for a conversion into a multi-frame context. It is one of the first methods implemented for nuScenes [6], achieving reasonable performance, so it's used by many a baseline [5, 52, 53, 50]. The PP pipeline was already explained and analysed in detail on Section 2.7.1 and illustrated in Figure 22, therefore it will not be reviewed again here.

5.3.1 Data Pre-processing

The input data must undergo a crucial process that transforms the information into the appropriate format to then be forward to the model.

The first step in making the data compatible with PyTorch is transforming the data into tensors, which are a type of data used as arrays or matrices to store information, in order to run more efficiently on GPU, despite also being able to run (inefficiently) on CPU.

Batch creation enables high parallelism during training, i.e., different point clouds can be handled simultaneously by the model, thereby drastically reducing training times.

So that the model knows which predictions in the learning task are accurate and how well it is performing, it is necessary to link ground truth bounding boxes to their corresponding frames during training and evaluation. Data augmentation is also a vital pre-processing technique as it can significantly boost performance.

Some of the pre-processing techniques can be adjusted in the configuration file, enabling, disabling or giving different intensities to each procedure, as illustrated in Figure 40.

```
batch_size: 3
preprocess: {
  max_number_of_voxels: 25000
  shuffle_points: false
  groundtruth_localization_noise_std: [0, 0, 0]
  groundtruth_rotation_uniform_noise: [0, 0]
  groundtruth_points_drop_percentage: 0.0
  global_rotation_uniform_noise: [0, 0]
  global_scaling_uniform_noise: [1.0, 1.0]
  global_translate_noise_std: [0, 0, 0]
  random_flip_x: false
  random_flip_y: True
  remove_environment: false
}
```

Figure 40: Processing sample from configuration file.

The configuration takes into consideration the following attributes:

- **Max Number of Voxels:** Maximum number of voxels/pillars that can be created for each point cloud;

- **Shuffle Points:** Points in a point cloud don't follow any particular order, so mixing them prevents the model from becoming dependent on points order when repeatedly learning from the same point cloud;
- **Object Noise:** It is a combination of various object related transformations (ground truth localisation noise, ground truth points drop percentage and ground truth rotation uniform noise), which may translate or remove some points related to the object and rotate the object left or right;
- **Global Rotation:** Rotate the entire point cloud around the ego-vehicle;
- **Global Scaling:** Shrinks or expands the whole point cloud;
- **Global Translate:** Translate the whole point cloud along the x, y and/or z axes;
- **Global Flip:** Flips the entire point cloud on the x and/or y axes, swapping left and right or front and back;
- **Remove Environment:** Because not every point is associated with an object, this process eliminates all such points.

NuScenes [6] has a peculiar manner of annotating its dataset, while other datasets (e.g., KITTI Tracking [13] and *Waymo* [45]) annotate every frame, nuScenes [6] only annotates one for every ten frames. This is due to the higher FPS and in order to reduce time and costs in producing a dataset with many sensors and data.

To circumvent the absence of annotated frames, methods can combine ten frames into a single point cloud, as illustrated Figure 41. Due to the movement of the objects and the ego-vehicle, this concatenation blurs the objects, erasing any temporal characteristics the objects may have in different frames.

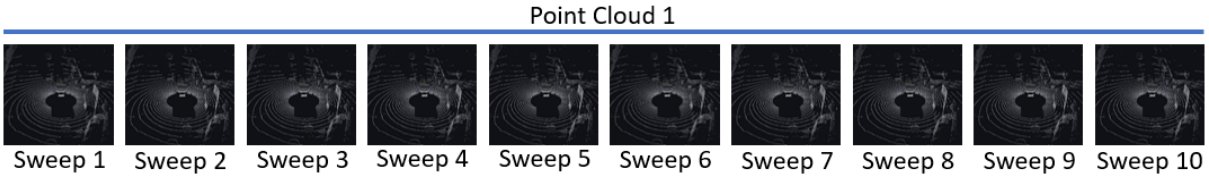


Figure 41: PP frames concatenation.

5.4 Temporal PointPillars

This work developed a Temporal PP, which is very similar to PP-REC [32]. As the name implies, PP is used as a baseline, integrating a temporal module to it and modifying the code accordingly so that the data flow retains its accurate temporal nature. Although PP was implemented both for KITTI [13] and later for nuScenes [6], only the latest has temporal characteristics, so this temporal approach will keep using nuScenes [6] for its training and validation. The process of comparing the baseline approach and the temporal approach, which adds a temporal module, is therefore possible, and enables the analysis of the single-frame and temporal strengths and drawbacks.

5.4.1 Pipeline

As stated in the introduction to this method, the pipeline closely resembles that of PP, maintaining the data pre-processing, voxel/pillar feature extractor, middle feature extractor, RPN and output post-processing modules, however adding the modules of memory bank and a temporal feature extrator, as presented in Figure 42. This pipeline is also utilised by PP-REC [32].

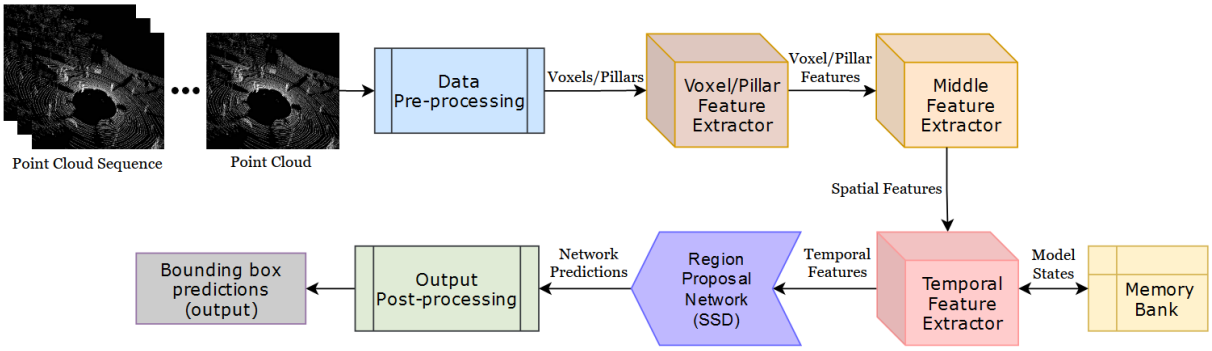


Figure 42: Temporal PP pipeline.

5.4.2 Data Pre-processing

The temporal method and PP are similar in terms of data processing. In regard to data augmentation, it was necessary to be particularly cautious with certain transformations as they might no longer realistic in a temporal context. For instance, only flipping some scenes of the temporal sequence would result in temporal inconsistencies. To avoid situations where, for example, only one of the temporal point clouds experiences a left-to-right flip, the data augmentation module was implemented with care to keep the random operations congruent with all the temporal point clouds using seeds. Some transformations, despite having been implemented, make little sense in a temporal context, such as global rotations, where the movement of the ego-vehicle and other objects in the scene causes inconsistencies.

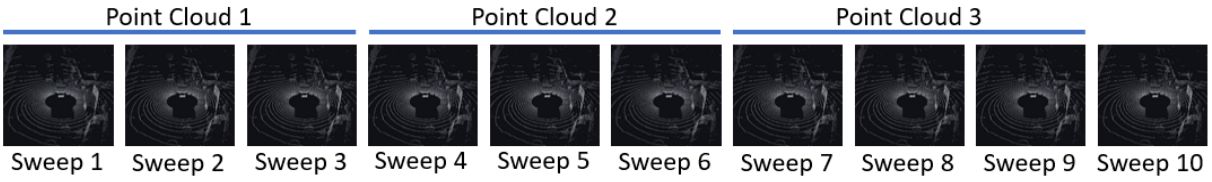


Figure 43: Temporal frames grouping processing.

