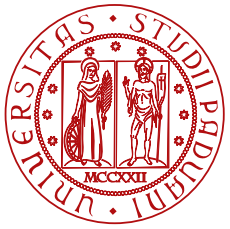

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Department of Information Engineering

Master Degree in ICT for Internet and Multimedia

Design and Evaluation of Data Dissemination Algorithms to Improve Object Detection in Autonomous Driving Networks

Master Candidate

Filippo Cenzi

Student ID 1234361

Supervisor

Prof. Marco Giordani

University of Padova

Co-supervisor

Dr. Paolo Testolina

University of Padova

Academic Year 2022/2023

13 April 2023

To my family

Abstract

In the last few years, the amount of information that is produced by an autonomous vehicle is increasing proportionally with the number and resolution of sensors that cars are equipped with. Cars can be provided with cameras and Light Detection And Ranging (LiDAR) sensors, respectively needed to obtain a two-dimensional (2D) and three-dimensional (3D) representation of the environment. Due to the huge amount of data that multiple self-driving vehicles can push over a communication network, how these data are selected, stored, and sent is crucial. Various techniques have been developed to manage vehicular data; for example, compression can be used to alleviate the burden of data transmission over bandwidth-constrained channels and facilitate real-time communications. However, aggressive levels of compression may corrupt automotive data, and prevent proper detection of critical road objects in the scene. Along these lines, in this thesis, we studied the trade-off between compression efficiency and accuracy. To do so, we considered synthetic automotive data generated from the SEmantic Large-Scale Multimodal Acquisition (SELMA) dataset. Then, we compared the performance of several state-of-the-art algorithms, based on machine learning, for compressing and detecting objects on LiDAR point clouds. We were able to reduce the point cloud by tens to hundreds times without any significant loss in the final detection accuracy. In a second phase, we focused our attention on the optimization of the number and type of sensors that are more meaningful to object detection operations. Notably, we tested our dataset on a sensor fusion algorithm that can combine both 2D and 3D data to have a better understanding of the environment. The results show that, although sensor fusion always achieves more accurate detections, using 3D inputs only can obtain similar results for large objects while mitigating the burden on the channel.

Sommario

Negli ultimi anni, i veicoli a guida autonoma stanno iniziando a produrre e inviare dati in quantità sempre maggiori, questo in proporzione al numero e alla risoluzione dei sensori di cui essi sono provvisti. Le auto possono essere dotate di telecamere, per ottenere una rappresentazione bidimensionale (2D) dell'ambiente, e di Light Detection e Ranging (LiDAR) che permettono invece di averne una versione tridimensionale (3D). A causa dell'enorme quantità di dati che i veicoli a guida autonoma possono immettere nella rete, è fondamentale che i dati vengano selezionati, archiviati ed inviati in modo adeguato. Nel corso degli anni sono state sviluppate varie tecniche per la gestione di dati in ambienti di tipo stradale; ad esempio, in caso di canali con banda limitata, alcuni algoritmi di compressione possono essere utilizzati per diminuire il carico sulla rete e facilitare la trasmissione dei dati in tempo reale. Tuttavia, applicando livelli troppo aggressivi di compressione, i dati potrebbero venire corrotti e di conseguenza impedire il corretto funzionamento di algoritmi per il rilevamento e riconoscimento di oggetti. In questa tesi, abbiamo messo a confronto la quantità di compressione e la accuratezza nel rilevamento di oggetti per trovare il giusto compromesso tra le due metriche. Per farlo, abbiamo utilizzato SEmantic Large-Scale Muiltimodal Acquisition (SELMA), un dataset sintetico dedicato alla guida autonoma. Abbiamo confrontato le prestazioni di diversi algoritmi in letteratura per la compressione e il rilevamento di oggetti su dati LiDAR. Applicando questi metodi, siamo stati in grado di ridurre le dimensioni dei file LiDAR anche di due ordini di grandezza senza riscontrare alcuna perdita significativa nella precisione di rilevamento finale. Nella seconda parte della tesi, ci siamo concentrati sull'ottimizzazione del numero e del tipo di sensori che possono essere di maggior impatto per il riconoscimento di oggetti. In particolare, abbiamo testato i nostri dati con un algoritmo in grado di combinare informazioni 2D e 3D per aumentare la precisione nella compresione della scena. I risultati hanno dimostrato che sebbene l'algoritmo di fusione possa raggiungere rilevamenti più accurati, utilizzando solo input 3D si possono ottenere risultati simili mitigando però il carico sulla rete.

Contents

List of Figures

List of Tables

List of Acronyms

LiDAR Light Detection And Ranging
SELMA SEmantic Large-Scale Multimodal Acquisition
V2V Vehicle to Vehicle
V2X Vehicle to Everything
V2I Vehicle to Infrastructure
HSC Hybrid Point Cloud Semantic Compression
ADS Autonomous Driving Systems
ODD Operational Design Domains
GPS Global Positioning System
IMU Inertial Measurement Unit
2D two-dimensional
3D three-dimensional
3GPP 3rd Generation Partnership Project
MAC Medium Access Control
DSRC Dedicated Short Range Communication
PLCP Physical Layer Convergence Protocol
PMD Physical Medium Access
CSMA/CA Carrier Sense Multiple Access with Collision Avoidance
STDMA Self-Organizing Time Division Multiple Access
BEV Bird's-Eye View
FV Front View
CNN Convolutional Neural Network
AP Average Precision
QP Quantization Parameter
CL Compression Level
SC Semantic Compression level
CARLA CAR Learning to Act
FoV Field of View
UDP User Datagram Protocol
PSNR Peak signal-to-noise ratio
SSD Single Shot Detector
RoI Region of Interest

IoU Intersection over Union

1 Introduction

Nowadays, autonomous cars are equipped with dozen of sensors, especially cameras and Light Detection And Ranging (LiDAR) sensors. The camera allows the vehicle to see the surrounding in a two-dimensional way via multiple sensors that can be pointed in different directions, focusing on the most representative views around the car. Instead, LiDAR sensors, with their three-dimensional representation of the environment, give the vehicle a complete vision of all nearby obstacles, and the relative distance to the vehicle. That is useful for setting up the correct timing to accelerate and brake in every situation. Both sensors are excellent in the matter of object detection: cameras are cheap, easy to install in the vehicle and produce valuable data while maintaining a compact file size; instead, LiDARs are bulkier and expensive, but they can produce a much more precise representation of the environment at the cost of having more complex and large files [1].

Every sensor in the car may produce data constantly during a journey. In the case of multiple cameras and LiDARs, the amount of information pushed through the transmission medium may be difficult to handle with standard communication technologies. In fact, in an autonomous driving environment, cars have to communicate with each other or with road side units, using, for example, the standard communication system employed for Vehicle to Vehicle (V2V) operations, which is IEEE 802.11p, or Vehicle to Infrastructure (V2I) networking, respectively [2]. Proper data selection and compression are crucial to alleviate the burden of data transmission through the channel. For example, if we center our attention on a standard point cloud with a frame rate of 30 fps, the resulting data rate can be over 500 Mbps [3], which is challenging for IEEE 802.11p that can offer a nominal rate up to 27 Mbps [4]. Therefore, given that other types of sensors may need to use the communication medium, the size of LiDAR files need to be reduced. In addition to the pure weight of LiDAR and camera files, we also have to address the problem of selecting the right combination of sensors to avoid the risk of sending redundant or unnecessary information. For instance, if a vehicle is provided with three LiDARs, one in front, one on top, and one in the back, we should consider if

every sensor is helpful to the detection or if we can gather almost the same information from only one. In conclusion, we must ensure we are pushing over the network only valuable information by choosing the right combination of images and point clouds without wasting any bands for unnecessary files [5].

In the first part of the work, we focused on researching the best compromise between detection accuracy and LiDAR file size, trying to find the compression parameters that lead to an optimal trade-off. Over the years, researchers studied and developed various compression algorithms to manage the large size of the point clouds and to fit them for real-time transmission. Some algorithms can reduce the point cloud size up to hundreds of times [6] compared to the raw file dimension. Among all algorithms, a novel method called Hybrid Point Cloud Semantic Compression (HSC) [7] can compress a point cloud up to 700 times to achieve real-time transmission. The algorithm exploits the semantic segmentation offered by RangeNet++ [8] and the compression capability of Draco [9]. Both frameworks are used to reduce the size of the final point cloud. The former is useful to add labels to the objects appearing in the scene, and maintains only valuable items crucial for detection, such as pedestrians and/or vehicles. The latter compresses the resulting point cloud using 11 different compression and 15 quantization levels. Using this compression method, we can reduce the network's burden when sending LiDAR files. However, the accuracy of the object detection is strictly related to the compression level on the point cloud. Therefore, we have to find a compromise between compression efficiency and the effect of this degradation on the quality of object detection.

To evaluate the trade-off between compression accuracy and efficiency over HSC, we trained and tested different object detection algorithms on the SEmantic Large-Scale Multimodal Acquisition (SELMA) dataset [10]. SELMA is a synthetic dataset for autonomous driving, created using an open-source simulator called CAR Learning to Act (CARLA) [11]. Among the datasets in the literature, we decided to use SELMA for its large amount of samples that were captured using several sensors, in different weather and daytime conditions. SELMA provides eight town environments for the acquisition that can be captured in three daytime and nine weather conditions. Moreover, seven cameras and three LiDAR are available to record each scene from different points of view. During our research, we applied HSC to the original SELMA dataset, to generate multiple compressed copies of the dataset based on different compression levels. Subsequently, we trained two state-of-the-art and well-designed three-dimensional (3D) detection algorithms, Pointpillars [12] and PV-RCNN [13], using as input the uncompressed version of the dataset. Then, Pointpillars and PV-RCNN have been tested on the compressed dataset to understand which configuration can achieve the best trade-off between compression accuracy and efficiency.

With our approach, we obtained impressive results in terms of compression without any significant loss in terms of object detection accuracy. Regarding the application of HSC on SELMA, we discovered that some compressed datasets could reach the

same final detection accuracy as the original one. To evaluate the accuracy, we used the Average Precision (AP), which is the most used metric for detection tasks. We reduced the file size by one order of magnitude while maintaining the AP equal to the uncompressed version. Moreover, if we can accept a slight loss in accuracy, we can push the compression even hundreds of times compared to the original dataset. In this way, we can easily fit LiDARs data stream in V2V transmission flows.

In the second part of the thesis, we focused on which sensors are the most important for the final detection and which information is worth to be sent through the communication medium. In fact, some sensors may shared the same view making their data redundant or not crucial for the final 3D object detection.

In the last few years, more advanced 3D detection algorithms started exploiting the advantages of both cameras and LiDARs. In particular, in this thesis, we used BEVFusion [14], an algorithm that performs the detection using both two-dimensional (2D) and 3D data. More specifically, BEVFusion converts images and point clouds features into the same Bird's-Eye View (BEV) domain to make them compatible. Then, they are merged to create a shared BEV feature map that can be used by multiple task-specific algorithms to resolve perception problems. Moreover, we exploit the code provided by the BEVFusion developers, which offers camera-only, LiDAR-only, and camera-LiDAR inputs to perform object detection based on the sensors' availability. Therefore, we trained and tested the algorithms using four cameras and three LiDARs using the SELMA dataset. The cameras are positioned in front, right, left, and back side of the car, allowing almost a 360-degree view. The three LiDARs lie on the top, front right, and front left side of the vehicle. First of all, we had to train the camera-only and LiDAR-only models. Afterward, we trained the BEVFusion model using the weights obtained from the previous two as a base line. The results showed how (and in which form) the accuracy of the object detection evolves as a function of the type (and number) of inputs/sensors.

Specifically, we observed that using only 2D information from cameras is not enough to perform 3D object detection. On the other hand, LiDAR-only and fused models output excellent results. We obtained the best performance using four cameras and the top LiDAR, but the configuration with only the top LiDAR can almost reach the same accuracy. In contrast, adding more LiDAR information does not increase the precision of the detection. Both LiDAR-only and sensor fusion configurations that use three LiDARs as input perform similarly to the counterpart with only one sensor. Finally, we tested the models applying HSC to the LiDAR, discovering that, as demonstrated in the previous part, even with compressed 3D data, we can obtain excellent detection performance. In conclusion, the results suggest that using only 3D inputs gives the best compromise between compression efficiency and detection accuracy. Therefore, the information gathered from the top LiDAR is the one we must prioritize sending through the network when we are in the presence of a constraint channel.

The thesis is organized as follows. The first chapter provides a general view of autonomous driving, discussing vehicular communication networks and sensors for perception tasks. It also describes related works regarding data compression and object detection. The second chapter is dedicated to explaining the tools used in the thesis. In particular, the structure of the SELMA dataset, compression algorithms, and 3D object detection models. In the third chapter, we describe how we combined the different tools to train and test the dataset and its compressed versions. The fourth chapter describes the results obtained in terms of a trade-off between compression and detection accuracy. The final chapter summarizes the results achieved and describes possible future works.

2 Background

The evolution of autonomous driving is necessary to lead our vehicular and transportation environment to be a safe and efficient space for drivers and pedestrians. Recent studies have proved that human errors cause 94% of road accidents [1], incentivizing researchers and companies to move faster in the field of assisted and automated driving. Autonomous Driving Systems (ADS) aim to improve mobility quality by preventing traffic accidents, mitigating traffic congestion, and reducing emissions. The growth of new computer vision technologies that exploit advanced deep learning techniques and the availability of new refined sensor modalities have enabled ADS to evolve fast in the last few years. In a future vehicular environment, vehicles have to share the information they gain from their sensors in a fast and reliable manner to ensure an efficient and secure transportation system. Therefore, vehicles must be equipped with tools that can select which information is crucial and how and when sent it through the network. This chapter briefly describes how ADS are structured. Then, we discuss about vehicular communication systems and how they impact environment sensing. Next, we have a section dedicated to the two most crucial sensors regarding vehicular object detection: cameras and LiDARs. The final section describes some related works.

2.1 Autonomous driving

In general, an autonomous car is a vehicle capable of sensing the road environment and performing every journey task without human intervention. Although, as reported in [1], we can distinguish the level of autonomous driving into five categories. First, we can define as a level zero vehicle, the one without any automation. Then we categorize as follows:

- Level one: is a primitive level of automation, like adaptive cruise control or anti-lock braking.
- Level two: partial automation that may comprehend emergency braking or colli-

sion avoidance.

- Level three: conditional automation, where the driver can focus on other tasks but must be ready in an emergency. This level can operate only in Operational Design Domains (ODD), such as highways.
- Level four: Human intervention is not needed from this level and above, but vehicles of this category can operate only in ODDs with a specific and known infrastructure.
- Level five: is a fully automated system that can operate in any condition and environment.

