Point-Cloud based 3D Object Detection and Classification Methods for Self-Driving Applications: A Survey and Taxonomy

Duarte Fernandes*, António Silva*, Rafael Névoa*, Cláudia Simões, Dibet Gonzalez, Miguel Guevara, Paulo Novais, João Monteiro, Pedro Melo-Pinto

***Abstract*— Autonomous vehicles are becoming central for the future of mobility, supported by advances in deep learning techniques. The performance of a self-driving system is highly dependent on the quality of the perception task. Developments in sensor technologies have led to an increased availability of 3D scanners such as LiDAR, allowing for a more accurate representation of the vehicle's surroundings, leading to safer systems. The rapid development and consequent rise of research studies around self-driving systems since early 2010, resulted in a tremendous increase in the number and novelty of object detection methods. After a first wave of works that essentially tried to expand known techniques from object detection in images, more recently there has been a notable development in newer and more adapted to LiDAR data works. This paper addresses the existing literature on object detection using LiDAR data within the scope of self-driving and brings a systematic way for analysing it. Unlike general object detection surveys, we will focus on point-cloud data, which presents specific challenges, notably its high-dimensional and sparse nature. This work introduces a common object detection pipeline and taxonomy to facilitate a thorough comparison between different techniques and, departing from it, this work will critically examine the representation of data (critical for complexity reduction), feature extraction and finally the object detection models. A comparison between performance results of the different models is included, alongside with some future research challenges.**

***Index Terms*— Autonomous Vehicles, Computer Vision, Deep Learning, Perception, LiDAR, 3D Object Detection Models**

I. INTRODUCTION

According to the World Health Organization (WHO) the average number of global road deaths per year is estimated to be around 1.2 and 1.35 million, and the number of people that suffers non-fatal injuries and/or disabilities is between twenty and fifty million [1], [2]. Besides, road traffic collisions are the leading cause of deaths of people aged between 5-29 years [1]. Both the research and industrial communities look to autonomous vehicles as a vital strategy to reduce the current traffic fatalities. A reduction of 75-80% of current traffic fatalities is expected as a consequence of the impact of the proliferation of autonomous vehicles [3]. However, the effect of autonomous vehicles will go behind the crash rates, since it is also expected to drastically reduce insurance costs (economic burden of road injuries is equivalent to an annual tax of 0.12% on global gross domestic product during 2015-2020 [4]), increase mobility for non-drivers, improve road efficiency and reduce the impact of human mobility on the environment due to the increase on fuel efficiency and emissions reduction [5].

To simplify communication and facilitate collaboration within technical and policy domains, the Society of Automotive Engineers (SAE) published a standard called J3016 "Levels of Driving Automation" in 2014. Here, a common taxonomy and definitions for automated driving are settled, and six levels of driving automation are defined based on the amount of driver intervention and attentiveness required [6]. The levels range from no driving automation to full driving automation. This standard, states that vehicle perception is a crucial component to allow vehicles to reach the highest levels for self-driving. Developments on the perception of vehicles have been focused on the tasks of detection of road markings, and detection of surrounding objects, such as other vehicles, pedestrians, cyclists and signs.

To effectively perform the tasks mentioned above, autonomous vehicles must collect crucial information from the surrounding environment and extract from it relevant knowledge to ultimately categorise data by their semantic meaning, and even predict their future states [7]. To do that, the perception system might use a single acquisition technology or multiple sensors to continuously scan and monitor the environment, similar to human vision and other senses [8]. It involves collecting, filtering and processing the raw data coming from multiple sensors [9].

Thanks to the rapid development of 3D sensing technologies, 3D scanners called LiDAR, which stands for Light Detection and Ranging - consisting in a remote sensing method that uses light in the form of a pulsed laser to measure ranges (variable distances) to the Earth - are becoming increasingly available and affordable. To precisely measure the distance between the sensor and surrounding obstacles while providing rich geometric, shape and scale information [10]–[12], this technology measures the time between emitting and detecting the reflected laser light to extract a metric space between objects and sensor [13]. Every LiDAR scan generates a 3D cloud of points, which consists of a graphical representation of the surrounding scenario where each point

*These authors contributed equally to this work

contains information regarding its Euclidean distance. This type of sensor is able to provide long-range detection capabilities, high resolution and good performance under different lighting conditions. However, several factors make the task of perception on point clouds very challenging, which significantly increases the likelihood of failure. Some of these factors are related to the limitations of sensors and complexity of the acquisition scenario, namely (1) the vast diversity of environments, with particular attention to the variation of luminosity up to the variety of weather conditions; (2) object occlusion and truncation, resulting in partial or complete invisibility of an object due to view-blocking between objects or parts of the object being out of the sensor range; (3) dissimilar representations of objects of different classes as a consequence of the object's size but also for objects of the same classes, as the object distance to the sensor affects the acquisition technology's output; and (4) performance reliability across the whole driving domains, with different structures and classes.

Although LiDAR sensors have been widely adopted in self-driving applications, there are other sensing solutions, such as those listed in Table I, that have been also explored for several purposes within the self-driving context. Camera-based solutions have the advantage of providing high-density pixel intensity information, which allows capturing shape and texture properties [13]. Camera-based sensors, such as monocular cameras, have the disadvantage of lacking depth information, while other camera-based sensors, such as stereo cameras and Time of Flight (ToF), provide this information at the cost of expensive computation and resolution [14], respectively. Moreover, the error of the depth measure provided by stereo camera increases with the distance exponentially and is more sensitive to changes in the lighting conditions than the ToF cameras.

Due to the abovementioned limitations, several solutions resort to RADAR - Short Range (SRR) and Long Range (LRR) – and Ultrasonic sensors to acquire depth estimations [7]. However, they offer lower resolution and smaller field of view angle than LiDAR sensors. Their specifications make them suitable for other purposes, such as determination of objects' velocity (RADAR), and short-range object detection or autonomous parking manoeuvres (Ultrasonic). Table I compares the advantages and disadvantages of the most commonly used sensors for scanning the vehicles' surrounding environment.

For the above reasons, it is possible to find in the literature deep learning solutions relying on sensor setups that combine LiDAR sensor with other sensing technologies. This approach aims at overcoming several shortcomings of the different sensing technologies since the different types of sensors react differently to environment degradations (light, magnetic, weather condition) and provide different sensing advantages. The most adopted multi-sensing setup architecture combines LiDAR with monocular cameras. Information from these two sensing solutions complement each other; LiDAR captures depth information over longer distances (thus are more suitable for 3D detection over long distances) even in adverse lighting conditions, while cameras provide information richer in texture than LiDAR sensors which aid deep learning models to better perform their object detection tasks.

TABLE I. COMPARISON OF EXTERNAL SENSORS USED FOR DRIVING AUTOMATION (ADAPTED FROM [7]).

	LiDAR	**SRR**	**LRR**	**Ultrasonic sensor**	**Camera Monocular**	**Camera Stereo**	**Camera ToF (IR)**
Range <2m (short range)	–	+	–	++	– –	++	– –
Range 2m up to 30m (nominal range)	++	+	++	– –	– –	–	– –
Range 30m up to 150m (long range)	+	0	++	0	– –	– –	– –
Field of View (FoV) angle <10º	++	+	+	– –	++	++	++
Field of View (FoV) angle >30º	++	–	– –	–	++	–	++
Angular Resolution	++	–	–	– –	++	++	++
Direct speed information	0	++	++	–	0	0	0
Working under rain	–	++	+	–	–	–	–
Working in fog and snow	– –	++	++	+	– –	– –	–
Working with dust on the sensor	–	++	++	++	0	0	0
Night vision	0	0	0	0	– –	– –	++
Sensitivity to Light	++	++	++	++	0	0	+
Direct Depth Information	++	+	+	++	0	–	+
Cost	–	+	+	++	+	–	– –

Caption:
(++) Perfectly adapted sensors;
(+) Sensors having good performances;
(−) Sensors potentially meeting the criteria but with possible disadvantages;
(− −) Sensors that can be used with adaptations and additional heavy treatments;
(0) Sensors unable to meet the criteria or is not not applicable;

Deep learning on point clouds has been attracting the attention of both industrial and academic communities and, in the last four years, the number of methods being proposed to address various problems related to point cloud processing has exponentially increased. The available public datasets have had a vital role in the boost of research in this area. However, the amount of annotated data is still limited and does not cover all potential scenarios that a self-driving vehicle will face. Besides the limitations imposed by the LiDAR scanner, deep learning applied to point clouds has also been facing several other challenges related to the (1) nature of the point cloud, as it holds high dimensional data of sparse and unstructured nature; (2) high demanding requirements in terms of performance, autonomous vehicles are expected to extract features from the point cloud and detect and classify objects in real-time - typically the scenario is scanned at 10 Hz, which means that models have an interval of 0.1 s to process each frame and output model predictions – and must be reliable and robust; (3) setup limitations, the processing units equipped on vehicles are resource-constrained, indicating that models will have to determine the points relation for extracting local geometric features by means of efficient computational models.

Although deep learning techniques dominate several research areas, such as computer vision for several applications, speech recognition, Natural Language Processing (NLP) and bioinformatics, the present document analysis deep learning methods proposed for processing 3D point clouds to detect and classify objects within the scope of self-driving applications.

A. Pipeline Architecture and Taxonomy

Figure 1 depicts an overview of the pipeline architecture used herein to describe the workflow carried out by any of the existing deep learning-based algorithms. This pipeline architecture allows us to generically describe the underlying principle of any 3D object detection algorithm at the same time that provides a systematic way to analyse and compare the several object detection models architectures.

The proposed pipeline structure comprises the following three blocks: (1) Data Representation, (2) Feature Extraction and (3) Detection Network modules. In block (1) the model consumes the point cloud provided by a LiDAR sensor and organises this information into a structure that allows the next block to process it more suitably according to the design choices. The existing research in literature follows one of two possible approaches: divide the point clouds into voxels, frustums, pillars or 2D projection perspectives, or simply consume it as a raw point cloud (designated point-based in Figure 1). The block (2), called Feature Extraction, is responsible for the extraction of different types of features, namely low-dimensional and high-dimensional features from point clouds to generate features maps that will be forwarded to the next block. Finally, the last block (3), called Detection Network, is a multi-task block to provide 3D object detection, namely object class prediction, regression of bounding boxes around detected objects, determination of object orientation and, in some cases, inference of object velocity. Moreover, this block typically implements another set of layers for feature extraction before performing 3D object detection, designated here as semantic context features, from the feature map outputted by block (2). As depicted in Figure 1, some of the existing works implement a two-stage architecture: the first stage detects object proposals and the second stage typically follows a structure similar to block (3) to improve those predictions. The later module is here called Prediction Refinement module.

To better understand and compare the most notable research contributions in this field of study, we leveraged the proposed generic pipeline to propose a taxonomy, which is depicted in Figure 2. This taxonomy categorises the existing works according to the design choices at each of the blocks identified in Figure 1, evidencing the diversity of solutions proposed in the literature for each pipeline's stage.

B. Contributions

Recent surveys [15]–[21]on object detection cover deep learning models for general domains, while to the best of our knowledge only the survey [13] has its focus on the advances made in this field of study, specifically for self-driving applications.

The former surveys do not contain state-of-the-art methods which provide novel solutions to handle the challenges found in the application of autonomous vehicles.

Due to the rapid development of 3D object detection methods for self-driving applications, proven by a tremendous rise in number and novelty of publications since early 2019, the number of methods and models covered by the later survey is very low. Therefore, this document will highlight and describe the most recent advances found in solutions targeted for the application of interest in the current paper.

This paper aims at offering a comprehensive review of recent progress in point cloud-based deep learning methods for performing 3D object detection (object classification, object localisation, 3D point cloud segmentation and object orientation classification tasks) within the scope of autonomous vehicles. The proposed pipeline depicted in Figure 1 aims at generically describing the operational behaviour of any 3D object detection model, while offering a common ground for the analysis of the main components of the different models. At the same time, the existing works in the literature are categorised according to their design choices for every pipeline stage, allowing us to propose a comprehensive taxonomy. This taxonomy aims to provide a comprehensive study and comparative analysis of state-of-the-art methods, per stage and in a systematic manner. Moreover, a comparative analysis between the models' performance is provided, along with insightful observations and hopefully, future research directions.

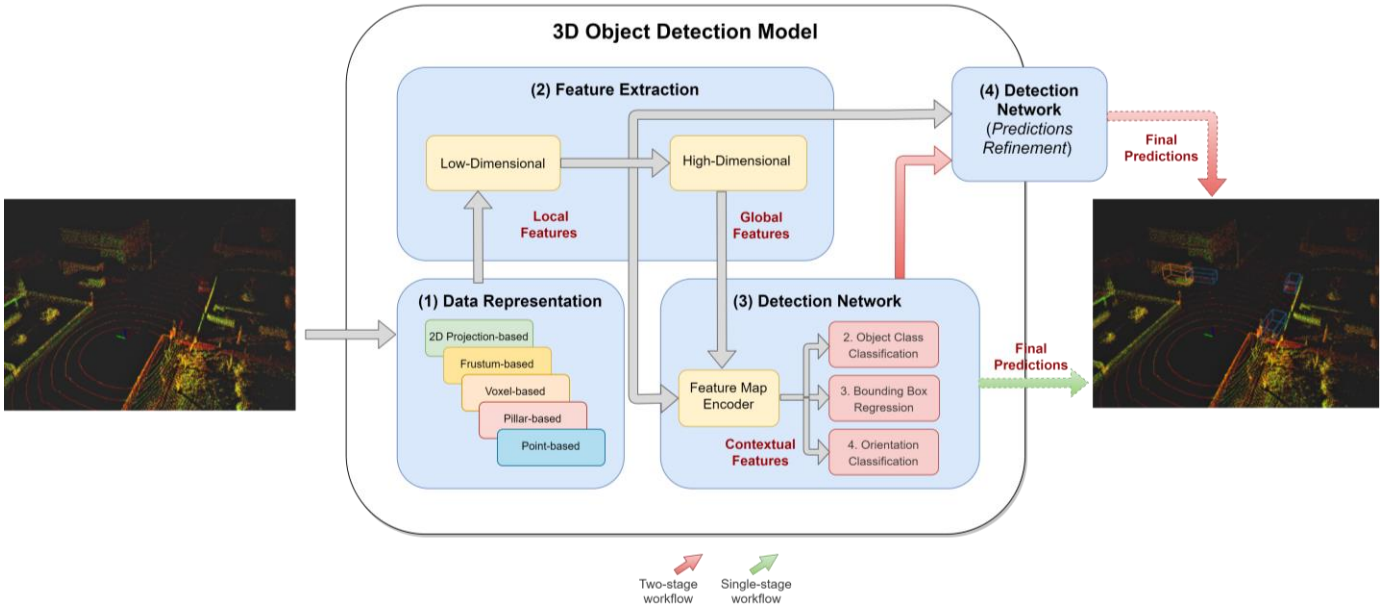

Figure 1. Architecture pipeline proposed to generically describe the operation principle and architecture choices of any 3D Object Detection addressed in the literature. Workflow of two-stage and a single-stage model are illustrated through a purple and green arrow, respectively.

C. Document Structure

This paper is structured into eight sections. In the present section, we described and compared the different sensors adopted in the context of autonomous vehicles according to their advantages and disadvantages, but also introduced the generic pipeline of 3D object detection models, and a taxonomy proposed to categorise and classify 3D object detection models. Each of the following three sections addresses a category identified in the proposed taxonomy; Section II offers a comprehensive analysis of the different approaches used to structure LiDAR-based data; Section III addresses the several options adopted in the literature to effectively extract the different types of features from LiDAR-based data; while Section IV addresses two different taxonomies. First, it covers the developments found in the literature regarding detection networks - where a taxonomy is proposed to represent and describe the developments achieved within the context of this model stage -, and a comparative study of state-of-the-art detection networks is provided. Second, it addresses the different Prediction Refinement Networks described in the literature.

The remaining sections, i.e. Section V and Section VI, offer a comparison between the 3D object detection models' performance, highlighting the impact of their design methodology and pipeline design choices. Section V addresses other development steps, not related to the pipeline design itself, which can have a deep impact on the model performance. Moreover, this section also addresses the most adopted benchmarks and respective features and limitations imposed in the process of training and validation of a 3D object detection model. Section VI, which is divided into two subsections, provides a comparative study of the state-of-the-art of fusion-based and LiDAR-based models, highlighting their similarities and differences in terms of design choices, as well as, a comparative analysis of the models' performance.

Before the final section, corresponding to the conclusion section, section VII highlights issues and challenges and points out some future research directions in the design of 3D object detection models.

II. Data Representation Approaches

Raw data from a LiDAR sensor is an unstructured point cloud with high dimensionality. With the prevalence of CNN-based (Convolutional Neural Networks [22]) object detection approaches, point cloud representations naturally require a structure where convolution operations can be efficiently applied. Next, we will present the main approaches in the literature for 3D point cloud representation.

Representative examples of such methodologies are the Sliding Shapes [23], Vote3D [24], VoxelNet [25], MFDS[26], MEGVII [27], 3D FCN [28], Vote3Deep[24], SECOND [29], Patch Refinement[30], Fast Point R-CNN[31], Voxel-FPN[32], PV-RCNN[33], HotSpotNet[67, 3DBN [34], Fusion of Fusion Net[35] and Point A^2 NET[36].

A. Point-based

Point-based approaches directly process the input point clouds and produce a sparse representation. Then, they extract a feature vector for each point by aggregating their neighbouring features. Models designed to handle raw point clouds first extract low-dimensional features from each point independently, and then get high-dimensional features by aggregating those low-dimensional features.

Qi et al. proposed a point-based approach, named PointNet, which represents unordered point clouds in a set of 3D points using a deep learning framework [37]. This model was extended to provide ways of extracting local and global features [38], and later

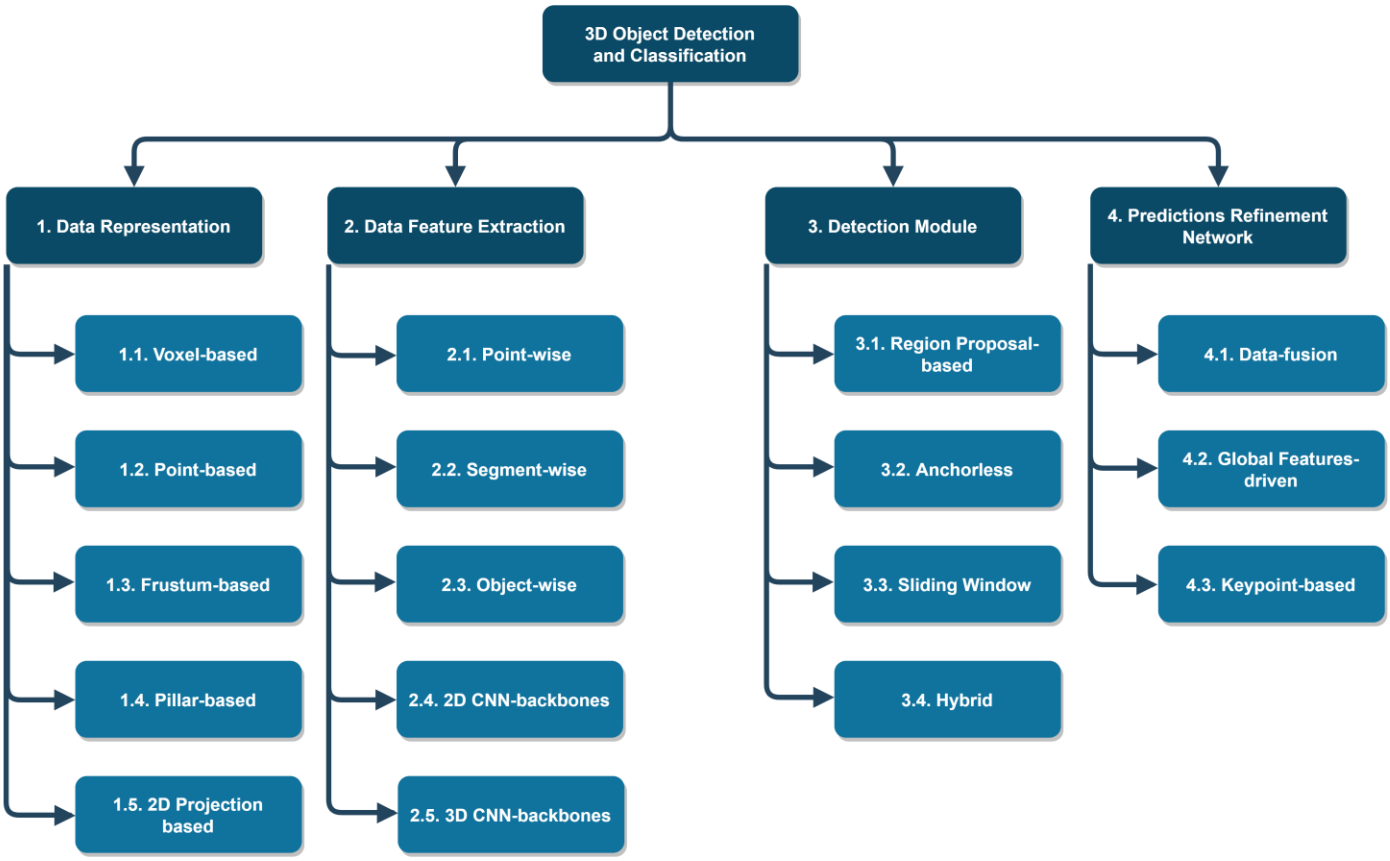

Figure 2. Proposed Taxonomy for description and comparison of the algorithms proposed for 3D detection using point clouds. The various contributions for deep-learning based object detection were categorised into four major categories, namely point-cloud representation, data feature extractor, Detection Module and Predictions Refinement Network.

has inspired other works, such as the Voxel Feature Extractor [25], to learn local features from each region of a point cloud structured as a volumetric representation. Several works, such as IPOD [39], STD [40], PointRCNN [10], PointRGCN[41], LaserNet[42], PointFusion[43], RoarNet[44] and PointPaiting [45], resort to this scheme of point cloud representation.

B. Voxel-based

A voxel is a volume element that represents a specific grid value in 3D space. Voxel-based approaches divide the point clouds into equally spaced 3D voxels in the three-dimensional cartesian coordinate system. Then, feature learning can be applied to extract features of a group of points within each voxel. This representation scheme results in the reduction of the point cloud's dimension, saving memory resources. Feature extractor networks specially designed to handle point clouds arranged in voxels have been proposed in the literature, e.g. Voxel Feature Extractor [25], whereas others initially designed for point-based point cloud representation have been directly applied or adapted to voxel-based point cloud representations in LiDAR-based 3D object detection models. Voxel-based representations help the feature extraction network to be computationally more efficient and reduce memory needs, as local/low- and global/high-dimensional features are extracted for a group of points, here called voxels, instead of extracting these features for each point individually.

C. Frustum-based

Frustum-based methodologies divide 3D point clouds into frustums. A frustum is a portion of a solid (usually a cone or pyramid) that lies between one or two parallel planes cutting it. Frustum PointNet [46], Frustum ConvNet [47] and SIFRNet [48] are examples of 3D object detector models that crop point cloud regions identified by an RGB image object detector. The resultant set of data presents the geometry of a frustum. We will name these regions Frustum-cloud hereafter.

D. Pillar-based

Pillar-based approaches organise point clouds in vertical columns, called pillars, before extracting features from it. Using pillars allows tuning the binning process in the x-y plane, thus excluding the z coordinate, such as in PointPillars [49].

E. 2D Projection-based

Besides using a 3D voxel representation, some approaches compact the information into a 2D projection, as a means to reduce the high computational cost of representing and processing 3D LiDAR data. Different works propose representing three-dimensional data in two dimensions, with several projection schemes being proposed, such as front view (FV), range view (RV),

and bird's eye view (BEV). In BEV, we have a top-down perspective of the scene, which means that the data is compacted along the y-axis. In the FV perspective, the data is compacted along the z-axis. In the case of RV, the data is projected in a 360-degree panoramic view, so the data is compacted along the x-axis.