As displayed in Figure 43, temporal point clouds are generated in a somewhat different fashion. In contrast to the PP [24] approach, which concatenates 10 sweeps into a single point cloud, the temporal method groups the sweeps into many point clouds. The number of frames for each point cloud can range from 1 to 10, with 1 resulting in the greatest number of point clouds (10), and 10 resulting in a single point cloud, which is no longer temporal. Utilising one and three sweeps per point cloud revealed to have the most significant results.

Grouping more than 8 point clouds for processing showed system instability, leading to crashes due to lack of memory, so with 1 sweep per point cloud, only a total of 8 sweeps can be used. In the case of 3 sweeps per point cloud, a total of 3 point clouds are created. A fourth point cloud is not created with a single sweep, nor is the extra sweep added to the last point cloud (technique used by [32]), because in a real case scenario each new point cloud always has the same number of sweeps.

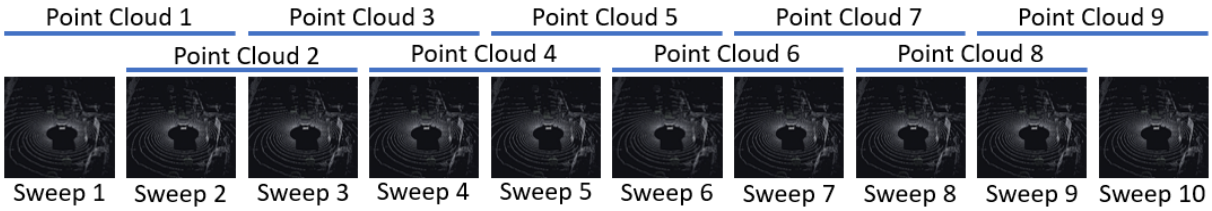


Figure 44: Temporal frames grouping processing with sliding window.

Figure 44 illustrates a sliding window mode in which each point cloud has at least two sweeps and each point cloud differs from the previous one by a single point cloud. However, despite [32] reporting better results in the Car class, these results failed to be replicated, resulting in inferior training results and a lengthy training period. Both are justified by the repetition of information, which tends to confuse architectures (e.g. LSTM) and makes them extremely slow when processing so much redundant information.

A code to use 20 sweeps, instead of 10, was also developed, but the greater the number of point clouds used, the more memory was consumed and the longer it took for the data to go through the entire pipeline, leading to extremely high training times. As a result of its computational limitations and the increase in training time, its benchmark was consequently constrained.

5.4.3 Memory Bank

Memory bank is the implementation's central storage, where a vast variety of data can be stored. This information is intended to be used during the processing of every new frame, and may be limited to saving the input data for use in later frames, which, in the case of LiDAR data processing, are the raw point clouds, or saving information created during the processing of previous frames. The information created in previous frames can be data directly required by subsequent processes (e.g., RNN's hidden state) or that would be generated repeatedly in each time step and always produce the same results (e.g., same point cloud feature extraction), thereby increasing processing time in each time step.

Only information regarding the hidden state of the RNNs, or the cell state in the case of the LSTM, is stored in the implemented memory bank. The decision to only store this information is due to the small size of the data, which is approximately 20 megabytes in GPU memory. Thus, not only will there be no issues with memory space, as the addition of information stored on the GPU is negligible, but the fast speeds will also be established, as the cost of passing memory from the CPU to the GPU is significant.

5.4.4 Temporal Module

Temporal module is composed of an RNN, and all the variants presented in Section 1.2.8, i.e., Standard RNN, LSTM and GRU were implemented and can be used interchangeably.

The information from earlier time steps stored in the memory bank, along with the spatial features that were extracted from the current time step, are input to the temporal module. This module aims to analyse all the data to derive temporal features. As a result, it is possible to enhance these temporal features by using relevant information found in earlier point clouds.

5.5 Benchmark and Discussion

In this section, comparisons will be made between the SOTA implementation PP [24], a previously implemented temporal adaptation of PP [32], and some of the developed implementations and experiences acquired during the course of this work.

The benchmark will emphasise the detection performance of three main classes: Car, Pedestrian and Motorcycle. Section 5.2.2 explains why these specific classes were selected. As evaluation metric, it will be used the official nuScenes [6] benchmark metric, distance AP, which is explained in Section 4.4 alongside with its thresholds, being 0.5, 1, 2, 4 and the average of them (all).

5.5.1 SOTA PP versus Our PP

As a first point of reference, in Table 23, we benchmark the official results obtained by the original PP, and the values obtained by our PP proof of concept.

Method	Class	AP@0.5	AP@1.0	AP@2.0	AP@4.0	AP@all
PP*	Car	62.9	73.07	76.77	78.79	72.88
	Pedestrian	**61.44**	**62.61**	**64.09**	**66.35**	**63.62**
	Motorcycle	9.25	12.9	13.69	14.11	12.49
PP (Ours)	Car	**63.8**	**77.63**	**81.71**	**83.15**	**76.57**
	Pedestrian	57.82	60.92	63.38	66.02	62.04
	Motorcycle	**19.51**	**26.49**	**27.15**	**27.71**	**25.22**

Table 23: Results between original PP [24] and our PP proof of concept. * represent approaches with self-declared AP values.

The code for PP (Ours) is nearly identical to PP [24], however, minor modifications were made on training to achieve the best results. For instance, the use of the entire training dataset as opposed to only half as used by PP [24] because it contains more quality data. Fine-tuning that primarily focuses on parameters found in the configuration file was also done to improve results.

Our proof of concept attains slightly different results than the original. On the one hand, it achieves significantly better results in the class Car, reaching 76.57% AP@all, an increase of nearly 4% AP@all, and

in the class Motorcycle, with more than double of the performance, from 12.49% to 25.22% AP@all. On the other hand, it achieves slightly lower performance on the class Pedestrian, which is the most relevant class on the dataset, getting 64.04% AP@all, 1.58 percent less than the original.

Overall, our PP achieves better results due to the significant increase in the Car and Motorcycle classes, despite the slight decline in the most important class.

5.5.2 PP-REC versus Our Temporal PP

Both PP [24] and PP-REC [32] served as inspiration for our temporal implementation. Our temporal approaches will be compared with relevant SOTA temporal implementation as a first benchmark, as presented in Table 24.

Method	Class	AP@all	IT (ms)
PP-REC-S*	Car	**75.79**	N/C
	Pedestrian	49.41	
	Motorcycle	N/A	
PP-REC*	Car	67.97	N/C
	Pedestrian	56.87	
	Motorcycle	N/A	
PP-REC (Ours)	Car	69.01	47
	Pedestrian	52.79	
	Motorcycle	7.86	
RNN PP (Ours)	Car	74.86	**27**
	Pedestrian	64.90	
	Motorcycle	15.25	
GRU PP (Ours)	Car	75.03	29
	Pedestrian	65.63	
	Motorcycle	14.39	
LSTM PP (Ours)	Car	75.16	29
	Pedestrian	**66.66**	
	Motorcycle	**19.72**	

Table 24: Results between PP-REC [32], our replication and our Temporal PP approaches. IT means inference time. * represent approaches with self-declared AP values. All methodologies were trained and evaluated with 3 sweeps per point cloud.

All methodologies were trained and evaluated with 3 sweeps per point cloud, however, while our approaches divide 9 sequential sweeps into 3 point clouds, PP-REC-S [32] uses a sliding window technique to generate 7 point clouds with a single sweep shift, resulting in duplicate and redundant information. This version of PP-REC [32] was chosen because the declared results were superior to those of other versions. Despite efforts to replicate similar results with the same parameters and even with more point clouds and finer pillars (made possible by code and memory optimisations), the sliding window method always produced the worst results. RNNs typically struggle with repeated and redundant information, so it is not surprising that the sliding window technique produces the poorest results in this instance.

Our replication of the PP-REC is not applying sliding window mode, as the values obtained are significantly lower than those of all other variations. In contrast, the values obtained by our replication are comparable to the PP-REC values, with approximately 1% increase in the Car class and approximately 4% decrease in the Pedestrian class.