Another classification we can do on autonomous vehicles is on their architecture, both in algorithmic design and connectivity types [1].

The algorithmic design is divided into two main categories: modular systems and end-to-end driving. A modular system comprises various sensory inputs, and an actuator gathers information from them to actuate an output. For example, modules can be localization and mapping, communication, perception, vehicle control, and human-machine interface. In contrast, end-to-end driving means the creation of ego-motion by a direct perception of the surrounding. In this approach, the vehicle exploits advanced deep learning techniques to generate a continuous flow of operation to be accomplished based on the sensory inputs.

The connectivity classification can be divided into ego-only vehicles and connected systems [1]. The first one outline all the vehicles equipped with every sensor and computational capability to guarantee their self-traveling without any intervention. The second type of classification describes a full ADS, where the road environment is filled up with sensors, placed both in vehicles or in static or dynamic objects in the surrounding, that can communicate with each other. Even though ego-only vehicles can offer safety on the roads, more work is needed to improve the system's efficiency. Therefore, researchers are putting more effort into creating an efficient connected system, which can improve even more safety, but it can also enhance any other transportation issue, like traffic congestion or energy consumption.

Autonomous vehicles are equipped with many sensors that can help to accomplish the two main tasks for a safe and efficient journey: localization and environment perception.

For localization, sensors like Global Positioning System (GPS) and Inertial Measurement Unit (IMU) can be advantageous in accomplishing the tasks. In addition, specific algorithms exploit other sensors like cameras and LiDARs to solve localization problems. For example, landmark search exploits the presence of landmarks on the road to localize the vehicle by point cloud matching, which compares a small portion of points around the vehicle with a priori point cloud map.

Just as important, environment perception means extracting crucial information from the surrounding that the vehicle uses to proceed safely on the road. Cameras

and LiDARs are the most common sensors, and various algorithms extract information from them to achieve object detection or semantic segmentation.

In conclusion, the vehicle exploits all these information to plan the route and make decisions during its journey. It creates a global plan to be able to reach its final destination efficiently. At the same time, locally, it tries to pay attention to the environment, for example, finding suitable trajectories to avoid obstacles.

2.2 Vehicular communication systems

In the last few years, researchers and automotive companies are investing a lot of funds and resources in Vehicle to Everything (V2X) communications. A reliable and robust transmission system is a fundamental piece in the automotive field, making roads a safer place to travel and improving the efficiency of the entire transportation system. Nowadays, the standard for vehicular transmission services is IEEE 802.11p, which enables robust and reliable communication. But on the other hand, the evolution of autonomous vehicles has moved the focus not only on latency and reliability but also on increasing throughput demand. This need for very powerful connections comes from the massive amount of data that sensors equipped on cars can produce. LiDARs, cameras, radars, and GPSs are just some of the sensors whose data may saturate an IEEE 802.11p link. Therefore, more recent studies have proposed using mmWave connections for vehicular transmission in combination with other wireless systems. Communication networks based on mmWave, with large bandwidths, complex antenna arrays, and directional beamforming, allow high-capacity channels, which fit well for V2V networking.

V2V communications are crucial in the evolution of road transportation utilities, allowing cars to handle in autonomy the traffic and safety management. Moreover, the 3rd Generation Partnership Project (3GPP) has named four main objectives for the next generation of vehicular systems, explained in the following lines.

Vehicle platooning is intended as a service used to virtually bind vehicles, allowing them to move simultaneously, even at high speed, on the highways. For these applications, connections like IEEE 802.11p fits well due to their latency and reliability efficiency.

Advanced driving allows vehicles to coordinate with each other for better traveling and traffic handling. In addition, cars can move autonomously by exchanging small messages with each other, also maintaining low latency to handle unpredictable events.

Extended-sensor enables the perception of a whole environment through the exchange of data gathered, for example, from cars sensors, pedestrians devices, or other tools spread along the vehicle path. Therefore, mmWave communications systems fit well for these types of data exchange due to the high throughput demand.

Remote driving enables remote control over vehicles located in dangerous places

or, for example, allows people unable to drive to be guided by a remote driver.

In conclusion, the vehicular environment evolves rapidly, and various communication features are needed to handle multiple types of network traffic. For example, IEEE 802.11p presents the characteristics to provide reliable and robust connections to deal with critical situations. At the same time, mmWave networks can offer high throughput channels to serve systems with high levels of data production.

2.2.1 IEEE 802.11p communications

The IEEE 802.11p standard supports the Physical (PHY) and Medium Access Control (MAC) layers of the Dedicated Short Range Communication (DSRC) transmission service. It can offer V2V data exchange at a nominal rate from 6 to 27 Mbps within a few hundred meters range. This standard uses the 5.9 GHz band with channel spacing of 20MHz, 10MHz, and 5MHz. IEEE 802.11p operates on about nine channels, with three dedicated respectively to security purposes, to avoid congestion, and to control the transmission broadcast and link establishment. The remaining six channels are allocated for bidirectional communication between different types of units.

The physical layer is responsible for hardware specification, bits conversion, signal coding, and data formatting. It is divided into two main sublayers: Physical Layer Convergence Protocol (PLCP), which handles the communications with the MAC layer, and Physical Medium Access (PMD), which is the interface to the physical transmission medium.

At the MAC layer, the purpose of the standard is to reduce the overhead needed to set up the transmission by removing the set of multiple handshakes that the standard IEEE 802.11 needs. An early version of 802.11p implements Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) at the MAC layer [15] to handle access to the wireless channel. However, in [15], researchers demonstrate that in the presence of high-density road traffic is more efficient to use Self-Organizing Time Division Multiple Access (STDMA) to guarantee real-time data traffic.

In general, IEEE 802.11p is an excellent standard regarding V2V transmission, but it loses some effectiveness when the communication needs a high-throughput channel.

2.2.2 Millimeter Wave communications

The new generation of cellular communication systems exploits bands between 10 GHz to 300 GHz, the so-called mmWave bands. These frequencies can also be suitable for future vehicular communications systems due to their ability to offer high bitrate. However, mmWave signals have a limited communication range, cannot pass through most solid materials, and may suffer a significant doppler spread. All these issues are common in a vehicular environment, and V2V transmissions try to overcome them by applying some ad-hoc practices.

An intelligent base station placement is necessary to address the short communication range. Channel characterization can help in the decision for the correct placement to be able to serve simultaneously as many vehicles as possible. Moreover, the vehicle antenna position has to follow some parameters to guarantee adequate reception and transmission, avoiding as many obstacles as possible. The vehicular environment is changing fast and needs a quick response between two communicating objects. Therefore, an optimal routing protocol design is required to avoid useless hops during transmission. MmWave technologies can exploit the antenna beamforming to overcome path loss. A precise beam alignment between the receiver and the transmitter can help strengthen the connection but also introduce overhead.

In conclusion, mmWaves are also suitable for vehicular communication thanks to their high throughput and, combined with the reliability of IEEE 802.11p, can make V2V communications more flexible.

2.3 Sensors on autonomous vehicles

Modern vehicles provide a considerable amount of information from all the sensors installed. Specifically, cameras and LiDARs output crucial data to guarantee a safe and comfortable journey. They can be exploited to achieve simple tasks like trajectory and parking assistance or more difficult ones like object detection or semantic segmentation.

Cameras can perceive the environment in a two-dimensional way enabling the usage of 2D computer vision algorithms for detection and segmentation tasks. In addition, cameras can also sense colors from the environment, which can be fundamental for vehicular tasks allowing, for example, traffic light recognition. Moreover, cameras are passive sensors, i.e., they do not emit any signal for measurements; hence they do not interfere with other systems. However, cameras may lose precision if weather and illumination conditions are bad, and obtaining depth information from camera frames

Figure 2.1: Example of camera image from SELMA dataset.

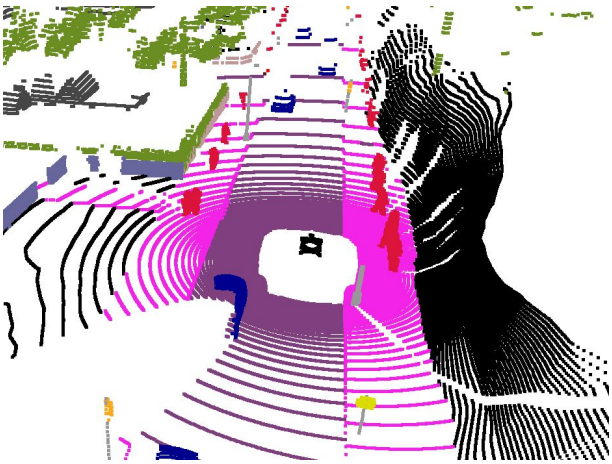

Figure 2.2: Example of LiDAR point cloud from SELMA dataset.

can be computationally demanding. Sensors like LiDARs can resolve the difficulties that cameras may encounter. They give a three-dimensional representation of the surrounding, allowing easy estimation of objects' distances. A LiDAR sensor emits infrared light waves to sense the environment, so it can be defined as an active sensor, thus, may encounter and create interference. Weather conditions such as fog or snow negatively impact the LiDAR's performance. Moreover, compared to radars, LiDARs work at a shorter distance but are more accurate. Figures 2.1 and 2.2 show two examples of cameras and LiDARs outputs.

In conclusion, both cameras and LiDARs have their pros and cons. Some advanced algorithms can exploit information from both types of sensors to achieve specific tasks.

2.4 Related works

In recent years, the increasing interest in the autonomous vehicles field has moved a lot in developing the technologies needed for this purpose. In the following paragraphs, we briefly describe the state of art technologies and tools behind the development of autonomous driving.

Compression of 3D data

In the last few years, the majority of the most precise vehicular detection algorithms have taken advantage of the characteristics offered by 3D data representation of the environment. LiDAR scans offer an accurate surrounding description but at the cost of storing large files. Therefore, many works have contributed to researching an excellent 3D point cloud compression approach. Previous works, such as [16] and [17], tried to project the point cloud into a 2D representation, but, apart from the real-time performances, on a full 3D representation of the environment, they are not really efficient. In other works, such as [18], the authors exploit the deep learning potential to extract features from data, but they don't deal very well with sparsity and the large dimension of the point cloud. Methods based on tree structures can hierarchically organize data and deal with sparsity, as is the case for example of [19]. HSC [7] exploits tree structures along with semantic segmentation to obtain a powerful and fast compressor, which also gives many choices for the compression level.

3D object detection

In the field of object detection, many works tried to deploy an efficient method to detect objects in a 3D environment. Due to the high cost of LiDAR sensors, researchers have spent significant effort on camera models to achieve 3D object detection. For example, FCOS3D [20] distributes the objects to different feature levels based on their 2D scales, and then regression targets are assigned according to the projected 3D center.

Inspired by the design of LiDAR-based detectors, BEVDet [21] and LSS [22] convert the camera features into a BEV to apply similar algorithms to the final feature map. CaDDN [23] improves the view transformer estimation by explicitly adding depth estimation supervision. In general, camera data is easy to access and use, but the lack of a third dimension makes predicting distances and sizes more challenging.

In contrast, LiDAR sensors allow a more accurate representation of the surrounding, but they are more challenging to manage. Some works like [24] and [25] have tried to use a 3D convolutional network, but it is inefficient in terms of computational speed. Another idea was to organize point clouds in voxels and encode the figure as a pseudo-image. This kind of architecture can be seen in works such as [26], [27], and [28]. In some recent works such as VoxelNet [29] and SECOND [30], they exploit PointNet [31] [18] to apply it to voxels which are then processed by 3D convolutional layers, a 2D backbone, and a detection head. One of the most efficient algorithm, exploiting the voxel organization paradigm, is Pointpillars [12], which obtains excellent detection performances while inferring at 62 Hz. Other researches, like PointFusion [32] and PointRCNN [13], apply methodologies that rely on points' characteristics, where the algorithms focus on extracting intrinsic features from single points. In this thesis, we decided to use PV-RCNN [13], an advanced detection algorithm that exploits both the characteristics of voxel structures and points feature extraction.

Most recent approaches started to exploit the characteristics contained in both 2D images and 3D point clouds by fusing them to obtain a whole representation of the environment. For example, we used BEVFusion [14], which extracts the features into a BEV from both images and point clouds, and fuses sensors together to improve detection accuracy.

Datasets for autonomous driving development

Almost every 3D detection algorithm needs a dataset where to train, provided with bounding boxes, each labeled with a specific class. Over the years, companies and universities created various datasets, which help researchers develop efficient algorithms and challenge each other to do better and better. The earliest and most popular 3D dataset is KITTI [33], created by Karlsruhe Institute of Technology and Toyota Inc. KITTI was captured in Karlsruhe urban environment, and it provides a set of 360 degrees LiDAR scans, frontal images, and localizations. In addition, each frame contains both 2D and 3D annotations. Other projects, like nuScenes [34] and Waymo [35], aim to expand the number of samples and sensor availability. NuScenes was the first to provide 360-degree sensor coverage and is rich in annotations. Waymo has similar characteristics to nuScenes, but improved the annotation frequency and sensors' field of view. In recent works, synthetic datasets have been developed due to their simple and low-cost building procedure. Also, they can easily be rich in terms of samples and annotations. In this work, we used SELMA [10], which is a dataset

gathered from a simulated environment. It contains a wide variety of sensors, samples, and environmental conditions to permit detection algorithms to have a comprehensive training set.

3 Tools

This chapter will describe the tools we used and combined to perform our trade-off evaluation. In particular, we will discuss the SELMA dataset and its characteristics, explaining why we choose it among all other datasets. Then, the chapter continues with a description of HSC and its features and how it is able to combine them to achieve excellent point cloud compression. Follows a general view of object detection algorithms and, in particular, why 3D object detection is crucial for vehicular environments. The chapter ends with a more detailed description of Pointpillars and PV-RCNN, which are LiDAR-based detection algorithms, and BEVFusion that fuses 2D and 3D data to perform perception tasks.

3.1 SELMA dataset

The SEmantic Large-Scale Multimodal Acquisition (SELMA) dataset [10] is a multimodal synthetic dataset created using the CARLA simulator. The goal of the dataset is to have large and diversified sets of vehicular environments captured by a consistent number of sensors to ensure good design, prototyping, and validation of autonomous driving models. The advantages of having a synthetic dataset lie in the ease of creating a considerable amount of data with desirable configurations without the cost of providing ground truths like semantic labels and bounding boxes.