VeloFCN [50] projects the point clouds to the front view, obtaining a 2D depth map. MV3D [51] also uses the projection representation, but to alleviate the information loss, they combine learned features extracted from multiple views, such as FV, BEV and camera view. However, the fused model has nearly linear computation cost concerning the number of input modalities, making real-time application infeasible. BirdNet [52] represents LiDAR data into cell encodings for BEV. The objective of this projection scheme is to meet real-time requirements inherent in self-driving. Other relevant works representing point clouds in a 2D BEV include HDNet[53], RT3D[54], PIXOR [55], AVOD[56], SCANet[57], ConFuse [58], MMF[59], Complex-Retina[60] and LaserNet++[61].

Some state the major rationale for the dimension reduction in BEV is that it is reasonable in the context of self-driving, as the objects of interest are on the same ground with typically small variance [1], [55], [56], [58]. Being the most popular choice when it comes to 2D LiDAR data representation, BEV has some other advantages over a front plane view (either RGB images or LiDAR points). First, the occlusion problem is solved, since objects in BEV keep the metric space, with different objects occupying different space areas, and not overlapping with each other [1], [2]. Second, objects preserve their physical sizes, having a small variance, which cannot be stated for a front plane view [1].

III. Data Feature Extraction Methods

The most crucial task carried out within the data object detector component depicted in Figure 1 is the extraction of features. It is fundamental to ensure an optimal feature learning ability in order to achieve optimal object detection performance. It is at this task that most of the research has been focusing on, developing strategies to improve the efficiency of the object detector models, with many methods being proposed for the extraction of features. As depicted in Figure 1, the type of features extracted by a feature extractor can be classified as (1) Local, (2) Global or (3) Contextual features. (1) Local features, also designated low-level features, are rich in detailed information about the localisation information of a point and usually are obtained at the earliest stage of the model pipeline. (2) Global features, also called high-level features, encode the geometric structure for a point and its corresponding neighbouring point clouds. A single network or combination of more than one network can be adopted to acquire the global context information. (3) Contextual features are extracted at the latest stage of the pipeline and are desired to be rich in both localisation and semantic information. This is the type of feature provided to the model's final tasks, carried out by the multi-task headers: Object Class Prediction, Bounding Box Regression, and Orientation Classification.

Due to the importance of relating different types of features along the pipeline, combinations of different feature extractors can be found at in 3D object detection models, forming compound solutions, whereas works might not include all the aforementioned feature extraction types in its pipeline.

Feature extractors will be categorised as (A) point-wise; (B) segment-wise, (C) object-wise and (D) CNN-based networks.

A. Point-Wise

Point-wise classification refers to the approaches that consider the whole point cloud as input, analysing and labelling each point individually [13], with PointNet [37] and PointNet++ [38] being the most well-known point-wise feature extractors. Point-wise feature extraction can be recursively used by segment-wise based networks [10], [40].

- PointNET

PointNet [37] which follows the architecture depicted in Figure 3, was initially proposed for direct extraction of global-features from point clouds and for producing classification and regression predictions. On the context of self-driving applications, object detection models adopt it as either a network responsible for individual extraction of local point features [25] or as a detection module network [24] for producing classification and regression predictions.

PointNet extracts global structures from spatial features of each point within a subset of points in Euclidean space. For this purpose, it implements a non-hierarchical neural network that comprises three main blocks: a max-pooling layer, a local and global information combination structure, and two joint alignment networks that align input points and point features. To extract point-wise features, a set of multi-layer perceptrons (MLP) is adopted. To provide permutation invariance, they apply a symmetric function to the extracted and transformed features, which sums up all representations and applies non-linear transformations, so the result does not depend on the order of input points. Although this network can be applied for extraction of geometric features and object classification purposes [37], it has serious limitations in capturing local structure information between neighbouring points, since features are learned individually for each point and the relation between points is ignored [16]

- PointNet++

PointNet++ was introduced as an extension of PointNet trying to overcome its main limitation. It proposes a hierarchical network for capturing fine geometric structures from the neighbourhood of each point, implementing a multi-scale feature learning structure called Prediction Pyramid (c.f. section III.E). The core of this solution is called the Set Abstraction layer and comprises two sub-layers, the Sampling and Grouping sublayer, which performs pre-processing tasks before applying the PointNet network sublayer. Thus, this grouping of sub-layers tries to address the lack of interpoint relationship knowledge in PointNet. At each set abstraction layer, a set of points is processed and abstracted by exploring metric space distances, leading to the production of a new set of fewer elements that is forwarded to the next level, allowing this pyramid structured extraction feature process to learn local features with increasing contextual scales [38]. Due to its simplicity and object geometric structure representation, PointNet++ is adopted in several works, such as PointRCNN [10], STD [40], among others. Moreover, it is also included at any stage of the generic pipeline depicted, for (1) classification of region proposals [10], [46], (2) segmentation [10], or (3) for refinement of the model's final predictions [10]).

This network was initially applied for the extraction of features from the whole point-cloud. However, some models opted by employing it for extraction of features from other point cloud representations, such as voxels where the aggregation is performed on voxel-wise features [33]. The set abstraction operation following a hierarchical paradigm introduced in PointNet++ has inspired several works to improve feature extraction by optimising the structure of this hierarchical feature learning as analysed in subsection III.E.

B. Segment-Wise

Since point-wise solutions increase object detection runtime, an alternative solution to reduce it has been introduced. Segment-wise feature extraction first segments the point cloud into several volumetric scale scenes before a point-wise classification model is applied to (each point of) the respective segment to extract volumetric features. An example of such methodology is the Voxel Feature Extractor (VFE), depicted in Figure 4, which was introduced as an extension of the PointNet concept to point clouds structured as volumetric scale scenes [13], [25], and is widely adopted in the literature. This feature extractor implements a segment-wise solution by stacking several Set Abstraction layers, allowing the extraction of features from points, that are further aggregated and concatenated, increasing the receptive field and adding more context to the extracted features. This allows the network to extract low-dimensional features (for instance, objects edges, per voxel). Unlike the point-wise feature extractor, segment-wise refers to feature extractors that are directly applied to a volumetric representation of the point cloud, as voxels [25], [29], [32] pillars [49] or frustums [46].

Examples of models adopting this network are VoxelNet [25], Second [29], Voxel-FPN [32], and HVNet [62]. In these works, point clouds are first structured in voxels, and then the feature extractor depicted in Figure 4 is applied. In [25], and [29], the feature extractor includes a network comprising two layers of Voxel Feature Encoding (VFE), to obtain a descriptive volumetric representation before learning the features through a Fully Connected Network (FCN). A VFE network takes the points of a voxel as input and encodes inter-point interactions, using an FCN – consisting in a linear layer, a batch normalisation layer [63] and a ReLU layer [22] – to extract features and get a point-wise feature representation. After that, an element-wise max pooling is used to obtain the locally aggregated features for each voxel. Finally, point-wise and element-wise features are concatenated and subjected to a second VFE layer to obtain the voxel-wise features. Other solutions that have opted for a volumetric representation and consequently a segment-wise based feature extractor might have adopted a different network. PointNet and PointNet++ are widely used in point-based point cloud representations, including by models that consume point clouds structured as volumetric representations [46].

Although the point-wise feature extraction and the segment-wise extraction processes are related, since the later methods resort to point-wise feature extractors, segmentation solutions when applied to volumetric point cloud representations tend to improve efficiency and inference time of the whole 3D object detection model, due to the data volume reduction [64]. Moreover, the features extracted from a segment containing several points can be more robust than a single point [65]. Statistics can help reduce errors and consequently allows for the extraction of more complex features for characterising local 3D shape information [36] used for 3D region proposals.

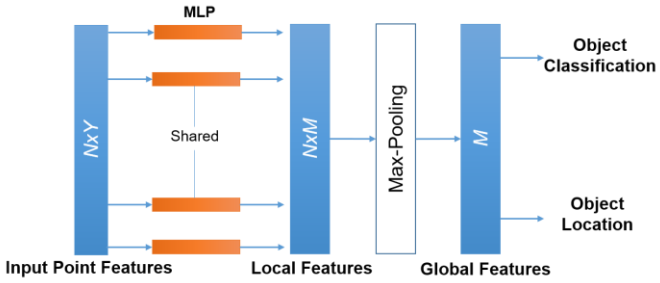

Figure 3. Architecture of the PointNet extractor network, where N refers to the number of points, Y the dimension of features (just Euclidean distance if adopted at early stages of model pipeline or semantic features if adopted as Detection block) and M is the number leaned features. The detection head here illustrated is not applied for the purpose of increasing the dimension of the localisation features of points.

To reduce the processing time and the imbalance of points between the voxels (consequently reducing the sampling bias and adding more variation to training) [25], [29], models randomly sample points, when voxels contain more than a pre-defined number of points, before extracting features. However, limiting the number of points in voxels that have significantly more points than the pre-defined limit, results in information loss and consequently the model's prediction behaviour might suffer from instability [32], [62]. On the other hand, when the number of points per voxel is significantly lower than the pre-defined limit, zero padding is applied [25], [29], which

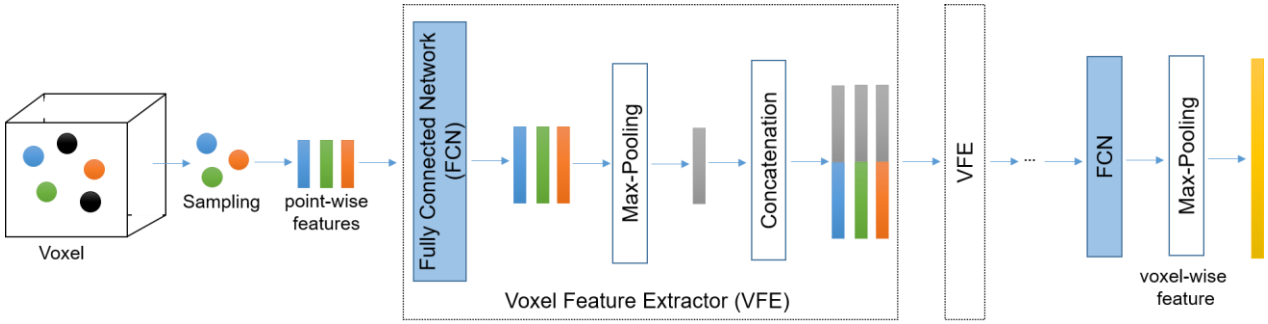

Figure 4. Architecture of the Voxel Feature extractor network.

increases the computation and memory resources required [32], [62] and consequently affecting models' inference time. The pre-defined limit number of points per voxel is 100 in research projects VoxelNet [25], Second [29], and PointPillars [49].

Another parameter that needs to be balanced is the voxel size, since it will affect the feature map size and consequently the computation and memory costs and model performance. Increasing the voxel size leads to smaller feature maps, improving inference time metric at the expense of the accuracy metric, whereas smaller voxels enable the extraction of finer geometry features that help in the localisation of objects, but penalise the inference time [32], [62]. In the literature, VFEs have been applied to generate features by either voxelization from a single voxel size [25], [29], or various voxel sizes [32], [62].

C. Object-Wise

Whereas previous methods focus on extracting features directly from points of a region of points without having previous knowledge whether this point or region belongs to an object, the object-wise methods consist of solutions that take advantage of mature 2D object detectors to filter a point cloud and detect objects in the image and then extrudes the obtained 2D bounding boxes to a 3D object bounding box. This approach allows the reduction of the search space to regions of interest following the dimension reduction principle. Moreover, the number of points processed is reduced, as only 3D regions contain objects of interest, while points of non-relevant regions (such as ground, sky, vegetation etc., for instance) are ignored[46]. Solutions might also combine multi-view representation of 3D point clouds [51], extracting object-wise features by projecting 3D objects/box proposals to the feature maps from multiple views. After detecting the object in a point cloud, point-based or segment-based features extractors can be applied to the output points or regions relative to the object, so that the detection and localisation process can be optimised for further methods. More detailed information about models that resort on data provided from combinations of LiDAR with other sensors is provided in subsection VI.A.

D. Convolutional Neural Networks

In 2012 [66], Krizhevsky et al. showed that CNN-based image classification solutions significantly outperformed the then existing solutions, namely SVM-based, for image classification and brought the attention of the research community to the potential of CNNs for tasks such as classification and potentially localisation of objects. Since then, several efforts have been made in the computer vision field to provide CNN-based architectures for the detection of objects with a better trade-off between performance and efficiency [54].

We will now address CNN-based solutions for detection of objects in point clouds. As will be shown, some of these solutions were first introduced in the literature to handle object detection in images and later adapted to handle point cloud specificities, while other methods were specifically designed to take advantage of the 3D space.

- 2D Backbone

The general pipeline for object detection relies on the usage of backbone networks, which act as a basic element to extract features. Most object detection models adopt CNN-based architectures since it allows for adaptive extraction of features without the necessity of applying the less general manual engineering processes. Commonly, most backbone networks used for object detection are the networks for image classification with an adapted architecture. Different models choose between deeper and densely connected or lightweight backbones according to the existing requirements related to accuracy versus efficiency.

In recent years, the popularity and effectiveness of CNNs have grown. Typically, CNN-based object detection models follow a similar architecture, where deeper networks are implemented to extract more complex and adapted features.

The interest in CNNs started with AlexNet, which was one of the first attempts to apply a CNN-based solution to image classification. Based on AlexNet, Simonyan et al. proposed the VGG model [67], which replaced the large filters used by AlexNet with smaller sized filters. This model increases the depth of the network by stacking a set of convolutional layers and has shown that increasing the depth of the network could improve the model's expression capability. However, it has consequences in the model training stage leading to optimisation problems. With the network depth increasing, the accuracy gets saturated and degrades rapidly due to the vanishing gradient problem. The same problem happens in GoogleNet [68] (also called Inception_v1), and generally in all architectures with a high number of layers.

To overcome this problem, He et al. proposed ResNet [69], which properly address the before-mentioned limitation by using skip connections (also referred as shortcut connections and residuals) while building deeper models, and thus not compromising the model's generalisation ability. The general idea of skip connections can be given by the following equation:

$$x_{l+1} = x_l + f_{l+1}(x_l, \theta), \quad (1)$$

Where x_l is the input feature map of the layer l and f_{l+1} are operations on the input x_l such as convolutions, normalisation or non-linear activation. $f_{l+1}(x_l, \theta)$ denotes the residual function of input x_l and each feature map of the deep layers is the sum of the shallow layers and the residual function. The idea of using skip connections is to skip the non-linear transformation and directly pass the values to the next layer. By using shortcut connections, gradients can be directly propagated from deep layers to shallow units, leading to significant reductions in training difficulty, thus allowing the increase in model depth without compromising the model's training capabilities. Also, ResNet-v2 [35] was the first attempt to utilise Batch Normalisation [63], which helps to increase even more the model's depth and provide better performance than the former version.

Further efforts were made to provide a network capable of fully utilising original features (from previous layers) in shallow layers since, during element-wise operations, these features will gradually disappear. DenseNet [70] concatenates the input with the residual output, instead of using element-wise operations. Other methodologies were proposed to increase model performance, efficiency or to reduce computation and memory costs (examples include ResNet [71], MobileNet [72]). Even though these models specialise on image classification tasks, they can be sub-optimal for use as detection tasks. To address this limitation, DetNet [73] and Hourglass Network [74] were designed to perform object detection. In current state-of-the-art, Hourglass network is widely used as a detection framework [75]–[77].

- 3D Backbone

The direct application of convolutions in a 3D space would be computationally inefficient and would severely increase the model's inference time, as 3D representation processing takes naturally longer than 2D representation. Adding to that fact that point clouds are sparse. Therefore, the direct usage of 3D representations looks a very time-consuming task. For these reasons, novel solutions have emerged to better handle and take advantage of point cloud sparsity, and efficiently extract features with more computationally efficient processes and faster runtime.

- Sparse Convolutional Networks

As previously stated, performing standard dense convolutions on sparse data, such as LiDAR point clouds, is a very inefficient process, considering most of the computational power is spent on spatial areas with no relevant information. Still, convolutional neural networks are powerful tools for understanding spatio-temporal data. Considering that point clouds are highly sparse data structures that can contain accurate spatial and categorical information, it becomes relevant to take advantage of its sparsity to speed up feature extraction, while minimising the number of points processed, reducing computational time and resources. A few approaches, such as [78], [79] have been proposed to address this problem by focusing computational power on the meaningful information present in the input data. In these approaches, a ground state is considered for hidden variables which receive no relevant value. This ground state must be calculated only once per forward pass during training and once for all passes during test time (it is generally non-zero because of the bias term). The only values the network is required to calculate are the hidden variables corresponding to active sites. A site in the input or hidden layers is considered active if any element in the feature vector differs from the ground state. Forward propagation is performed by calculating a feature matrix, composed of a list of row vectors, one for the ground state and one for each active spatial location in the layer, and a pointer matrix to store the number of each corresponding row.

In [80], the author realised that the convolution operations defined in the aforementioned methods [78], [79] were not optimised to accommodate the sparsity of the input data, leading to a continuous dilation of the data that would end with the consequent reduction of sparsity, rendering the method

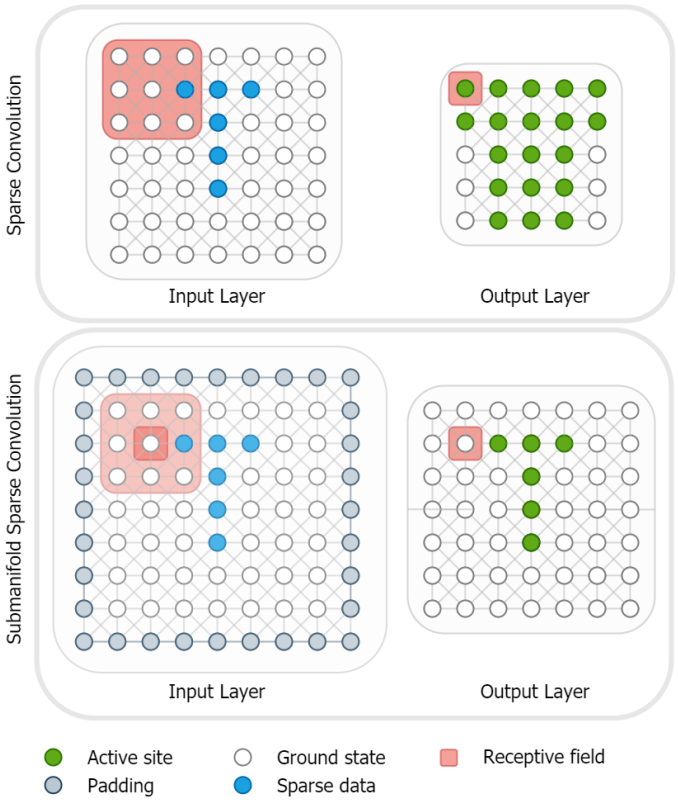

Figure 5 - Comparison between Sparse Convolution (SC) and Submanifold Sparse Convolution (VSC) operations.

impractical for modern deep networks. To allow sparse data to be efficiently processed whilst retaining a greater degree of sparsity, the authors propose two distinct sparse convolution operations, both largely based on the mechanisms of [78], [79] a Sparse Convolution (SC) and a Valid Sparse Convolution (VSC), also referred to as a Submanifold Sparse Convolution. A comparison between these two operations is depicted in Figure 5. The SC operation computes the set of active sites in the same way as a regular convolution but discards the ground state from non-active sites. Though this operation offers computational benefits, it does not perform well on deeper networks because of dilation. In order to deal with this problem, the VSC operation takes a different approach to the input data. First, the input is padded so that the output retains the same dimensions as the input, and then it restricts an output to be active if and only if its central site in the receptive field is active. Sparse max-pooling and average-pooling operations are also optimised to work on sparse data, as well as a deconvolution operation, defined as the inverse of the SC operation by inverting the connection between input and output. These convolutions are implemented using a feature matrix, containing one row per active site, and a rule generation algorithm using a hash table to store the location and row for all active sites.

- CNN networks based on voting scheme

An alternative approach for dealing with spatial sparsity is presented in [8] and [65] where a feature-centric voting scheme is proposed to extract features from 3D space while reducing the number of floating-point operations when compared to a standard 3D convolution.

First, the point cloud is discretised into a sparse 3D grid. A fixed-dimensional feature vector containing a binary occupancy value, mean and variance of the reflectance and three shape factors is then extracted for each occupied cell. Since cells in empty space are not stored, this leads to a sparse representation of the 3D space. To perform sparse convolutions via voting each non-zero vector casts a set of votes, weighted by the filter weights, to its surrounding cells in the output layer, as defined by the receptive field of the filter. The voting weights are obtained by flipping the convolutional filter along each dimension, and the result of this operation is the sum of the votes falling in each cell of the output. Biases in this scenario are constrained to be non-positive to retain sparsity and only need to be added to non-empty cells.

In [64], the sparse voting scheme is proven to be mathematically equivalent to a convolution on a sparse grid, and since the filter only needs to be applied to the occupied cells in the input grid, rather than convolved over the entire grid, the computational cost is proportional only to the number of occupied cells, rather than the total number of cells in the 3D grid. Given the similarity of the result of performing a regular sparse convolution and a voting-based one, one can assume that the problem of dilation also applies to this algorithm. While the input point cloud is sparse, with every convolutional layer, the regions of non-empty cells are dilated approximately by the receptive field size of the corresponding filters in the layer. In an attempt to compensate for this, the work in [24], proposes the use of non-linear activation functions to help maintain sparsity by only allowing features with a value greater than zero to cast votes in the next sparse convolutional layer. Figure 6 shows a graphical representation of the mathematical equality of the Submanifold Sparse Convolution and Voting operations.

- Graph Convolution Network

Although GCNs are not new, they have only gained attention in the last two years, since they are regarded as being an extension of CNNs for irregular or non-Euclidean structured data [41], which allows overcoming some of the limitations of point cloud representation methods. GCNs perform operations similar to CNNs, where the model learns features by inspecting neighbouring nodes. However, CNNs were designed to operate on regularly structured data.

In its generic architecture, GCNs transform points into a graph by defining points as nodes and specifying connections between them, which are here designated as edges. To extract features on a per-node basis, GCNs apply graph convolutional operations to aggregate features from a set of nodes of the specified node's neighbourhood.

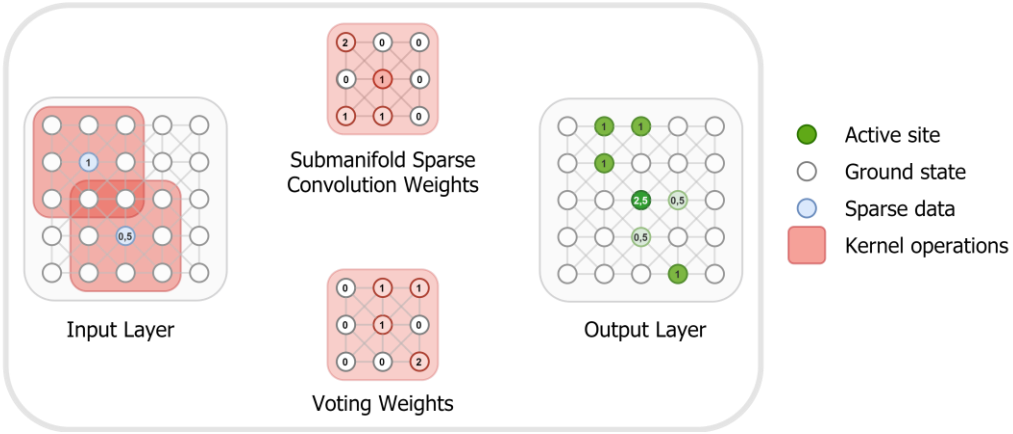

Figure 6 - Demonstration of the mathematical equality of the Submanifold Sparse Convolution and Voting Operation (adapted from [24]).