The values obtained by the PP-REC-S are odd since it receives fewer sweeps, the information on the pillars has less resolution, and training and evaluation use repeated information. The decrease in results for the Pedestrian class partially justifies the increase for the Car class, indicating that the model prioritises the Car class more than the Pedestrian class.

The variants implemented exclusively in this work, i.e. (Standard) RNN PP, GRU PP, and LSTM PP, achieved comparable results. All reached around 75% for the Car class, with the LSTM PP having a slight edge in the Pedestrian class, with an increase of more than 1% compared to the other two, and achieving the greatest difference in the Motorcycle class, with an increase of more than 4% compared to the other two.

To ensure a fair comparison, all inference times were measured on the same hardware. Despite the fact that the PP-REC also employs a LSTM, our methods are significantly faster due to major code optimisations. When comparing PP-REC with our equivalent LSTM method, there is a significant 18ms difference. The RNN PP is the fastest of our implementations, while the other two have equivalent inference times with each other.

Theoretically, all implementations are real-time as long as their inference time is less than 50ms, which is the rate at which the nuScenes [6] sensor collects data. However, the hardware utilised is significantly more powerful and energy-intensive than that typically found in autonomous vehicles. Not only do our temporal approaches have a significant advantage in terms of prediction quality, but with their edge in speed are also much easily implemented in a real world situation.

Our LSTM PP is the best of all temporal implementations, as it achieves the highest values in the Pedestrian and Motorcycle classes, with a non-significant decrease in the Car class compared to the values declared for the PP-REC-S. LSTM PP is also the second-fastest variant, behind RNN PP by 2 milliseconds, however the improvement in performance justifies its preference.

5.5.3 RNN versus LSTM

The longer the sequences, the more RNNs are susceptible to the vanishing and exploding gradient problem, therefore, a comparison was made between two sequence sizes for the Standard RNN and LSTM. As shown in Table 25, the sizes consist of 3 point clouds (9 sweeps with 3 sweeps per point cloud) and 8 point clouds (8 sweeps with 1 sweep per point cloud).

Using 3 sweeps per point cloud, i.e., sequences of reduced length, RNN PP and LSTM PP do not differ significantly in terms of performance, despite a slight advantage for LSTM PP, as explained in the previous comparison.

Method	Class	AP@0.5	AP@1.0	AP@2.0	AP@4.0	AP@all	IT (ms)
RNN PP (3sweepPerPC) (Ours)	Car	61.93	75.74	79.93	81.82	74.86	27
	Pedestrian	60.5	63.79	66.34	68.96	64.90	
	Motorcycle	10.32	16.4	16.99	17.28	15.25	
LSTM PP (3sweepPerPC) (Ours)	Car	**63.17**	75.82	79.83	81.81	75.16	29
	Pedestrian	**62.34**	**65.51**	**68.1**	**70.68**	**66.66**	
	Motorcycle	**13.81**	**20.49**	**21.99**	**22.59**	**19.72**	
RNN PP (1sweepPerPC) (Ours)	Car	57.23	72.73	77.36	79.5	71.71	**25**
	Pedestrian	45.39	48.39	50.77	54.15	49.68	
	Motorcycle	5.44	9.35	9.85	10.13	8.69	
LSTM PP (1sweepPerPC) (Ours)	Car	61.65	**76.4**	**80.72**	**82.73**	**75.38**	27
	Pedestrian	60.64	63.55	66.04	68.7	64.73	
	Motorcycle	11.23	19.69	20.85	21.28	18.26	

Table 25: Results between our (Standard) RNN PP, and our LSTM PP with different values for sweeps per point clouds. IT means inference time.

When using 1 sweep per point cloud, the length of the sequences increases. It is clear from looking at the RNN PP results that they have decreased significantly, as was to be expected. The Pedestrian class has decreased by an astonishing amount (more than 15% AP@all) demonstrating the difficulty Standard RNNs have in dealing with the issue of vanishing and exploding gradient, while the LSTM PP model proved effective in combating the issue, as both approaches produced similar outcomes.

It should be taken into account that for different thresholds the results may be superior for different variants, as is the case in the Car class for the two LSTM PP variants, where the 3 sweeps per point cloud variant is superior only at threshold 0.5, with a difference of more than 1%, whereas the 1 sweeps per point cloud variant wins in all other thresholds, including the most important (AP@all), but with an almost negligible advantage of 0.22%.

Methods that use 3 temporal sweeps per point cloud have access to more information, as a total of 9 temporal sweeps are used, whereas in the case of 1 temporal sweep per point cloud, only 8 point clouds are created, resulting in a total of 8 temporal sweeps, as there is a limit of 8 point clouds for training not to crash.

The variants that use one sweep per point cloud can truly be considered real-time, as they transform newly acquired sweeps into point clouds for processing, whereas in the case of three sweeps per point cloud, a buffer of three sweeps must be filled prior to creating the point cloud. With 1 sweep per point cloud, the point clouds are processed every 50 milliseconds, however with 3 sweeps per point cloud, they are limited by the capture of those three sweeps, only being processed every 150 milliseconds.

The LSTM PP with 1 sweep per point cloud was appointed as the best overall, as it can run and process point clouds in 50 milliseconds or less, achieving second best overall results with less total sweeps/information. Although the RNN PP with the same number of sweeps per point cloud is 2 milliseconds faster, this advantage is more than outweighed by the fact that the AP results are significantly worse across all classes and thresholds.

5.5.4 PP versus LSTM

As a final benchmark, the original PP [24] will be compared to the implementation that achieved the best results in this study, LSTM PP. For a clearer understanding of the advantages and disadvantages of each variation, various sweeps per point cloud will be presented in Table 26.

Method	Class	AP@all	IT (ms)
PP (10sweepsPerPC) (Ours)	Car	**76.57**	31
	Pedestrian	62.04	
	Motorcycle	**25.22**	
	Bicycle	0	
PP (3sweepsPerPC) (Ours)	Car	73.60	29
	Pedestrian	52.88	
	Motorcycle	17.01	
	Bicycle	0	
LSTM PP (3sweepsPerPC) (Ours)	Car	75.16	29
	Pedestrian	**66.66**	
	Motorcycle	19.72	
	Bicycle	0.3	
PP (1sweepPerPC) (Ours)	Car	67.61	**27**
	Pedestrian	41.95	
	Motorcycle	11.53	
	Bicycle	0	
LSTM PP (1sweepPerPC) (Ours)	Car	75.38	**27**
	Pedestrian	64.73	
	Motorcycle	18.26	
	Bicycle	**0.31**	

Table 26: Results between original PP [24], and our LSTM PP with different values for sweeps per point clouds. IT means inference time.

The Bicycle class was added to Table 26 as it is quite similar to the Motorcycle class and provides additional information about model predictions.

As demonstrated in the previous benchmark, LSTM PP does not change significantly with the use of fewer sweeps per point cloud, despite using less information. However, the performance of the original PP decreases drastically when the number of sweeps per point cloud is decreased, as it only uses the information from the most recent point cloud. Comparing the variant with 10 sweeps per point cloud to the variant with 1 sweeps, this produces a significant reduction of over 20% AP@all for the class Pedestrian.

PP with 10 sweeps per point cloud achieves comparable results to LSTM PP with 3 sweeps per point cloud, despite the fact that LSTM PP employs one fewer sweep in total for each prediction. PP outperforms LSTM PP in the Car class by more than 1% and in the Motorcycle class by more than 5%. However, when analysing the most important class on the dataset, i.e., Pedestrians, LSTM PP outperforms PP, by more than 4.5%. PP is unable to locate the Bicycle class well enough for the value of AP to rise, whereas the LSTM PP, despite being negligible, already reveals a value greater than 0. The LSTM PP model tries to

predict the Bicycle class more precisely at the cost of not predicting the Motorcycle class as well, as they are physically quite similar. It should be noted that the PP variants have the advantage of applying more data augmentation techniques during training that do not make sense in the temporal case, such as the scene front and back flip, which improves performance.