There are a lot of state-of-the-art synthetic datasets in the literature, like GTA5 [36] or SYNTHIA [37]. However, semantic understanding models trained on these synthetic datasets may suffer when they are applied to real world scenarios. In contrast, SELMA stands out for its excellent performance achieved in semantic perception tasks in both domains. Even if models are trained over synthetic SELMA data, they can achieve similar performances when tested in a real environment. These results are possible thanks to its variety of sensors and environmental conditions, which make the dataset really useful for autonomous driving training purposes.

Moreover, the sensors available for the acquisition are:

- 7 RGB cameras, with a resolution of 5120x2560, downsampled to 1280x640 pixels and a Field of View (FoV) of 90 degrees. The downsample is needed to achieve a ×4 anti-aliasing enhancement.
- 7 depth cameras with a resolution of 1280x640 and a FoV of 90 degrees.
- 7 semantic cameras with a resolution of 1280x640 and a FoV of 90 degrees.
- 3 semantic LiDARs with 64 vertical channels, generating 100000 points per second, with a range of 100 meters.

Figure 3.1 shows the positioning of the sensors on the simulation car.

SELMA also offers 27 different environmental conditions offered by the combination of 3 daytime which are Noon, Sunset, and Night and 9 whether conditions which are Clear, Cloudy, Wet, Wet and Cloudy, Mid Fog, Hard Fog, Soft Rain, Mid Rain, and Hard Rain. Some examples are shown in the figure 3.2.

The final dataset comprises 30909 samples, acquired across eight virtually generated towns, placed in specific preselected locations called waypoints. These waypoints lie at 4 meters from one another, and at each position, the dynamic objects of the road, like cars, pedestrians, trucks, and bicycles, are placed randomly in the scene.

In conclusion, the SELMA dataset fits perfectly for the work of this thesis due to its ease of access, variety of scenes, variety and quantity of sensors, and high compatibility for real-world testing.

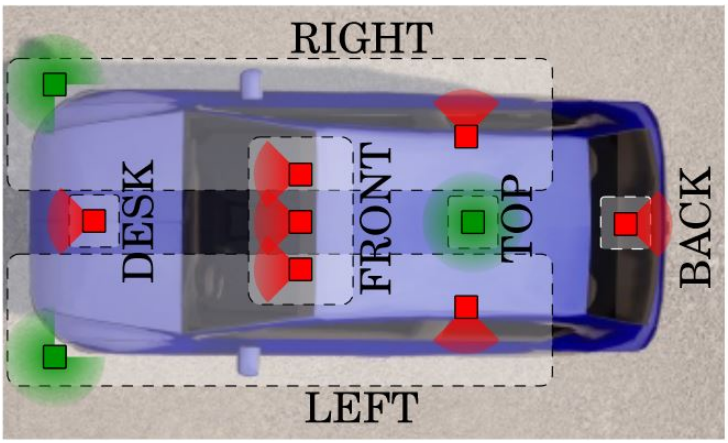

Figure 3.1: Sensors positioning on the simulation car. Green sensors are LiDARs and red ones are cameras.

Figure 3.2: SELMA scenarios examples. Columns are in order: Sunset, Noon, and Night, while rows are: Clear, Cloudy, Wet, Hard Fog, and Mid Rain.

3.2 Hybrid Point Cloud Semantic Compression

Hybrid Point Cloud Semantic Compression (HSC) [7] is a novel algorithm aiming to improve compression efficiency for 3D LiDAR point clouds by exploiting two tools. The first one is RangeNet++ [8], a convolutional neural network capable of obtaining a fast and efficient semantic segmentation of the point cloud. The second one is Draco [9], an algorithm developed by Google to provide compression at very high speed using KD-tree data structure to organize 3D data. These frameworks are combined to achieve real-time compression performance with limited accuracy degradation. In particular, HSC removes not crucial points from the point cloud based on the labels computed by RangeNet++. Then, it applies Draco compression to the point cloud using specific compression parameters. The final result is impressive for both file size reduction and encoding/decoding speed.

RangeNet++

RangeNet++ [8] aims to achieve fast and accurate semantic segmentation by using a projection-based 2D Convolutional Neural Network (CNN) to process the input point

clouds. The approach consists of four main steps:

- Transformation of the input LiDAR scan into a so-called range image, which corresponds to a spherical representation of the point cloud;
- Applying a 2D fully convolutional semantic segmentation: an encoder does considerable downsampling, followed by an upsampling operation performed by the decoder. During the upsampling, the decoder also adds an output stride of the encoder to recover some high-frequency information. After the encoding-decoding procedure, the algorithm performs a set of 1x1 convolutions, and a softmax function processes the output;
- The method performs a semantic transfer from 2D to 3D to recover the original point cloud information;
- Finally, a 3D post-processing procedure obtains an efficient range image by processing the blurry outputs provided by the 2D semantic segmentation. Then, a GPU-based KNN search operates directly on the input point cloud to clean the result. This final operation allows finding the closest k points in the 3D scan of a given point.

Draco

Draco [9] is a software designed by Google to compress 3D data at very high speed while dealing with sparsity. The algorithm splits the point cloud from the center, alternating the axes each round to have a more balanced space-partitioning tree. At each iteration, the encoder takes the number of points in the smaller half, subtracts it from the total number of points, and divides it by two, obtaining the current encoded version of the number of points. Draco allows compressing point clouds with 15 levels of quantization and 11 levels of compression. Compression accuracy and quantization errors have a proportional behavior: if the quantization level grows, the compression quality increases, but at the cost of larger files, whereas higher quantization levels mean more compression but lower decompression speed and quality.

Implementation

As a result, we can obtain a powerful compression algorithm with the combination of the two previous frameworks. Furthermore, we can obtain up to 495 different compression levels by changing the parameters. In fact, the algorithm has 3 degrees of freedom regarding the semantic segmentation and 15 and 11, respectively, for quantization and compression. The three semantic filtering levels are:

- HSC-0: the algorithm does not use RangeNet++, so the original point cloud remains unaltered.
- HSC-1: after the segmentation, static objects are removed from the scene and deleted from the point cloud.
- HSC-2: with the last level, the only elements that remain in the point cloud are

the dynamic and important ones, such as pedestrians and vehicles. So, all the less critical elements that cameras can also provide are removed from the 3D scene.

Draco takes in input the output of the semantic segmentation filtering and compresses the point cloud by tuning two parameters: the quantization and the compression level. Beyond the compression levels, these two parameters are also important for the compression and decompression speed. The more compression we ask to the algorithm, the more time is needed to encode and decode the point cloud.

Compression efficiency and accuracy

In general, the standard communication methods can not handle transmitting raw LiDAR point clouds over the network. Even if we limit the transmission sampling to 10 fps, the traffic generated can be ten times bigger than the standard nominal rate. HSC can compress the point cloud up to 700 times [7] with tolerable accuracy degradation, allowing it to reach real-time transmissions over a V2V network. Furthermore, using HSC-2, the segmentation filtering allows encapsulating the LiDAR scan in a single User Datagram Protocol (UDP) packet, which is crucial for emergencies or time-critical situations. HSC also performs well in terms of encoding and decoding times, completing the compression and decompression in less than 75 ms, even for the most aggressive configurations. To conclude, looking at the Peak signal-to-noise ratio (PSNR), HSC can always maintain it above 50 dB, which is typically an acceptable value for wireless transmission quality loss.

3.3 Object detection algorithms

Object detection is a computer vision task whose objective is to detect instances of particular objects of a specific class on a 2D or 3D digital representation of the environment. Object detection algorithms aim at two specific goals: the first includes classification and localization accuracy, while the second one focuses on the detection speed. With these two goals in mind, researchers have developed algorithms that apply their influence in many fields, such as robotics, transportation, security, etc. Early studies implemented algorithms based on various computer vision techniques, helping to move the first steps for object recognition over 2D images. However, with the fast growth of machine learning and deep learning techniques, object detection has evolved rapidly and become an integral part of many computer vision tasks.

In recent years, easier access to more powerful computational capabilities enables object recognition over more complex and heavier data types, like LiDAR or radar scans. Therefore, the progression of 3D object detection algorithms on recognition tasks significantly improves specific services that need a deep understanding of the environment, such as self-driving cars.

In this thesis, we focus primarily on 3D object detection algorithms to exploit the capabilities and precision that sensors like LiDARs may offer to the final understanding of the surrounding. We can mainly identify three different classes of 3D object detection algorithms [38]:

- Projection-based methods: which project 3D point clouds over a 2D plane, enabling well-established 2D Convolutional Neural Networks to be used even for 3D tasks. There are two subcategories for projection-based methods: Front View (FV)-based methods, where the 2D map is obtained by a cylindrical projection of the point cloud, and BEV-based methods in which the whole point cloud is projected over the ground plane.
- Voxel-based methods: which discretize the whole 3D space into a fixed-size voxel grid, in which each voxel contains some point to make the irregular form of point clouds more structured. Voxelization may help to maintain the 3D shape of objects, but what is challenging is the sparsity of point clouds.
- Point-based methods: which preserve the point cloud's natural geometry without rearranging its spatial characteristics like the previous two methods. These kinds of methods extract local and global features of the points to be able to process the point cloud directly.

In this thesis, we tested our dataset using three 3D object detection algorithms: Pointpillars [12], which belongs to the voxel-based class, and PV-RCNN [13], which fuses the advantages of voxel-based and point-based methods, and BEVFusion [14] as a point-based detection algorithm. The following sections outline how these algorithms work.

3.3.1 Pointpillars

Pointpillars [12] is a 3D object detection algorithm aiming to fast and accurately detect objects in LiDAR scans by learning features on pillars. By pillars, we mean vertical structures that contain points of the 3D scan that have the purpose of regrouping information to make the irregular shape of the point cloud easier to understand for the machine.

Pointpillars structure consists of three main phases: an encoder that gathers features information from the point cloud to convert it into a pseudo-image, a 2D convolutional backbone to process the pseudo-image and a detection head to obtain the 3D bounding boxes. See figure 3.3.

The first step is crucial because to apply a 2D CNN we have to flatten the 3D information coming from the point cloud. In the beginning, the algorithm discretizes the point cloud into a set of pillars in the x-y plane, obtaining for each pillar five parameters (xc, yc, zc, xp and yp), three describing the distance from the arithmetic mean of all points of the pillar and two denoting the offset from the pillar center.Then, all the information gathered from the previous step are saved into a dense tensor of size (D, P, N), where D is the dimension of each augmented point (composed by its intrinsic characteristics x, y, z, and reflectance r, and its pillar information), P is the number of non-empty pillars, and N is the number of points per pillar. In the following step, the algorithm applies a simplified version of PointNet [31] to obtain a sized tensor (C, P, N), where C is a high-level feature representation of D, followed by a maximization operation which gives in output a tensor of size (C, P). In the last step, features are scattered back to the original pillar location to create a pseudo image of size (C, H, W), where H and W denote the height and width of the canvas.

The backbone structure has two subnetworks: one to downsize the pseudo-image into a smaller resolution and one to upsample the features. The top-down part comprises a sequence of blocks (S, L, F), each operating at stride S, having L 3x3 2D convolution layers with F output channels. Next, the F features are upsampled using a transposed 2D convolution, and the final output is a concatenation of the outcome of each stride.

In the final step, Single Shot Detector (SSD) matches the priorboxes to the ground truth using 2D Intersection over Union. Finally, bounding box height and elevation are used as additional regression targets to detect 3D boxes.

Pointpillar has been evaluated on the KITTI dataset, obtaining excellent performances in both BEV and 3D detection while running at 62 Hz, reaching a good tradeoff between fast encoding and accuracy on the detection.

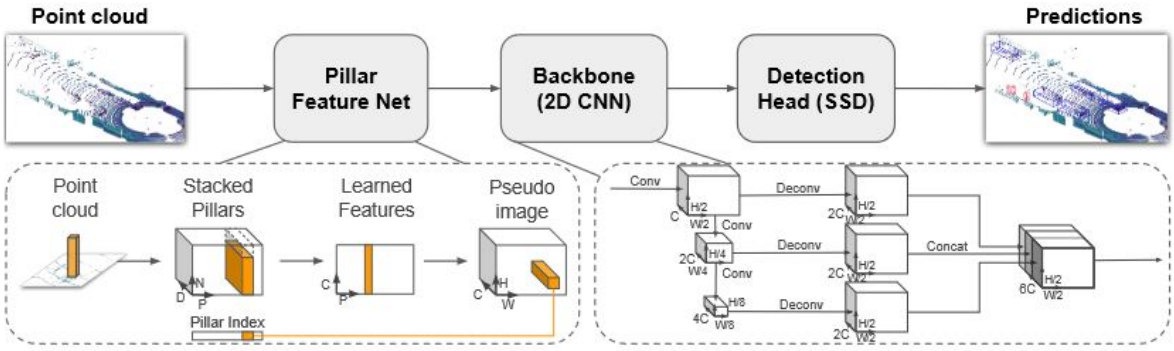

Figure 3.3: Pointpillars network structure [12].

3.3.2 PV-RCNN

PointVoxel-RCNN (PV-RCNN) [13] is a novel framework that enables high performances for the 3D object detection task by exploiting both the functionality of point-based and voxel-based algorithms. Moreover, voxel-based methods help with the discretization of the point cloud, allowing to obtain a straightforward representation to be processed by CNNs. In contrast, point-based approaches enable the preservation of point location, mitigating the loss of information carried by voxelization of the point cloud.

PV-RCNN can efficiently integrate the advantages of the two approaches by introducing two novel operations: the voxel-to-keypoint scene encoding and the point-to-grip Region of Interest (RoI) feature abstraction.

Firstly, the algorithm divides the input points into voxels of size (L, W, H) and calculates the voxel feature as the mean of point-wise features of all points inside it. Then, the framework applies a 3 x 3 x 3 3D sparse convolution to downsample the point cloud until a scale of 8x. Finally, to obtain a 2D BEV, the features of the 8x downsampled vector are stacked along the Z axis to obtain an L/8 x W/8 bird's-eye feature map.

PV-RCNN presents voxel-to-keypoint scene encoding that is useful to aggregate voxel at different neural network layers into a smaller number of keypoints. First, the Furthest-Point-Sampling algorithm subsamples the point cloud obtaining a small set of n keypoints uniformly distributed around non-empty voxels. Next, a Voxel Set Abstraction module aggregates the voxel-wise features into a final small set of keypoints, which are weighted, considering that foreground objects should contribute more than background ones.

In the final portion of the framework, each 3D proposal (RoI) generated by the 3D voxel CNN is aggregated to the weighted keypoint features to obtain an accurate proposal refinement. Therefore, a keypoint-to-grip RoI feature abstraction performs a set of abstraction operations to aggregate keypoint features to the 6 x 6 x 6 grid obtained within each 3D proposal. Finally, the output of each grid is vectorized and transformed by a two-layer multi-layer perceptron, obtaining the overall proposal.