Recent developments in GCNs enabled the introduction of a set of features, such as (1) residual skip connections, allowing the implementation of deeper networks [81], [82], which facilitates the gradient flow between layers; (2) dynamic receptive fields where edge connections are dynamically updated from layer to layer [83], resulting in a different graph of relations; and (3) dilatation, which enables the increase of the receptive field to an optimal size that better gathers feature information by skipping neighbours without increasing the number of model parameters [82]. All these features enable performance improvements of GCN based models in tasks such as classification and segmentation [81], [84]. According to the reported results in PointRGCN, feature (1) and (3) improved the GCN performance by 5.4% and 5.1%, respectively.

Since GCNs are described as being computational and memory intensive networks [41] and driverless vehicle applications demand very time-efficient 3D object detection models, the use of graph representations on LiDAR inputs has been limited to module (4) of the generic model pipeline. This means that this approach is only applicable to dual-stage models. The number of points and features received as input by module (4) is much lower than the original point cloud as this data refers to only points that belong to a proposal outputted by module (3). Graph representations of LiDAR point clouds in driverless vehicle applications are here categorized in (1) graph representation *per-proposal*; and (2) *per-frame*. In the former solution, the points that contain geometric information of a proposal are represented in the format of a graph, and contextual features are extracted through per-proposal feature aggregation based on a GCN. In the latter category, all proposed/detected objects in a frame are represented in a graph, where each proposal/object corresponds to a GCN node and edges between nodes are settled. These edges are then leveraged to extract contextual information across all proposals, consequently refining all proposals. The application of this network has shown success in tasks such as segmentation, the task of proposal classification and localisation refinement [41].

Examples of graph convolution layers developed and applied for the context of autonomous vehicles are EdgeConv [85] and MRGCN [81]. EdgeConv layers take as input local geometrical features, encoded by the feature difference between a given neighbour's features and the current node's features, and global geometry, which refers to each node's centre feature. Then, this input data is processed by an MLP, which means that an MLP is performed for every given neighbour's features. The MRGCN is described as a simplified version of the former layer, aiming at reducing its computational and memory budget. In this type of layer, an MLP is only performed once, since aggregation of all neighbour features is previously performed, with further concatenation with global geometry features and MLP operation following. Prediction refinement methods are analysed in-depth in subsection

IV.D.

E. Feature Extraction Paradigms in 3D Object Detection Models

The process of extracting features along the model pipeline varies from model to model. Models implementations might opt for a compound feature extractor as depicted in Figure 1 (where different methods are used to extract local and global features) or by a single feature extractor. The former approach explores the advantages brought by different methods, whereas the latter approach intends to improve the inference time of block (2). Moreover, networks for the extraction of features at either block (2) or (3) of the model pipeline can follow different hierarchical structures in the extractor network aiming at extracting information without losing localisation information, which is important for better classification, and more often, 3D object localisation.

Regarding the architectural structure of feature extractors, several paradigms to detect and classify objects in multi-scale and different aspect ratios have been adopted in the literature. Since objects, so-called foreground objects in point clouds, have a large variance in scale and aspect ratios, the need to implement networks able to learn hierarchical features increased. Some solutions, such as [86] and [87], proposed a very intuitive but computationally heavy approach, where different detector and classifier heads are implemented and trained for each network layer. Each detector head takes the responsibility of detecting objects in a certain range of scales according to the inputted image size. Therefore, several copies of the original image but with different resolution sizes of have to be generated and fed to their respective decoder. Instead, most recent detection encoders have opted for updating the decoders' architecture by increasing the number of layers of a deep network, which allows for the generation of feature maps with different resolutions. Shallow layers are usually more robust in detecting small objects - its features provide spatially-rich information and have smaller receptive fields resulting in higher resolution - whereas deep layers are more suitable for detecting large objects - layers with semantic-rich features, known for being more robust to illumination and translation [21], with larger receptive fields but lower resolution. The first multi-scale feature learning model based on a pyramid structure and deep learning came from [12] and was later adapted for 3D in [28], where authors implemented a fully connected network t(FCN) hat ensured a bottom-up path to extract high-level features from an inputted single feature scale as shown in Figure 7. Pyramid solutions have as its main advantage the ability to detect objects of different sizes without needing to perform point cloud cropping. The multi-scale feature learning schemes can be categorised by the existence or absence of lateral connections in its pipeline.

The structures illustrated in Figure 7 refer to feature extractors with a single path and can be divided into:

(i) Prediction Pyramid, in which predictions are carried out over a single feature map, which is obtained after several layers of feature extraction [11], [12], [54], [88] on the finest level of the pyramid structure.

(ii) Multiple Prediction Pyramid, where predictions are made from multiple layers (each layer is responsible for a certain object scale), leading to a high amount of overall outputs (the final prediction is made by merging all detection results from different feature maps) which results in a computationally expensive process [89];

(iii) Integrated Features, also called deep layer integrations, where predictions are only performed over a final feature map. This approach takes advantage of the concept of skip connections [69], [90] (to fuse the earlier output layers, by skipping one or

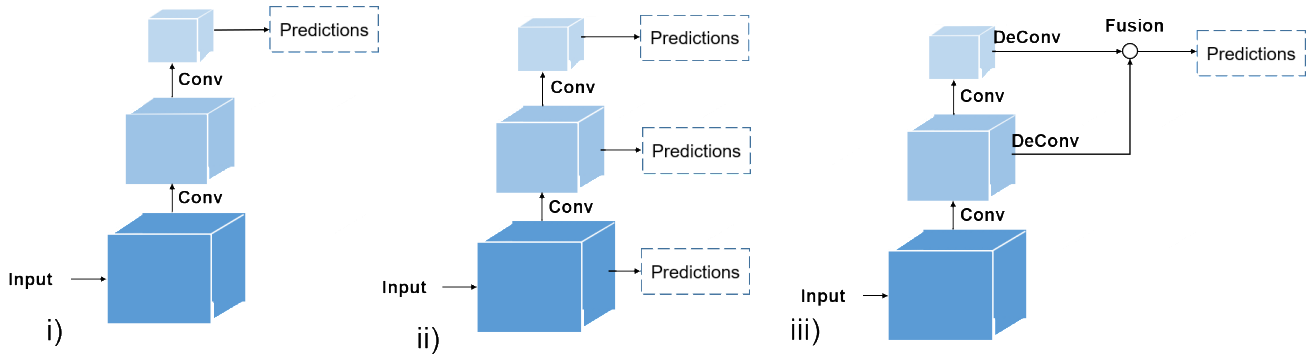

Figure 7. Different paradigms for multi-scale feature learning without lateral connections for 3D object detection in point clouds: i) Prediction Pyramid; ii) Multiple Prediction Pyramid; and iii) Integrated Features;

more layers, with the output of other remaining stacked layers) to combine spatially rich features (with high resolution but smaller receptive field) from shallow layers and semantic-rich features (with lower resolution but larger receptive field) from deep layer features. The fusion of features with different scales can be made by means of concatenation [11], [25], [29]–[31], [42], [55], [91] or element-wise summation [11], [25], [29]–[31], [55], [91]. The deconvolutional layers on multiple feature maps are performed to improve their resolutions by up-sampling features from deeper layers [31]. By reducing the number of upsampling operations, this solution achieves fast detection speeds.

Architectures featured in Figure 7 tend to lose information due to the introduction of position information errors in the structure's middle layers as a result of the existence of a single route, and multiple downsampling operations [35]. Inspired by [92], several works have proposed schemes based on feature fusion and multiple-path to improve the model's ability to generate accurate location information and consequently, reliable predictions [35]. Precise locations of features can be directly passed from the finer levels of the bottom-up maps via the lateral connections to the top-down maps, representing a major improvement over the unprecise locations outputted by the middle layers. Moreover, the top-bottom path passes the 3D semantic context downward, helping to produce high-level 3D semantic feature maps at all pyramidal levels. The possible features encoders structures that follow this principle are depicted in Figure 7 and are as follows.

(i) Encoding-Decoding Feature Pyramid with multiple predictions, this fusion scheme adopts lateral connections to merge bottom-up and top-down features [93]. Each lateral connection transmits feature maps of the same spatial size from the bottom-up to the top-down path for enriching the scale-invariant features' top-down path. This structure decodes feature maps from various scales and performs predictions/extract and outputs features at all levels of the top-down path. Thus, multiple scale-dependent classifiers are learned on these feature pyramids. The low-resolution but semantically strong feature maps are upsampled to the size of the upper pyramidal level by using the deconvolutional layers, and subsequently concatenated with the corresponding feature maps of the same spatial size;

(ii) Multi-Input Feature Pyramid implements a pyramid structure that explores the advantages brought from the previous multi-scale feature learning scheme. However, this scheme takes more than one input, corresponding to feature maps with different sizes as a result of the application of a data feature extraction method to the same point cloud, but assigned to several 2D/3D grid sizes. Consequently, the number of points per grid cell (which might be a voxel) will vary, resulting in different segment-wise feature maps with different sizes [62]. Small grid cell sizes capture finer geometry features, richer in localisation information, and thus suitable for detection of small objects, whereas larger grid cells result in smaller feature maps suitable for detection of large objects, since it covers larger receptive fields. Hence, the features extracted throughout the pyramid structure are concatenated with the inputted segment-wise features, improving the pyramid structure to preserve more localisation information and generate richer contextual information [32].

(iii) Encoding-Decoding Feature Pyramid, similar to the solution (i), but only a single classifier is learned on these feature pyramids in order to improve the inference time of the solution [36], [40];

(iv) Fusion on Fusion combines the structure of the Encoding-Decoding Feature Pyramid scheme and Integrated Features scheme to fuse features of multiple scales and concatenate all those features (after assuring that their sizes are uniform by deconvolution) of each top-down path layer [34], [35]. Here, the prediction task is performed just once and not at all layers of the top-down path, which improves the inference time, while the detection efficiency might be optimised due to a high dimensional feature map.

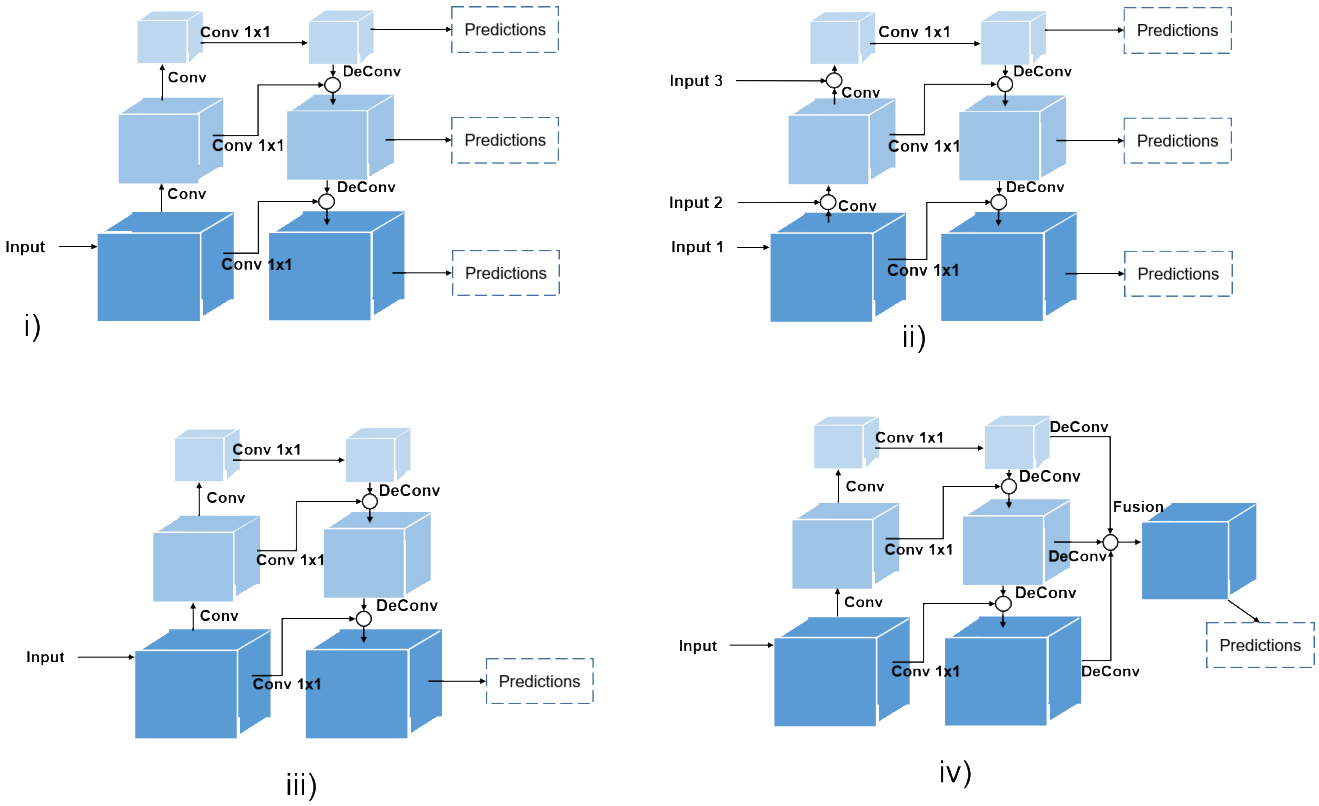

Figure 8. Different paradigms for multi-scale feature learning with lateral connections for 3D object detection in point clouds: i) Encoding-Decoding Feature Pyramid with multiple predictions; ii) Multi Input Feature Pyramid; iii) Encoding-Decoding Feature Pyramid and iv) Fusion on Fusion. Note that the depth of these structures varies from model to model, the block called Predictions only exist when feature extractor with respective structure is employed at the end of the model pipeline.

Summing up, the abovementioned structure paradigms only refer to the feature learning architecture, since models can select a point-wise, segment-wise or object-wise learning approach to extract and aggregate features through the structure paths (even though convolutions and deconvolutions are represented in Figure 7 and Figure 8). For instance, STD and Part A2 Net, stack several sparse convolutions and sparse deconvolution layers to implement a multi-scale feature learning scheme, but PointRCNN stacks several Set Abstraction layers to perform the bottom-up path while the deconvolution layers are implemented utilising fully connected layers.

Moreover, solutions might use a single extractor for learning global features or a compound solution. Table II summarises the design choices regarding the feature extraction approaches adopted by object detection models.

IV. Detection And Prediction Refinement Networks

Detection networks, identified in the general pipeline as a block (3), are expected to encode features from block (2) to generate contextual features and output the model predictions. The several proposals for block (3) found in the literature can be divided according to the architecture of the detection network. The process of detection might follows a single- or dual-stage architecture. Moreover, other detection network specifications allow us to distinguish the several networks addressed in the literature. In Figure 9, a full taxonomy of detection networks is proposed, where networks are described according to their architecture specification (c.f. subsection IV.A); their detection setting, i.e. how the localisation of the objects in the point cloud is displayed (c.f. in subsection IV.B); the technique followed to generate the region proposals/final predictions (c.f. subsection IV.C); the multi-scale feature learning scheme adopted (c.f. subsection III.E); and finally, methods used to refine the models' region proposals/final predictions (c.f. subsection IV.D), when applicable.

This section addresses the several methods adopted to build a detection network for 3D object detection models and covers the Prediction Refinement Network developments.

A. Detector Network Architecture

Regarding the Detector Network architecture, solutions can be organised into two classes (c.f. Figure 10), namely (i) **dual-stage detectors** such as R-CNN, Faster R-CNN, PointRCNN, PointRGCNN or Patch, and (ii) **single-stage decoders**, such as YOLO [94], SSD [89], PV-RCNN [33] among others.

Table II. Features Extractors adopted by object detection models that uniquely resort on LIDAR sensors.

Data Representation	Model (Year)	Local Features Extractor	Global Features Extractor		Contextual Features Extractor	
			Network	Multi-scale Feature learning	Network	Multi-scale Feature learning
Volumetric	3D FCN (2016) [28]	N.A.	3D Convolution layers	Prediction Pyramid	N.A.	N.A.
	VoxelNet (2017) [25]	Voxel Feature Extractor	3D Convolution Layers	Prediction Pyramid	2D Convolution layers (RPN)	Integrated Features
	Vote3Deep (2017)[24]	N.A.	3D CNN networks based on voting	Prediction Pyramid	3D Conv. Layer	Prediction Pyramid
	SECOND (2018)[29]	Voxel Feature Extractor	3D Sparse Convolution Network	Prediction Pyramid	2D Convolution layers (RPN)	Integrated Features
	Patch Refinement (2018)[30]	Voxel Feature Extractor	2D Convolution layers	Integrated Features	N.A.	N.A.
	PointPillars (2018)[49]	PointNet-based	2D Convolution layers	Integrated features	SSD	Multiple Prediction Pyramid
	Fast Point R-CNN (2019) [31]	PointNet-based	3D Convolutions layers	Prediction Pyramid	2D Convolution layers (RPN)	Integrated Features
	VOXEL-FPN (2019)[32]	Voxel Feature Extractor	N.A.	N.A.	2D Convolution layers(RPN)	Multi Input Feature Pyramid
	PV-RCNN (2019)[33]	PointNet-based	3D Sparse Convolution Network	Prediction Pyramid	2D Convolution layers (RPN)	Integrated Features
	MEGVII (2019)[27]	Light data augmentation	Submanifold and 3D sparse convolutions	Prediction Pyramid	2D Conv. & Deconv. Layers (RPN)	Integrated Features
	HotSpotNet (2019)[95]	Light data augmentation	3D Submanifold and 3D sparse convolutions	Prediction Pyramid	A single 2D Convolution Layers	Prediction Pyramid
	3DBN (2019) [34]	N.A.	3D Sparse Convolution Network	Integrated Features	3D Convolution layers	Fusion on Fusion
	Fusion of Fusion Net (2020) [35]	Voxel Feature Extractor	3D Sparse Convolution Network	Prediction Pyramid	2D Conv. & Deconv. layers	Fusion on Fusion
	Point A2 Net (2020) [36]	N.A.	3D Sparse Conv. & Decon. layers	Encoding-Decoding Feature Pyramid	3D Sparse Convolution layers (RPN)	Integrated Features
	HVNet (2020) [62]	Voxel Feature Extractor	N.A	N.A	2D Convolution layers(RPN)	Multi Input Feature Pyramid
Points	IPOD (2018) [39]	Point-Wise	PointNet++	Encoding-Decoding Feature Pyramid	PointNet++-based	Prediction Pyramids
	STD (2019) [40]	N.A.	PointNet++ based	Encoding-Decoding Feature Pyramid	PointNet	N.A.
	PointRCNN (2019)[10]	Point-wise	PointNet++	Prediction Pyramid	PointNet++ based	Prediction Pyramid
	PointRGCN (2019)[41]	Point-wise	PointNet++	Prediction Pyramid	GCN-based	N.A.
	LaserNet (2019) [42]	Light Data augmentation	2D Conv. & Deconv. layers	Integrated Features	N.A.	N.A.
Projection	Vehicle FCN detection (2016) [50]	Light Data augmentation	2D Conv. & Deconv. Layers	Encoding-Decoding Feature Pyramid	N.A.	N.A.
	HDNet (2018)[53]	Data Augmentation (ground estimation)	2D Conv. & 2D Conv.	Integrated Features	2D Convolution layers	Prediction Pyramid
	BirdNet (2018)[52]	Data normalization and augmentation	2D Conv. layers (VGG-16)	Prediction Pyramid	2D Convolution layers (RPN of Faster R-CNN)	Integrated Features
	RT3D (2018) [54]	N.A.	2D CNN Backbone (Resnet50)	Prediction Pyramid	2D Convolution layers (RPN of Faster R-CNN)	Integrated Features
	Pixor (2019)[55]	Light Data augmentation	2D Conv. & 2D Deconv. layers	Encoding-Decoding Feature Pyramid	2D Convolution layers	Prediction Pyramid

Dual-stage models first generate a sparse set of region proposals using a proposal generator, for instance, a Region Proposal Network (RPN) which will be later described in detail (c.f. subsection IV.B). In dual-stage models, this proposal generator identifies a set of regions to be classified and located further. Therefore, RPNs are followed by a multi-task head to score the regions per class and consequently determine which regions contain an object and are therefore set as the final output of the model. The output of this process is a set of regions called intermediate proposals. The extraction of features by mean of several processes in cascade (along the block (2) and (3) of our model pipeline) might lead to some loss of localisation precision in features. For this reason, some research works have decided to add a second stage, block (4) in the model pipeline depicted in Figure 1, as a way to improve mainly the localisation precision of the model. This stage comprises a block called Prediction Refinement Network as illustrated in Figure 1 and the classification and regression head to compensate those losses by combining different types of features to produce refinement results, as analysed in detail in section IV.D, and ultimately output the final bounding boxes.

Single-stage object detectors do not separate region proposals from the classification and bounding box regression blocks. It rather integrates all those processes as a set of connected layers and directly performs object classification and the final bounding box prediction on each location of the feature maps without the cascaded region classification step and bounding box refinement [10], [21]. For all these reasons, one-stage detectors are commonly more time-efficient and consequently have greater applicability to real-time object detection, whereas two-stage detectors tend to achieve better model accuracy.

B. Detection Settings

Object localisation is done either using rectangular cuboid or segmentation-masks (by leveraging point-wise features provided by the feature extractor component represented in the overview of the generic pipeline depicted in Figure 1).

Regarding the first approach, also called bounding-box-level localisation, the detection of objects consists in drawing a rectangular cuboid, called bounding box, to localise an object in a point cloud, and tight localisation prediction is desired for each object within the scenes. Different techniques might be used to draw and refine this bounding box over an object regarding the type of detection encoder category. For instance, [25], [29], [35], [49] rely on pre-defined sizes of bounding boxes which are set in the proposal regions step, and then their size and orientation are further refined.

Segmentation mask solutions, also called mask-level localisations can be found in models that consume the point cloud as point-based data representation. In this detection setting, the models aim at learning to classify points as foreground or background points. Therefore, the result of this model are objects segmented by a pixel-wise mask (or as hotspots/voxel-wise masks as implemented in [95]) instead of less precise coarse bounding boxes. However, for visualisation purposes, models that choose this detection scheme tend to use these masks to regress bounding boxes around it.

Detection Encoders that follow the bounding box-level localisation typically require as input a feature map and rely on bounding box annotations in training datasets to optimise its regression ability.

Mask-level localisation algorithms take as input point-wise features encoded by the backbone point cloud network and therefore require ground-truth segmentation masks for the training process. As datasets typically provide 3D bounding box annotations for object classes, these masks need to be obtained from those annotations. For instance, points inside 3D boxes are considered points that belong to the representation of an object, also known as foreground points [10], [41]. Regarding the performance evaluation of the detector, both approaches use the IoU (c.f. section V.C) between the outputted bounding box and ground truth. Since mask-level methods do not output a bounding box, models have to implement a stage in their architecture responsible for regressing a bounding box over the instanced object/segmentation masks as depicted in Figure 10.

C. Detector Module Techniques

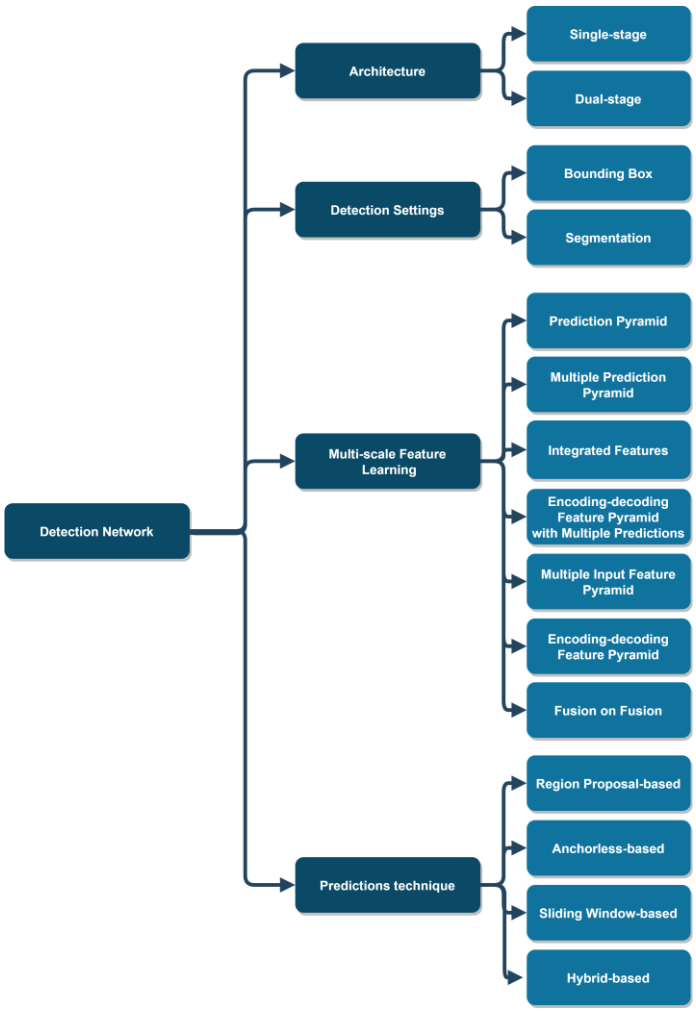

Figure 9. Proposed taxonomy for block (3) of object detection model pipeline called Detectors Network for Object Detection in Point Clouds.