In terms of inference times, both implementations use the same amount of sweeps per point cloud, with the 1 sweep per point cloud implementation being 4 milliseconds faster than the 10 sweeps per point cloud solution. The temporal module in LSTM PP has an inference time of less than 2ms, however, due to code optimisations, LSTM PP maintains the same inference time as PP when using the same number of sweeps per point cloud.

For training and evaluation, for both temporal and single point cloud models, all sequence sweeps (up to 8 point clouds) are fetched and stored in memory. However, the temporal approach would not need to store any information from previous point clouds, considering that this procedure is forced due to nuScenes [6] annotations being performed once every 10 sweeps. Only the hidden state and cell state would be stored for use by the RNNs. In contrast to the PP, which must store the previous 10 sweeps in CPU memory, concatenate them into a single point cloud, and process it from the start, these states maintain a much smaller size and are always ready to be used when a new sweep arrives, requiring neither fetch time nor pre-processing. In the perspective of AD real world implementation, the temporal model may be even faster than the single point cloud, as CPU memory usage and memory management are minimal. These times would vary greatly, as they were not accounted for in the benchmark.

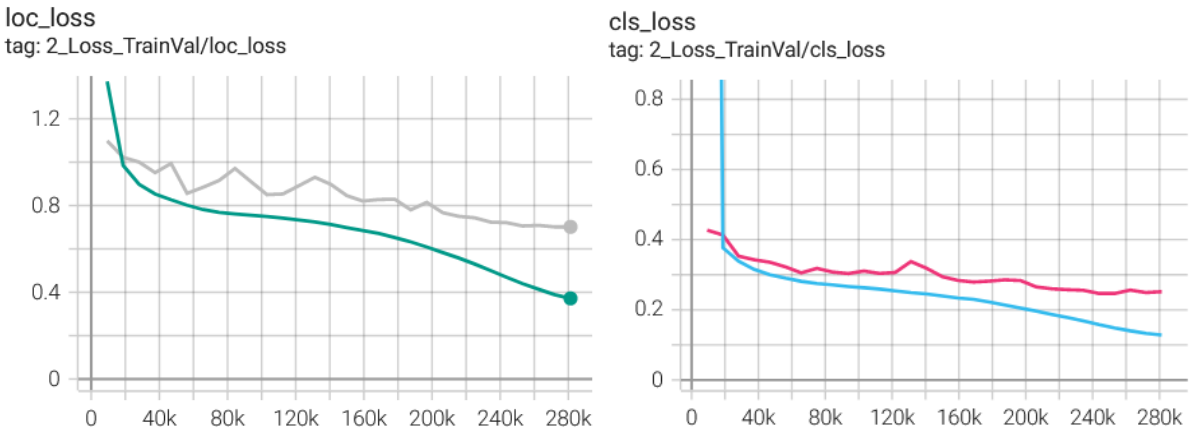
All models have advantages and disadvantages, but LSTM PP with 1 sweep per point cloud can be considered the most balanced, and an in-depth analysis will be presented in the Section 5.5.5. This method achieves the lowest inference time and memory usage by not requiring the sweeps to be kept in memory, and instead storing the less memory-intensive hidden states and cell states. On the other hand, the results are quite comparable to the best values obtained by other models in every class, despite dealing with less information (8 sweeps in total). Thus, when dealing with 9 or 10 sweeps in total, it would be able to achieve even better results.

5.5.5 Model Learning Analysis

To track the evolution of the model's training, a library called tensorboard was utilised. It enables a real-time visualisation on an interactive website by adding the appropriate lines of code and inserting the values of the variables you wish to track. Thus, it was possible to monitor the evolution of values such as classification and localisation loss, accuracy of the model's RPN, the AP in the various classes and thresholds, as well as memory utilisation, among many others.

As an example, some tensorboard data from LSTM PP with 1 sweep per point cloud will be analysed, where the values are presented after every epoch, or every 9350 steps.

Model losses are the most crucial parameters because they can be examined to determine whether or not the model is learning correctly. In this instance, as shown in Figure 45, our aim is to minimise



(a) Localisation loss progression.

(b) Classification loss progression.

Figure 45: Losses progression. Green and blue colours refers to training loss and grey and red colours to validation loss.

two losses: a loss related to the model's estimation of an object's location, and a loss related to the classification of the object. For localisation it was used smooth L1 loss [29] and for classification a sigmoid focal loss [26].

Normally, it is preferable for the training loss to be greater than the validation loss, but it is difficult to combat model overfitting due to the limitations of data augmentation and other factors. Overfitting happens when the model starts to match perfectly only on the training data as a consequence of learning training patterns too well, while providing negative performances on unseen data (e.g., validation data). Even in PP training, this trend is similar. Consequently, it remains crucial that the validation loss decreases, and when this decrease becomes stagnant, it indicates that the model has ceased to learn.

These losses take into consideration the average for all existing classes providing the means to assess this results in-depth, there is also information on location loss per class, in case it is necessary to visualise with more granularity, as is the case of Figure 46.

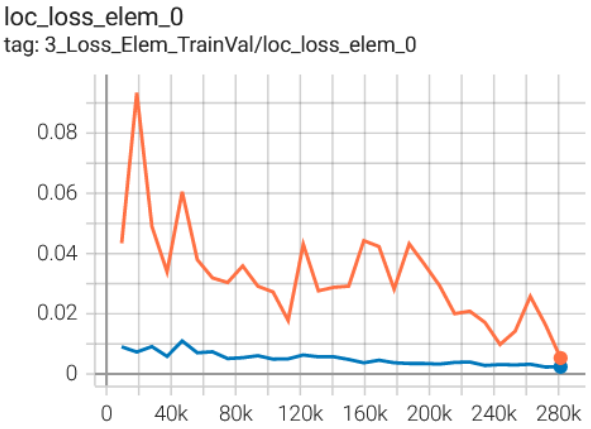


Figure 46: Localisation loss progression for Car class. Orange colour refers to training loss and blue colour to validation loss.

In the case of the Car class location, it can be seen that the validation line (blue) is always below the training line (blue) and is still decreasing, indicating that there is still room for improvement. However, as mentioned at the beginning of this section, this training was terminated due to time constraints.

As shown in Figure 47, it is also possible to track the evolution of the module responsible for the prediction, i.e. the RPN.

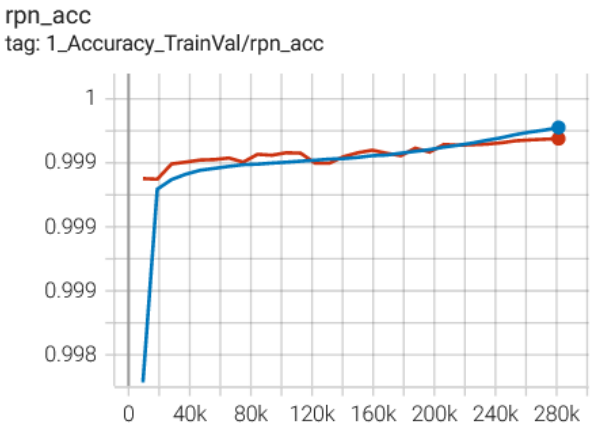
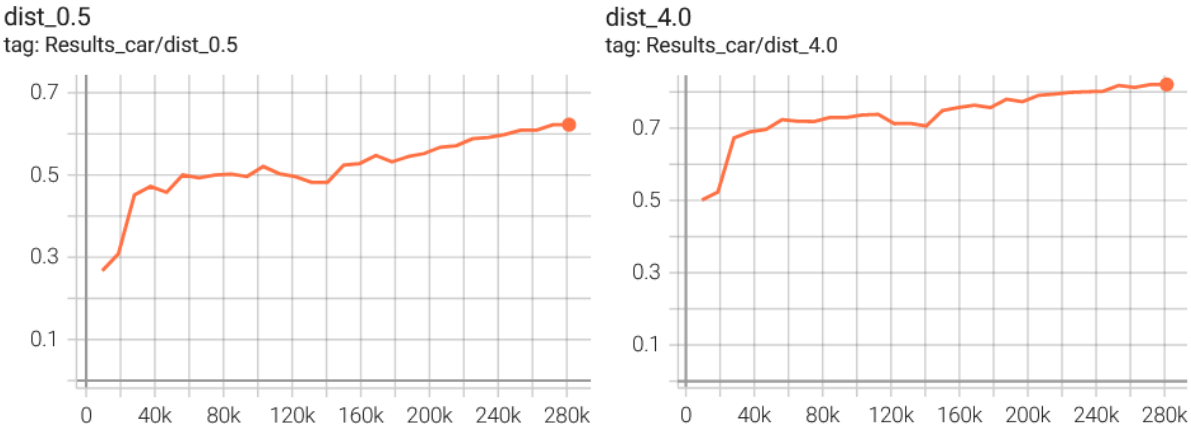