In conclusion, the confidence prediction is performed by adopting the 3D Intersection

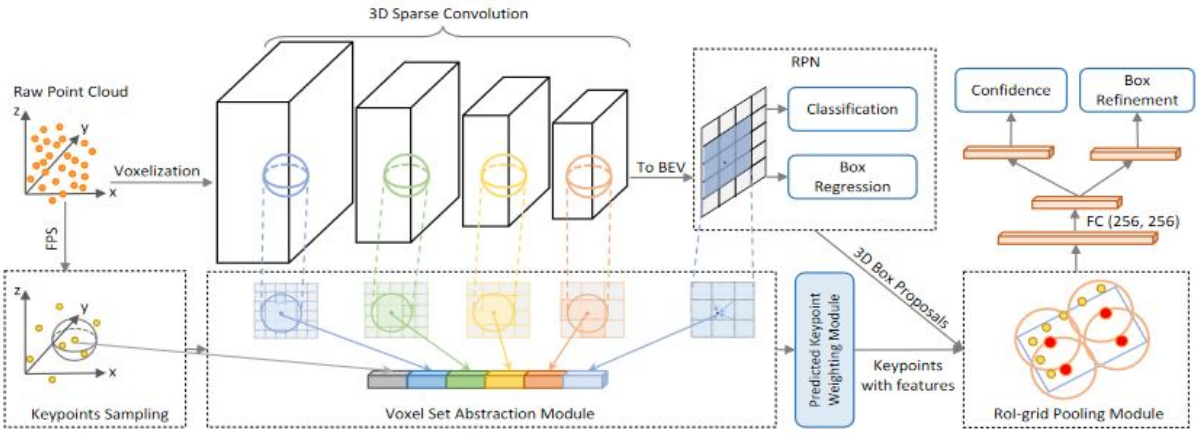

Figure 3.4: PV-RCNN network structure [13].

over Union (IoU) between RoIs and ground-truths, while the final box refinement is obtained using the traditional residual-based method and optimized by the smooth-L1 loss function.

PV-RCNN has been validated over both KITTI and Waymo datasets obtaining a significant improvement compared to the previous state-of-art algorithms.

3.4 BEVFusion

BEVFusion [14] is a multi-sensors and multi-modal features detection algorithm that works in a shared BEV. It uses 2D and 3D information from cameras and LiDARs to convert all features into a common BEV space while maintaining both geometric and semantic density. Moreover, it helps to support the results of most perception tasks since the output of many algorithms is naturally obtained in BEV view.

The algorithm is divided into three main phases: the features extraction into the BEV map, the fully-convolutional fusion, and the multi-task heads application as shown in figure 3.5.

The BEVFusion developers decided to use a BEV view because it is an efficient way to preserve geometric and semantic information from both cameras and LiDARs. Specifically, LiDAR-to-BEV projection flattens LiDAR features along the height dimension while maintaining the geometric proportions without distortions. Instead, camera-to-BEV traces every pixel into the 3D space, which leads to a dense BEV representation that retains complete semantic information from the cameras. Researchers of BEVFusion discover the camera-to-BEV transformation to be the bottleneck for the entire model runtime. Algorithms like LSS [22] and BEVDet [21] explicitly predict the discrete depth distribution of each pixel. The resulting camera feature point cloud has a size of NHWD, where N is the number of cameras, (H,W) is the camera feature map size, and D is the number of discrete depths. As a result, the BEV pooling is really inefficient and slow and could create a camera feature map two orders of magnitude denser than a LiDAR feature point cloud. Therefore, the BEVFusion solution adopts

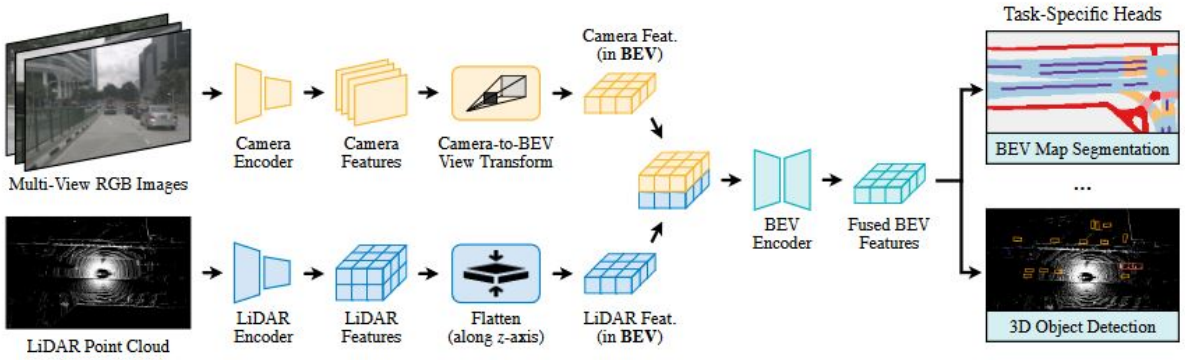

Figure 3.5: BEVFusion network structure [14].

a pre-computation and an interval reduction to reduce the BEV pooling latency more than 40 times.

In the pre-computation step, the association between each point in the camera feature point cloud and the BEV grid is done a priori. The algorithm precomputes the 3D coordinate and the BEV grid index of each point, and during inference, all features need to be reordered according to the map calculated before. The algorithm also uses an interval reduction strategy to accelerate the camera-to-BEV mapping. Specifically, a specialized GPU kernel parallelizes directly over BEV grids assigning a GPU thread to each grid that calculates its interval sum and writes the result back. These two strategies help to reduce the time dedicated to camera-to-BEV conversion from 500 ms to 12 ms, allowing the algorithm to spend only 10% of the time on this task compared to the 80% needed before. Once the algorithm obtains the features in the BEV domain from the camera and the LiDAR, it performs a fully-convolution fusion to combine together the information coming from both sensors. Even if LiDAR and camera features after the transformation are in the same BEV space, they can be misaligned. Therefore, BEFVfusion exploits a convolutional-based BEV encoder to align all the features in a compatible way.

The final step involves applying multi-task heads to the final fused BEV feature map. BEVFusion can be applied to most 3D perception tasks. For example, the BEVFusion paper shows its application in 3D object detection using a class-specific center heatmap to predict the location of the objects and a few regression heads to estimate size and rotation. Moreover, the algorithm was tested in segmentation by formulating the problem as a multiple binary semantic segmentation task.

In conclusion, BEVFusion is a powerful tool for almost every 3D perception task, and excellent are the results obtained in 3D object detection and BEV map segmentation.

4 Implementation

In this chapter, we will discuss the implementation of the various strategies of compression and sensor selection applied to the SELMA dataset to obtain an optimal transmission packet. Moreover, the data sent over the transmission channel should be compressed to avoid network congestion while maintaining all the intrinsic properties of the sensor files to perform accurate object detection at the receiver. Firstly, we show how to compress LiDAR scans efficiently using HSC, and then we discuss how to evaluate the object detection accuracy on compressed point clouds exploiting Pointpillars and PV-RCNN. In the second part, we tested BEVFusion on SELMA to understand how adding or removing sensors from the input of the object detection algorithm affects the final recognition.

4.1 Evaluation metrics

Before we describe the procedure of our work, we have to define the metrics we used to compare our results in terms of compression efficiency and object detection accuracy. For compression, the two most crucial parameters are the final file size of the pointcloud and the time needed to encode and decode it. For detection accuracy, the most widely used metric in the field of object detection is the Average Precision (AP). These metrics are briefly described in the following lines.

Compression metrics

Since pointclouds put a strain on communication systems, the most critical aspect to consider is the number of bytes that will be exchanged between vehicles. Therefore, we measured the average amount of bytes that a point cloud occupies. As explained in [7], a raw LiDAR file can weigh 3.2 MB, but in this work, for simplicity, we discarded all the other information on the original scan to maintain only the location of the points: x, y, and z. That makes more effortless the collection of the size of the compressed dataset,

because we did not need to store other information during the encoding process. Then, we also collected the encoding and decoding speed, which Draco already computes during both phases, and gave it as output alongside the compressed and decompressed file. We could not calculate the time needed for the semantic filtering, applied in the first part of the HSC algorithm, because the dataset is already provided with the semantic labels. Therefore, to take also that into account, we used the semantic filtering statistics in the worst-case scenario computed in the experiments of HSC [7].

Detection metrics

Before we describe the AP, we need to explain how a detection algorithm can distinguish good guesses from wrong guesses. The metric used to declare whether the algorithm finds an object or not is the IoU. The IoU aims to compute how well the prediction boxes, gathered from the detection algorithm, overlap with the ground-truth objects. Moreover, as its name suggests, the IoU is calculated as the difference between the area of the intersection of the two boxes over the area of their union, figure 4.2. There are two main categories of IoU: the one which is done in 2D, like FV or BEV, that utilize the area of the boxes, and another one that computes the IoU in 3D using the volume of the boxes. The AP exploits the IoU to obtain its two main parameters: precision and recall. The precision describes our predictions' accuracy, while the recall defines how well we find all the positive detections.

These two metrics are computed as follows:

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

After we set an IoU threshold, we can define as:

- True Positive (TP) if we correctly label and categorize an object, and its IoU is above the threshold;
- False Positive (FP) if we label or categorize an object that we should not have;
- True Negative (TN) if we correctly do not label or categorize an object;
- False Negative (FN) if we do not label or categorize an object that we should have or we do, but its IoU is below the threshold.

For each object detected by the algorithm, we add it to the list to compute precision and recall at each step to obtain a recall-precision function. As we can observe in the figure 4.1, what we obtain, after a maximization phase to smooth the pattern, is a step function. The general definition of the AP is the area under the recall-precision function, computed as:

$$AP = \int_0^1 p(r)\, dr$$

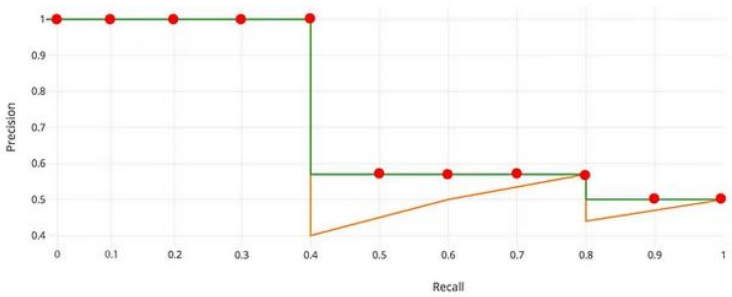

Figure 4.1: Example of a recall-precision function with 11 interpolation points [1].

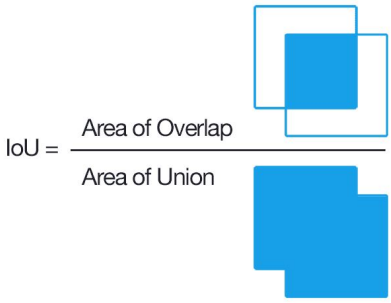

Figure 4.2: Intersection over Union [2].

However, the most used version in the literature is the interpolated AP. We set a certain number of points on the recall axis, commonly 11 or 40, and we take the average of the precision values in these points, which will be our AP.

$$AP = \frac{1}{11} \sum_{r \in \{0.0, 0.1, \ldots, 1\}} p_{interp}(r)$$

$$AP = \frac{1}{40} \sum_{r \in \{0.0, 0.025, \ldots, 1\}} p_{interp}(r)$$

As also suggested by KITTI [33] developers, the most precise version is the one with 40 interpolation points.

4.2 HSC compression on SELMA

The first step of our work was to obtain a set of compressed datasets exploiting HSC capabilities starting from a subset of the SELMA dataset. We choose the compression configurations trying to have a comprehensive set of characteristics to differentiate the datasets by their final file size and their compression and decompression speed. However, we pushed only a little on the quantization parameter to avoid the point cloud from being irremediably corrupted. HSC compresses the original dataset using the input parameters to set up the correct Draco and semantic filtering configuration. Finally, the algorithm's output gives us some interesting statistics, like the file size and the encoding and decoding times, together with the decompressed files used for the next object detection phase.

4.2.1 Dataset filtering

SELMA offers excellent characteristics for object detection purposes due to its large number of samples (around 30000), its diversity of weather conditions and environ-

[1] https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173
[2] https://www.v7labs.com/blog/mean-average-precision

ments, and the possibility of choosing between a considerable number of sensors. In this thesis, we decided to take a subsample of the entire dataset to make data easier to handle for both the compression and the object detection procedure. Furthermore, to validate the optimal characteristics of SELMA compared to other datasets in the literature, we decided to employ the same number of samples as by KITTI [33], which is probably the most famous and utilized dataset for vehicular detection purposes. KITTI offers a dataset containing about 7500 samples with top LiDAR and front camera files, labels, and calibration. Therefore, moving in this direction, we randomly took a total of 7794 samples from the first five towns' environments, allowing scenes from every daytime and weather condition. The dataset split is shown in table 4.1.

Split	**Town01**	**Town02**	**Town03**	**Town04**	**Town05**	**Total**
Train	258	182	617	1282	1570	3909
Val	253	176	616	1277	1563	3885

Table 4.1: This table shows how many samples for each town were collected for train and val splits for HSC testing. Total number of sample is 7794.

4.2.2 Compression configurations

HSC requires three parameters for its compression procedure: a Quantization Parameter (QP), a Compression Level (CL), and a Semantic Compression level (SC). After some preliminary tests, we discovered that the parameters affecting the final file size more are the QP and the SC parameters. Instead, varying the CL level does not significantly change the compression accuracy and efficiency. For example, as shown in figure 4.3, if we change the compression level by increasing or decreasing it by two, the final file size does not change by more than 5%. At the same time, figures 4.4 and 4.5 show that by changing QP by 2 or 3 and varying the SC level, the point cloud dimension is reduced by over 50%. After we made these considerations, we focused mainly on the QP and SC parameters and ended up having twelve different configurations to be used by the HSC algorithm. In the end, we obtained a spread set of configurations going from the uncompressed dataset with about 1 MByte of file size to the most aggressive compression, which led to a file dimension of about 6 KBytes. Table 4.2 shows all the various arrangements of quantization, compression, and semantic filtering parameters.

QP	0	0	0	14	14	11	14	11	9	11	9	9
CL	0	0	0	5	5	5	5	5	5	5	5	5
SC	0	1	2	0	1	0	2	1	0	2	1	2

Table 4.2: The table show for each column a configuration for one of the compressed dataset created ordered by average file size.