This subsection addresses the main region proposal techniques adopted in the literature. We will consider four main categories, namely Region Proposal-based, Sliding Window-based, Anchorless Detectors, and Hybrid Detectors

- Region Proposal-based Frameworks

R-CNN [91] was the first model to address the detection task using region proposals in a two-stage detection pipeline. It generates Regions of Interest (RoI) using a low-level algorithm, called Selective Search [96], to produce 2k region proposals per image. The Selective Search sets off generating sub-segmentations to generate many candidate regions and following a bottom-up grouping recursively combine similar regions into larger ones to provide more accurate final candidate proposals. Each of these regions is independently submitted to the CNN module proposed in [66] to produce dimensional features. The outputted feature map is then fed to an SVM classifier to predict the object class within the candidate RoI. Along with object class prediction, the algorithm also predicts four values which are offset values of the bounding box. This project was ground-breaking at several levels, since it implemented a hierarchical feature representation and outperformed the existing state-of-the-art. It has shown that deep architectures lead to more accurate models [21], [97]. However, it also has strong shortcomings due to the adopted selective search algorithm: (1) it is not able to properly handle complex contexts as it relies on low-level visual clues, (2) regions generation is a very time-consuming task (≈2 seconds to extract region proposals) and it is inefficient (most of the proposed regions are redundant); (3) training this network is a very time-consuming task as 2000 regions per image need to be classified (model runtime is over 40 seconds for each test image); (4) it does not take advantage of GPU acceleration; (5) the training process is also expensive in terms

of space since the features extracted from the proposed regions are saved in memory.

Fast R-CNN [11] also adopts the same Selective Search algorithm above described. However, it does not feed CNN modules with every region proposal. Instead, the whole input image feeds the CNN to generate feature maps, which are hence set as an input of the selective search algorithm. Proposed regions are subjected to a feature generation stage to obtain features of the possible objects (region proposals) using a RoI pooling to reshape it so that they can be fed into a fully connected network. This network comprises two layers: a classification layer to predict which class the proposed region might belong to; and a regression layer to determine the bounding box offset values. This solution is much faster since it does not require 2k region proposals to be fed to the CNN module for every image. However, it is still considerably slower since the Selective Search algorithm is time-consuming and runs on CPU computation.

Faster R-CNN [12] proposes generating region proposals using a network called Region Proposal Network (RPN). This module aims at learning, for every feature map, whether an object is present at its corresponding location by resorting to anchors. Unlike Selective Search, RPNs consist of a separate and trainable module that predicts a set of bounding boxes concerning reference boxes of multiple sizes and aspects ratios. Therefore, for each location on the feature extractor output, this model defines a set of anchors with different aspect ratios and sizes on the corresponding input image or point cloud location.

In Faster R-CNN, by default, three scales and three aspects ratios are used, resulting in 9 anchors per location. This approach significantly reduced the number of anchors compared to solutions that preceded it, where the direct application of the detection modules techniques of 2D object detectors in a 3D space is impractical. For instance, research work AVOD [56], generates more than 80k anchors whereas the R-CNN generates 2k anchors per RGB image frame [56], [91], showing that the huge 3D search space negatively affects the run-time of a model [10]. After setting up anchors, a feature extractor is applied and the resulting anchors' features are further processed by a multi-task network head. Since it follows a dual-stage architecture responsible for extracting contextual features as depicted in Figure 10, anchor coordinates will be extracted to tune each bounding box.

RPNs were widely adopted in object detection tasks of top-performing frameworks either for object detection in RGB images [51], [93], [98], [99] or in point clouds [25], [27], [39], [49], [52], [54], [29]–[36]. This network has important properties, such as (1) cost-effective solution as it handles objects with multiple scales relying on images or point clouds on a single scale; (2) all filters within an RPN have the same size which avoids the need for creating either pyramids of images or point clouds (to extract a feature map for each one of the scaled images), or pyramids of filters with multiple scales; and (3) it is translation invariant, as an algorithm that computes region proposals follows a fully convolutional network, which ensures this property [92]; and (4) as discussed before, solutions resorting on this technique to detect objects in point clouds reduce the model's size and inference time since the number of anchors is reduced.

The maximum number of anchors adopted in this detection technique is class-dependent since a fixed number of anchors is settled per class. Typically, two or four anchors are adopted per class[29], [31], [51]. However, in [12], the authors followed a different approach where the same number of anchors is always adopted, regardless of the number of different classes the model is trained to detect.

In [28], deep modifications have been made to the original RPN architecture (also inspired by the improvements in R-FCN [100]) to extend the fully convolutional network-based detection techniques to 3D (including parallel processing) for application on point cloud data. These novel RPNs require a feature map as input (a sparse 4D tensor where dimensions correspond to channel, height and width of the 3D tensor), and uses only one anchor to determine regions proposals since objects to be detected are of approximately fixed size (or, as in related works [25], [29], where anchor geometry is determined based on the means of the sizes and centre locations of all ground truths in the training set) instead of generating thousands of anchors. Moreover, this solution implements a multi-scale feature learning paradigm based on a prediction pyramid, c.f. Figure 7. Therefore, these solutions opted by learning hierarchical features in different layers so that different scale information is captured. These solutions require as input the feature map, a sparse 4D tensor (channel, height and width of the 3D tensor), provided by the feature map encoder and outputs

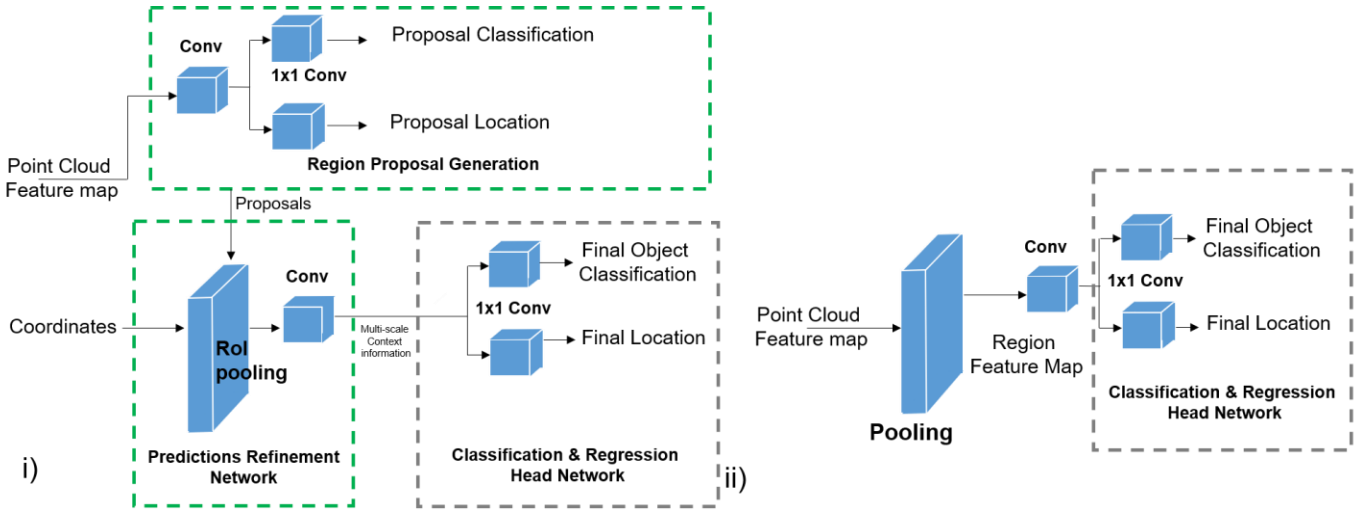

Figure 10. Generic representation of the architecture of i) dual and ii) single stage detectors.

the probability score map and a bounding box regression map. The improved RPN uses three blocks of fully convolution middle layers to downsample the inputted feature map and a block responsible for concatenating all features from each of the convolution blocks after being upsampled (to ensure that all feature maps outputted by each convolution block have the same size). The output of this block consists of a high-resolution feature map, which is then fed to two 1x1 convolution layers responsible for performing object detection (estimate the probability of object or not object for each proposal) and bounding box regression (estimates the 3D coordinates of the anchor centre location, length, width, height and orientation of the box).

Subsequent projects, such as VoxelNet [25] and SECOND [29] adopted the RPN architecture from [28], but with small changes in the sub-networks. They use two anchors per class with the same size but different rotations (0 and 90º). During the training process, anchors are assessed as positive or negative according to their IoU with the ground truth.

Single anchor-based RPNs like in [25] follows a feature extraction and feature representation architecture based on the fusion on fusion multi-scale learning paradigm described in subsection III.E. As seen in SECOND's RPN, the required input is a dense feature map from a Sparse CNN.

PIXOR [55] proposes a detection encoder following the same principle of the abovementioned RPNs, but the context feature extractor is deeper, and a single anchor over a BEV representation of the point cloud is adopted.

Fast Point R-CNN [31] proposes an end-to-end RPN, as both CNN backbone and RPN are integrated into a single network, called VoxelRPN, that follows an Integrated-Feature multi-scale learning paradigm to extract a small number of initial proposals (one anchor with four pre-defined orientations is adopted). Furthermore, a lightweight PointNet called RefinerNet takes region proposals as inputs for final box refinement. Unlike previous solutions where feature maps are taken as input, VoxelRPN consumes a voxelised point cloud.

The Single Shot Detector (SSD) follows a feed-forward convolutional network approach to perform 3D object detection [89]. This task is achieved by scoring the presence of object class instances in a set of proposed boxes. First, a feature map is extracted from the input using as backbone VGG16 [67] and an auxiliary structure, consisting of a set of convolutional feature layers to produce detections of objects at multiple scales. Unlike single shot detectors frameworks that work with single-scale feature maps, such as Overfeat [101] and YOLO [94], bounding box proposals in SSD are very similar to the schemes adopted in Faster R-CNN [12], i.e. based in the anchor box paradigm. However, SSD applies 3x3 convolutions on feature maps with different resolutions instead of using anchors to poll features, leading to a more efficient discretisation of the output space of bounding boxes. This multi-scale learning paradigm is widely called Prediction Pyramid, in which predictions taken from multiple feature maps are used to detect objects at multiple scales [21]. For each bounding box, the SSD scores the shape offsets for all classes of interest and confidences for all object categories. Such in Faster R-CNN and YOLO, before the training process of the SSD framework, it is mandatory to assign the information of the ground truth to a default bounding box to train the network accordingly. The SSD framework was inspired by the anchor-based strategy adopted in MultiBox [102], Faster R-CNN [12] and multi-scale representation [103], as the ground truth, is matched with the default box that results in the best Jaccard overlap [102]. Therefore, the SSD solution has two output layers, a classification layer, where softmax probabilities are generated for all categories (confidence loss metric); and a regression layer that relies on the Smooth L1 loss [11] for regressing the bounding box targets. Thus, the final loss of the proposed solution is the weighted sum of the regression and confidence loss.

In [30], authors follow an architecture similar to R-CNN [91], where an RPN determines the location of objects in a BEV perspective. However, the object detection and improvement of bounding box regression is carried out by a Prediction Refinement Network, called Local Refinement Network (LRN), which leverages the region proposals determined by the RPN to perform feature extraction, then regress bounding boxes and classify objects over those initial proposals.

- Sliding Window-based

The sliding window method was largely adopted in the development of object detection in computer vision. However, this method is rarely used for the detection of objects in point clouds. Some models such as [101], [104], [105], have tested this solution, but results revealed that the window search in 3D is a very exhaustive task (as a large number of candidate windows – many of them redundant – are required) leading to extremely heavy computation and, consequently, very time-consuming models (> 10s).

Nevertheless, Vote3D tries to overcome the shortcoming of sliding window schemes exploring the sparse nature of the point clouds by replacing sparse convolutions with a technique called voting, to significantly reduce computation time (<0.5s) [64].[64]. It first discretises the point cloud into small voxels, and then a sliding window is spatially executed on these feature maps. Every sliding window location contains several voxels and is seen as a potential region proposal, getting a score denoting the possibility of this region containing an object. This score is calculated as the sum of votes from only the occupied voxels that fall within the detection window, where the voxel's vote is outputted by a linear classifier (SVM) that receives as input the voxel weights and the whole feature map. With this sparse voting scheme, the filter only needs to be applied to the occupied cells in the input grid, rather than convolved over the entire grid as in standard convolutions. Finally, Non-Maximum Suppression [106] is applied to remove potential duplicate regions proposals. These models' results are above the average performance of state-of-the-art, which led to the introduction of Vote3Deep network [24]. This solution explores this voting scheme in a deeper network which, by explicitly exploiting the sparsity encountered in the input to improve the model performance achieves a performance improvement up to 40% compared to Vote3D, but penalising the inference time.

- Anchorless Detectors

Anchorless Detectors are by nature solutions that follow a mask-level detection setting. Szegedy et al. in [68] is a reference of how to reformulate the task of object detection as a Deep Neural Network (DNN)-based regression. This solution, which was developed for object detection over RGB images, outputs several masks of the objects or parts of the objects, in a multi-scale fashion, followed by a simple bounding box inference to generate proposals, also called point-wise proposals [31]. One of the problems found in the previous solutions is when there is overlapping between objects in a frame [97]. Several solutions have been proposed such as HotSpotNet [95], PointRCNN [10], PIXOR [55], VoteNet [107], SGPN [108] and 3D-BoNET [109], which are able to offer better performance when point cloud objects suffer occlusion or truncation. Anchorless Detectors do not use anchor-based regression. Instead, each point contributes to the 3D geometry reconstruction. Most of the aforementioned solutions were designed to detect indoor static objects. Only the single-stage detectors, HotSpotNet and PIXOR, and the dual-stage detector PointRCNN have been successfully applied in self-driving applications.

During the training and inference processes of anchor-based models, a large number of predefined bounding boxes are generated. For instance, the AVOD generates around 80-100k anchor boxes. These approaches might lead to heavier computation. At the same time, some pre-defined statistical-driven parameters are needed (typically obtained through time-consuming tasks). Some examples are anchor orientation and size needed to regularise the training targets and stabilise training. For instance, in SECOND, fixed-size anchors are determined based on the means of the sizes and centre locations of all ground truths in the KITTI dataset. Besides, a threshold value of IoU matching to the ground truth must be set to decrease the number of region proposals. However, this approach will aggravate the imbalance of positive and negative predictions

To overcome the downsides of region-proposal frameworks, HotSpotNet introduced a detection encoder called HotSpot-as-Object that takes feature maps gridded in voxels as input and interprets any object within a point cloud as a set of voxel samples forming a hotspot proposal [95]. Hotspot assignment is the most diverse process of this solution. This approach takes advantage of each annotated ground truth bounding box to determine the hotspots that lie in the respective object. First, objects are partitioned into quadrants according to the relative hotspot location to the centre of the object. Thereafter, two sets of points, called effective and ignoring box, are constructed. These boxes enclose high confidence points of lying in objects and points that are within a confusing area between object and background, respectively. Therefore, at the training stage, each voxel is classified as hotspot or non-hotspot, and binary classifier is applied for training [110]. This approach allows the solution to detect the presence of occluded objects since leftover hotspots are leveraged to extract 3D geometry information and consequently to assess the presence of mostly occluded objects. Besides the task of object detection, HotSpotNet model also learnt to determine in which object quadrant a hotspot fits in. According to the authors, this model feature offers a better overall 3D object detection performance, regardless of the type of class. Authors stated that the object-part relation information helps in the process of hotspot estimation, improving the convergence time of this detector. To train the quadrant classification subnetwork, authors train it with a binary cross-entropy loss function [111]. Regarding its architecture, they implemented the following three subnetworks: (1) a subnetwork optimised to predict the probability of each hotspot being part of an object category; (2) a subnetwork to detect objects and regress bounding boxes; and (3) one last subnetwork to predict the quadrant of each hotspot.

PointRCNN [10] introduces a detection encoder called bottom-up 3D proposal generation, showing that anchor-free solutions can also follow a scheme similar to region proposal frameworks, i.e. segmentation tasks are performed to generate regions to be further subjected to refinement by means of RoI pooling. Thus, this solution implements a two-stage detection module. Similarly to the approach adopted at HotSpotNet, PointRCNN transforms the object detection task in a point cloud segmentation problem but differently since it processes raw point clouds rather than dividing the point cloud in voxels. Moreover, bottom-up 3D proposal generation does not segment objects as a set of hotspots/voxels, instead, it segments the whole point cloud into the foreground and background points to further generate point-wise region proposals. To do so, two processes in parallel are performed: the segmentation process and the 3D box generation through the inclusion of a segmentation head for estimating the foreground mask and one box regression head for generating 3D proposals. The segmentation head forces the network to capture contextual information by learning how to segment the points that lay within an object through the processing of pointwise features. Every point in a point cloud must be classified as foreground or background before the training stage, with points categorised as background being ignored. As for the location of objects, this model follows the mask-level detection setting, thus 3D box generation is regressed according to the location of the foreground points. Once again, only this type of points is treated as a regression target. This solution adopts a bin-based 3D box regression and several 3D boxes are generated. These are fed to the bottom-up 3D proposal generation's second stage, where bounding boxes and classification head are refined.

PointRGCN[41] implements a model very similar to PointRCNN but the box refinement is replaced by a GCN-based scheme.

Point A^2 [36] which is inspired by the PointRCNN pipeline, takes advantage of a voxel-based feature learning method with better recall than point-based learning methods [36], by replacing the PointNet++ backbone with a 3D voxel CNN backbone inspired by the backbone of SECOND [29]. Therefore, the segmentation head found in PointRCNN receives as input data the discriminative point-wise features provided by a sparse convolution network. The results show that this approach leads to better performance in both BEV and 3D detection trials.

- Hybrid Detectors

Hybrid detectors, such as STD [40], rely on both anchors and point masks to get region proposals. A dual-stage model is introduced, where region proposals are obtained through a bottom-up proposal generation network called Proposal Generation Module, which can be divided into two tasks. First, anchors are settled by seeding every point with a spherical anchor and the result is filtered to eliminate redundant anchors (final number of anchors is typically ≈500). Secondly, semantic context features and coordinates that fit within the previous output (region proposals without orientation/spherical anchors), feed a PointNet network to predict classification scores, regression offsets and orientations.

While traditional anchor-based region proposal solutions pre-define the orientation of the cuboid anchors, these models also aims at predicting their orientations. For this purpose, authors introduce the concept of spherical anchors that not only preserve accurate location information but also allow reducing the number of anchors to about 50%. Moreover, the number of anchors has been reduced even more thanks to the abovementioned filtering strategy, which consists of utilising the Non-Maximum Suppression (NMS) method. As described in subsection V.B, this method requires the use of the anchors IoU. For this purpose, the authors implemented the PointNet++ model to extract semantic context features and compute the abjectness score for each point, assigning a final score per anchor.

D. Prediction Refinement Network (PRN)

Dual-stage object detection models, such as Patch [30], Fast Point-RCNN [31], PV-RCNN [33], Part-A^2 net [36], STD [40], PointRCNN [10], and PointRGCN [41] naturally take RoI features of each box proposal from RPNs and then optimize the imperfect bounding box proposals from previous stages by predicting and fixing the size and location (centre and orientation) residuals relative to the input bounding box predictions. The Prediction Refinement stage also called bag-of-specials [112], is not commonly found in single-stage networks. However, since this type of network typically offers lower precision but better inference time, some research works have added a prediction refinement network to the end of the single-stage object detection model [33] to overcome this limitation. Due to the initial low number of predictions, the prediction refinement network is a very time-efficient process.

Traditional Dual-stage approaches, such as F-PointNet [113], Faster R-CNN [12], or single-stage ones, for instance, VoxelNet [25], SECOND [29] refine the region proposal by computing the semantic features obtained from the model feature extractor. However, consecutive convolutions and downsampling operations hinder the precise localisation signal that originally exists in point clouds and that is fundamental to ensure an optimal object localisation. Although for localisation purposes, a few background points within the bounding box predictions might be acceptable, for the implementation of SLAM -Simultaneous Localisation and Mapping (relevant in the context of self-driving), it might significantly reduce the solution efficiency.

Recent advances in the Refinements stage have shown that the inference process can also benefit from important context information with precise localisation information. Research works upon 2018 usually consider all information provided by the first stage (semantic features) as well as the raw point cloud (so that point coordinates can be preserved and augmented) for a better localisation quality as showcased through the generic pipeline in Figure 1. The output of the first stage (RPN output), which is obtained by extracting features through a sequence of convolutions (as shown in the subsection dedicated to multi-scale feature learning), captures the local geometric structure of objects and gradually gathers them in a hierarchical way, leading to a much larger receptive field to improve the prediction task. Consequently, prediction refinement subnetworks leverage those features as well as the raw point cloud features (and/or local features) to learn more localisation information with enlarged receptive fields.

The Refinement process can be described as a four-fold process. First, RoI's interior points are randomly chosen and coordinates are converted to canonical coordinates to ensure translation and rotation invariance [31], [36], [40] and proper local spatial feature learning [10], [41]. Second, additional information might be included, for instance, PointRCNN, PointRGCN and Part A^2 Net include also the distance of the point to the sensor, along with some extra features (laser reflection intensity and point segmentation mask) and canonical coordinates of local feature points to improve local feature learning. This parameter compensates losses on depth information of far-away objects that have much fewer points than nearby objects. In [10], [30], [31], [40], [41], opt by expanding the RoI to include more points per proposal that bring extra context from objects surrounding the region proposal. Third, local and global features are concatenated and fed into a contextual feature network – which might be based on the structure of PointNet [31], PointNet++[10], GCN-based [41] or VFE network [30], [40] - to extract richer contextual features [10], [31], [41]. Local features might be subject to a block responsible for encoding them to the same dimension of the global features, as shown in PointRCNN where a lightweight version of the PointNet is employed. The result of the later step is a discriminative feature vector, which is used in the last step of the refinement network. Here, the outputted feature vector is fed into a confidence classification and box refinement head that relies on either traditional residual-based methods [25], [40], versions optimised by smooth-L1 loss function for angle regression loss [29], [32], [33], amodal box estimation PointNet [40], or bin-based 3D box regression losses [10], [36], [41].

PointRGCN set the PointRCNN model as the core of its pipeline and suggested some changes to the latter stage of the generic pipeline, i.e. block (4), to refine the PointRCNN outputs. In this context, two GCN-based PRN blocks are suggested called (1) R-GCN, which consists on a GCN comprising MRGCN based layers [81]; (2) C-GCN, which follows an architecture based on EdgeConv layers. In order to analyse the PRNs performance when combined with the PointRCN model, two deep learning models were proposed: (1) R-GCN and (2) PointRGCN. The former model set the GCN (1) as the PRN of its model, followed by a *per-proposal* graph representation; whereas in PointGCN, authors implemented a model that contains a PRN block which combines those two GCN-based PRNs, namely R-GCN and C-GCN following the paradigm *per-proposal* and *per-frame* graph representation.

These features are first subjected to a GCN-based block, called C-GCN, before arriving at the regression head. The C-GCN block shares contextual information between all RoIs of the scene to detect more consistent and coherent bounding boxes. In this context, the network first gathers the features from all RoIs into a single graph, resulting in a single representation of the current frame that embeds contextual information between the objects, and feeds it into a network comprised of three layers of EdgeConv [85] to perform the second context information extraction of the refinement network. Finally, global features and RoI's local features are concatenated.