Figure 47: RPN accuracy progression. Blue colour refers to training accuracy and Orange colour to validation accuracy.

At the end of the first epoch, both the training and validation accuracy of RPN are already exceeding 0.99 percent. In spite of this, it is possible to observe a rising trend in both lines, with the training line only surpassing the validation line completely at the 220 thousand step.

As shown in Figure 48, the progression of the AP@dist for the different thresholds can also be directly observed in a graph for a more intuitive understanding of the evolution of detection accuracy.



(a) AP progression with 0.5 dist threshold for Car class. (b) AP progression with 4 dist threshold for Car class.

Figure 48: AP progression with 0.5 and 4 distance thresholds for Car class.

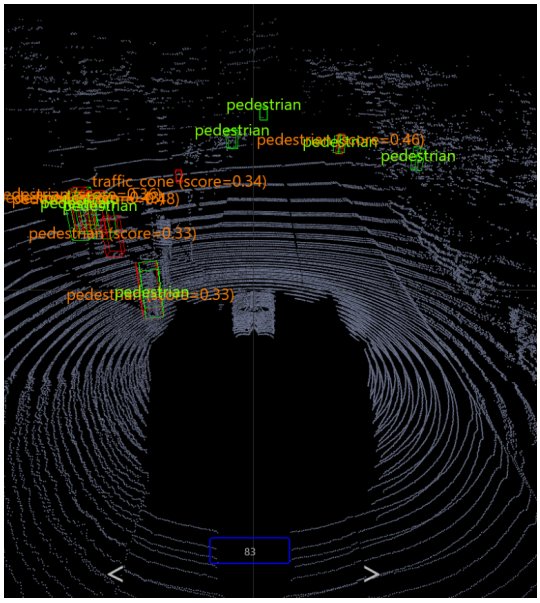
Considering that detections with a distance threshold of 4 metres (up to 4 metres) also include detections with a distance of less than 0.5 metres, the 4 metre threshold value is always higher. As expected, due to the still reducing loss, the AP for the Car class exhibits an upward trend in both thresholds. Thus, it

is demonstrated once more that this model has the potential to achieve even higher results with additional training.

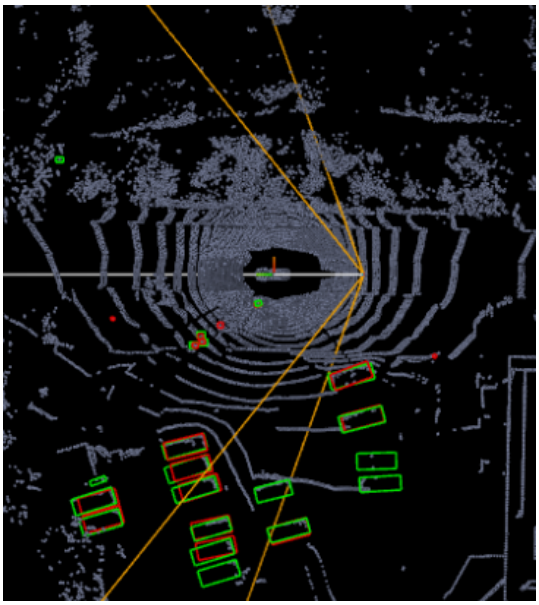
5.6 Visualisation and Demonstration

Through a visualiser, the dataset, its annotations, and model predictions may be evaluated qualitatively. Even though the base code was already provided, modifications were necessary to display more helpful information.

As displayed in Figure 49, the dataset information can be analysed interactively in different views.



(a) Interactive view for point cloud 83 from validation set, with ground-truth and predictions.



(b) BEV generated from point cloud 83 from validation set, with ground-truth and predictions.

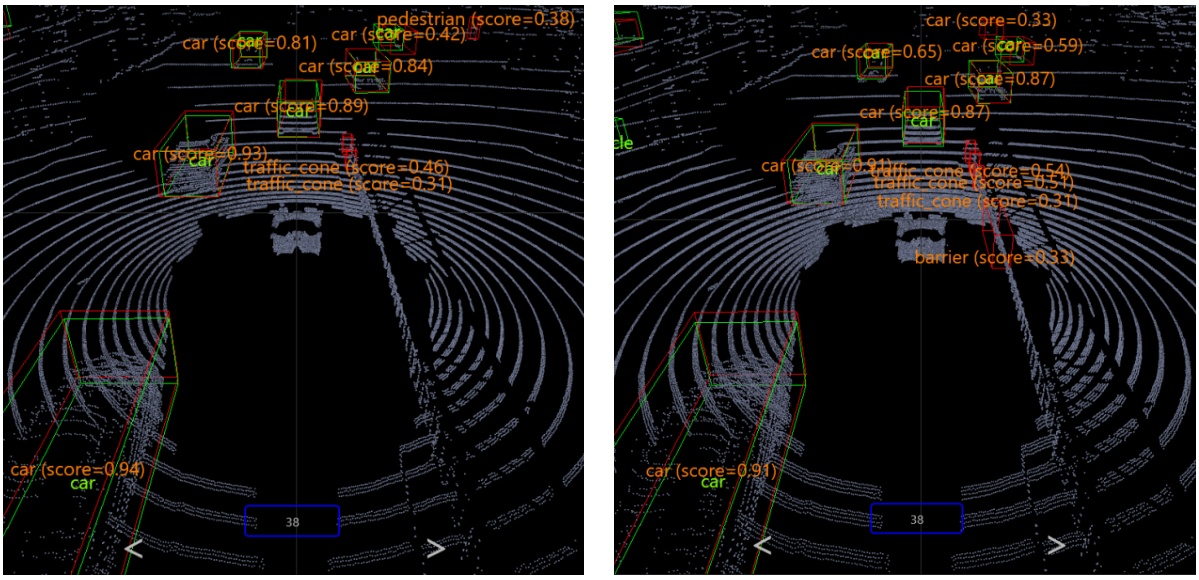
(c) Image related to point cloud 83 from validation set.

Figure 49: Visualiser with different views. Green bounding boxes refer to ground-truth and red bounding boxes refer to LSTM PP predictions. Score refers to the model confidence on the prediction.

In the case of the image (Figure 49c) and BEV (Figure 49b), it is only possible to zoom in and out, while in the case of the interactive view of the point cloud (Figure 49a), it is allowed to navigate through the whole 3D environment and examine it from various viewpoints.

The various views complement each other where the others fall short. Image (Figure 49c) manages to provide information that is more attractive to the sight, assisting us in perceiving the environment of the scene to be assessed and in determining the objects that appear in front of the ego-vehicle. The BEV (Figure 49b) can provides a top-down, 360-degree view of the whole point cloud, making it easy to spot and compare human annotations and model predictions, but these are only visible in two dimensions. The interactive point cloud view (Figure 49a) enables navigation in a 3D world and provides a superior perspective for analysing object points, bounding box dimensions, and comparing ground-truth and predictions.