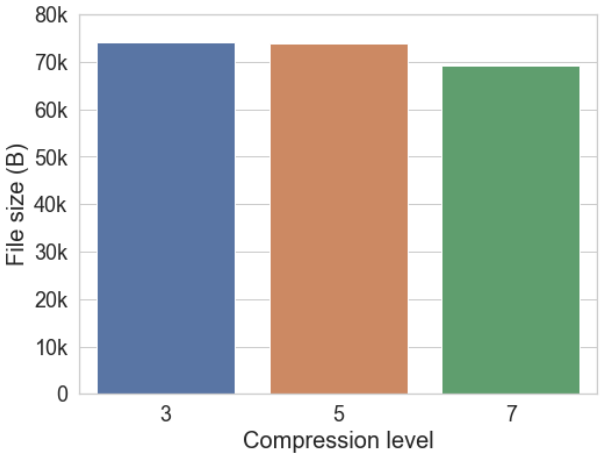

Figure 4.3: CL variation with QP=14 and SC=1 fixed.

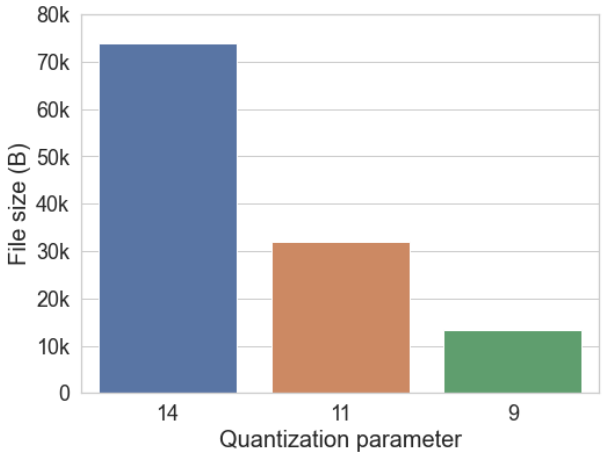

Figure 4.4: QP variation with CL=5 and SC=1 fixed.

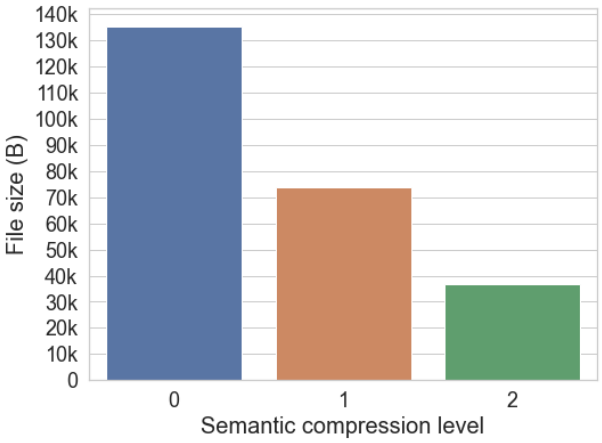

Figure 4.5: SC variation with QP=14 and CL=5 fixed.

Figure 4.6: File size change with respect to CL, QP, and SC.

4.2.3 HSC IMPLEMENTATION

HSC is open-source. The source code is publicly available on GitHub [3] and has been tested on the Semantic KITTI dataset [33]. Therefore, we modified the python script to make it compatible with SELMA. The algorithm, as input parameters, asks for a point cloud with the standard coordinates x, y, and z, and it also needs the corresponding label for every single point in the scan. Semantic KITTI format gives a point cloud in .bin format and a separate file for every label in .label format. The conversion to SELMA was simple, given that .ply files provided by the dataset have a five coordinates format (x, y, z, i, l), where *i* identifies the instance label and *l* stand for the semantic label. However, the parameter *l* of for each point has its own standard to identify each semantic object in the scene. Therefore, we performed a mapping operation on these labels to make them compatible with the Semantic KITTI standard.

Once the input is prepared, the algorithm applies semantic filtering to it based on

[3] https://github.com/signetlabdei/HSC

the SC parameters and saves the new point cloud with the desired composition into a .ply file. Finally, Draco takes this file and applies compression and decompression to output both the encoded version and the decoded one. This final decompressed version corresponds to the version the receiver will retrieve from the packet transmitted, assuming that the channel did not corrupt it. Therefore, the object detection algorithm will evaluate this point cloud in the next step of our implementation. In conclusion, from the Draco procedure, we extrapolate some other meaningful statistics, like the compression and decompression speed and the file size of the compressed data. This last statistic indicates how much LiDAR scans will impact the network.

4.3 Pointpillars and PV-RCNN on compressed dataset

Pointpillars and PV-RCNN are among the most popular algorithms for 3D object detection; therefore there are many implementations of both algorithms. For example, OpenPCDet[4] is an open-source repository that implements many 3D object detection algorithms, which are tested and validated on the most famous autonomous vehicle datasets, like Waymo [35], Nuscenes [34], and KITTI [33]. Moreover, we choose to make OpenPCDet our reference repository for object detection because it allows taking customized datasets as input. Specifically, it asks for standard three-dimensional .npy files as pointclouds and labels in the format of x, y, z, dx, dy, dz, yaw, category. Where x, y, and z indicate the center of the bounding box in the standard 3D coordinates; dx, dy, and dz are, respectively, the length, width, and height of the bounding box; yaw is the angle around the z-axis in radians; and the category is the label associated with each object. In the following two sections, we will explain how we implemented the training process and, finally, how we performed finals tests together with the output format.

4.3.1 Object detection training

In the OpenPCDet repository, the developers provide a comprehensive list of pre-trained models for almost every object detection model and every dataset. However, even if some datasets may share some common characteristics, training an object algorithm on them may require a different network structure. Therefore, the optimal solution for coherent results is to train each model from scratch on our dataset.

Firstly, from SELMA labels, we have to obtain the eight parameters (x, y, z, dx, dy, dz, yaw, category) described before to make them compatible with the custom dataset builder provided by the repository. For each sample, the coordinate system is centered on the ego vehicle, with the x-axis pointing forward, the y-axis pointing right, and the z-axis pointing upwards. Instead, SELMA has its coordinate system in a global format;

[4] https://github.com/open-mmlab/OpenPCDet

therefore, we needed to map each bounding box parameter to the proper position by using the so-called waypoints. Waypoints identify a specific location and rotation of the ego vehicle within a particular town and let us transform each bounding box's global coordinates into local ones.

Another preliminary step was to filter all bounding boxes based on two parameters: the minimum number of points within the box and the object's distance from the sensor. The most common choice for these two parameters is to take objects with a minimum of 5 points and at a distance of 60 meters. The first limitation comes from the fact that, as we can imagine, entities composed of many points represent the object characteristics more adequately than objects with few points. The second one sets a limitation on the distance because as we go far from the sensor, the LiDAR accuracy decreases, both for intrinsic reasons and for environmental characteristics. In fact, LiDARs have higher accuracy but a limited range compared to other sensors like radars. Moreover, weather conditions, like fog, snow, and rain, may hurt LiDAR performances.

Once we concluded the setup of the dataset, we trained both Pointpillars and PV-RCNN on the 3909 samples previously filtered. The experiment was conducted using a server equipped with an NVIDIA GeForce RTX 2080 with 8 GB of RAM. The algorithms are burdensome regarding memory usage, so we needed to tune some parameters to make them fit our specifications. Firstly we increased the voxel size from 0.05x0.05x0.1 to 0.16x0.16x0.1. This operation led to bigger voxels, consequently, to a slight loss of precision, but at the same time, allowed to relieve the network load. Then, we had to decrease the batch size to 2 to fit it in the 8GB GPU. The training lasted 10 hours for Pointpillars and almost 24 hours for PV-RCNN to reach 80 epochs where the training loss started to converge as shown in figure 4.7.

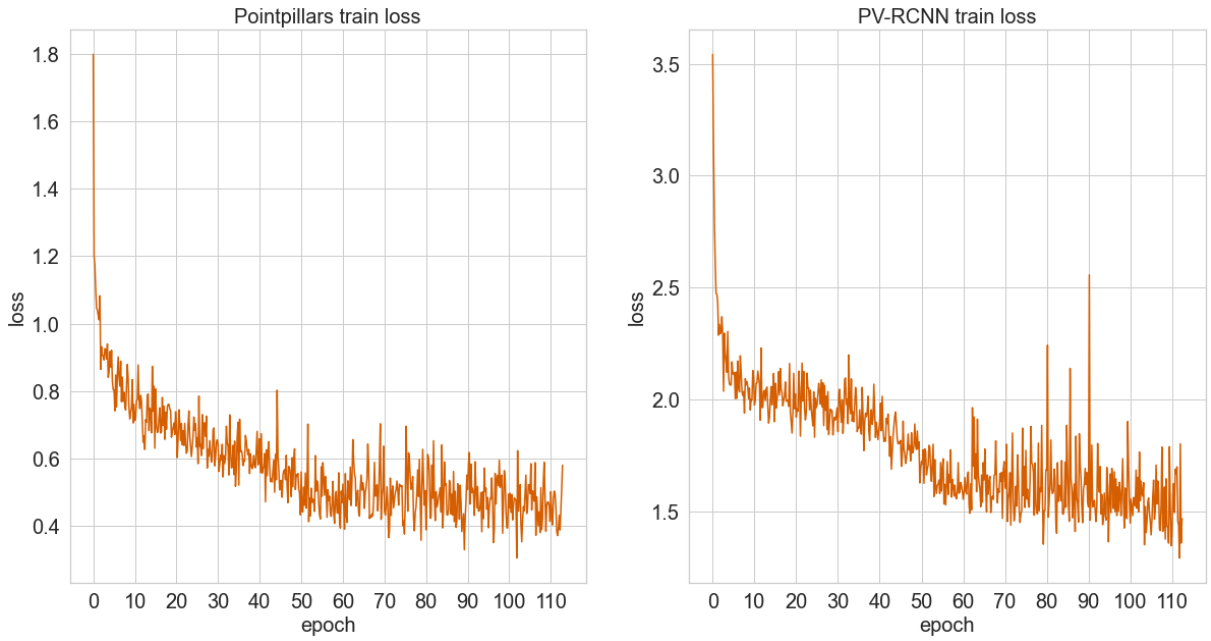

Figure 4.7: Train loss for Pointpillar (left) and PV-RCNN (right).

4.3.2 Object detection testing

The final step of the procedure consists of testing every compressed version of the original dataset with the two models obtained before. The labels remain the same for each configuration while we load the point cloud files into the test folder of OpenPCDet. We decided to keep results only for two classes: cars and pedestrians, because they are the most common and most relevant on the scene. Moreover, cars are representative of large objects, in contrast with pedestrians, which depict small entities. This choice helps us to validate the compression effectiveness for both macro categories.

In conclusion, we gathered all accuracy results that OpenPCDet gives in the output, which includes the AP for each class varying based on some parameters. Furthermore, the accuracy may be computed by:

- using 11 or 40 points to interpolate the precision-recall function;
- computing the Intersection over Union having bounding boxes in 2D in FV, in 2D in BEV, or directly in 3D;
- using different IoU thresholds, like 0.7 and 0.5 for cars and 0.5 and 0.25 for pedestrians.

4.4 BEVFusion for sensors selection

As already said at the beginning of the thesis, the two most important sensors for perception tasks, like object detection and semantic segmentation, are cameras and LiDARs. In early detection works, cameras were the first target for 3D object detection because they are cheap, and the computational capabilities needed to handle 2D images are smaller than the ones needed for 3D files. However, images in 2D are not good at calculating the distance between objects because of their intrinsic characteristics and the lack of a third dimension. Therefore, LiDARs started to take their space in 3D object detection. Although 3D files are heavy and complex structures, they guarantee a detailed representation of the vehicular environment, making them a good choice for object detection algorithm implementation. In more recent studies, researchers started to develop some algorithms that can exploit the most robust characteristics present in both cameras and LiDARs to help one another have a whole and precise environment representation.

In this thesis, we used BEVFusion, an algorithm that can perform 3D perception tasks by fusing characteristics from 2D images and 3D point clouds. The algorithm projects all the features into a shared BEV to make them compatible and then be processed by the same final detection heads. BEVFusion repository offers different tools to train and test some detection and segmentation algorithms. In particular, it can perform 3D object detection using a camera-only method, a LiDAR-only method, and a fusion method, employing the LiDARs and the cameras jointly. Comparing the results these algorithms gave us was useful in understanding which sensors are more relevant for the final detection, allowing efficient selection of data sent over the network.

This section will explain how we implement BEVFusion for SELMA from the dataset conversion to the implementation of the various algorithms.

4.4.1 SELMA into nuScenes

BEVFusion is a powerful algorithm for object detection purposes, and the developers made available a repository on Github[5] implemented in python and C++ based on mmdetection3d [39]. The code is well structured but lacks in terms of dataset acceptance. Indeed, the only dataset supported for training and testing the detection algorithms is nuScenes [34]. NuScenes is a well-designed dataset for object detection purposes, but it has its own way of storing its labels, calibration, and annotation files. Therefore, since the entire repository is implemented to accept, elaborate, train, and test nuScenes, we decided that the easiest thing to do was to convert our SELMA dataset into a similar format. In this way, we avoided rewriting from scratch the majority of BEVFusion scripts.

[5] https://github.com/mit-han-lab/bevfusion

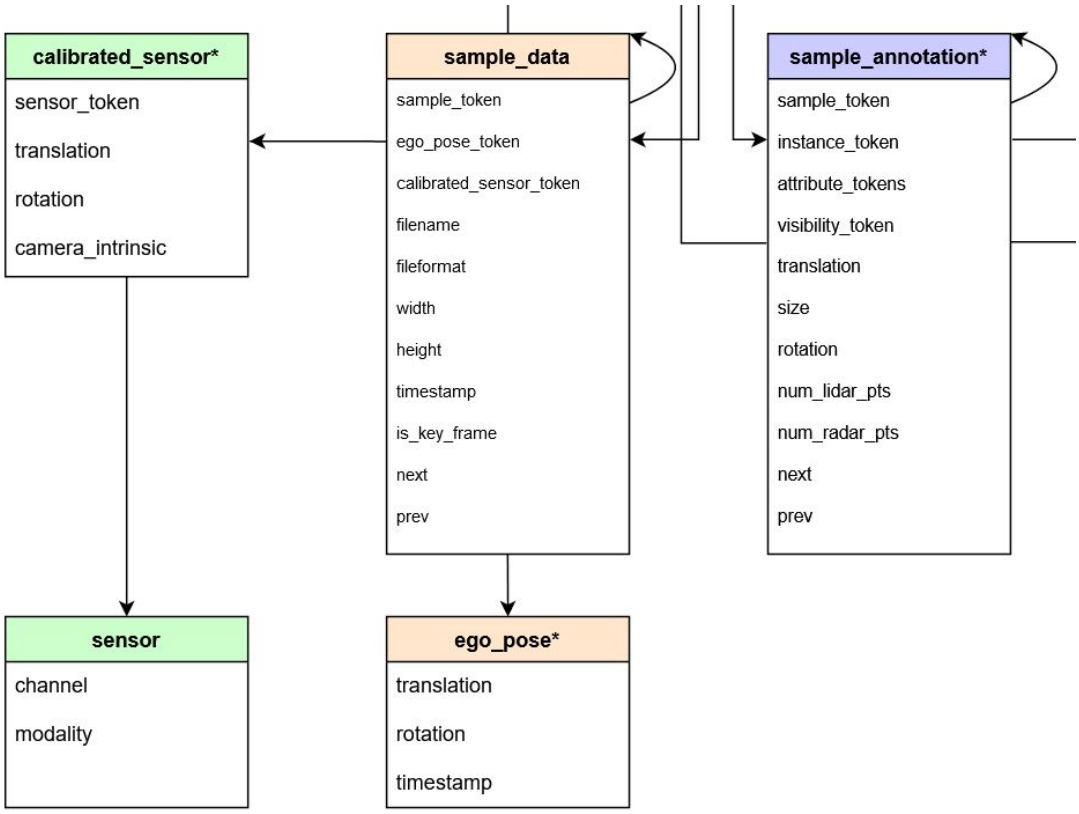

Figure 4.8: Example of most important tables in the nuScenes format, their links and their parameters.