Whereas Prediction Refinement Subnetworks addressed in PointRCNN [10], PointRGCN[41], Fast Point-RCNN [31], etc. rely on the object detection pipeline, since it consumes features from previous stages as depicted in Figure 1, research projects Patch [30] and STD [40] favour the independence of the prediction refinement. The purpose of this approach is to take advantage of its independence to improve the training process (since it can be focused) and model performance assessment (subnetworks ability can be assessed without the influence of an RPN), and enable the application of additional augmentation techniques [30], such as Surface Sampling, Random Cropping, as well as standard augmentation techniques such as scaling and mirroring of the object individually. Therefore, local features and higher-level features need to be extracted from region proposals identified by an RPN. To do so, these subnetworks follow an architecture analogous to the architecture of the RPNs, while respecting the refinement methodology. Both Patch and STD discretize the RoI in voxels and might concatenate features of canonical coordinates and semantic features of these points [40], or just preserve the absolute point coordinates before performing local feature extraction based on the VFE block [25] for each voxel. STD concatenates features of canonical coordinates, and semantic features of these points are used for each voxel, whereas Patch preserves the absolute point coordinates. Finally, a fully connected network is applied to generate feature maps of different resolutions, which are then concatenated after assuring that they have the same dimensions.

Research project PV-RCNN [33] consist of an object detection model that follows the traditional pipeline of single-stage models. However, in order to optimize its final object location, the authors included a Prediction Refinement Network to the edge of the pipeline, called RoI-Grid pooling module. Unlike the aforementioned solutions, this model summarizes the whole scene into a small number of keypoints. Therefore, instead of consuming canonical locations and RPN outputted features for context feature extraction, a light-weight PointNet version with pooling is fed with RoI's point features (containing the relative point position to neighbouring keypoints) and the global features extracted from a Voxel Set Abstraction (VSA) module. This model is inspired by the aggregation abstraction process addressed in PointNet++ [38]. It aimed at encoding multi-scale semantic features from the 3D CNN feature volumes. To do this, the features of each keypoint are aggregated by grouping the neighbouring voxel-wise features.

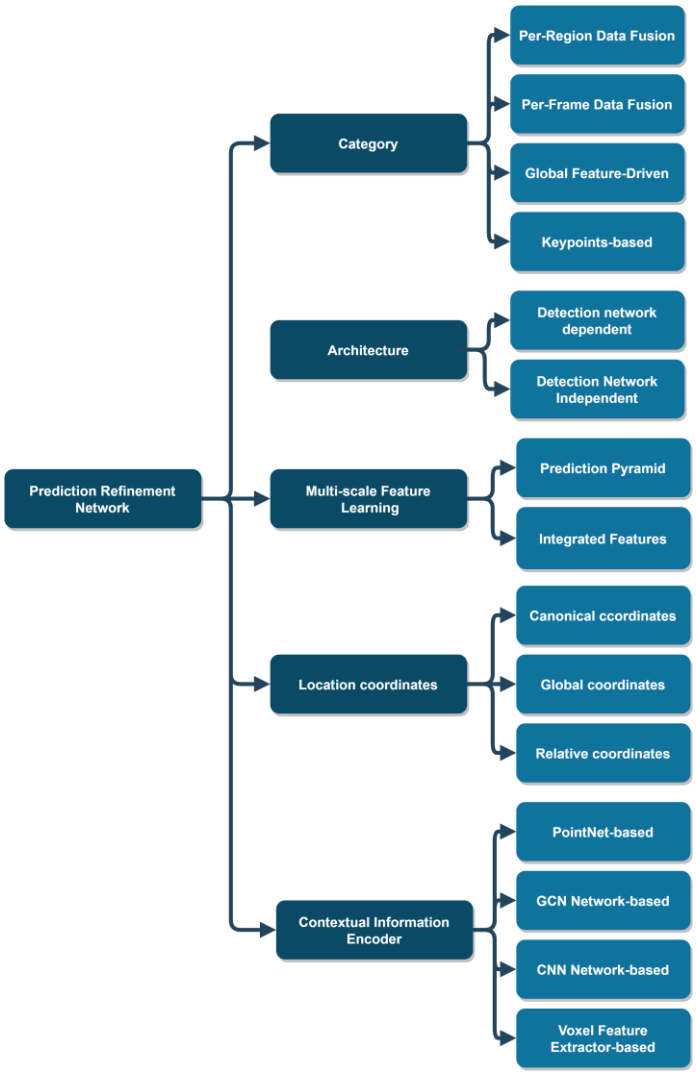

Figure 11. Proposed taxonomy for block (4) of object detection model pipeline called Prediction Refinement Network (PRN).

Projects might also opt by enlarging the area of the region proposal received from the previous stage, in order to increase the number of points and add more context to the region. Projects [30], [41] conducted experimental trials by testing and comparing the performance of the same model by varying the enlarging margin from 0.5m up to 2m in steps of 0.5m. Results show that enlarging the region proposal area leads to significant improvements, with an increase of average precision for hard difficulty on KITTI of nearly 7%. Results allow us to conclude that a higher margin does not necessarily mean better performance, as the model's maximum performance is observed for a margin of 1 meter in [41].

Summing up, refinement subnetworks aim at capturing each contextual information with lower losses of information, which is achieved by incorporating the advantages of local and global features. The former features preserve accurate location information with flexible receptive fields, whereas global features are extracted from Integrated Features, Encoding-Decoding Feature Pyramid, Multi Input Feature Pyramid, or Fusion on Fusion multi-scale feature learning methodologies that aggregate features with different receptive fields. Figure 11 displays a taxonomy proposed to study, analyse, and compare the models' design choices displayed in Table III. This taxonomy classifies PRNs addressed in the literature in four possible categories: (1) Per-Region Data fusion, which refers to the abovementioned models where localisation features from points within region proposals and global features are concatenated and aggregated to extract richer contextual

Table III. Detailed description of Predictions Refinement Network addressed in the literature and adopted by 3D Object detection models.

Year	Model	Category	Independent from Pipeline?	Type of Location coordinates	First stage features provider	Multi-scale Feature learning scheme	Contextual Information Extractor (block 4)
2019	PointRCNN [10]	Per-Region Data Fusion	No	Canonical coordinates	PointNet++	Prediction Pyramid	PointNet++
2019	R-GCN only [41]	Per-Region Data Fusion	No	Canonical coordinates	3D voxel CNN	N.A	GCN-based
2019	PoinRGCN [41]	Per-frame Data Fusion	No	Canonical coordinates	3D voxel CNN	N.A	GCN-based
2019	STD [40]	Per-Region Data Fusion	Yes	Canonical coordinates	PointNet++	N.A	Voxel Feature Extractor
2019	Fast Point-RCNN [31]	Per-Region Data Fusion	No	Canonical coordinates	2D Convolution layers (RPN)	Integrated Features	PointNet-based
2019	Patch [30]	Per-Region Data Fusion	Yes	Global coordinates	N.A	Integrated Features	Voxel Feature Extractor + 3D & 2D convolutions layers
2019	PV-RCNN [33]	Keypoints-based	No	Relative coordinates	2D Convolution layers (RPN)	Prediction Pyramid	PointNet++
2020	Part-A^2-anchor[36]	Per-Region Data Fusion	No	Canonical coordinates	3D Sparse Convolution layers (RPN)	Integrated Features	3D Sparse Convolution
2020	Part-A^2-free[36]	Per-Region Data Fusion	No	Canonical coordinates	3D Sparse Convolution layers (RPN)	Integrated Features	3D Sparse CNN

information; (2) Per-Frame Data Fusion, where features across all region proposals in a frame are gathered to ensure contextual consistency in the road scene to end up refining all proposals; (3) Global feature-driven, which refers to the traditional single-stage model, or dual-stage where the refinement process only consumes the semantic features provided by the feature extractor located at the first stage; and (4) Keypoints-based [33]. Subsection IV.D offers a description of the Prediction Refinement approach adopted by each model, where the following model's information is featured: (1) Refinement network category; (2) its independence to the model pipeline; (3) the source of the global features; (4) the type of location coordinates and extra parameters included in the set of local features; and (5) the model adopted to operate as the context information extractor.

V. Methodologies for the development of an Object Detection Model

As analysed before, several choices need to be made during the development of a 3D object detection model, such as the methods adopted at each stage of its pipeline. However, Figure 12 shows a set of other aspects that require special attention as they can have a strong impact on model performance. In this context, learning and testing strategies can be applied, assessed against different benchmarks, and compared with other model's performance regarding several metrics. This section addresses some approaches that can be followed as part of the model development methodology to improve its effectiveness.

A. Benchmarks

The self-driving platforms used to build multimodal datasets are multi-sensor moving platforms designed to research the following tasks: stereo, optical flow, visual odometry/SLAM, 3D object detection and 3D tracking [114].

These platforms were used to collect data by driving around one or more cities having dense traffic and highly challenging driving situations [115]. The routes used were selected to capture a diverse set of locations (urban, residential, nature and industrial), time (day and night) and weather conditions (sun, rain and clouds). The dataset A2D2, provided by AUDI, and nuScenes, provided by APTIV are, among the set of datasets analysed in Table IV, the ones that offer more diversity of scenarios, with data being acquired in three different cities for the former dataset and two cities for the later, but in different countries. However, A2D2 only comprises data acquired during the day, but different weather conditions are still considered.

As shown in Table IV, all platforms included sensors such as RGB cameras, LiDAR, GPS, motion sensors, as well as radars. In this context, the A2D2 benchmark stands out, since it includes additional sensors to provide more information about car states, such as translational/rotational speed and acceleration, steering wheel angle, throttle and brakes. Some pioneer research works, such as [116]–[118], have leveraged this additional information by providing it as input signals to their models, in combination with perception information. Their aim is generating end-to-end models to output commands for control of the vehicle's steering. This task is less demanding in terms of the sensor's data annotation. All mentioned setups ensure a horizontal field of view (FoV) of 360º through a single LiDAR sensor equipped at the top of the vehicle [119]–[121] or several LiDAR sensors with shorter FoV settled at several strategic points of the vehicle (for instance front, sides and rear) [122], [123]. Solutions also differ in terms of the LiDAR sensors adopted, since LiDARs with 16 [123] (capturing approximately 50k points per cycle), 32 [114] (70k points per frame) and 64 (more than 100k points per sensor reading) [119], [120], [122] beams/channels are adopted. However, the sensor measurement range is similar among the solutions, between 70 and 100 meters. Sensors comprised by the setup are typically

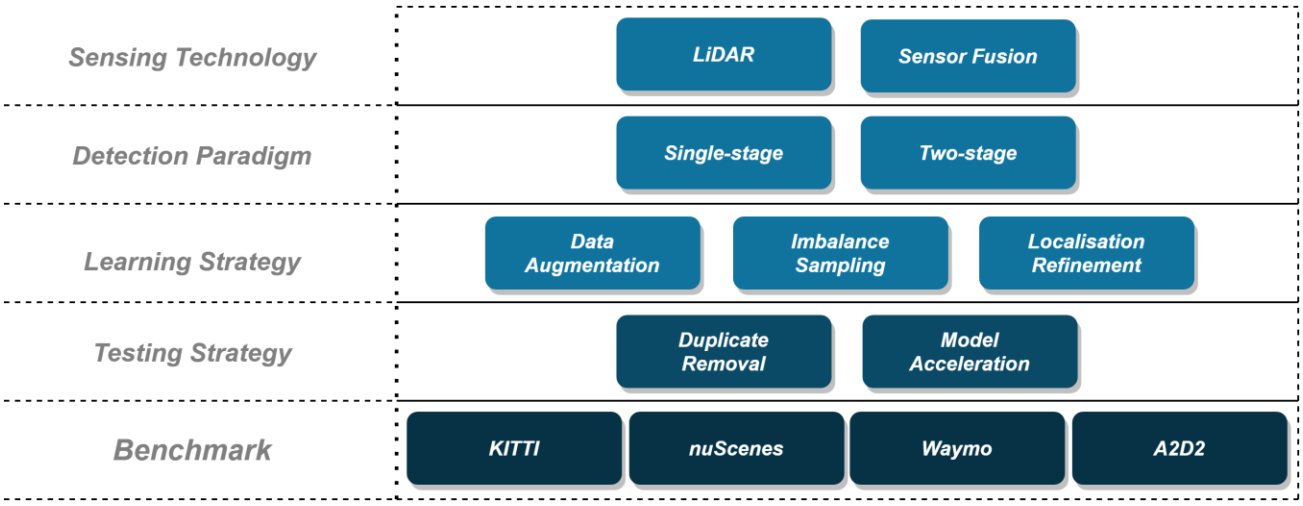

Figure 12. Taxonomy of key methodologies in the development of a 3D object detection model.

triggered at ten frames per second by the LiDAR (when facing forward) with shutter time adjusted dynamically (maximum shutter time of 2ms). Waymo [122] and nuScenes [121] acquired data at a rate of 2Hz and 20Hz, respectively.

Although KITTI [119], [120], nuScenes [121], Waymo [122] and A2D2 [123] address other aspects such as sensor calibration and synchronisation, odometry, etc., we will only focus on perception-based information.

In this context, the solution that provides more annotated frames and labelled objects is the Waymo dataset since it has an acquisition frequency of 2 Hz. A2D2 and KITTI offer much less annotated frames and labelled objects. These datasets use a bounding box-based labelling methodology, which encapsulates seven different characteristics: x, y, z, width, length, height, and yaw angle, for each object. One of the main limitations of these datasets is the disproportionate ratio of observations in each class, which forces models to implement solutions to ensure a balance between the number of objects per class at the training stage, as further analysed in subsection V.B. For instance, on the Waymo dataset, there are about 6.1M labelled vehicles against just 2.98M, and 67k labelled pedestrians and cyclists, respectively [122].

All benchmarks listed in Table IV organise datasets for training and validation purposes. For instance, the KITTI benchmark consists of 7.481k training images and 7.518k test images as well as the corresponding point clouds, comprising a total of 80.256k labelled objects [119], [120]. It is also essential to notice that benchmarks use different metrics to rank solutions. In subsection V.C more information can be found about these metrics. These benchmarks also comprise different classes, for instance, KITTI

Table IV. Comparison of the setup and which conditions data was collected in multimodal sensor approaches. Table edited from [123].

	Kitti [119], [120]	**NuScenes**[121]	**Waymo** [122]	**A2D2** [123]
Lidar Sensor	1 (64 channels)	1 (32 channels)	1+4 aux. (64 channel)	5 (16 channel)
Horizontal FoV(degrees)	360º	360º	360º	360º
Cameras	4 (0.7 MP)	6 (1.4 MP)	3(2.5 MP)+ 2 (1.7 MP)	6(2.3 MP)
Vehicle Bus Data	GPS+IMU	-	Velocity and angular velocity	GPS, IMU, steering angle, brake, throttle, odometry, velocity, pitch roll
Location	urban, one city (Karlsruhe)	urban, two cities (Boston and Sinagpore)	3 urban regions (USA)	urban, highways, country, roads, three cities in Germany
Hours	day	day, night	day, night	day
Weather	sunny, cloudy	various weather	various weather	various weather
Objects	3D	3D	3D, 2D	3D, pixel
Last Update	2015	2019	2019	2020
Nº classes	3 (car, pedestrian and cyclist)	Up to 23 (“animal”, “human.pedestrian.adult”, “vehicle.bicycle” or “vehicle.emergency.police”, “vehicle.moving”, “pedestrian.standing” or “pedestrian.moving”, etc.)	4 (vehicle, pedestrian, cyclist and sign)	14 (car, truck, pedestrian, cyclist, Van, Bus, Trailer, motorcycle, Emergency vehicles, animals among others)
Annotated Frames	20k	40k	230k	12k
3D Boxes	200k	1.4M	12M	N.S
Size (Hours)	1.5	5.5	6.4	N.S
Frames per second	10	20	2	10
Average points per frame	120k	34k	177k	N.S

consists of three classes - car, pedestrian and cyclist, whereas nuScenes [121] comprises 23 classes of objects, encoded a certain hierarchy that enables identification as specific objects from a broader group. Finally, Waymo and A2D2 comprise 4 and 14 different classes, respectively.

Table IV offers a comparison between the most recent and widely adopted benchmarks, however, there are other current benchmarks, such as H3D and AS LiDAR. Information about these benchmarks is available in [124] and [125], respectively.

B. Learning Strategies

- Data Augmentation

In this section, we will discuss strategies for data augmentation during the training stage. These strategies are generally applied to obtain a more balanced dataset, which can contribute to better object detection performance and prevent overfitting.

NuScenes and KITTI are the most commonly used datasets for object detection while being severely imbalanced. Several strategies have been proposed to alleviate this issue, with some being applied over the whole point cloud [30]; point cloud and ground-truth boxes [29], [49], or just ground-truth boxes [10], [24], [25], [52], [55], [95], [107]. According to the authors in [30], per-object object transformations (ground-truth boxes transformation) might introduce self-occlusion artefacts. For this reason, projects perform ground-truth transformations, for instance [10], [29], [49], as a means of verifying whether new sampled boxes are overlapping existing ground-truth boxes. It allows for avoiding physically impossible outcomes. Therefore, cases, where instances collide with each other, are removed.

Data augmentation can be applied following two strategies [24], [25], [52], [55], [107]: (1) on-the-fly augmented training data, where data does not need to be saved onto a disk or (2) extract points within ground-truth boxes of vehicle categories from an external point cloud to another point cloud [10], [27], [41], [49], [30].

Regarding the transformations of the ground truth instances, different transformations have been adopted, namely translation of original ground truth instances on the x-, y- and z- axis [24], [25], [29], [49], [52], [55], [95], rotation over the z-axis [29], [49], [55], [95] and instance scaling [10].

Regarding global transformations, the following changes have been applied in the literature, random global mirroring along the x-axis[49], global rotations to the z-axis [30], [49] and scaling; and global translation to x-, y- and z-axis [49]

To avoid few ground truths in rare categories, most of the projects have adopted one of the following strategies: (1) GT-AUG [29], which has been widely adopted [10], [27], [29], [49], where firstly a database with all instances of a category with the associated label and point cloud is generated. Then, ground truth instances are randomly sampled and associated with other point clouds; (2) duplicating the samples of a category according to its fraction of all categories [27]. Thus, the categories with fewer instances have more instances duplicated.

- Imbalance Sampling

While training a detector, typically a large number of RoIs are proposed to generate final box predictions. However, most RoIs are just background images (negative samples). In two-stage detectors, many negative samples are filtered out and reduced using cascaded classifiers. On the other hand, the one-stage detectors don't include a filter strategy and don't provide cascaded classifiers, which introduces high imbalance between foreground (positive) and background (negative) samples. Thus, the background samples that dominate the gradients during training will be easier to classify, while the objects become harder to classify. Several methodologies were introduced to solve this issue.

As mentioned before, most two-stage detectors such as R-CNN and Fast R-CNN provide a filter strategy where the majority of negative samples are rejected. In Fast R-CNN [11], the negative proposals are randomly sampled to reduce the ratio between negative-positive samples. It rejects most negative samples and generally keeps 2k RoI for further classification. The remaining proposals are then randomly sampled to keep the ratio of positive and negative at most 1:3. On the other hand, the single-stage detector SSD [89] proposed a hard negative sampling strategy, which involves sorting the negative examples using the highest confidence loss for each default box and pick the top ones. It allows reducing the ratio between the negative and positive samples at most 3:1. This random sampling strategy can solve the imbalance between positive-negative samples but cannot fully utilise the information of negative samples. While hard samples can increase detection accuracy, negative samples can have rich context information about the image.

Thus, the typical solution is hard negative mining [126]. In [127], a hard negative mining methodology, similar to SSD, called online hard example mining (OHEM) is proposed. This methodology automatically selects hard examples for training, ignoring categorical information and only considers difficult information, which means that the ratio of foreground-background samples is not fixed. Thus, for this model, difficult samples play a more important role than class imbalance in object detection.

RetinaNet [110] introduced the focal loss, an effective single-stage loss function to solve this problem of imbalance between negative and positive samples using hard negative mining. Instead of discarding negative samples, they proposed a methodology to suppress the gradients of easy negative samples by penalising these samples. This includes assigning parameters that represent the importance weight of each example. The focal loss function is as follows:

$$FL(\rho_t) = -\propto_t (1 - \rho_t)^{\gamma} \log(\rho_t), \tag{2}$$

Where ρ_t is the model's estimated probability, $\propto$ and γ are the parameters to control the importance of weight. Suppressing the gradient signals of easy samples during the training process leads to the model focusing more on hard samples.

Inspired by RetinaNet, Li et al. [128] proposed a gradient harmonizing mechanism (GHM) to balance the contribution of each kind of example to the gradients. It focuses on the harmony of the gradient contribution of different examples. It performs statistics on the number of examples with similar attributes, considering their gradient density and then applies a harmonizing parameter to each example accordingly. Thus, the cumulated gradient produced by easy examples can be largely down-weighted as well as the outliers (affect model stability since their gradients may have a large discrepancy from the other common examples). This contributes to more efficient and stable training. For this purpose, they define a reformulation of the loss function for classification loss, named GHM-C loss and for bounding box regression GHM-R loss.

- Localisation Refinement

Object detection models addressed after 2018 are focused on significantly improving detection efficiency, sometimes sacrificing inference time. What was before the final predictions are now seen as intermediate predictions which are subjected to a refinement process, turning models into a dual-stage approach. In subsection IV.D, the most recent developments in prediction refinement are discussed, showing that poor results in object localisation, i.e. residual errors in the bounding box regression, and orientation, are the consequence of the depth of the models since several layers for extraction of features are applied which weaken the geometric structure localisation information. Therefore, several solutions have opted to fuse coordinates of each interior point with global features from the previous stage. Analogously, the sequencing process of features extractors conducted in the first stage might be repeated within the Prediction Refinement Network, as shown in Patch [30]. However, this time the whole point cloud is not processed, as only the information of the regions referent to the predictions are processed. The former single-stage PV-RCNN [33] shows the gain in efficiency achieved by the Prediction Refinement Network. According to authors, although the inference time will increase, this subnetwork is time-efficient as the number of proposals and consequently, the number of points of interest, are typically very low. Moreover, solutions such as Patch [30] and STD [40] implement a refinement network fully independent of the previous pipeline stage, which increases the inference time but brings more options in terms of either training and testing approaches so that better results are achieved. Another technique to improve the effectiveness of this process is the enlargement of the region proposal identified at the first stage of the model pipeline as shown in PointRCNN [41] and Patch Refinement [30].

C. Evaluation Metrics

Assessing the performance of a computer vision model requires evaluation metrics to be defined according to the specific task at hand. Object detection metrics, especially those applied to the self-driving task, rely on object-oriented 1) detection, 2) localisation, 3) classification and 4) inference time metrics.

- Detection Metrics

To determine the quality of the detection process, a few notions are introduced: True positives (TP), the predicted objects that belong to the ground truth; True negatives (TN), which describe the lack of detection of non-existent objects; False positives (FP), referring to the prediction of objects that do not exist in the ground truth; and False negatives (FN), cases in which the model does not detect an object that exists in the ground truth. False negatives and true positives can also be represented on a confusion matrix. This structure is introduced since the selected elements by themselves do not give enough information on the quality of the overall object detection model. Precision and Recall, two widely adopted object detection metrics, can be derived from these categories. Precision, which is given by equation (3), describes the ratio of true positives relative to the selected objects, i.e. the percentage of selected objects that are correctly identified.

$$Precision = \frac{TP}{TP + FP}; \tag{3}$$

Recall expresses the ratio of current detections relative to the number of existing ground-truth objects of a class and is given by equation (4) as follows:

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

- Localisation Metrics

To perform object detection, the objects detected by the model, also need to be correctly localised in space. As such, a metric that considers the localisation effectiveness becomes of interest to this application. Since these metrics are usually linked to specific benchmarks, each with their criteria, different metrics to the same purpose can be found in the literature. For instance, in the KITTI Benchmark [114], a metric called Intersection over Union (IoU) is adopted for evaluating the location, orientation and size of the box compared to the ground truth bounding box. Distinctively, the nuScenes Benchmark [115] resorts to three different metrics for this task: Average Translation Error (ATE), Average Scale Error (ASE) and Average Orientation error (AOE).