When we zoom in on the image (Figure 49c), we can see some pedestrians in the distance, although those to the left of the ego-vehicle are not visible. Even though these pedestrians are present on the BEV, it is hardly noticeable if the ground-truth and predictions are coherent. In the interactive view, we can assess how well they match, e.g., the pedestrian closest to the sensor has a well-matched bounding box prediction despite the model's low confidence. On the other hand, it is more difficult for the model to predict pedestrians at a greater distance since they are presented with fewer points, with only one out of the four pedestrians being correctly predicted.



(a) LSTM PP predictions.

(b) PP predictions.

Figure 50: Comparative views for point cloud 38 from validation set, with ground-truth and predictions.

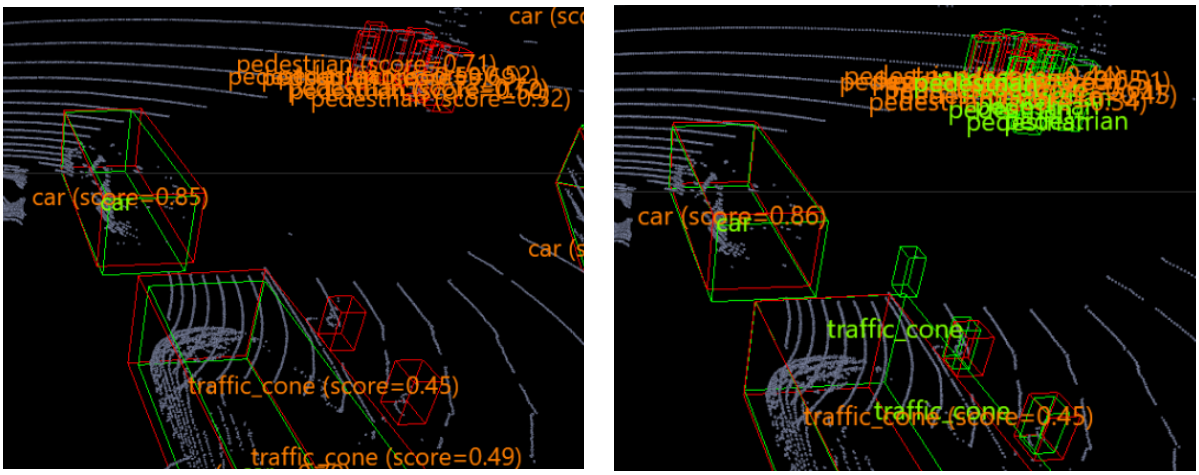
As displayed in Figure 50, when comparing Car class predictions of LSTM PP and PP, very similar results are obtained. The main distinction between the two is found in the prediction score/confidence. Typically, LSTM PP has greater confidence in correct predictions and less confidence in incorrect predictions than PP, which is a good advantage to LSTM PP.

There are fences to the right of the ego-vehicle, however they are not annotated as objects. Due to similarities with barriers and traffic cones, during the training phase, models attempt to predict fences as barriers and traffic cones as there is no fence class. The model should not be penalised for detecting these objects, because in the context of AD, it is advantageous to know there is a fence and the ego-vehicle

should not go against it.

As stated in 4.1, the dataset is equally as important as a good architecture implementation, so it is essential that they have some sort of synergy. Both the data capture and ground-truth annotations may contain errors. Consequently, the model aims to find patterns in the training data in order to reach the point of understanding that, occasionally, a detection identified as incorrect is not necessarily wrong, taking into account all of the extremely similar good examples that have previously been identified as such.

Figure 51 illustrates an instance in which the ground-truth annotations of the dataset are not only incorrect, but also inconsistent between different point clouds from the same sequence.



(a) Point cloud 315 from validation set.

(b) Point cloud 317 from validation set.

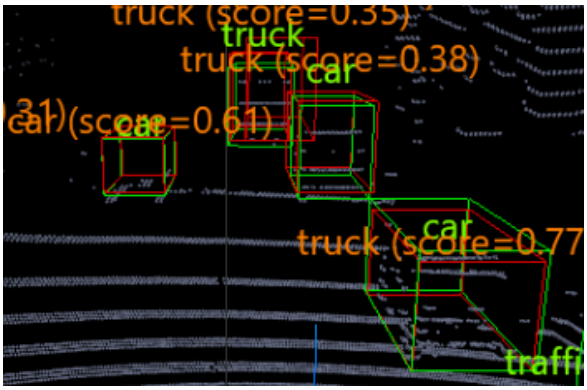
Figure 51: Scene annotation differences from the same sequence.

Figure 51a is a section of point cloud 315, and it can be seen in the upper right that the model detected a set of pedestrians despite the fact that there is no ground truth for these. Moving on to the Figure 51b that represents the point cloud 317, the model's detection is comparable, but this figure already contains ground-truth annotations for the Pedestrian class. The same can be seen in the lower portion of the two figures, where the model detects traffic cones, but the ground-truth is only added in the second one, including a ground-truth annotation (not detected by the model) of an object of the class Traffic Cone composed of only two points. These large inconsistencies between different point clouds can also be generalised to different scenes, thereby this challenge increases the model's difficulty in learning when and how to detect objects.

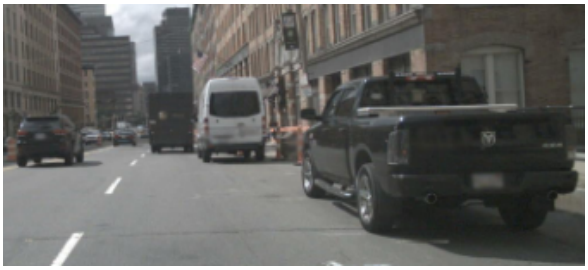
Figure 51 also depicts how well pedestrians are detected by the model. Even though the model can detect their generalised position when they are grouped closely together, it is challenging for it to determine which points correspond to each pedestrian's bounding box. Despite the fact it is difficult to track individual pedestrians when they are grouped together, it is still possible to have a general idea of their location in order to avoid collisions between the ego-vehicle and pedestrians.

Less closely related to dataset errors is the topic of classes that are similar to one another, even after the 23 total classes were grouped into 10 evaluated classes. For instance, the classes Car, Bus, Truck,

Trailer, and Construction Vehicle should have similar features, with only minor differences in attributes such as dimension. Occasionally, it is even difficult for humans to distinguish between these classes.



(a) Point cloud 1194 from validation set.



(b) Image related to point cloud 1194 from validation set.

Figure 52: Similar classes ground-truth and prediction comparison.

Figure 52 captures a case in which four vehicles were identified, three of which are annotated as cars and one as a truck, whereas the model classifies three of them as trucks and one as a car, as indicated in Figure 2 52a. If we add Figure 52b to the equation, we are able to see beyond the points and bounding boxes and analyse the vehicles as our eyes are accustomed to seeing them.

In the image (Figure 52b), although the vehicle on the left appears to be a car and has the dimensions of a car, the three vehicles on the right have distinctive features. The vehicle closest to the sensor has larger dimensions and bigger wheels than a typical car, whereas the car in white has dimensions that are comparable to the truck in front. Even though the objects' annotations are understandable, their classification by the model is also understandable, particularly when the dataset is known having some inconsistencies.

Chapter 6

Conclusion and Future Work

AI is becoming a very important aspect in people's lives, now appearing also in the theme of AD. AVs have been explored by several companies, with the objective of achieving a fast and safe implementation of these systems. These systems take advantage of key tasks such as OD, to create a strong foundation for AV scenarios. It is necessary to use reliable perception techniques, as the safety of these vehicles' users and the ones sharing the environment with, is of high priority. Due to the high fidelity of the LiDAR sensors, they are one of the most suitable for the machine's perception of the environment.