NuScenes organizes information mostly into JSON files, which are linked to each other as a relational database. Each item is identified by its unique primary token, and foreign keys may be used to link to a token into another table. Only some files present in the dataset are helpful for detection purposes, but to make the data loading clear and error-free, we have to insert also not useful ones. However, SELMA does not include every information that nuScenes have. Therefore, some JSON files like log.json and map.json are copied and pasted from the original nuScenes just to avoid errors. These files contain only information like the type of ego vehicle, the date and the location of the acquisition, and the link to the map image in BEV, which are not relevant to the final training and testing of the algorithm. The most important tables to obtain from SELMA are calibrated_sensor, ego_pose, sample_data, and sample_annotation. Calibrated_sensor, as the name suggests, contains all the details regarding how sensors are placed and oriented. Ego_pose is tracing the position on the map of the vehicle that captures the scenes. Sample_data stores the file information captured for each scene, whether it is an image or a LiDAR scan. Finally, the sample_annotation table contains the bounding box information for every object. In figure 4.8, we can partially see the database's appearance.

Almost every information gathered from SELMA can be simply mapped into the tables in nuScenes format, but there are two exceptions. The first one regards the pre-filtering of the bounding boxes. As for Pointpillars and PV-RCNN, we have to remove

all the bounding boxes that are too far or contain only a few points, which makes them less representative of their class. Therefore, as before, we keep only boxes within a 60 meters diameter and with at least 5 points. The second conversion we had to make was regarding the rotation format. The standard for the rotation in the SELMA dataset is to express it in yaw-pitch-roll format. At the same time, nuScenes uses a coordinate system orientation as a quaternion (w,x,y,z). Luckily, there are many python tools to do the conversion in both directions.

Finally, after we prepared the conversion script, we had to organize the training and testing dataset. The filtering of the SELMA was different compared to the one in the previous section. For BEVFusion, we decided to take only scenes at noon in clear conditions. The main reason was that cameras are more susceptible to weather and light conditions. Therefore using clean environments helped during the training phase. Instead, as before, we tried to keep our total number of samples around 7500 and used scenes from the first five towns of SELMA. As a result, the total number of scenes is 7487, and table 4.3 shows how the training and testing set is divided.

Split	**Town01**	**Town02**	**Town03**	**Town04**	**Town05**	**Total**
Train	818	375	853	851	847	3744
Val	816	381	846	848	852	3742

Table 4.3: This table shows how many samples for each town were collected for train and val splits for the BEVFusion testing. Total number of sample is 7487.

4.4.2 Metrics

This section briefly discusses the metrics adopted in this second part of the thesis. The evaluation of the performances is similar to the previous part, but we used some specific metrics found within the BEVFusion repository to interpret the results better.

Compression metrics

As before, the compression efficiency for 3D data is given by the file size, in Bytes, of the LiDARs files, taking into account only x, y, and z coordinates. The same is for 2D images, for which we counted the average file size of all outputs of the various cameras. In general, all cameras and all LiDARs have the same resolution, respectively. Therefore, as we add new sensors for detection, we just have to multiply the average file size by the number of sensors.

Detection metrics

The BEVFusion implementation, after it computes the predicted bounding boxes, exploits the evaluation procedure offered by the nuScenes devkit repository[6]. As

[6] https://github.com/nutonomy/nuscenes-devkit

explained in [34], their evaluation method is based on the average precision procedure, but they used a different discriminant to decide whether a prediction is a true or false positive. As discussed in section 4.1, most detection algorithms use the IoU to estimate if the prediction is correct. In contrast, nuScenes uses its own method, which consists in counting a true positive if the distance between the centers of the predicted bounding box and the ground truth lies below a certain threshold. Given a list of distance threshold $\mathbb{D} = \{0.5, 1, 2, 4\}$, the final AP for a given class is computed as follows:

$$AP = \frac{1}{|\mathbb{D}|} \sum_{d \in \mathbb{D}} AP_d$$

This version of the AP helped in some situations to better understand how the model performed, particularly in the camera-only detection.

Additionally, we implemented the standard version of the AP to ensure an adequate series of results according to the literature. The IoU was taken in BEV to be coherent with the BEVFusion feature representation. We used 0.5 and 0.3 as IoU thresholds for cars and pedestrians, and we interpolated the recall-precision function using 40 points.

4.4.3 Camera-only implementation

As explained before, BEVFusion developers made available on the repository a model that performs 3D object detection using 2D information in camera images. The model used to perform this operation is Swin Transformer [40], a general-purpose backbone for computer vision. Swin Transformer organizes images in hierarchical feature maps starting from small patches and subsequently merging them into bigger ones while going deeper into the transformer layers. This backbone also has the advantage of having linear computational complexity to image size. Another clever implementation of the algorithm is its shifted window which has much lower latency than the sliding window approach, making the model more suitable for real-time

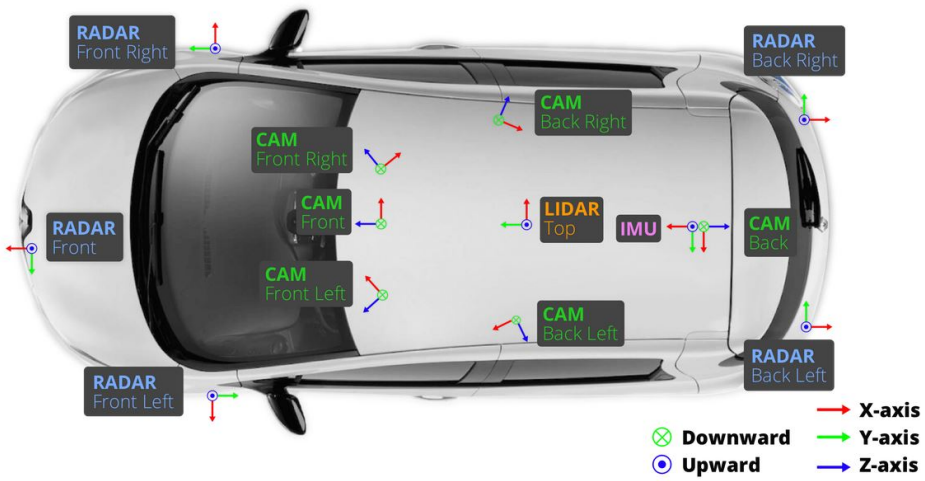

Figure 4.9: Sensor positioning on the nuScenes car.

purposes.

Thanks to our conversion of SELMA into nuScenes format, we needed only to adjust a few parameters in the training script to make it work for our dataset. We only had to make major modifications regarding the sensor differences between the two datasets. Since nuScenes have six cameras (front, front right, front left, back, back right, and back left), we have to modify the script to accept the camera setup of SELMA. Indeed, our dataset has only four major camera directions: front, right, left, and back. However, even with fewer cameras, we can almost reach a 360-degree view of the surrounding. Figure 4.9 and figure 3.1 show the respective positions of their sensors.

After some preliminary tuning of some parameters, we were able to train the camera-only model on an NVIDIA RTX A5000. The GPU is more powerful than the one used for HSC implementation and has a larger memory of 24 GB. However, even if computational capabilities were increased, the camera-only training phase lasted more than 24 hours to run 60 epochs; that was the time needed to reach stable results as shown in figure 4.10. This slowdown probably comes from the model's complex structure and some code inefficiency.

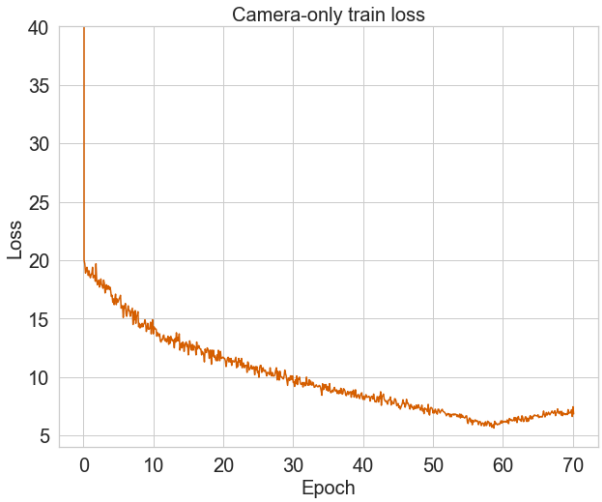

Figure 4.10: Train loss for camera-only model.

4.4.4 LiDAR-only implementation

BEVFusion also offers a LiDAR-only implementation for 3D object detection and semantic segmentation. The object detection model used in the repo is VoxelNet [29], a framework that learns a discriminative feature representation from point clouds and predicts 3D bounding boxes. VoxelNet creates equally spaced 3D voxels and, via a 3D convolution, aggregates local features for each voxel. The algorithm uses a regional proposal network that takes in input the voxelized representation of the point cloud and outputs the detection results.

As for the camera-only, we trained the model on an NVIDIA RTX A5000; this time, the training lasted 28 hours. As a result, the algorithm reached convergence after about

20/25 epochs as shown in figure 4.11. In the final testing phase, we used the model to evaluate the detection performances in both cases of single or multiple LiDARs. Given that the algorithm does not allow input from more than one point cloud, we had to perform a map operation to store all the information in one LiDAR file. We took the top LiDAR as the base file and added points from other sensors, ensuring to map each point according to the sensor calibration. We decided to evaluate one configuration with the LiDAR information gathered only from the top LiDAR and another containing data from all three available sensors. We did not evaluate the case with two sensors because the front left and front right LiDARs contain only information of the left and right part of the scene respectively. Therefore any combination of two sensors creates an unbalanced 3D representation of the environment.

4.4.5 Sensor fusion implementation

To conclude our work, the only part that needs to be added is related to the final sensor fusion detection. We used BEVFusion to merge together information gathered from SELMA four cameras and three LiDARs. As explained in the previous chapter, BEVFusion collects and maps 2D and 3D features into the same BEV representation to subsequently apply the detection method on the shared feature map. The model is really complex, and it is the main reason that pushed us to switch our GPU for the training. In fact, we could not load even the empty model with our previous GPU with 8GB of RAM. Moreover, using a batch size of 4 samples, the algorithm occupies more than 22 GB in our new NVIDIA RTX A5000, almost filling the 24 GB available. Even if BEVFusion is heavier than the other detection algorithms we used, it didn't take much time to train. In only 10 hours, it was able to train 10 epochs, reaching excellent performances. As we can observe in figure 4.12, the training loss is mostly flat because the weights from the camera-only and the LiDAR-only models are loaded, reducing

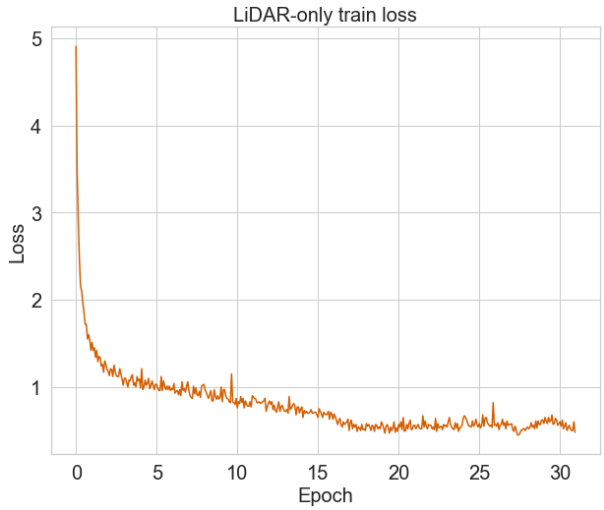

Figure 4.11: Train loss for LiDAR-only model.

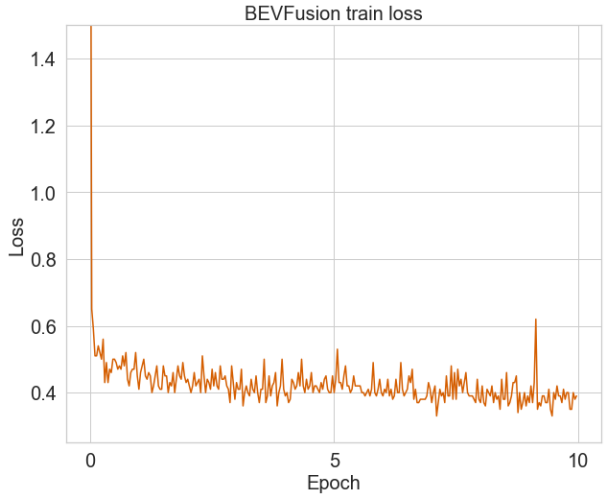

Figure 4.12: Train loss for BEVFusion model.

the training time for the BEVFusion model, as if it was pretrained.

We trained our model using four cameras to have coverage of 360-degree in the 2D space, and we used the LiDAR top as 3D information source. Our first test was with the same configuration as the train, with four cameras and one LiDAR. Subsequently, we tested using all the available sensors (adding the information from the other two LiDARs) to understand if merging multiple point clouds improves the final object detection. As a final test, we add HSC to the process by compressing the LiDAR file to understand its impact on the fusion model.

5 Results

In this chapter, we will discuss the performance achieved by HSC both in terms of compression efficiency and the ability to preserve crucial information available for detection purposes. After evaluating a number of compression configurations, we identified the optimal trade-off between compression ratio and accuracy. In the second part, we show how the usage of multiple sensors can affect object detection accuracy. We trained our dataset using camera-only, LiDAR-only, and BEVFusion models to evaluate which are the best sensors to keep for 3D object detection.