- Classification Metrics

The most used metric for model comparison in object detection is mean Average Precision (mAP). The Average Precision (AP) metric is obtained using a Precision-Recall curve and is the equivalent to the integral of the precision as a function of recall. Generally, when evaluating object detection using the average precision metric, an acceptance criterion is used for each bounding box and this threshold varies according to each benchmark. Regardless of the thresholding method used, the AP is usually averaged over a set of classes, and matching thresholds of IoU, for KITTI, or distance, for nuScenes, resulting in the mean Average Precision (mAP).

- Inference Time Metric

Object detection applications such as self-driving, along with the detection quality aspect of the model, are also highly dependent on its timing characteristics. To account for the time that a model takes to perform the detection task a metric called Inference Time, usually given in ms or Hz, is adopted in most benchmarks.

VI. Object Detection Models Comparison

A. Fusion-Based Solutions

Several methods, such as Frustum PointNet (F-PointNet) [46], Multi-View 3D Object Detection (MV3D) [51], PointFusion [43], Aggregate View Object Detection network (AVOD) [56], RegiOn Approximation Refinement Network (Roarnet) [44], ContFuse [58], Multimodal Fusion Detection System (MFDS) [26], Multi-task Multi-sensor Fusion (MMF) [59], SCANet [57], PointPainting [45], SIFRNet [48], Complex-Retina [60] and LaserNet++ [61] propose a combination of images and LIDAR data to improve object detection accuracy. It allows for performing object detection in difficult scenarios, such as classifying small objects (pedestrians, cyclists) or distant objects, which is one of the limitations found in LiDAR-based object detection models. On the other hand, cameras provide dense pixel information and an order of magnitude more measurements than LiDAR, however, no depth information is provided, which turns the task of 3D prediction from 2D images over long distances very challenging. Fusion-based models complement information from cameras with the rich depth object information provided by a LiDAR sensor. The information below describes several models addressed in the literature that explore the advantages of both types of sensors for the detection of objects within the application of autonomous vehicles. In contrast, Table V summarizes the components comprised in those solutions. Also, we provide Table VI to compare the results of the 3D object detection models achieved in KITTI benchmark for 3D AP (in percentage) on KITTI test set for easy (E), moderate (M) and hard (H) difficulties in three categories, namely pedestrian, car and cyclist. These difficulty classes are related to the level of truncation, occlusion and bounding box size of the objects, in which hard (H) means objects with high occlusion and truncation, as well as small bounding box size.

It is important to note that we categorise these research projects into three categories according to the type of data fusion performed. Thus, projects belong to one of the following categories: (1) Decision Level fusion, (2) Feature Level and (3) Decision-Feature Level fusion. The first type, called (1) Decision Level Fusion, regards to models that typically perform independent detections (LiDAR and RGB detection) and then combine both inputs to generate a single set of superior output detections. On the other hand, (2) Feature Level Fusion methods project LiDAR data into a 2D space and then process the RGB image and LiDAR projection inputs with a feature extractor. After that, they combine both features to complement the representation of the scenario. Along with these two types, there is (3) Decision-Feature Level fusion, in which image detectors are used to delimit 3D search space and create detections from LiDAR point clouds. This method uses LiDAR data and image detections jointly, to provide richer 3D detections.

As shown in Table V, most of the methods in the literature typically use the feature level fusion category, which means that LiDAR data is projected onto a 2D space. Therefore, they project LiDAR data in BEV, in a single view manner, while the model MV3D [51] produces a multi-view representation of 3D point clouds. Its idea is to combine features from the front view, BEV, and camera view to alleviate the information loss. Models that fall in this group set CNNs as its feature extractor to extract features from both RGB images and LiDAR point cloud, with ResNet-50 and VGG-16 being the preferred backbones. Then, the features are fused and projected onto the respective view space using a fusion network. Afterwards, a detection header network is responsible for producing BEV detections. Within this group, models such as PointFusion and PointPainting, instead of using a projection scheme, they process the LiDAR point cloud directly using PointNet and PointNet++, respectively. PointFusion [43] combines point-wise features provided by PointNet in 3D point clouds, image features using a ResNet architecture, and combines both outputs in a fusion network. PointPainting [45] uses an image semantic segmentation network to extract features as class probability scores and projects them along with the LiDAR point clouds by appending the score to each point. Afterwards, a detection header provides 3D oriented boxes.

Other models, resort to the usage of off-the-shelf 2D object detectors to propose 2D regions from the image. Generally, the models that use this type of data fusion, named Decision-feature level fusion, lift these 2D regions to a 3D point cloud, becoming frustum proposals, also called Frustum-cloud. Instead of using 2D regions to propose frustums, RoarNet [44] uses a RoarNet 2D network based on 2D CNNs, to propose 3D regions. MFDS also uses 2D object detectors to propose a set of possible objects, but they pair these detections with LiDAR point cloud clusters, which are formed based on Euclidean distance [26]. The approaches presented here, typically use PointNet++ or PointNet to learn point cloud features and segment object instances. Point-wise features inside each frustum or region are aggregated in a 2D feature map and then forward passed to a header network that estimates 3D oriented boxes.

Table V. Design choices of the Fusion-based models addressed in the literature for the application of driverless cars.

Data Representation	Model (year)	Data Fusion Type	Data Feature Extractor		Detection Encoder				
			RGB	Point Cloud	Feature-Map Encoder	Category	Architecture	Multi-scale Feature Learning	Detection Settings
Volumetric	MFDS (2019) [26]	Decision-feature Level Fusion	2D CNN Backbone (MobileNetv1)	Point-wise	2D CNN-based	Region Proposal-based	Single-stage	Multiple Prediction Pyramid	Bounding-box
	Sang-Il et al. (2017) [129]	Decision level Fusion	2D CNN Backbone	2D CNN Backbone	2D CNN-based	Region Proposal-based	Single-stage	Integrated Features	Bounding-box
Points	PointFusion (2017) [43]	Feature Level Fusion	2D CNN Backbone (ResNet)	Point-wise (PointNet)	PointNet-based	Sliding Window-based	Two-stage	Prediction Pyramid	Bounding-box
	RoarNet (2018) [44]	Decision-feature Level Fusion	2D CNN	Point-wise (PointNet)	PointNet-based	Sliding Window-based	Two-stage	Prediction Pyramid	Bounding-Box
	PointPaiting (2019) [45]	Feature Level Fusion	2D CNN Backbone (DeepLabv3+ for Kitti and ResNet-50 for nuscenes)	Point-wise (PointNet++)	PointNet-based	Anchorless detector	Two-stage	Integrated Features	Mask-level
Projection	MV3D (2017) [51]	Feature Level Fusion	2D CNN Backbone (VGG16)	CNN (VGG16) BEV	2D CNN-based	Sliding Window-based	Two-stage	Prediction Pyramid	Bounding-box
	AVOD (2018) [56]	Feature Level Fusion	2D CNN (VGG16)	CNN (VGG16) BEV	2D CNN-based	Sliding Window-based	Two-stage	Encoding-Decoding Feature Pyramid with multiple predictions	Bounding-box
	SCANet (2019)[57]	Feature Level Fusion	2D CNN Backbone (VGG16)	CNN (VGG16) BEV	2D CNN-Based	Sliding Window-based	Two-stage	Integrated Features	Bounding-box
	ContFuse (2018) [58]	Feature Level Fusion	2D CNN Backbone (ResNet-18)	CNN (ResNet-18) BEV	2D CNN-based	Region Proposal-based	Single-stage	Encoding-Decoding Feature Pyramid with multiple predictions	Bounding-box
	MMF (2019) [59]	Feature Level Fusion	CNN (ResNet-18)	CNN (Custom ResNet) BEV	2D CNN	Region Proposal-based	Two-stage	Encoding-Decoding Feature Pyramid with multiple predictions	Bounding-box
	Complex-Retina (2019) [60]	Feature Level Fusion	CNN (VGG16)	CNN (VGG16) BEV	2D CNN-based	Region Proposal-based	Single-stage	Encoding-Decoding Feature Pyramid with multiple predictions	Bounding-box
	LaserNet++ (2019) [61]	Feature Level Fusion	CNN (ResNet)	CNN (ResNet)	2D CNN-based	Anchorless detector	Single-stage	Integrated Features	Mask-level
Frustrum	F-PointNet (2018) [46]	Decision-feature Level Fusion	2D CNN Backbone	Point-wise (PointNet++)	PointNet-based	Region Proposal-based	Two-stage	Encoding-Decoding Feature Pyramid with multiple predictions	Mask-level
	Frustrum ConvNet (2019) [47]	Decision-feature Level Fusion	2D CNN Backbone (ResNet-50)	Point-wise (PointNet)	PointNet-based	Region Proposal-based	Two-stage	Integrated Features	Mask-level
	SIFRNet (2019) [48]	Decision-feature Level Fusion	2D CNN Backbone (ResNet-50)	Point-wise (Custom PointNet++)	PointNet-based	Region Proposal-based	Two-stage	Integrated Features	Mask-level

On the other hand, Sang-Il et al. [129] proposed a decision level model that processes LiDAR point clouds and RGB images independently. It uses a segmentation process in each pipeline (image RGB and LiDAR point cloud) to produce the object-region proposals that are fused and fed to CNN models that classify the object proposals.

Most projects use RPN, either with Faster R-CNN [51], [43], [56] or with another network [44], [57]. Typically, RPN-based approaches use sliding window proposals and thus rely on the multi-scale feature learning paradigm based on the prediction

pyramid. RPNs are applied to generate 3D object proposals or 2D object crops using the feature maps as input. These regions are passed to a detection network that performs dimension refinement, orientation estimation and class prediction. AVOD uses this RPN architecture in an FPN structure. Its objective is to use a multi-scale feature learning paradigm based on Encoding-Decoding Feature Pyramid with multiple predictions. Other projects [58], [59] use a combinational approach similar to an FPN to fuse point-wise features from LiDAR and image streams and project them to their respective view stream. However, they do not include RPN networks. MFDS uses SSD, which was one of the first attempts at using a CNN in Multiple Prediction Pyramid structure.

Most projects that use PointNet or PointNet++ to do instance or object segmentation tasks follow Mask-level detection setting. Only RoarNet and PointFusion, which use the aforementioned networks, follow a detection setting based on the bounding box. It is important to refer that Mask-level settings are computationally more expensive when compared with bounding box-level.

Since fusion-based methods rely on two different types of data sources, synchronisation and calibration between them are required, which might make the solution more sensitive to sensor failures. Then, there is an increase in costs associated with the use of an additional sensor. As shown in Table VI generally, these methods achieve good performance results; however, models [45], [46], [51], [113] are computationally inefficient, presenting inference times higher than 170 ms. One limitation that makes these solutions run slow when compared, for example, with LiDAR-only methods is the significant amount of image input that they process. Moreover, Fusion-based models are generally composed of two-stage pipelines which are less time-efficient when compared with single-stage models. Another possible limitation is the usage of RPNs to select candidate bounding boxes, as used in [44], [51], [56], [57]. With a large number of unknown candidate boxes being generated and more detailed classification and regression being conducted, the model's inference time increases. Although these approaches achieve good performance results, typically they rely heavily on off-the-shelf 2D object detectors and thus in 2D detection performance. Also, they cannot take advantage of 3D information to generate more accurate bounding boxes.

Along with this, some projects that rely on combining features from multiple views, such as MV3D [51] can suffer information loss since feature sizes are inconsistent and need to be normalised, which can limit detection performance. Also, the models that resort to decision and feature level fusion have their weaknesses. The decision fusion models can be less accurate since these

Table VI. Comparison of 3D Object Detection Models results on the KITTI test 3D detection benchmark - Fusion-Based Models.

Data Representation	**Model (year)**	**Inference Time (ms)**	**Cars**			**Pedestrians**			**Cyclist**		
			E	**M**	**H**	**E**	**M**	**H**	**E**	**M**	**H**
Volumetric	MFDS (2019) [26]	108.3	-	-	-	-	-	-	-	-	-
Points	PointFusion (2017) [43]	-	77.92*	63*	53.27*	33.36*	28.04*	23.38*	49.34*	29.42*	26.98*
	RoarNet (2018) [44]	100	84.25	74.29	59.78	-	-	-	-	-	-
	PointPaiting (2019) [45]	400	82.11	71.10	67.08	50.32	40.97	37.87	77.63	63.78	55.89
Projection	MV3D (2017) [51]	360	74.97	63.63	54	-	-	-	-	-	-
	AVOD (2018) [56]	80	76.39	66.47	60.23	36.10	27.86	25.76	57.19	42.08	38.29
	SCANet (2019) [57]	90	76.09	66.30	58.68	50.66	41.44	36.60	67.97	53.07	50.81
	ContFuse (2018) [58]	60	83.68	68.78	61.67	-	-	-	-	-	-
	MMF(2019) [59]	80	88.40	77.43	70.22	-	-	-	-	-	-
	Complex-Retina (2019) [60]	90	-	-	-	-	-	-	-	-	-
	LaserNet++ (2019) [112]	38	-	-	-	-	-	-	-	-	-
Frustrum	F-PointNet (2018) [46]	170	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01
	Frustrum ConvNet (2019) [113]	470	87.36	76.39	66.69	52.16	43.38	38.80	81.98	65.07	56.54
	SIFRNet (2019) [48]	-	85.62*	72.05*	64.19*	69.35*	60.85*	52.95*	80.87*	60.34*	56.69*

Caption:

*[1] AP results on Kitti validation set.

E – easy, M – moderate, H – hard.

systems rely on LiDAR and image features separately. On the other hand, feature-level fusion presents limitations on finding equivalent representations for each type of data.

B. LiDAR-Based Solutions

Fusion-based models also rely on RGB images for detection of objects; however, a variety of modern 3D object detectors have relied only on LiDAR data, as it provides good geometric cues while offering depth information which makes 3D localisation easier. Some of the most notable works in object detection for self-driving can be seen in Table VII.

Some of these models project the point clouds to BEV, creating a very compact representation. This transformation is reasonable as objects are on the same ground. Furthermore, there is no loss of information since the height information can be handled as another 2D channel like performed in RGB channels processing models. The main advantage regards the possibility of treating the point cloud in the same manner as RGB images. Methods designed for handling RGB images have been applied to the point cloud context in a straightforward manner, e.g. Faster R-CNN, SSD, R-CNN or 2D CNN Backbones. These solutions are often described as a general framework for the detection of objects over arbitrary 2D structures, as is the case of a BEV point cloud representation. Models following this representation omit the size along the z-axis, making this method more computationally efficient since 2D instead of 3D CNN backbones can be adopted. Table VIII shows that these methods are fast (detection rate higher than 10 Hz for all projection methods). As expected, regarding 3D detection benchmarks, they present results much lower than those found in projects that do not omit one of the spatial dimensions. The model with the best performance among the projection-like models reported results much lower than Volumetric and Point data representation-based models. Although it might have been expected that projection models were able to outperform other types of models in BEV detection benchmarks, reported results have shown that projection models are still less efficient. For instance, for detection of cars on BEV using the KITTI Benchmark, Birdnet achieved AP of 60%, 84% and 57%, Pixor 87%, 81% and 77%, whereas STD ensures AP metric values near to 89%, 95% and 86% for moderate, easy and hard level, respectively. Therefore, omitting one dimension does not necessarily lead to an increase in performance in terms of inference time, nor better results in evaluations of the model in BEV.

Volumetric representations might demand more computationally expensive processes as a consequence of the structuration of the point cloud in voxels, pillars or frustums, leading to a larger number of points being processed. For instance, for 3D CNNs, several research projects have addressed this topic by introducing methodologies or techniques to reduce complexity. As a result, 3D detection models have improved, as showcased by the performance comparison between projection and volumetric models. For instance, Frustum-based solutions such as Patch-refinement [30] or F-PointNet [46], do not input the whole point cloud to a 3D CNN. Instead, they run a less resource-demanding solution for detection of objects, which can be RGB image processing or detection of objects over a BEV representation of the original point cloud. Thus, only the points contained within these regions are forwarded for the subsequent stages, where finally 3D CNNs are applied. Sometimes regions might be enlarged to encompass more contextual information.

Regarding models that structure point clouds in voxels or pillars, such as PointPillars [49], these might opt by shifting from 3D CNNs to 2D convolutions, reducing the space and time complexity of point feature extraction. To do so, these models compress these 3D features into 2D maps, i.e. projecting features into pseudo-image feature maps, to reduce the computational cost. As the height dimension is smaller and contains less information than the remaining dimensions of the tensors, some projects opt by applying larger strides along the height dimension. This results in 4D tensors, where the channel dimension is included, that are transformed into 3D tensors that do not include height dimension. They become 2D feature maps with channels that can be processed by any standard 2D convolution detection network. The inference time of SECOND [29] and VoxelNet [25] shows the impact of this strategy on its performance. Although SECOND follows the same architecture of VoxelNet, its global feature extractor reduces the dimension of the global feature maps, resulting in a 3D object detection five times faster than VoxelNet. This strategy made the SECOND model feasible for self-driving applications in terms of real-time processing requirements. VFE-based solutions opt by generating single or multi pseudo-image feature maps. Instead of balancing the voxel size (as performed in single pseudo-image feature maps, which leads to information loss due to the maximum number of points imposed per voxel), HVNet [62] and VOXEL-FPN [32] models generate multiple pseudo-image feature maps from versions of the original point cloud structured in voxels of different sizes. This allows the extraction of finer geometric features, rich in localisation information from smaller voxels, and features richer in context information, covering larger receptive fields with small feature maps. Thus, the flexibility between model speed and accuracy can be explored. As shown in Table VIII, the models performing 3D Object Detection from several pseudo-image feature maps, such as HVNet [62] and VOXEL-FPN [32] were able to improve their accuracy with regards to single pseudo-image-based solutions, such as VoxelNet [25], SECOND [29] and PointPillars [49], by slightly sacrificing their inference time. For instance, HVNet outperforms SECOND, VoxelNet and PointPillars by over 1.62%, 12.47%, and 3.27%, respectively in moderate difficulty, whereas its inference time is 11ms and 15ms higher than SECOND and PointPillars, respectively.

Outputted denser feature maps can be provided as input for region proposal networks as adopted by SECOND, VoxelNet, Fast Point R-CNN, PV-RCNN, MEGVII, HotspotNet and Fusion on Fusion Net. At the same time, these solutions might opt for a 3D convolution network specially designed to explore the sparse nature of point clouds, as analysed in subsection III.D to improve effectiveness while reducing the complexity and improving inference time.

Table VII. Description and comparison of 3D Object Detection models regarding the methods set as blocks of the generic pipeline.

Data Representation (Block 1)	Model	Architecture	Data Feature Extraction (Block 2)	Detection encoder (Block 3)			PRN Network (Block 4)
				Category	Multi-scale Feature learning	Detection settings	
Volumetric	3D FCN (2016) [28]	Single-stage	3D CNN Backbones	Anchorless	N.A	Mask-level	N.A.
	VoxelNet (2017) [25]	Single-stage	Compound solution	Region Proposal-based	Integrated Features	Bounding Box	N.A.
	Vote3Deep (2017)[24]	Single-stage	3D CNN Backbones	Anchorless	Prediction Pyramid	Bounding Box	N.A.
	SECOND (2018)[29]	Single-stage	Compound solution	Region Proposal-based	Integrated Features	Bounding Box	N.A.
	Patch Refinement (2018)[30]	Dual-stage	Compound solution	Region Proposal-based	Integrated Features	Bounding Box	Per-Region Data Fusion
	PointPillars (2018)[49]	Single-stage	Compound solution	Region Proposal-based	Multiple Prediction Pyramid	Bounding Box	N.A.
	Fast Point R-CNN (2019) [31]	Dual-stage	Compound solution	Region Proposal-based	Integrated Features	Bounding Box	Per-Region Data Fusion
	VOXEL-FPN (2019) [32]	Single-stage	Segment-wise	Region Proposal-based	Multi Input Feature Pyramid	Bounding Box	Global features-driven
	PV-RCNN (2019) [33]	Dual-stage	Compound solution	Region Proposal-based	Integrated Features	Bounding Box	Keypoints-based
	MEGVII (2019) [27]	Single-stage	3D CNN Backbones	Region proposal-based	Integrated Features	Bounding Box	N.A.
	HotSpotNet (2019) [95]	Single-stage	3D CNN Backbones	Anchorless	Prediction Pyramid	Mask-level	N.A.
	3DBN (2019) [34]	Single-stage	3D Sparse Convolution Network	Region Proposal-based	Fusion on Fusion	Bounding Box	N.A.
	Fusion of Fusion Net (2020) [35]	Single-Stage	Compound solution	Region proposal-based	Fusion on Fusion	Bounding Box	N.A.
	Point A^2-anchor (2020)[36]	Dual-stage	3D CNN Backbones	Region Proposal-based	Integrated Features	Bounding Box	Per-Region Data fusion
	Point A^2-free (2020)[36]	Dual-stage	3D CNN Backbones	anchorless	Integrated Features	Mask-level	Per-Region Data fusion
	HVNet (2020) [62]	Single-stage	Segment-wise	Region Proposal-based	Multi Input Feature Pyramid	Bounding Box	Global features-driven
Points	IPOD (2018) [39]	Single-stage	Segment-wise	Region Proposal-based	Prediction Pyramids	Bounding Box	N.A.
	STD (2019) [40]	Dual-stage	Segment-wise	Anchorless	N.A.	Mask-level	Per-Region Data Fusion
	PointRCNN (2019)[10]	Dual-stage	Segment-wise	Anchorless	Prediction Pyramid	Mask-Level	Per-Region Data Fusion
	PointRGCN (2019)[41]	Dual-stage	Segment-wise	Anchorless	N.A.	Mask-level	Per-Region Data Fusion
	LaserNet (2019) [42]	Single-stage	3D CNN Backbones	Anchorless	N.A.	Bounding Box	N.A.
Projection	Vehicle FCN detection (2016) [50]	Single-stage	2D CNN Backbone	Anchorless	Encoding-Decoding Feature Pyramid	Bounding Box	N.A.
	HDNet (2018)[53]	Single-stage	2D CNN Backbone	Anchorless	Prediction Pyramid	Bounding Box	N.A.
	BirdNet (2018)[52]	Dual-stage	2D CNN Backbone	Region Proposal-based	Integrated Features	Bounding Box	N.A.
	RT3D (2018) [54]	Dual-stage	2D CNN Backbone	Region proposal-based	Prediction Pyramid	Bounding box	N.A.
	Pixor (2019)[55]	Single-stage	2D CNN backbone	Anchorless	Prediction Pyramid	Bounding Box	N.A.

As observed in Table VII, some projects such as SECOND[29], VoxelNet [25], PointPillars [49], Fast Point R-CNN [31], etc. rely on compound feature extraction followed by an RPN network instead of a single feature extractor backbone. This pipeline takes advantage of both point-wise/segment-wise and CNN-based feature extractors. As mentioned before, solutions that use point-based feature extractors to generate point-wise/voxel-wise features, such as PointNet, PointNet++ or Voxel Feature Extractor can produce pointwise or global features without requiring any prior knowledge of topology. Along with learned features, which preserve accurate location information, points could be further voxelized or not structured [40]. For instance, Voxel Feature Extractor groups points into voxels based on their local coordinates, which is a process similar to those performed by PointNet-based architectures, as performed in PointNet++. In this method, linear transformations and max-pooling are conducted in each voxel cell to aggregate features. Sparse convolution networks collect voxel-wise features to generate descriptive information that might be forwarded for an RPN or directly for a classification and regression head network. Some projects adopt a single feature extractor as a backbone, instead of a compound solution, as is the case of models that directly process the raw point cloud, called point-based solutions, as well as solutions that structure point clouds in a volumetric representation. For instance, Part A^2 Net [36] does not rely on PointNet++ or VFE to extract point-wise or local features, instead, the point-wise information is directly subjected to a sparse convolutional network to extract global features. Point A^2 free authors state that sparse convolutional networks learn more discriminative point-wise features from the point cloud.