Recent developments in AD and OD were presented in Chapter 1 to justify the motivations and challenges behind the development of this dissertation. An explanation about the main topics referred throughout this work were also presented. It is mentioned and described how the various levels of automation are divided in AD, in addition with the types of perception that these machines may employ to become autonomous. A quick overview of the themes of AI, ML and DL, as well as how they are related, is introduced. Finally, it is discussed the general concept of OD along with the different nuances (i.e., 3D and temporal OD) that are relevant to this work.

SOTA methods related to 3D temporal OD are revealed in Chapter 2. A brief analysis of their pipelines and innovative techniques is carried out, as well as the results they were able to achieve. These methods had different types of focus. For instance, LSTM [20], 3D-MAN [58], PP-REC [32] and PC-TCNN [55] are all temporal models, although PC-TCNN [55] was developed for tracking, and the SFD, SE-SSD, SECOND and PP are single-frame models that use very different approaches. The creation of a temporal algorithm can benefit greatly from all of their uniqueness.

Chapter 3 presents the research project objectives for this work. A work plan is created that includes what has already been achieved as well as what is expected to be addressed in the next phase of the dissertation.

Chapter 4 introduces some fundamental concepts for developing an OD algorithm. Three of the most

important datasets in the field of AD are examined, selecting nuScenes [6] as the chosen dataset due to temporal advantages, where most temporal SOTA solutions are evaluated through its benchmarks. A quick analysis is conducted of the possible representations of LiDAR data, which have a significant impact on the performance of the developed DL models. A possible implementation of a temporal OD pipeline is proposed and explored, as well as several key strategies for reducing the inference time of the temporal algorithm, which tends to be slower due to the larger number of point clouds to be processed. Finally, the most basic metrics are discussed, but also how they contribute to the mAP@IoU and AP@dist evaluation metrics, which are the most commonly utilised in the context of OD.

The development of the temporal 3D OD solution is described in detail in Chapter 5. It describes the hardware and software used to run the training and testing of all models. To further validate the usage of the nuScenes dataset, it is provided additional information, such as its sensor data and available classes. Since PP was used as a baseline during the benchmark, this is explained in greater detail for better comprehension. The developed temporal implementation is then discussed in depth, as it was one of the primary goals of this dissertation, going through its pipeline, data pre-processing, memory bank, and temporal module. The benchmark is conducted by taking into account the values declared by implementations that are significant for the comparison, as well as the values obtained during the development of this work, for both existing and newly developed implementations. It is described how the training and validation of the models were monitored and analysed, followed by a visualisation of the results, with a focus on single frame and temporal models, as well as dataset nuances.

6.1 Contributions

The research and development of this work led to some major contributions to the AD topic:

- Research and analysis of the main datasets on AD, e.g., KITTI [13], nuScenes [6] and *Waymo* [45];
- Research and analysis of relevant SOTA methodologies;
- NuScenes [6] deep analysis with visual quality demonstrations;
- Study and development of three temporal module variants, i.e., Standard RNN, LSTM and GRU;
- Development of an end to end temporal implementation, which achieves promising results and inference times;
- Qualitative and quantitative benchmark of multiple implementations, including single-frame and various temporal variants.

6.2 Risks and Limitations

The research and development of a work like this comes with its set of risks and limitations.

During research, very few successful SOTA temporal implementations as a solution for AD were found. Those who were selected did not provide the developed code or a comprehensive explanation of how each module of the implementation operates.

Aside from the LSTM implementation [20] for the *Waymo* [45] dataset, which did not provide the corresponding code, RNNs or other temporal approaches were not explored in the context of AD using LiDAR.

Very few datasets were ideal for this task of temporal 3D OD using LiDAR. The most popular datasets are KITTI OD [13], which lacks temporal information, *Waymo* [45], which is relatively new as a dataset, resulting in a lack of implementation on it, and nuScenes [6], which focuses more on multi-modal information than temporal information, providing annotations every 500 milliseconds instead of the usual 100ms.

The absence of repositories introduced an element of uncertainty into the possibility of obtaining results. Fortunately, not only was one of the most popular baselines (PP [24]) adapted from KITTI OD [13] to a temporal dataset (nuScenes [6]), but the first adaptation for a temporal solution was also discovered to serve as a benchmark.

nuScenes [6], the selected dataset, is not exempt from having issues, as all datasets do. Several inconsistencies were discovered in the ground-truth annotations, which compromises the quality of any implemented detection methodology.

Due to the annotation features of the dataset, the temporal training time increases by a factor of two to three, thereby stifling and slowing down the rate of development and producing results.

The hardware on which the implementations were developed presented its own challenges. The combination of the increased training time for temporal implementations and the time constraints of each train impacted the progress.

6.3 Next Steps

This study has already produced reliable results for a temporal implementation of 3D OD, thereby enhancing an existing SOTA implementation. However, there is always room for additional investigation into the subject.

Since the implemented temporal module was the primary focus of this study, it was the one that required the most fine-tuning and hyper-parameterisation. Although parameters in other modules were also briefly investigated, care was taken when modifying them so that a comparison with the baseline single frame would remain meaningful. In the future, these other modules can be investigated further to maximise the benefits of the temporal module.

Various other types of temporal modules may also be explored further. In this study, three distinct types of RNNs were examined, however, other RNN-alike architectures and other temporal approaches can be

tested. Transformers [51], which have proven to be quite capable in the context of language processing, are an example of an additional method that can be investigated.

Moreover, the temporal module can be adapted to temporally analyse other types of information. Instead of processing the spatial features, the temporal module can directly receive the unprocessed point cloud to learn the temporal characteristics of the cloud's points.

Bibliography

[1] *API for SemanticKITTI*. https://github.com/PRBonn/semantic-kitti-api/blob/master/README.md. Accessed: 2022-01-24.

[2] Anurag Arnab et al. *ViViT: A Video Vision Transformer*. 2021. doi: 10.48550/ARXIV.2103.15691. url: https://arxiv.org/abs/2103.15691.

[3] *Artificial Intelligence (AI)*. https://www.ibm.com/cloud/learn/what-is-artificial-intelligence. Accessed: 2022-01-24.

[4] *Beyond the pixel plane: sensing and learning in 3D*. https://thegradient.pub/beyond-the-pixel-plane-sensing-and-learning-in-3d/. Accessed: 2022-01-24.

[5] Prarthana Bhattacharyya, Chengjie Huang, and Krzysztof Czarnecki. *SA-Det3D: Self-Attention Based Context-Aware 3D Object Detection*. 2021. doi: 10.48550/ARXIV.2101.02672. url: https://arxiv.org/abs/2101.02672.

[6] Holger Caesar et al. *nuScenes: A multimodal dataset for autonomous driving*. 2020. arXiv: 1903.11027 [cs.LG].

[7] Junyoung Chung et al. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling*. 2014. doi: 10.48550/ARXIV.1412.3555. url: https://arxiv.org/abs/1412.3555.

[8] *Deep learning vs. machine learning – What's the difference?* https://levity.ai/blog/difference-machine-learning-deep-learning. Accessed: 2022-01-24.

[9] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[10] Jiajun Deng et al. "Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection". In: *arXiv:2012.15712* (2020).

[11] Pedro Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. USA: Basic Books, Inc., 2018. isbn: 0465094279.

[12] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. doi: 10.48550/ARXIV.2010.11929. url: https://arxiv.org/abs/2010.11929.

[13] 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*. 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[14] Andreas Geiger et al. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[15] Ross Girshick. *Fast R-CNN*. 2015. doi: 10.48550/ARXIV.1504.08083. url: https://arxiv.org/abs/1504.08083.

[16] *Google's Self-Driving Cars: A Quest for Acceptance*. https://www.popsci.com/cars/article/2013-09/google-self-driving-car/. Accessed: 2022-01-24.

[17] Benjamin Graham. *Spatially-sparse convolutional neural networks*. 2014. doi: 10.48550/ARXIV.1409.6070. url: https://arxiv.org/abs/1409.6070.

