Comparison of major LiDAR data-driven feature extraction methods for autonomous vehicles

Fernandes, Duarte * Névoa, Rafael *
dduartefernandes@gmail.com
Silva, António Simões, Cláudia Monteiro, João
Novais, Paulo Melo-Pinto, Pedro

Abstract

Object detection is one of the areas of computer vision that has matured very rapidly. Nowadays, developments in this research area have been playing special attention to the detection of objects in point clouds due to the emerging of high-resolution LiDAR sensors. However, data from from a Light Detection and Ranging (LiDAR) sensor is not characterised by having consistency in relative pixel densities and introduces a third dimension, raising a set of drawbacks. The following paper presents a study on the requirements of 3D object detection for autonomous vehicles; presents an overview of the 3D object detection pipeline that generalises the operation principle of models based on point clouds; and categorises the recent works on methods to extract features and summarise their performance.

1 Introduction

Deep Learning research area has been witnessing tremendous growth, leading to developments that allow its implementation in a wide range of applications. It has been mostly used in object detection and classification tasks, with autonomous driving systems as one of its main targets. These deep learning algorithms for object detection can be implemented following a specific Neural Network architecture and adopt a technology, of which RGB cameras are the most widely used. However, this technology has some disadvantages - prone to adverse light and weather conditions and no depth information is provided - that have been hampering the 3D object detection models of achieving a fully reliable and feasible solution. For these reasons, Light Detection and Ranging (LiDAR) sensing technology has drawn the attention of both academic and industrial community. It offers 360°Field-of-View and introduces a third dimension that allows precise distance measurements, etc [7]. To achieve a fully

*Both authors contributed equally to this work

safety-critical system for autonomous vehicles, 3D object models must meet the following fundamental requirements: (1) real-time operation, which is feature sensor-driven; and (2) detection of a wide range of classes with high accuracy and (3) . These requirements are limited by state-of-the-art of LiDAR sensors. For instance, LiDAR sensors Velodyne VLS-128 and Velodyne HDL-64E are able to offer a frame rate up to 20 Hz [7]. Thus, inference time lower than 50 ms must be imposed to 3D object detection models. Moreover, models must detect also objects smaller than cars, such as cyclists or pedestrians, requiring a high density of points. Velodyne VLS-128 and Velodyne HDL-64E provide a point cloud with 3 and 1.3 million points, respectively, resulting in point clouds with different sparsity as shown in 1. However, computing more points will naturally affect negatively the inference time of a network. Assuring these requirements is one of the main challenges of 3D object detection models.

Although a 3D object detection is composed by several blocks, the key design to enable real-time operation and also accuracy is the process of feature enconding [6]. This block is the only block of the model pipeline that directly processes all input points from the point cloud for feature extraction that feeds the following block. Therefore, it is expected that the feature extractor are fast enough at processing the data to assure that satisfactory inference timed are achieved. However, as LiDAR beams are narrow by nature, sensors are likely to disregard narrow objects (e.g. lampposts or even persons) [10]. In order to overcome this limitation, current sensing technologies tend to either increase the number of reading when scanning the sensor surrounding or increase the LiDAR sensors resolution. As the number of points increases, it is expected to improve algorithms accuracy, but it might sacrifice the inference time of the solution. Therefore, features extractors must apply mechanisms to exclude points with no relevant information and assure that only extracted features that are meaningful in the context of current vehicle's surrounding scenario are forwarded to the following block. This article will pay special attention to the feature extractors addressed in the literature, analysing how techniques suggested on noteworthy research projects have evolved to better explore the nature of point clouds and optimise their performance metrics.

This paper is structured as follows: Section 2 highlights the main research challenges and describes the generic pipeline of a 3D object detection model; in section 3, we categorise feature extractors according to the architecture adopted, review the models addressed in the literature, and compare its performance on a benchmark dataset; and Section 4 provides a brief summary and concludes this document.

2 LiDAR-based Object Detection Challenges

The LiDAR sensor is becoming a key element in self-driving cars, mainly because of its long-range detection abilities, its high resolution and the good performance under different lighting conditions. Several approaches developed research based on LiDAR data to provide real-time object detection and identification as will be

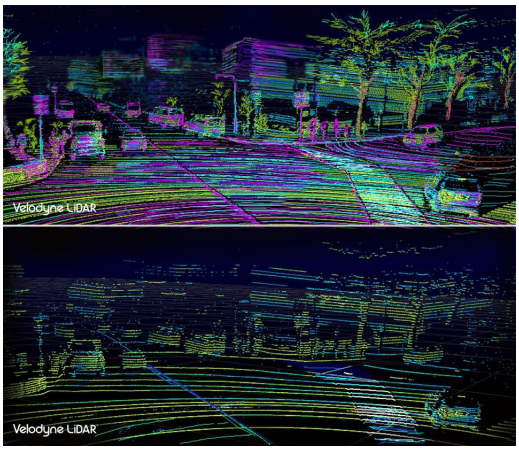

Figure 1: Point Clouds of the same scene obtained by two different sensors. The top image displays a point cloud with 3 million points, while the bottom image is the result of a frame with 1.3 millions points. Image from [2]

further shown. Thus, research lines were guided to the development of suitable object detector algorithm architectures for self-driving cars, using LiDAR sensing technology. However, LiDAR data consists of high-dimensional unstructured point clouds of sparse nature, which affect the solution both methodologically and computationally.