5.1 Object detection efficiency on compressed LiDAR files

At the end of the object detection test over the entire set of datasets, we gathered all the information we needed to have a panoramic on how the HSC affects point clouds. Specifically, we tried to assess the loss of information of the different compression levels. We decided to compute the AP using 40 points to interpolate the precision-recall function, the IoU comparison in 3D, and IoU thresholds of 0.7 for cars and 0.5 for pedestrians. Other definitions of AP may lead to different values, but we tried to draw general conclusions.

Firstly, we will talk about how car detection, performed by PV-RCNN, is affected by the reduction of file size by the HSC algorithm. As we can observe in figure 5.1, while we maintain the quantization parameter above 11, the performances are excellent, and the AP slightly deviates from the value obtained for the uncompressed dataset. However, the situation changes for a quantization level equal to 9, where the AP considerably drops by 20 points. Moreover, we can understand that at this level, the quantization is so high that Draco irremediably corrupts even relatively large objects like cars. Finally, the last thing we can observe is that the AP slightly decreases by fixing QP and CL and increasing only the SC parameter. This is because some cars with a high occlusion may be challenging to recognize by the semantic procedure, so they are removed from the

point cloud during the semantic filtering.

Similarly, in figure 5.1, we can observe that, for the PV-RCNN model, the AP on pedestrians behaves the same way as cars, with two minor differences. Firstly, we understand that the quantization affects the AP more for pedestrians since they are relatively small objects, and high quantization makes them hard to reconstruct. Therefore, the AP drops by 65% to 80% in the AP results for pedestrians detection compared to the uncompressed dataset. Secondly, by increasing the SC parameter, we obtain an opposite effect compared to cars. Since small objects like pedestrians are composed of a low number of points, they are more challenging to be detected. This can easily lead to a higher number of false positives, which will affect the precision and, consequently, the AP itself. Therefore, by applying semantic filtering on the point cloud with HSC, we were able to remove some noise from the background, reducing the possibility of wrongly guessing pedestrians.

In this section, we only explained the results for PV-RCNN because, as we can see from figure 5.1, the performances of Pointpillars are different but behave in the same way. Numerical results are shown in table 5.1.

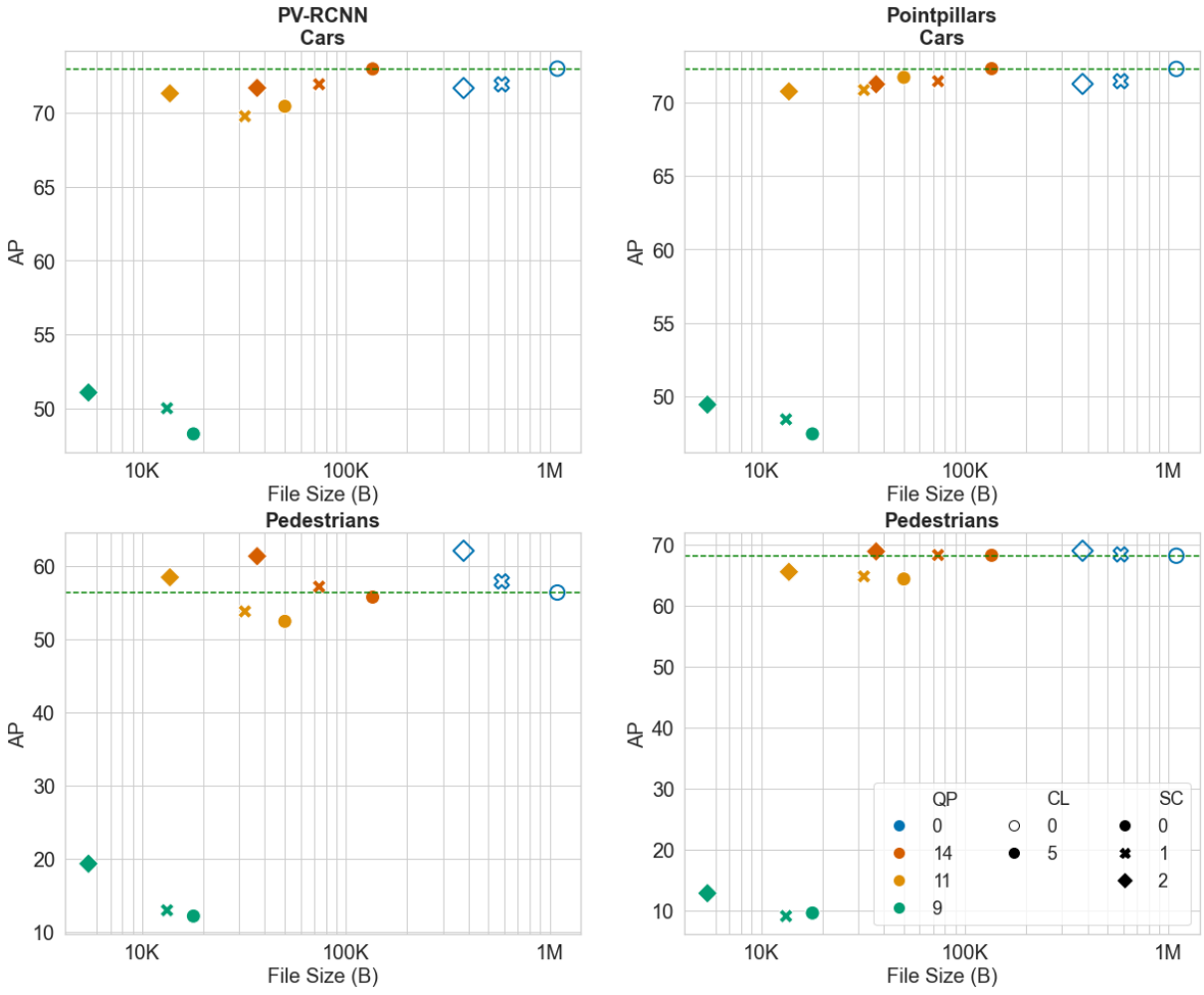

Figure 5.1: AP for PV-RCNN (left) and Pointpillars (right) on every HSC compressed dataset. First row shows the AP for cars and second row shows the AP for pedestrians.

				PV-RCNN		Pointpillars	
QP	**CL**	**SC**	**File size (KB)**	**Car AP**	**Ped AP**	**Car AP**	**Ped AP**
0	0	0	1066.3	72.99	56.33	72.29	68.24
0	0	1	569.0	71.93	57.84	71.46	68.46
0	0	2	369.4	71.67	62.03	71.27	69.05
14	5	0	132.2	72.97	55.7	72.31	68.31
14	5	1	72.2	71.93	57.13	71.46	68.36
11	5	0	49.0	70.43	52.4	71.72	64.43
14	5	2	35.8	71.68	61.3	71.25	68.95
11	5	1	31.2	69.76	53.76	70.86	64.86
9	5	0	17.4	48.26	12.17	47.44	9.67
11	5	2	13.4	71.31	58.41	70.76	65.6
9	5	1	12.9	50.0	12.96	48.43	9.16
9	5	2	5.3	51.07	19.31	49.44	12.9

Table 5.1: Numerical results for PV-RCNN and Pointpillars on every compressed dataset, ordered by average file size.

The last consideration we can make is on the compression and decompression speed. Since we do not have available the time needed to infer the semantic labels with Rangenet++, we use an average value suggested by [7] of 56 ms. This value is added to the encoding time if the dataset exploits the semantic filtering in the compression procedure. As shown in figure 5.2, we can notice that most of the time is needed for the semantic inference, while all configurations that have an SC parameter of 0 needed a maximum of dozen milliseconds to be compressed. On the contrary, configurations with SC equal to 0 are slightly slower in decoding speed, as shown in figure 5.2. In general, HSC is performing well, maintaining the compression speed around a maximum of 60 ms and the decompression speed below 6 ms.

In conclusion, if we want to decide which configuration is the right to use, we have to distinguish by the network capabilities available and the performances we

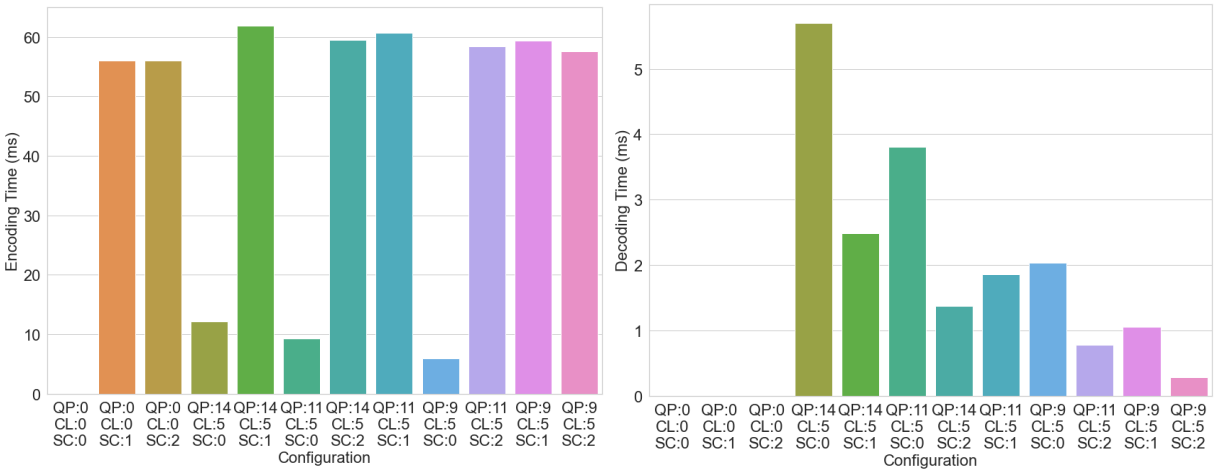

Figure 5.2: Encoding (left) and decoding (right) speed for each HSC compression configuration.

want to achieve. Moreover, if we want to reach performances not less than the ones of the original dataset and we do not have strict network constraints, we can use the configuration with QP=14, CL=5, and SC=0. With these parameters, the AP is not reduced for neither cars nor pedestrians. However, the network flow that we will create, for example, sending 30 frames per second, produces about 32 Mbps, which exceeds the IEEE 802.11p capabilities but can be handled by future mmWave connections.

However, if we are working on constraint networks and we can accept a sligth loss on the AP, our choice easily falls on QP=11, CL=5, and SC=2. While the final AP remains near the original one, the size of the point cloud is a hundred times smaller, around 13 KB. If we push this information over the network, for example, every 100 ms, the traffic produced is around 1 Mbps, standard V2V transmission methods can also handle that. Furthermore, if we have 2 LiDARs, one in front and one in the back, we can easily fit both LiDAR scans into a UDP payload.

5.2 Object detection efficiency using multiple sensors

In this second part, we tested our models with several configurations of sensors. In particular, we were interested in selecting a group of sensors that guarantee a compact file size while achieving higher object detection accuracy.

The first model we are going to discuss is the camera-only. The results obtained using only four cameras to perform 3D object detection are not good as expected. As figure 5.3 shows, the AP is not even comparable to the counterparts with only LiDARs or sensor fusion. Moreover, the performances are far from the ones obtained

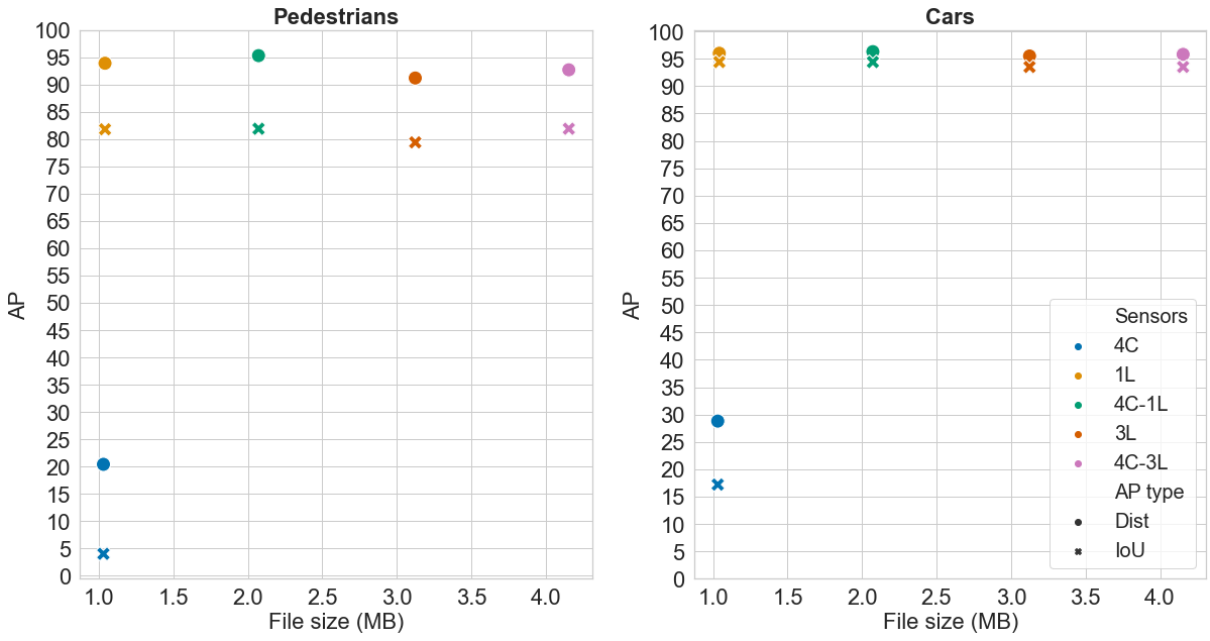

Figure 5.3: AP for multiple sensors object detection. Pedestrians AP in the left and car AP in the right.

Figure 5.4: Front view (left) and BEV (right) of predicted bounding boxes of camera-only model.

by BEVFusion developers on nuScenes, and the differences between the two datasets may be the reason for this accuracy loss. The algorithm pre-processes the input images by scaling and normalizing them using values taken from ImageNet [41]. Since SELMA produces synthetic images, this pre-processing may affect the final result. Moreover, since the final field of view of nuScenes and SELMA is not the same, due to the different numbers and positions of cameras, the features that the algorithms are able to retrieve may differ.