Solutions that follow a Point-based data representation such as IPOD [39], STD[40], PointRCNN [10], PointRGCN [41] follow a detection setting based on Mask-level, which means that bounding box proposals are generated directly from the segmented foreground points. By learning to segment the foreground points, the point-cloud network is forced to capture contextual information for making accurate point-wise predictions, which is also beneficial for 3D box generation. Therefore, the first stage of such a model learns to segment foreground and background points that further extrapolate a proposal bounding box for each foreground point. In dual-stage models, both proposals and segmentation features might be used by the prediction refinement block.

Although PointRGCN implements a GCN-based network, it aims at improving the prediction refinement process. Since current GCN-based feature extraction methods are memory intensive, they are not suitable for application at early stages of the pipeline as feature extractors [41]. However, results observed in Table VIII suggest that improvements are required for feature extractors that resort on GCNs to meet the real-time processing requirements imposed by self-driving applications.

Region Proposal-based strategies are not memory efficient since they aim at generating 3D proposals from global features by resorting on pre-defined anchors with different orientations and sizes per class, at each spatial location, increasing the number of saved parameters significantly. In contrast, anchorless strategies typically follow a bottom-up scheme by segmenting foreground points to simultaneously generate a single bounding box and 3D proposals from each predicted foreground point. Therefore, this strategy avoids the need for a large number of 3D anchor boxes in the whole scene, which allows for the implementation of lightweight models that are more memory efficient and have lower calculation costs. The difference in memory and run-time efficiency between region proposal-based and anchorless methods is more clear in multiclass object detection models application, as different classes in 3D object detection generally require more anchors in the scenes, for instance, anchor size of pedestrians might be l = 0.8m, w = 0.6m, h = 1.73m, cyclist anchor size might be l = 1.76m, w = 0.6m, h = 1.73m. In contrast, cars might be l = 3.9m, w = 1.6m, h = 1.5m [49] and consequently, cannot share the same anchor. In anchorless solutions, the point-wise features previously extracted are shared for generating proposals for several classes, as a solution only needs to calculate the 3D size residual relative to the corresponding average object size. The performance of anchorless methods seems to be lower for large-sized objects such as cars, as residuals from the object surface points to the object centre are larger, turning the task of regressing more challenging. However, according to the experiments conducted in Part A2 Net [36], anchorless solutions achieve comparable performance to region proposal-based solutions when detecting small-sized objects, such as pedestrians or cyclists. Region proposal-based models achieve higher recall improvements thanks to the application of its pre-defined anchors for each class to cover the whole 3D scene. In contrast, in large-size objects, better performance is achieved as anchors are closer to the centre locations of objects.

As featured in Table I, most of the projects have followed a Region proposal-based scheme, especially those that opted for volumetric data representation. In contrast, point and projection data representations generally use an anchorless solution. Region proposal-based methods tend to achieve better results, as revealed by Table VIII. Still, anchorless solutions might be strongly explored in the near future, as they seem to have a lot of potential for application in real scenarios. These types of scenarios include objects with different dimensions, where fast detections are mandatory and embedded hardware is usually resource constrained.

As shown in Table VII, most of the projects utilise a discretisation process, which inevitably introduces quantisation artefacts with resolution decreasing to the number of bins in the voxel map. When associated with successive convolution and downsampling operations, this will weaken the precise localisation signal that originally exists in point clouds and are provided by point-wise based feature extractors. Region-proposal based solutions are widely constructed based on traditional RPNs, R-CNN [91], Faster R-CNN [12] or SSD [89]. These solutions ignore low-level features as shallow feature maps are not considered; however, these are computationally less expensive and minimize latency. For instance, research projects VoxelNet [25], SECOND [29], Patch Refinement [30], Fast Point R-CNN [31], PV-RCN [33], MEGVII [27], 3DBN [34], Part A2 Net [36], RT3D [54] and BirdNet [52] concatenate multi-scale feature maps through the connection of all or parts of the convolutional layers output. More recently, developments in RPNs led to concatenating these output feature map pairs before passing them to the subsequent layers, as low-resolution. Still, semantically strong feature maps are upsampled to the size of the upper pyramidal level by using the deconvolutional layers that are then concatenated to the corresponding feature maps of the same spatial size. These approaches follow the idea proposed by Long et al. [18]. The performance table shows us that combining features from lower and higher layers improves the prediction task, in particular for small objects.

Most of the projects in Table VII have opted for a single-stage architecture without a second stage fully dedicated to re-score the box proposals outputted by the first stage and refining their locations. As expected, single-stage models are faster but achieve lower 3D object detection performance if compared to dual-stage (this is featured in Table VIII). However, recent projects such as Point-RCNN[10], Fast Point R-CNN[31], Patch Refinement[30], STD [40], Point A^2 Net [36], PV-RCNN [33], PointRGCN [41] and RT3D [54] implement a second-stage that significantly improved the 3D detection performance. For instance, Patch Refinement compared the performance of a model following the structure of the VoxelNet/SECOND but tested two configurations: (1) model with an additional stage to perform proposal refinement; (2) model without performing proposal refinement. Results show very significant improvements in moderate and hard difficulty (4% and 9% of AP, respectively). However, this last stage can also be described according to the way received data is represented. Most of the models directly consume the region proposal

Table VIII. Comparison of 3D Object Detection Models results on the KITTI test set 3D detection benchmark - LiDAR-Only.

Data Representation	Model (Year)	Inference Time (ms)	Cars			Pedestrians			Cyclist		
			E	M	H	E	M	H	E	M	H
Volumetric	3D FCN (2016) [28]	1000	-	-	-	-	-	-	-	-	-
	VoxelNet (2017) [25]	225	77.47*[1]	65.11*[1]	57.73*[1]	39.48*[1]	33.69*[1]	31.51*[1]	61.22*[1]	48.36*[1]	44.37*[1]
	Vote3Deep (2017)[24]	1100	-	-	-	-	-	-	-	-	-
	SECOND-V1.5 (2018) [29]	20	84.65	75.96	68.71	-	-	-	-	-	-
	HR-SECOND (2018) [29]	110	84.78	75.32	68.70	45.31	35.52	33.14	75.83	60.82	53.67
	Patch Refinement – Patches - EMP (2018) [30]	50	89.84	78.41	73.15	-	-	-	-	-	-
	Patch Refinement – Patches (2018) [30]	150	88.67	77.20	71.82	-	-	-	-	-	-
	PointPillars (2018)[49]	16	82.58	74.31	68.99	51.45	41.92	38.89	77.10	58.65	51.92
	Fast Point R-CNN (2019) [31]	60	85.29	77.40	70.24	-	-	-	-	-	-
	VOXEL-FPN (2019) [32]	50	85.64	76.70	69.44	-	-	-	-	-	-
	PV-RCNN (2019) [33]	80	90.25	81.43	76.82	52.17	43.29	40.29	78.60	63.71	57.65
	MEGVII (2019)[27]	-	-	-	-	-	-	-	-	-	-
	HotSpotNet-Dense (2019)[95]	-	88.12*[1]	78.34*[1]	73.49*[1]	47.14*[1]	39.72*[1]	37.25*[1]	79.09*[1]	62.72*[1]	56.76*[1]
	HotSpotNet-Direct (2019)[95]	-	86.49*[1]	77.74*[1]	72.97*[1]	51.29*[1]	44.81*[1]	41.13*[1]	77.70*[1]	63.16*[1]	57.16*[1]
	3DBN (2019) [34]	130	83.77	73.53	66.23	-	-	-	-	-	-
	Fusion of Fusion Net (2020) [35]	50	84.15	74.45	66.97	49.44	41.21	36.42	75.36	59.65	53.03
	Point A^2-anchor (2020)[36]	80	87.81	78.49	73.51	53.10	43.35	40.06	79.17	63.52	56.93
	Point A^2-free (2020)[36]	80	88.48*[3]	78.96*[3]	78.36*[3]	70.73*[3]	64.13*[3]	57.45*[3]	88.18*[3]	73.35*[3]	70.75*[3]
	HVNet (2020) [62]	31	87.21*[3]	77.58*[3]	71.79*[3]	69.13*[3]	64.81*[3]	59.42*[3]	87.21*[3]	73.75*[3]	68.98*[3]
Points	IPOD (2018) [39]	20	79.75*[2]	72.57*[2]	66.33*[2]	56.92*[2]	44.68*[2]	42.39*[2]	71.40*[2]	53.46*[2]	48.34*[2]
	STD (2019) [40]	80	87.95	79.71	74.16	53.29	42.47	38.35	78.69	61.59	55.30
	PointRCNN (2019)[10]	100	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
	R-GCN only (2019) [41]	160	83.42	75.26	68.73	-	-	-	-	-	-
	PointRGCN (2019) [41]	260	85.97	75.73	70.60	-	-	-	-	-	-
	R-GCN (2019) [41]	160	83.42	75.26	68.73	-	-	-	-	-	-
	C-GCN (2019) [41]	147	83.49	73.62	67.01	-	-	-	-	-	-
	LaserNet (2019) [42]	30	-	-	-	-	-	-	-	-	-
Projection	Vehicle FCN detection (2016) [50]	-	-	-	-	-	-	-	-	-	-
	HDNet (2018) [53]	50	-	-	-	-	-	-	-	-	-
	BirdNet (2018) [52]	99	40.99	27.26	25.32	22.04	17.08	15.82	43.98	30.25	27.21
	RT3D (2018) [54]	90	23.74	19.14	18.86	-	-	-	-	-	-
	Pixor (2019) [55]	35	-	-	-	-	-	-	-	-	-

Caption:

*[1] AP results reported only in the authors article on the Kitti test set.

*[2] AP results reported on Kitti online test set using 11 recall positions.

*[3] AP results on Kitti validation set.

E – easy, M – moderate, H – hard.

features received, whereas some projects, such as STD or Patch Refinement, subdivide each proposal into equally spaced voxels. This has the advantage of making models more flexible since the performed RoI pooling can also be applied to models following a Point-based data representation. Extracted point-wise features can be converted into regular or sparse voxel features, allowing the encoding of the position-specific geometry and semantic features of 3D proposals, which are further processed by efficient sparse convolution networks. Furthermore, models might share features between stages, e.g. models Point-RCNN[10], Fast Point R-CNN [31], Patch Refinement [30], Point A2 Net [36], PV-RCNN [33], PointRGCN [41] and RT3D [54], or their stages can be independent of each other without sharing of information, as implemented in Patch Refinement [30] and STD [40]. According to the performance results featured in Table VIII, the later scheme might deteriorate the inference time but improve the 3D object detection efficiency of models. It occurs due to the fact that the model's stages can be individually trained and assessed, and additional augmentation techniques can be carried out.

Moreover, multi-scale and aggregation of intra-proposal features and aggregation of different features are beneficial to improve the quality of the predicted 3D boxes. For instance, research project Point RCNN demonstrated that the generation of context features through the concatenation of local and global features slightly improves the 3D object detection performance (metric AP) up to 2% [10]. Not aggregating contextual information significantly affects model performance, especially on scenarios where the number of points available is too low, for instance, when objects are occluded or far away from the sensor. In this scenario, the

multi-scale feature learning scheme takes an important role. Their lateral connections and paths offer richer semantic information, even for small-sized objects. Moreover, the concatenation of semantic information with local features (which carry rich localisation information) enables the extraction of context information, which can be processed by classifiers and proposal refinement modules. Recent projects have adopted multi-scale feature learning schemes based on a dual path at block (4) of the model pipeline (c.f. Figure 1) as an essential strategy to improve their model performance as shown in Table III.

VII. Research Challenges and Opportunities

The last two decades have witnessed tremendous advances in self-driving, with significant advances due to increases in available computing power and cost reduction in sensing and computing technologies, resulting in the maturing technological readiness level of fully autonomous vehicles. Autonomous vehicles are expected to play a vital role in the future of mobility, as they offer the potential for additional safety, increased productivity, greater accessibility, better road efficiency, and a positive impact on the environment [9].

Self-driving has opened many new challenges for the automotive industry, and the development of sensors capable of capturing real-time 3D point clouds that represent the environment has opened a plethora of possibilities in the field of the automotive industry. A critical issue involved is 3D object detection and, specifically, 3D object detection using non-structured point clouds such as those generated by LiDAR sensors. Autonomous perception has traditionally focused on engineered features. Still, the use of learned features has gained increased importance and can be considered today's most competitive solution for object detection, as this literature review evidenced. But deep learning-based approaches present several challenges, very different from earlier times. In this section, we will put forward a set of (major) challenges yet to be addressed, or solved, and identify some possible opportunities for research paths in this context.

A. On the sparsity of the data and on the extraction of features

LiDAR point clouds are inherently sparse. While self-driving strives for fast methods, dealing with this sparsity using standard convolutional methods has proven to be computationally inefficient. For this purpose, recent backbones leverage the advantages of submanifold and spatially sparse convolutions. However, the main focus has been on the pipeline as a whole, and no new deep learning backbones have been proposed in recent years. Recent developments on graph CNNs [41], [130] represent a new path to explore, whereas the voting CNN-based solutions did not receive too much attention from the research community even though it is computationally more efficient than traditional convolutional solutions.

Novel convolutional methods have recently emerged; Depthwise Separation Convolutional Kernels [131], which was developed to accelerate the speed of a convolutional operation and help meet real-time purposes, while also facilitating model integration in low-compute edge devices. Based on this, a new paradigm was released called MixConvs [132], where they explore multiple groups of different large-sized kernels to improve the model's accuracy and efficiency. In [131], the authors propose another paradigm called Switchable Atrous Convolutions, which improves detection accuracy by changing the size of the receptive field when needed (e.g. for smaller objects), without increasing computation, by using different atrous rates and automatic switches. They also introduce the use of Recurrent Feature Pyramid Networks, in which one FPN's top-down predictions feed the bottom-up input of another FPN, in order to extract richer features at various feature map sizes. Although these solutions were proposed for RGB data, they can be explored for improving the efficiency of ConvNet-based solutions for 3D Object Detection.

Another growing trend that stems from the Natural Language Processing side of deep learning are attention-based models. Attention mechanisms have been making their way into computer vision, with new papers combining mature ConvNet-based backbones with attention. This trend has been mostly adopted in 2D based object detection models, such as [132], [133], but is already part of some novel 3D object detection pipelines, as is the case of [62], [130], [134]. While there have been efforts to replace ConvNets with attention models, towards more general and less biased models, there are still very few papers on this topic for RGB, such as [135], [136] and none regarding point cloud-based object detection.

B. On the data representation

Data representation is up to now the way data has been (pre) processed to speed up the whole procedure, either "compressing" it to 2D or by considering a coarser representation (such as voxels, pillars). Together with the sparse nature of data, data representation is yet not fully explored in order to open the way for time-efficient methods that are needed for real-time applications.

C. On the Occlusion and Truncation issues

Occlusion is one of the factors that can increase failure rates and is characterised by a situation when one object blocks the view of another, resulting in partial or complete invisibility of the object. Fusion-based solutions have the advantage of leveraging the RGB data to detect objects under such conditions; however, in LiDAR-based solutions, the number of research works addressing this issue is very low. Moreover, practically only Mask-level detection setting-based models introduced solutions to this problem since typically, anchor regression is not directly applied; instead, each point contributes to the 3D geometry reconstruction. Some works, for instance, HotSpotNet [95], implemented a network also called part-based models, that aims at learning to determine if a region (called hotspot) is part of an object. Therefore, each hotspot individually predicts its likelihood of the object's presence. Thus, even when an object is mostly occluded or partially truncated, some hotspots are still able to indicate the presence of the

object and contribute to the 3D geometry information.

However, most of the projects follow a bounding box level detection setting. Thus, research on this issue is required, as in real scenarios, it is very likely that objects with high levels of occlusion or truncation exist.

D. On model training

One characteristic of supervised learning is the need for large amounts of annotated data which, for the case of computer vision and especially for LiDAR point clouds, is not easy to obtain. Moreover, having a large and diverse training dataset covering all relevant scenarios is key to achieve the accuracy implied by safety-critical operations, such as self-driving. The existing public datasets [119]–[123] do not provide enough examples of real-world edge-case scenarios, and as a consequence, the current models lack testing under more challenging conditions. These datasets are also specific to the LiDAR sensor used to capture the data, which limits the available data to train the object detector models. Data augmentation might help to overcome this problem, as well as the use of transfer learning [137], [138]. There have been several recent works, such as [139]–[141], proposing new paths for data augmentation in computer vision that, so far, was not adapted and applied to datasets made of point clouds.

High-quality annotation of data requires huge human effort, thus becoming a very costly and time-consuming task. By finding the right data to label, costs can be reduced, while also aiming for a performance increase. Active learning [142] is a powerful technique used to improve data efficiency by allowing the machine learning algorithm, in this case, the object detector, to choose the data from which it learns. There are several works published on this technique, such as [143]–[149]. In [149], the main component of the system is an image-level scoring function that evaluates the informativeness of each new unlabelled image. The experimental results showed solid performance improvements for automatic data selection, with performance improvements when compared to a manual selection process done by experts.

Even though recent results are promising, more research is necessary to validate both of these techniques with LiDAR point clouds and the self-driving scenario.

E. On the evolution of multimodal perception

LiDAR-based solutions have been achieving increasingly better performance results; however, LiDAR measurements have less semantic information than RGB cameras, data is sparse by nature, and sensor range is limited between 50 and 100 meters. In contrast, RGB sensors offer an increase in range, with radar even doubling it. RGB cameras also offer semantically richer information with better localisation on the image plane. Therefore, not only multimodal setups are complementary but also offer redundant solutions to overcome failures and to handle the limitations imposed by adverse conditions and blind spots [121]. Regardless, multimodal solutions bring challenges related to the calibration and synchronization of the several sensors used. More research is required to implement fusion-based solutions, able to surpass single sensor models performance.

F. On the possibilities for motion information integration

Integration of motion (and tracking) information such as in the exploratory work in [150], may further speed up the models and enhance their robustness and accuracy, especially against problems due to truncation and object occlusion.

On this context, the Simultaneous Localisation and Mapping (SLAM) models have the ability to predict acquisition system and dynamic objects' motion and respective position estimation [151], [152]. These models might be worth exploring as a way to improve the effectiveness of LiDAR-based object detection models.

G. On the transparency and explainability of the models

Besides the technological challenges, self-driving presents legal and ethical problems that need special attention, with a great emphasis on safety. The use of deep learning approaches for object detection hinders a full understanding of the underlying models and the correspondent internal decision sequences. Deep learning models are viewed as black-box methods, where the causal relationship between input and output is not (entirely) understood. Therefore, more research and new models for increased transparency and explainability are needed [153].

VIII. CONCLUSION

This survey addresses the problem of Point-Cloud based 3D object detection with autonomous driving applications in mind. In recent years, the outperformance of deep learning models for object detection together the development of 3D sensing and computing technology resulted in steep developments of the perception stage for autonomous driving. On the other hand, the increasing availability of LiDAR sensors and its superior accuracy and field of view angle when compared to radar-based solutions has paved the way to their use for 3D object detection, with or without sensor-fusion.

In this work, we have introduced a generic 3D object detection pipeline, offering a common ground for the analysis of the main components of the different models, and for the thorough study and comparative analysis of state-of-the-art methods in each stage of the pipeline. For the better categorisation of the existing works considering their design choices according to the introduced pipeline, we also have proposed a comprehensive taxonomy. A central aspect of all models – the feature extraction block -, was analysed in more detail taking into consideration the classification/localisation quality vs. inference time trade-off. The detection

module due to its coverage was further refined to include specific aspects such as the network architecture or the detection techniques used. The comparative analysis of the state-of-the-art object detection methods was performed for LiDAR or fusion-based (including LiDAR) solutions to meet today´s autonomous driving research needs. Besides the block-based systematic study of the different existing methods, we were able to identify some existing shortcomings and relate them with the actual stage of the object detection pipeline, summarising some challenges of deep learning-based approaches for LiDAR point clouds and possible directions for future work.

This work allows us to realize that the potential of the methodologies is not yet fully explored and that a complete understanding of the model in its different stages is fundamental to address it. Novel methods such as anchorless detectors, combinations of one- and two-stage detectors for increased detection accuracy and improved post-processing NMS, represent some of the improvements on the existing models. Nevertheless, there is a growing need for more accurate real-time methods which enable these methods applications in edge devices. Recent studies have been mainly focused on the implementation of novel solutions for blocks 3 and 4 aiming at achieving highly accurate and efficient detectors. New deep learning architectures, new methods for extraction and combination of rich features, exploration of new data representations were some of the challenges that recent solutions have been facing. Other issues, such as the interpretability of the models, complex perception scenarios, small or occluded objects, and negative-positive imbalance sampling are still major challenges in 3D object detection for self-driving showing that, despite all the recent advances in object detection for autonomous driving, there is still much development to be done.

ACKNOWLEDGEMENT

This work is supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 037902; Funding Reference: POCI-01-0247-FEDER-037902].

REFERENCES AND FOOTNOTES

A. References

[1] W. H. Organization, “Global status report on road safety 2018,” 2018.
[2] M. M. Peden and P. Puvanachandra, “Looking back on 10 years of global road safety,” *Int. Health*, vol. 11, no. 5, pp. 327–330, Sep. 2019, doi: 10.1093/inthealth/ihz042.
[3] Society for Risk Analysis., “Self-driving cars must reduce traffic fatalities by at least 75 percent to stay on the roads.,” 2018. [Online]. Available: https://www.sciencedaily.com/releases/2018/05/180530132959.htm.
[4] S. Chen, M. Kuhn, K. Prettner, and D. E. Bloom, “The global macroeconomic burden of road injuries: estimates and projections for 166 countries,” *Lancet Planet. Heal.*, vol. 3, no. 9, pp. e390–e398, Sep. 2019, doi: 10.1016/S2542-5196(19)30170-6.
[5] T. Litman, “Autonomous Vehicle Implementation Predictions Implications for Transport Planning,” 2020.
[6] Society of Automotive Engineers, “SAE Standards News: J3016 automated-driving graphic update,” 2019.
[7] D. Gruyer, V. Magnier, K. Hamdi, L. Claussmann, O. Orfila, and A. Rakotonirainy, “Perception, information processing and modeling: Critical stages for autonomous driving applications,” *Annu. Rev. Control*, vol. 44, pp. 323–341, 2017, doi: 10.1016/j.arcontrol.2017.09.012.
[8] J. Van Brummelen, M. O’Brien, D. Gruyer, and H. Najjaran, “Autonomous vehicle perception: The technology of today and tomorrow,” *Transp. Res. Part C Emerg. Technol.*, vol. 89, Mar. 2018, doi: 10.1016/j.trc.2018.02.012.
[9] S. Pendleton *et al.*, “Perception, Planning, Control, and Coordination for Autonomous Vehicles,” *Machines*, vol. 5, no. 1, p. 6, Feb. 2017, doi: 10.3390/machines5010006.
[10] S. Shi, X. Wang, and H. Li, “PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud,” *Comput. Vis. Pattern Recognit.*, Dec. 2018, [Online]. Available: http://arxiv.org/abs/1812.04244.
[11] R. Girshick, “Fast R-CNN,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2015 Inter, pp. 1440–1448, 2015, doi: 10.1109/ICCV.2015.169.
[12] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
[13] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, “A Survey on 3D Object Detection Methods for Autonomous Driving Applications,” *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, 2019, doi: 10.1109/tits.2019.2892405.
[14] O. Elkhalili *et al.*, “A 64×8 Pixel 3-D CMOS time of flight image sensor for car safety applications,” *ESSCIRC 2006 - Proc. 32nd Eur. Solid-State Circuits Conf.*, pp. 568–571, 2006, doi: 10.1109/ESSCIR.2006.307488.
[15] L. Jiao *et al.*, “A Survey of Deep Learning-Based Object Detection,” *IEEE Access*, vol. 7, no. 3, pp. 128837–128868, 2019, doi: 10.1109/access.2019.2939201.
[16] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, “Deep Learning for 3D Point Clouds: A Survey,” *Comput. Vis. Pattern Recognit.*, Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.12033.
[17] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, “Deep Learning on Point Clouds and Its Application: A Survey,” *Sensors*, vol. 19, no. 19, p. 4188, Sep. 2019, doi: 10.3390/s19194188.
[18] D. Griffiths and J. Boehm, “A review on deep learning techniques for 3D sensed data classification,” *Comput. Vis. Pattern Recognit.*, Jul. 2019, doi: 10.3390/rs11121499.
[19] A. Ioannidou, E. Chatzilari, S. Nikolopoulos, and I. Kompatsiaris, “Deep Learning Advances in Computer Vision with 3D Data,” *ACM Comput. Surv.*, vol. 50, no. 2, pp. 1–38, Jun. 2017, doi: 10.1145/3042064.
[20] M. M. Rahman, Y. Tan, J. Xue, and K. Lu, “Recent Advances in 3D Object Detection in the Era of Deep Neural Networks: A Survey,” *IEEE Trans. Image Process.*, vol. 29, pp. 2947–2962, 2020, doi: 10.1109/TIP.2019.2955239.
[21] X. Wu, D. Sahoo, and S. C. H. Hoi, “Recent Advances in Deep Learning for Object Detection,” *Neurocomputing*, 2019, doi: 10.1016/j.neucom.2020.01.085.
[22] J. Gu *et al.*, “Recent Advances in Convolutional Neural Networks,” *Comput. Vis. Pattern*, Dec. 2015, [Online]. Available: http://arxiv.org/abs/1512.07108.
[23] S. Song and J. Xiao, “Sliding Shapes for 3D Object Detection in Depth Images,” in *Computer Vision -- ECCV 2014*, 2014, pp. 634–651.
[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,” 2017.