[18] Benjamin Graham and Laurens van der Maaten. *Submanifold Sparse Convolutional Networks*. 2017. doi: 10.48550/ARXIV.1706.01307. url: https://arxiv.org/abs/1706.01307.

[19] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[20] Rui Huang et al. *An LSTM Approach to Temporal 3D Object Detection in LiDAR Point Clouds*. 2020. arXiv: 2007.12392 [cs.CV].

[21] *Intersection over Union (IoU) for object detection*. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/. Accessed: 2022-01-24.

[22] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2014. doi: 10.48550/ARXIV.1412.6980. url: https://arxiv.org/abs/1412.6980.

[23] Ankit Laddha et al. *MVFuseNet: Improving End-to-End Object Detection and Motion Forecasting through Multi-View Fusion of LiDAR Data*. 2021. arXiv: 2104.10772 [cs.CV].

[24] Alex H. Lang et al. *PointPillars: Fast Encoders for Object Detection from Point Clouds*. 2019. arXiv: 1812.05784 [cs.LG].

[25] Yiyi Liao, Jun Xie, and Andreas Geiger. *KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D*. 2021. arXiv: 2109.13410 [cs.CV].

[26] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2017. doi: 10.48550/ARXIV.1708.02002. url: https://arxiv.org/abs/1708.02002.

[27] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[28] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. doi: 10.1007/978-3-319-46448-0_2. url: https://doi.org/10.1007%5C%2F978-3-319-46448-0_2.

[29] Yuliang Liu and Lianwen Jin. "Deep Matching Prior Network: Toward Tighter Multi-Oriented Text Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[30] João M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. url: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf.

[31] John McCarthy. "What is Artificial Intelligence?" In: (Jan. 2004).

[32] Scott McCrae and Avideh Zakhor. "3d Object Detection For Autonomous Driving Using Temporal Lidar Data". In: *2020 IEEE International Conference on Image Processing (ICIP)*. 2020, pp. 2661–2665. doi: 10.1109/ICIP40778.2020.9191134.

[33] Warren Mcculloch and Walter Pitts. "A Logical Calculus of Ideas Immanent in Nervous Activity". In: *Bulletin of Mathematical Biophysics* 5 (1943), pp. 127–147.

[34] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA, USA: MIT Press, 1969.

[35] Jiquan Ngiam et al. *StarNet: Targeted Computation for Object Detection in Point Clouds*. 2019. arXiv: 1908.11069 [cs.CV].

[36] Ilias Panagiotopoulos and George Dimitrakopoulos. "An empirical investigation on consumers' intentions towards autonomous driving". In: *Transportation Research Part C: Emerging Technologies* 95 (2018), pp. 773–784. issn: 0968-090X. doi: https://doi.org/10.1016/j.trc.2018.08.013. url: https://www.sciencedirect.com/science/article/pii/S0968090X1830086X.

[37] Kedar Potdar, Chinmay Pai, and Sukrut Akolkar. *A Convolutional Neural Network based Live Object Recognition System as Blind Aid*. Nov. 2018. doi: 10.13140/RG.2.2.34494.54085.

[38] Charles R. Qi et al. *Frustum PointNets for 3D Object Detection from RGB-D Data*. 2017. doi: 10.48550/ARXIV.1711.08488. url: https://arxiv.org/abs/1711.08488.

[39] Charles R. Qi et al. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2016. doi: 10.48550/ARXIV.1612.00593. url: https://arxiv.org/abs/1612.00593.

[40] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning internal representations by error propagation". In: 1986.

[41] *Selecting the Right Bounding Box Using Non-Max Suppression (with implementation)*. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/. Accessed: 2022-01-24.

[42] Shili Sheng et al. *A Case Study of Trust on Autonomous Driving*. 2019. arXiv: 1904.11007 [cs.HC].

[43] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. *PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud*. 2019. arXiv: 1812.04244 [cs.CV].

[44] Kiwoo Shin, Youngwook Paul Kwon, and Masayoshi Tomizuka. *RoarNet: A Robust 3D Object Detection based on RegiOn Approximation Refinement*. 2018. doi: 10.48550/ARXIV.1811.03818. url: https://arxiv.org/abs/1811.03818.

[45] Pei Sun et al. *Scalability in Perception for Autonomous Driving: Waymo Open Dataset*. 2020. arXiv: 1912.04838 [cs.CV].

[46] Pavel Tokmakov, Karteek Alahari, and Cordelia Schmid. *Learning Video Object Segmentation with Visual Memory*. 2017. arXiv: 1704.05737 [cs.CV].

[47] *Tracking 3D LIDAR Point Clouds Using Extended Kalman Filters in KITTI Driving Sequences*. https://www.semanticscholar.org/paper/Tracking-3D-LIDAR-Point-Clouds-Using-Extended-in-Maalej-Sorour/caeccc5f201983486a3b4ffe0584ef1f1b576f38. Accessed: 2022-01-24.

[48] John R Treat et al. *Tri-level study of the causes of traffic accidents: final report. Executive summary.* Tech. rep. Indiana University, Bloomington, Institute for Research in Public Safety, 1979.

[49] A. M. TURING. "I.—COMPUTING MACHINERY AND INTELLIGENCE". In: *Mind* LIX.236 (Oct. 1950), pp. 433–460. issn: 0026-4423. doi: 10.1093/mind/LIX.236.433. eprint: https://academic.oup.com/mind/article-pdf/LIX/236/433/30123314/lix-236-433.pdf. url: https://doi.org/10.1093/mind/LIX.236.433.

[50] Michael Ulrich et al. *Improved Orientation Estimation and Detection with Hybrid Object Detection Networks for Automotive Radar*. 2022. doi: 10.48550/ARXIV.2205.02111. url: https://arxiv.org/abs/2205.02111.

[51] Ashish Vaswani et al. *Attention Is All You Need*. 2017. doi: 10.48550/ARXIV.1706.03762. url: https://arxiv.org/abs/1706.03762.

[52] Sourabh Vora et al. *PointPainting: Sequential Fusion for 3D Object Detection*. 2019. doi: 10.48550/ARXIV.1911.10150. url: https://arxiv.org/abs/1911.10150.

[53] Tai Wang, Xinge Zhu, and Dahua Lin. *Reconfigurable Voxels: A New Representation for LiDAR-Based Point Clouds*. 2020. doi: 10.48550/ARXIV.2004.02724. url: https://arxiv.org/abs/2004.02724.

[54] Xinshuo Weng et al. *3D Multi-Object Tracking: A Baseline and New Evaluation Metrics*. 2020. arXiv: 1907.03961 [cs.CV].

[55] Hai Wu et al. “Tracklet Proposal Network for Multi-Object Tracking on Point Clouds”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*. Ed. by Zhi-Hua Zhou. Main Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 1165–1171. doi: 10.24963/ijcai.2021/161. url: https://doi.org/10.24963/ijcai.2021/161.

[56] Xiaopei Wu et al. *Sparse Fuse Dense: Towards High Quality 3D Detection With Depth Completion*. 2022. url: https://openreview.net/forum?id=SoiF5R9z6zQ.

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

[58] Zetong Yang et al. *3D-MAN: 3D Multi-frame Attention Network for Object Detection*. 2021. arXiv: 2103.16054 [cs.CV].

[59] Wu Zheng et al. *CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud*. 2020. doi: 10.48550/ARXIV.2012.03015. url: https://arxiv.org/abs/2012.03015.

[60] Wu Zheng et al. *SE-SSD: Self-Ensembling Single-Stage Object Detector From Point Cloud*. 2021. arXiv: 2104.09804 [cs.CV].

[61] Yin Zhou and Oncel Tuzel. *VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection*. 2017. doi: 10.48550/ARXIV.1711.06396. url: https://arxiv.org/abs/1711.06396.

[62] Yin Zhou et al. *End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds*. 2019. arXiv: 1910.06528 [cs.CV].