Despite the low AP, the camera-only model helps us understand how cameras perform in 3D object detection. We can better understand the results by looking at figure 5.4, which plots the predicted bounding boxes by the camera-only model (in front view and BEV). As we can observe, the algorithm can detect most objects, but the main problem is that it struggles to get the proper distance and rotation of the bounding box. We can also deduce it from the numerical results by looking at the difference between the AP computed using the center distance and the one using the IoU in table 5.2. In fact, the difference between the two metrics is always larger for the camera-only model because the AP based on the IoU is more affected by the mispositioning of the bounding boxes. It is even more evident for pedestrians because, for small objects, a slight misalignment can entirely corrupt their IoU. This behavior is understandable because images have only two dimensions, making collecting distances and positions difficult.

Below, since the performances are similar, we will discuss the LiDAR-only and BEVFusion models together. As we can observe in figure 5.3, both models achieve excellent results for cars and pedestrians, with the fusion one being slightly better. The best performances can be achieved by having four cameras and one LiDAR, but the configuration with only one LiDAR can reach similar AP values. We can also see how well the LiDAR-only method is performing from figure 5.5. As we can observe, the bounding boxes are more precise and perfectly aligned with the corresponding object. Moreover, objects that are difficult to detect by the camera-only model due to partial occlusion are now well captured.

In contrast, if we add LiDAR information to the input (by using 3 LiDARs), we are

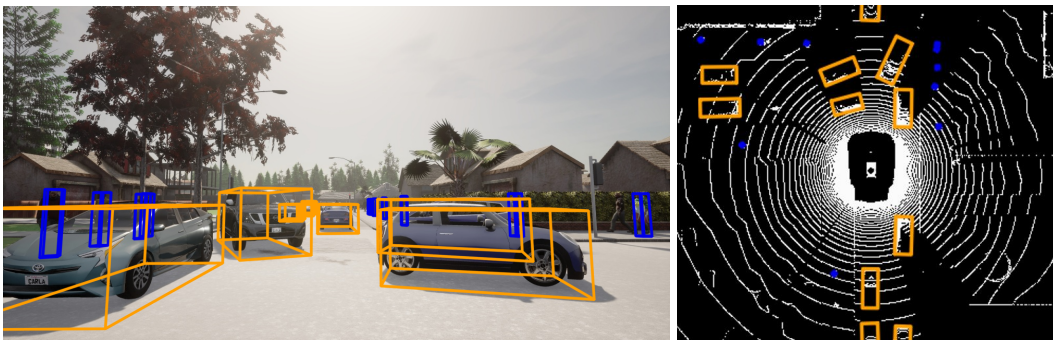

Figure 5.5: Front view (left) and BEV (right) of predicted bounding boxes of LiDAR-only model.

# Cameras	# LiDARs	File size (KB)	Car AP	Ped AP	QP	CL	SC
4	3	4254.5	93.5/95.8	81.9/92.7	/	/	/
/	3	3199.0	93.5/95.5	79.4/91.2	/	/	/
4	1	2121.9	94.4/96.3	81.9/95.3	/	/	/
4	1	1187.8	94.4/96.3	81.9/95.3	14	5	0
4	1	1068.9	94.2/95.9	76.8/86.3	11	5	2
/	1	1066.3	94.4/96.0	81.8/93.9	/	/	/
4	/	1055.5	17.2/28.8	4.0/20.4	/	/	/
/	1	132.2	94.4/96.0	81.8/93.9	14	5	0
/	1	13.37	91.8/95.3	74.2/84.0	11	5	2

Table 5.2: Numerical results for various sensors configuration, ordered by average file size. The AP contains both IoU and distance versions (IoU/Dist).

not gaining any advantage. In fact, due to the high density of points in the point cloud, the feature map created by the algorithms may be different, resulting in worse detection accuracy. The best solution to resolve the issue is to re-train the models using three LiDARs as input. However, we must also consider that adding too much data to the point cloud is slowing the detection a lot, increasing the computational time by more than seven times.

Another observation we can make looking at the two figures is that, as we can expect, pedestrians are more challenging to locate precisely. As we can see, the difference between the distance version of the AP and the IoU one is significant for pedestrians. As explained before, for small objects like pedestrians, a slight misalignment of the bounding box hardly affects the IoU.

As a final comparison for our models, we tested the LiDAR-only and the BEVFusion using HSC to compress the point cloud. Specifically, we used two configurations that we found to be the most suitable to use in the previous section. The first one has a QP=14, CL=5, and SC=0, and as before, it is the perfect choice when we want to maintain the same accuracy compared to the uncompressed one. As figure 5.6 shows, this configuration does not alter the final AP for both models. The second one has a QP=11, CL=5, and SC=2, and as expected from previous tests, the models lose some precision. This time, the loss is more accentuated compared to the results of the previous section,

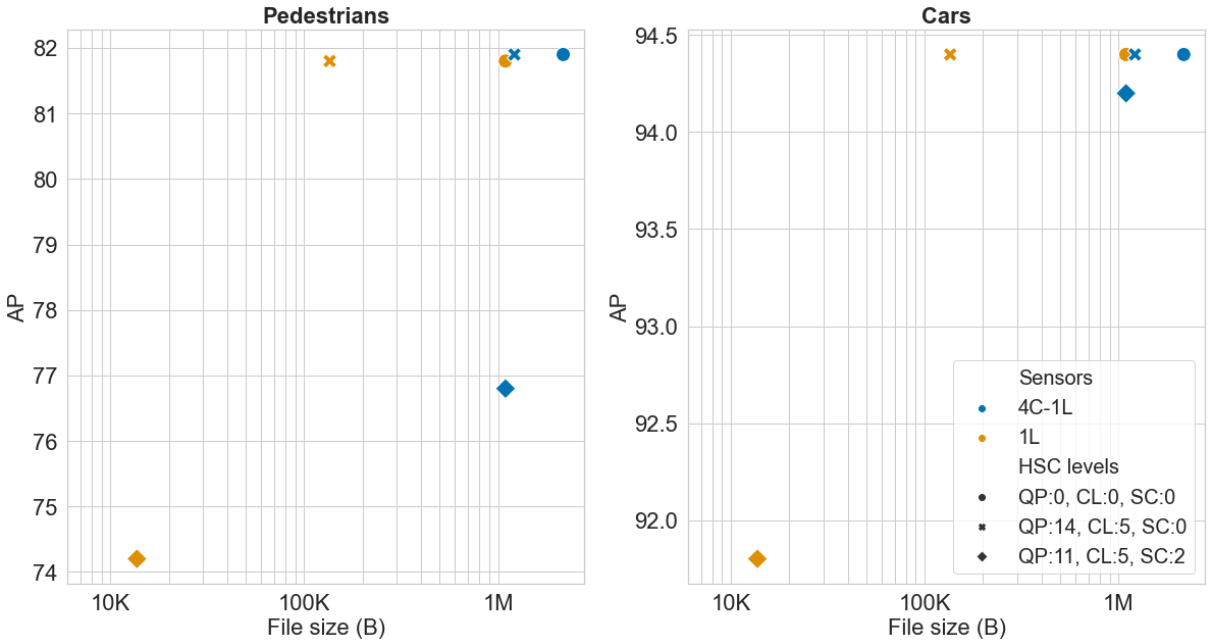

Figure 5.6: AP results for LiDAR-only and BEVFsuon models using compressed point clouds. Pedestrians AP in the left and car AP in the right

which can be caused by the change of the detection algorithm. However, we can observe from the figures that by adding 2D information, the BEVFusion model can help recover some data that may have been lost during the point cloud compression.

In conclusion, BEVFsion is a robust algorithm able to achieve excellent performance almost in every situation. Even with compressed LiDAR files, its detection accuracy remains the best. However, sending both camera and LiDAR information may overload the channel making reliable communications harder to achieve. For this reason, in most cases, using only 3D data is enough to achieve excellent detection accuracy while mitigating the burden of the channel. Moreover, using soft levels of HSC compression, we can reduce the point cloud size by one order of magnitude while ensuring detection performance similar to the one obtained with BEVFusion. Furthermore, if we push the LiDAR compression even more and we choose an adequate 3D object detection algorithm, we can obtain excellent compression efficiency with good detection accuracy.

6

Conclusions and Future Works

The purpose of this study consists in evaluating compression strategies to alleviate the burden on the communication network without affecting object detection accuracy. This approach may help future V2V or V2I networks to be efficient in their data dissemination but with the guarantee of providing each vehicle with a solid and consistent environment representation. Furthermore, this straightforward information sharing will transform our transportation system into a fully connected environment capable of improving traffic and safety efficiency. In the beginning, since LiDARs, among all sensors, are the ones that produce more data, we studied how to reduce their file size. We created a set of compressed datasets, starting from SELMA, based on different HSC compression levels. The goal was to evaluate the impact of the HSC algorithm on object detection models and to determine the best configuration to use based on the tradeoff between compression and detection accuracy.

We validated the results using two object detection models, Pointpillars and PV-RCNN. The outcomes suggest that HSC can perform excellent compression while maintaining unaltered most crucial information. For example, using a QP equal to 14, a CL equal to 5, and SC equal to 0, the point cloud became almost ten times smaller, but the AP remains equal to the uncompressed version. Furthermore, using more aggressive configurations, like the one having QP, CL, and SC equal to 11, 5, and 2, respectively, the point cloud size is reduced by two orders of magnitude. The AP slightly drops, but this configuration is a good compromise between compression and accuracy.

In the second part of our work, we focused on the impact of using multiple sensors on 3D object detection. We conducted a comprehensive evaluation of various sensor configurations utilizing camera-only, LiDAR-only, and BEVFusion models. Our findings suggest that specific sensors are more relevant for detection tasks while other sensors are just adding redundant data that do not impact the final detection. We proved that even if sensor fusion algorithms may achieve the best detection performances, using

a single LiDAR with a 360-degree view can reach similar results while alleviating the network burden.

In conclusion, if we want to perform 3D object detection in a multiple-vehicle environment, a 3D point cloud is the data we would like to exchange between road objects. Therefore, a sensor like the top LiDAR, which contains information from all directions, should have the highest priority in the presence of a constraint channel.

Future works

In the thesis, we tested our models only on cars and pedestrians, the most representative objects in a vehicular environment, and those who appear more frequently in the scene. In future works, we plan to extend the testing and validation of our results to all the vehicular classes. In fact, a comprehensive view of all classes helps better estimate our approach's effectiveness.

Moreover, our algorithms have been evaluated only on static data. Future works may expand the research by testing the compression also in a simulated traffic environment. Using ad-hoc simulation tools, we can reproduce a typical road environment, also simulating vehicular transmissions. In this way, we can prove our approach's effectiveness in the presence of V2V and V2I transmission services and their relative constraint channels.

References

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[2] M. Giordani, A. Zanella, T. Higuchi, O. Altintas, and M. Zorzi, "Performance study of lte and mmwave in vehicle-to-network communications," in *2018 17th Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2018, pp. 1–7.

[3] C. Cao, M. Preda, and T. Zaharia, "3d point cloud compression: A survey," in *The 24th International Conference on 3D Web Technology*, New York, NY, USA, 2019, pp. 1–9.

[4] M. Giordani, A. Zanella, T. Higuchi, O. Altintas, and M. Zorzi, "On the feasibility of integrating mmwave and ieee 802.11p for v2v communications," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–7.

[5] V. Rossi, P. Testolina, M. Giordani, and M. Zorzi, "On the role of sensor fusion for object detection in future vehicular networks," in *2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)*, 2021, pp. 247–252.

[6] F. Nardo, D. Peressoni, P. Testolina, M. Giordani, and A. Zanella, "Point cloud compression for efficient data broadcasting: A performance comparison," in *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2022, pp. 2732–2737.

[7] A. Varischio, F. Mandruzzato, M. Bullo, M. Giordani, P. Testolina, and M. Zorzi, "Hybrid point cloud semantic compression for automotive sensors: A performance evaluation," 2021.

[8] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet ++: Fast and accurate lidar semantic segmentation," Nov. 2019, pp. 4213–4220.

[9] Google. (2017). "Draco 3d data compression," [Online]. Available: `https://github.com/google/draco`.

[10] P. Testolina, F. Barbato, U. Michieli, M. Giordani, P. Zanuttigh, and M. Zorzi, "Selma: Semantic large-scale multimodal acquisitions in variable weather, daytime and viewpoints," in *IEEE Transactions on Itelligent Transportation Systems (T-ITS)*, 2023.

[11] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," 2017. arXiv: 1711.03938 [cs.LG].

[12] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," 2018.

[13] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "Pv-rcnn: Point-voxel feature set abstraction for 3d object detection," 2019.

[14] Z. Liu, H. Tang, A. Amini, X. Yang, H. Mao, D. Rus, and S. Han, "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[15] K. Bilstrup, E. Uhlemann, E. G. Strom, and U. Bilstrup, "Evaluation of the ieee 802.11p mac method for vehicle-to-vehicle communication," in *2008 IEEE 68th Vehicular Technology Conference*, 2008, pp. 1–5.

[16] B. Wu, A. Wan, X. Yue, and K. Keutzer, "Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud," 2017.

[17] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," 2020.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.

[19] L. Huang, S. Wang, K. Wong, J. Liu, and R. Urtasun, "Octsqueeze: Octree-structured entropy model for lidar compression," 2020.

[20] T. Wang, X. Zhu, J. Pang, and D. Lin, "Fcos3d: Fully convolutional one-stage monocular 3d object detection," 2021. arXiv: 2104.10956 [cs.CV].

[21] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "Bevdet: High-performance multi-camera 3d object detection in bird-eye-view," 2022. arXiv: 2112.11790 [cs.CV].

[22] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d," 2020. arXiv: 2008.05711 [cs.CV].

[23] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander, "Categorical depth distribution network for monocular 3d object detection," 2021. arXiv: 2103.01100 [cs.CV].

[24] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, "Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks," 2016.

[25] B. Li, "3d fully convolutional network for vehicle detection in point cloud," 2016.

[26] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," 2016.

[27] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3d proposal generation and object detection from view aggregation," 2017.

[28] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-yolo: Real-time 3d object detection on point clouds," 2018.

[29] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," 2017.

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

[31] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2016.

[32] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3d bounding box estimation," 2017.

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

[34] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "Nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[35] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.

[36] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., ser. LNCS, vol. 9906, Springer International Publishing, 2016, pp. 102–118.

[37] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3234–3243.

[38] Y. Wu, Y. Wang, S. Zhang, and H. Ogai, "Deep 3d object detection networks using lidar data: A review," *IEEE Sensors Journal*, vol. 21, no. 2, pp. 1152–1171, 2021.

[39] M. Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection," 2020.

[40] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," 2021.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

Acknowledgments

I would like to express my deepest gratitude to my thesis and internship advisor Prof. Marco Giordani, who gave me the opportunity to work on this project and was always available to provide me with some advice. I'm also extremely grateful to my thesis co-advisor Paolo, who helped me during the development of the project and offered me support in the writing of the thesis.