[25] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 4490–4499, 2018, doi: 10.1109/CVPR.2018.00472.

[26] M. Person, M. Jensen, A. O. Smith, and H. Gutierrez, "Multimodal Fusion Object Detection System for Autonomous Vehicles," *J. Dyn. Syst. Meas. Control. Trans. ASME*, vol. 141, no. 7, pp. 1–9, 2019, doi: 10.1115/1.4043222.

[27] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu, "Class-balanced Grouping and Sampling for Point Cloud 3D Object Detection," *Comput. Vis. Pattern Recognit.*, pp. 1–8, 2019, [Online]. Available: http://arxiv.org/abs/1908.09492.

[28] B. Li, "3D Fully Convolutional Network for Vehicle Detection in Point Cloud," *Comput. Vis. Pattern Recognit.*, Nov. 2016.

[29] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors (Switzerland)*, vol. 18, no. 10, pp. 1–17, 2018, doi: 10.3390/s18103337.

[30] J. Lehner, A. Mitterecker, T. Adler, M. Hofmarcher, B. Nessler, and S. Hochreiter, "Patch Refinement -- Localized 3D Object Detection," *Comput. Vis. Pattern Recognit.*, Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.04093.

[31] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast Point R-CNN," *Comput. Vis. Pattern Recognit.*, Aug. 2019, [Online]. Available: http://arxiv.org/abs/1908.02990.

[32] B. Wang, J. An, and J. Cao, "Voxel-FPN: multi-scale voxel feature aggregation in 3D object detection from point clouds," *Comput. Vis. Pattern Recognit.*, Jun. 2019.

[33] S. Shi *et al.*, "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection," *Comput. Vis. Pattern Recognit.*, Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.13192.

[34] X. Li, J. Guivant, N. Kwok, Y. Xu, R. Li, and H. Wu, "Three-dimensional Backbone Network for 3D Object Detection in Traffic Scenes," *Comput. Vis. Pattern Recognit.*, 2019.

[35] L. Wang, X. Fan, J. Chen, J. Cheng, J. Tan, and X. Ma, "3D object detection based on sparse convolution neural network and feature fusion for autonomous driving in smart cities," *Sustain. Cities Soc.*, vol. 54, p. 102002, Mar. 2020, doi: 10.1016/j.scs.2019.102002.

[36] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network," *Comput. Vis. Pattern Recognit.*, Jul. 2019, [Online]. Available: http://arxiv.org/abs/1907.03670.

[37] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," 2017.

[38] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[39] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive Point-based Object Detector for Point Cloud," *Comput. Vis. Pattern Recognit.*, Dec. 2018, [Online]. Available: http://arxiv.org/abs/1812.05276.

[40] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "STD: Sparse-to-Dense 3D Object Detector for Point Cloud," *Comput. Vis. Pattern Recognit.*, Jul. 2019, doi: 10.1109/ACCESS.2019.2939201.

[41] J. Zarzar, S. Giancola, and B. Ghanem, "PointRGCN: Graph Convolution Networks for 3D Vehicles Detection Refinement," *Comput. Vis. Pattern Recognit.*, Nov. 2019.

[42] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An Efficient Probabilistic 3D Object Detector for Autonomous Driving," *Comput. Vis. Pattern Recognit.*, Mar. 2019.

[43] A. Xu, Danfei ; Anguelov, Dragomir;Jain, "PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation.," *IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 244–253, 2018, [Online]. Available: http://arxiv.org/abs/1711.10871.

[44] K. Shin, Y. P. Kwon, and M. Tomizuka, "RoarNet: A Robust 3D object detection based on region approximation refinement," *IEEE Intell. Veh. Symp. Proc.*, vol. 2019-June, no. Iv, pp. 2510–2515, 2019, doi: 10.1109/IVS.2019.8813895.

[45] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential Fusion for 3D Object Detection," 2019.

[46] C. R. Qi and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data Charles," *Comput. Vis. Pattern Recognit.*, 2018.

[47] Z. Wang and K. Jia, "Frustum ConvNet : Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection," pp. 1–8.

[48] X. Zhao, Z. Liu, R. Hu, and K. Huang, "3D Object Detection Using Scale Invariant and Feature Reweighting Networks," *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 9267–9274, 2019, doi: 10.1609/aaai.v33i01.33019267.

[49] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," *Mach. Learn.*, 2018, [Online]. Available: http://arxiv.org/abs/1812.05784.

[50] B. Li, T. Zhang, and T. Xia, "Vehicle Detection from 3D Lidar Using Fully Convolutional Network," *Comput. Vis. Pattern Recognit.*, 2016.

[51] C. Xiaozhi, M. Huimin, W. Ji, L. Bo, and X. Tian, "Multi-View 3D Object Detection Network for Autonomous Driving," *Comput. Vis. Found. Videos*, pp. 1907–1915, 2017, [Online]. Available: https://www.youtube.com/watch?v=ChkgSvxAvMg.

[52] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, "BirdNet: A 3D Object Detection Framework from LiDAR Information," *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2018-Novem, pp. 3517–3523, 2018, doi: 10.1109/ITSC.2018.8569311.

[53] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD Maps for 3D Object Detection," in *Proceedings of The 2nd Conference on Robot Learning*, 2018, vol. 87, pp. 146–155, [Online]. Available: http://proceedings.mlr.press/v87/yang18b.html.

[54] Y. Zeng *et al.*, "RT3D: Real-Time 3-D Vehicle Detection in LiDAR Point Cloud for Autonomous Driving," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3434–3440, 2018, doi: 10.1109/LRA.2018.2852843.

[55] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D Object Detection from Point Clouds," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 7652–7660, 2018, doi: 10.1109/CVPR.2018.00798.

[56] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," in *IEEE International Conference on Intelligent Robots and Systems*, Dec. 2018, pp. 5750–5757, doi: 10.1109/IROS.2018.8594049.

[57] H. Lu, X. Chen, G. Zhang, Q. Zhou, Y. Ma, and Y. Zhao, "Scanet: Spatial-channel Attention Network for 3D Object Detection," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 2019-May, pp. 1992–1996, 2019, doi: 10.1109/ICASSP.2019.8682746.

[58] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep Continuous Fusion for Multi-sensor 3D Object Detection," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11220 LNCS, pp. 663–678, 2018, doi: 10.1007/978-3-030-01270-0_39.

[59] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3D object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, pp. 7337–7345, 2019, doi: 10.1109/CVPR.2019.00752.

[60] M. Li, Y. Hu, N. Zhao, and Q. Qian, "One-Stage Multi-Sensor Data Fusion Convolutional Neural Network for 3D Object Detection," *Sensors*, vol. 19, no. 6, p. 1434, Mar. 2019, doi: 10.3390/s19061434.

[61] G. P. Meyer, J. Charland, D. Hegde, A. Laddha, and C. Vallespi-Gonzalez, "Sensor Fusion for Joint 3D Object Detection and Semantic Segmentation," *Comput. Vis. Pattern*, Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.11466.

[62] M. Ye, S. Xu, and T. Cao, "HVNet: Hybrid Voxel Network for LiDAR Based 3D Object Detection," *Comput. Vis. Pattern Recognit.*, Feb. 2020.

[63] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Comput. Vis. Pattern*, Feb. 2015, [Online]. Available: http://arxiv.org/abs/1502.03167.

[64] D. Zeng Wang and I. Posner, "Voting for Voting in Online Point Cloud Object Detection," Jul. 2015, doi: 10.15607/RSS.2015.XI.035.

[65] E. Che, J. Jung, and M. J. Olsen, "Object Recognition, Segmentation, and classification of Mobile Laser Scanning Point Clouds: A State of the Art Review," *Sensors*, vol. 19, no. 4, p. 810, 2019.

[66] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Neural Inf. Process. Syst.*, vol. 60,

no. 6, pp. 84–90, May 2012, doi: 10.1145/3065386.
[67] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *Comput. Vis. Pattern Recognit.*, Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.1556.
[68] C. Szegedy *et al.*, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
[69] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 2015.
[70] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks,” *Comput. Vis. Pattern Recognit.*, Aug. 2016, [Online]. Available: http://arxiv.org/abs/1608.06993.
[71] S. Xie, R. Girshick, and P. Doll, “Aggregated Residual Transformations for Deep Neural Networks http://arxiv.org/abs/1611.05431v2,” *Cvpr*, 2017, doi: 10.1109/CVPR.2017.634.
[72] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” 2017.
[73] Z. Li, C. Peng, G. Yu, X. Zhang, Y. Deng, and J. Sun, “DetNet: A Backbone network for Object Detection,” pp. 1–17, 2018.
[74] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation,” in *European conference on computer vision*, 2016, pp. 483–499.
[75] H. Law and J. Deng, “CornerNet: Detecting Objects as Paired Keypoints,” *Eccv2018*, vol. 11218 LNCS, pp. 765–781, 2018, doi: 10.1007/978-3-030-01264-9_45.
[76] X. Zhou, D. Wang, and P. Krähenbühl, “Objects as Points,” *Comput. Vis. Pattern*, 2019, [Online]. Available: http://arxiv.org/abs/1904.07850.
[77] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, “CenterNet: Keypoint triplets for object detection,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2019-Octob, pp. 6568–6577, 2019, doi: 10.1109/ICCV.2019.00667.
[78] B. Graham, “Spatially-sparse convolutional neural networks,” *Comput. Vis. Pattern Recognit.*, Sep. 2014, [Online]. Available: http://arxiv.org/abs/1409.6070.
[79] B. Graham, “Sparse 3D convolutional neural networks,” *Comput. Vis. Pattern Recognit.*, May 2015, [Online]. Available: http://arxiv.org/abs/1505.02890.
[80] B. Graham, M. Engelcke, and L. van der Maaten, “3D Semantic Segmentation with Submanifold Sparse Convolutional Networks,” *Comput. Vis. Pattern Recognit.*, Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.10275.
[81] G. Li *et al.*, “DeepGCNs: Making GCNs Go as Deep as CNNs,” *Comput. Vis. Pattern Recognit.*, Oct. 2019, [Online]. Available: http://arxiv.org/abs/1910.06849.
[82] G. Li, M. Muller, A. Thabet, and B. Ghanem, “DeepGCNs: Can GCNs Go As Deep As CNNs?,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 9266–9275, doi: 10.1109/ICCV.2019.00936.
[83] Y. Wang and J. M. Solomon, “Deep Closest Point: Learning Representations for Point Cloud Registration,” *Comput. Vis. Pattern Recognit.*, May 2019, [Online]. Available: http://arxiv.org/abs/1905.03304.
[84] L. Wang, Y. Huang, Y. Hou, S. Zhang, and J. Shan, “Graph Attention Convolution for Point Cloud Semantic Segmentation,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019, pp. 10288–10297, doi: 10.1109/CVPR.2019.01054.
[85] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic Graph CNN for Learning on Point Clouds,” *ACM Trans. Graph.*, vol. 38, no. 5, 2019, doi: 10.1145/3326362.
[86] F. Yang, W. Choi, and Y. Lin, “Exploit All the Layers: Fast and Accurate CNN Object Detector with Scale Dependent Pooling and Cascaded Rejection Classifiers,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2129–2137, doi: 10.1109/CVPR.2016.234.
[87] Y. Liu, H. Li, J. Yan, F. Wei, X. Wang, and X. Tang, “Recurrent Scale Approximation for Object Detection in CNN,” Jul. 2017, [Online]. Available: http://arxiv.org/abs/1707.09531.
[88] P. O. Pinheiro, L. Tsung-Yi, R. Collobert, and P. Dollàr, “Learning to Refine Object Segments,” *Comput. Vis. Pattern Recognit.*, 2016.
[89] W. Liu *et al.*, “SSD: Single shot multibox detector,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, 2016, pp. 21–37.
[90] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, “Deep Layer Aggregation,” *Comput. Vis. Pattern Recognit.*, Jul. 2017, [Online]. Available: http://arxiv.org/abs/1707.06484.
[91] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Comput. Vis. Pattern*, Nov. 2013, [Online]. Available: http://arxiv.org/abs/1311.2524.
[92] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 3431–3440, doi: 10.1109/CVPR.2015.7298965.
[93] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature Pyramid Networks for Object Detection,” *Comput. Vis. Pattern Recognit.*, Dec. 2016.
[94] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” *Comput. Vis. Pattern Recognit.*, Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.02640.
[95] Q. Chen, L. Sun, Z. Wang, K. Jia, and A. Yuille, “Object as Hotspots: An Anchor-Free 3D Object Detection Approach via Firing of Hotspots,” *Comput. Vis. Pattern Recognit.*, Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.12791.
[96] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, “Selective Search for Object Recognition,” *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013, doi: 10.1007/s11263-013-0620-5.
[97] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, “Object Detection With Deep Learning: A Review,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.
[98] Z. Cai and N. Vasconcelos, “Cascade R-CNN: Delving into High Quality Object Detection,” *Comput. Vis. Pattern Recognit.*, Dec. 2017.
[99] T. Kong, A. Yao, Y. Chen, and F. Sun, “HyperNet: Towards Accurate Region Proposal Generation and Joint Object Detection,” Apr. 2016.
[100] J. Dai, Y. Li, K. He, and J. Sun, “R-FCN: Object Detection via Region-based Fully Convolutional Networks,” *Comput. Vis. Pattern Recognit.*, May 2016.
[101] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” *Comput. Vis. Pattern*, Dec. 2013, [Online]. Available: http://arxiv.org/abs/1312.6229.
[102] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, “Scalable Object Detection using Deep Neural Networks,” *Comput. Vis. Pattern Recognit.*, Dec. 2013.
[103] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick, “Inside-Outside Net: Detecting Objects in Context with Skip Pooling and Recurrent Neural Networks,” Dec. 2015.
[104] G. E. Hinton, A. Krizhevsky, and S. D. Wang, “Transforming Auto-Encoders,” in *ICANN*, 2011, pp. 44–51, doi: 10.1007/978-3-642-21735-7_6.
[105] G. W. Taylor, I. Spiro, C. Bregler, and R. Fergus, “Learning invariance through imitation,” in *CVPR 2011*, Jun. 2011, pp. 2729–2736, doi: 10.1109/CVPR.2011.5995538.
[106] A. Neubeck and L. Van Gool, “Efficient Non-Maximum Suppression,” in *18th International Conference on Pattern Recognition (ICPR’06)*, 2006, pp. 850–855, doi: 10.1109/ICPR.2006.479.
[107] C. R. Qi, O. Litany, K. He, and L. J. Guibas, “Deep Hough Voting for 3D Object Detection in Point Clouds,” *Comput. Vis. Pattern Recognit.*, Apr.

2019.
[108] W. Wang, R. Yu, Q. Huang, and U. Neumann, “SGPN: Similarity Group Proposal Network for 3D Point Cloud Instance Segmentation,” *Comput. Vis. Pattern Recognit.*, Nov. 2017, [Online]. Available: http://arxiv.org/abs/1711.08588.
[109] B. Yang *et al.*, “Learning Object Bounding Boxes for 3D Instance Segmentation on Point Clouds,” *Comput. Vis. Pattern Recognit.*, Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.01140.
[110] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal Loss for Dense Object Detection,” Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.02002.
[111] A. Creswell, K. Arulkumaran, and A. A. Bharath, “On denoising autoencoders trained to minimise binary cross-entropy,” Aug. 2017.
[112] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” Apr. 2020.
[113] Z. Wang and K. Jia, “Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection,” *Comput. Vis. Pattern Recognit.*, pp. 1–8, Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.01864.
[114] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361, doi: 10.1109/CVPR.2012.6248074.
[115] APTIV, “NuScenes Benchmark.” https://www.nuscenes.org/object-detection?externalData=all&mapData=all&modalities=Any (accessed Jul. 20, 2020).
[116] M. Bojarski *et al.*, “End to End Learning for Self-Driving Cars,” *Comput. Vis. Pattern Recognit.*, Apr. 2016, [Online]. Available: http://arxiv.org/abs/1604.07316.
[117] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López, “Multimodal End-to-End Autonomous Driving,” *Comput. Vis. Pattern Recognit.*, Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.03199.
[118] H. Haavaldsen, M. Aasboe, and F. Lindseth, “Autonomous Vehicle Control: End-to-end Learning in Simulated Urban Environments,” *Comput. Vis. Pattern Recognit.*, May 2019, [Online]. Available: http://arxiv.org/abs/1905.06712.
[119] KITTI, “The KITTI Vision Benchmark Suite.” http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d (accessed May 12, 2020).
[120] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013, doi: 10.1177/0278364913491297.
[121] H. Caesar *et al.*, “nuScenes: A Multimodal Dataset for Autonomous Driving,” 2019, [Online]. Available: http://arxiv.org/abs/1903.11027.
[122] P. Sun *et al.*, “Scalability in Perception for Autonomous Driving: Waymo Open Dataset,” *Comput. Vis. Pattern Recognit.*, Dec. 2019.
[123] J. Geyer *et al.*, “A2D2: Audi Autonomous Driving Dataset,” *Comput. Vis. Pattern Recognit.*, Apr. 2020.
[124] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, “The H3D Dataset for Full-Surround 3D Multi-Object Detection and Tracking in Crowded Urban Scenes,” *Comput. Vis. Pattern Recognit.*, Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.01568.
[125] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, “TrafficPredict: Trajectory Prediction for Heterogeneous Traffic-Agents,” *Comput. Vis. Pattern Recognit.*, Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.02146.
[126] M. Bucher, S. Herbin, and F. Jurie, “Hard negative mining for metric learning based zero-shot classification,” in *European Conference on Computer Vision*, 2016, pp. 524–531.
[127] A. Shrivastava, A. Gupta, and R. Girshick, “Training region-based object detectors with online hard example mining,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 761–769.
[128] B. Li, Y. Liu, and X. Wang, “Gradient Harmonized Single-Stage Detector,” *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 8577–8584, 2019, doi: 10.1609/aaai.v33i01.33018577.
[129] S. Il Oh and H. B. Kang, “Object detection and classification by decision-level fusion for intelligent vehicle systems,” *Sensors (Switzerland)*, vol. 17, no. 1, pp. 1–21, 2017, doi: 10.3390/s17010207.
[130] Q. He, Z. Wang, H. Zeng, Y. Zeng, S. Liu, and B. Zeng, “SVGA-Net: Sparse Voxel-Graph Attention Network for 3D Object Detection from Point Clouds,” *Comput. Vis. Pattern Recognit.*, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.04043.
[131] S. Qiao, L.-C. Chen, and A. Yuille, “DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution,” *Comput. Vis. Pattern*, Jun. 2020, [Online]. Available: http://arxiv.org/abs/2006.02334.
[132] J.-S. Lim, M. Astrid, H.-J. Yoon, and S.-I. Lee, “Small Object Detection using Context and Attention,” *Comput. Vis. Pattern Recognit.*, Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.06319.
[133] W. Li, K. Liu, L. Zhang, and F. Cheng, “Object detection based on an adaptive attention mechanism,” *Sci. Rep.*, vol. 10, no. 1, p. 11307, Dec. 2020, doi: 10.1038/s41598-020-67529-x.
[134] A. Paigwar, O. Erkent, C. Wolf, and C. Laugier, “Attentional PointNet for 3D-Object Detection in Point Clouds,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2019, pp. 1297–1306, doi: 10.1109/CVPRW.2019.00169.
[135] J.-B. Cordonnier, A. Loukas, and M. Jaggi, “On the Relationship between Self-Attention and Convolutional Layers,” *Comput. Vis. Pattern*, Nov. 2019, [Online]. Available: http://arxiv.org/abs/1911.03584.
[136] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, “Stand-Alone Self-Attention in Vision Models,” *Comput. Vis. Pattern Recognit.*, Jun. 2019, [Online]. Available: http://arxiv.org/abs/1906.05909.
[137] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A Survey on Deep Transfer Learning,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11141 LNCS, pp. 270–279, 2018, doi: 10.1007/978-3-030-01424-7_27.
[138] L. Torrey and J. Shavlik, “Transfer learning,” in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, 2010, pp. 242–264.
[139] T. Dao, A. Gu, A. J. Ratner, V. Smith, C. De Sa, and C. Ré, “A kernel theory of modern data augmentation,” *Proc. Mach. Learn. Res.*, vol. 97, p. 1528, 2019.
[140] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, “RandAugment: Practical automated data augmentation with a reduced search space,” *Comput. Vis. Pattern Recognit.*, Sep. 2019, [Online]. Available: http://arxiv.org/abs/1909.13719.
[141] S. Rajput, Z. Feng, Z. Charles, P. L. Loh, and D. Papailiopoulos, “Does data augmentation lead to positive margin?,” *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 9273–9282, 2019.
[142] B. Settles, “Active Learning Literature Survey,” 2010.
[143] S. Roy, A. Unmesh, and V. P. Namboodiri, “Deep active learning for object detection,” 2018.
[144] K. Chitta, J. M. Alvarez, and A. Lesnikowski, “Large-Scale Visual Active Learning with Deep Probabilistic Ensembles,” *Comput. Vis. Pattern Recognit.*, Nov. 2018.
[145] O. Sener and S. Savarese, “Active Learning for Convolutional Neural Networks: A Core-Set Approach,” *Mach. Learn.*, Aug. 2017.
[146] C.-C. Kao, T.-Y. Lee, P. Sen, and M.-Y. Liu, “Localization-Aware Active Learning for Object Detection,” *Comput. Vis. Pattern Recognit.*, Jan. 2018.
[147] H. H. Aghdam, A. Gonzalez-Garcia, J. van de Weijer, and A. M. López, “Active Learning for Deep Detection Neural Networks,” Nov. 2019.
[148] C.-A. Brust, C. Käding, and J. Denzler, “Active Learning for Deep Object Detection,” Sep. 2019.
[149] E. Haussmann *et al.*, “Scalable Active Learning for Object Detection,” Apr. 2020.
[150] W. Luo, B. Yang, and R. Urtasun, “Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 3569–3577, doi: 10.1109/CVPR.2018.00376.

[151] R. Jian *et al.*, “A Semantic Segmentation Based Lidar SLAM System Towards Dynamic Environments,” in *International Conference on Intelligent Robotics and Applications*, 2019, pp. 582–590, doi: 10.1007/978-3-030-27535-8_52.
[152] C. Debeunne and D. Vivet, “A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping,” *Sensors*, vol. 20, no. 7, p. 2068, Apr. 2020, doi: 10.3390/s20072068.
[153] A. B. Arrieta *et al.*, “Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI,” *Artif. Intell.*, Oct. 2019.