LiDAR sensors produce unstructured data containing typically around 10^6 3D points per 360-degree sweep, which impose large computational costs. Processing and inferring objects in a LiDAR point-cloud will directly impact the inference time of the model, which can make the solution unsuitable for real-time applications. Also, point clouds come with non-uniform density in different areas, which introduces a significant challenge for point set feature learning [9].

To allow a better trade-off between accuracy and inference time, several design choices were adopted. The resulting design solutions led to an appropriate downstream detection pipeline, which includes the following stages: **(1) LiDAR data representation**: Organise the point clouds into a structure that allows further computations;**(2) Data Object Detector**: According to the data representation, this stage aims to perform at least two tasks: feature map extraction and detection of objects of interest;**(3) Multi-task header**: This aims at performing object class prediction and bounding box regression.

In the next section, we will provide a proper description of a generic 3D object detection pipeline based on LiDAR data.

2.1 Generic 3D Object Detection Pipeline

Figure 2 depicts the generic pipeline architecture used by 3D object detection algorithms. This pipeline architecture evidences the diversity of solutions and design choices in each architecture stage. As mentioned before, this architecture comprises three stages: (1) Data Representation, (2) Data Object Detector, and (3) Multi-task header.

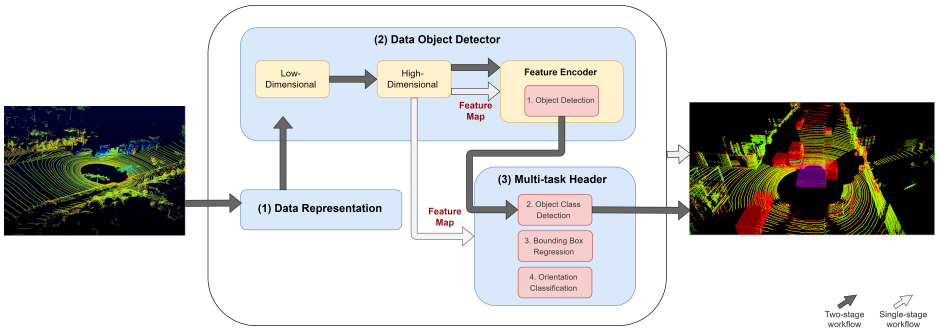

Figure 2: Generic architecture pipeline proposed for description of the operation principle of any 3D object Detection algorithm.

In the *Data Representation* stage, the raw point clouds are organised into a structure that allows the next block to process it more suitably according to their design choices. The existing models have adopted the following methodologies: voxels, frustums, pillars or 2D projections. The *Data Object Detector* will receive point clouds in a structure according to the before-mentioned representations and will perform at least two tasks: feature map extraction and detection of objects of interest in these point clouds. Several methodologies have been adopted to extract low- and high-dimensional features from point clouds to produce the feature map. Some research works opt by hand-crafted feature encoders, others use deep network architectures to apply convolutions and extract features. Finally, the *multi-task header* performs object class prediction, bounding box regression, and determination of objects orientation. These tasks are accomplished using the feature maps and the objects of interest generated by the *Feature Encoder*.

3 Feature Extractors

3.1 CNN-based

Convolutions networks are one of the preferred techniques used for extracting features from Spatio-temporal data as evidenced by the large of projects adopting it. Standard "dense" implementations of convolutional networks are

a well-matured technique, with multiple variations derived from extensive research studies, achieving high performance when applied to dense data. Applying these convolutional architectures to sparse data, such as LiDAR point clouds, is a very inefficient process. Considering that moving from two- to three-dimensional space, the number of points to process increases significantly and the higher dimensional space is, the higher the probability of relevant input data being sparse, it makes sense to take advantage of this spatial sparsity to speed up the feature extraction process. This minimises the number of points to be processed, reducing computational time and resources.

In [4], an approach for dealing with sparsity in 2D image classification and online handwriting recognition is presented, wherein a ground state is considered for hidden variables which receive no meaningful input, thus only having to be calculated once per forward pass during training and once for all passes during test time. Consequently, only the value of the hidden variables that differ from their ground state must be calculated, memoizing the convolutional and max-pooling operations. The forward propagation of the network is performed by calculating, for each layer, a feature matrix – composed of one-row vector for the ground state and one for each active spatial location in the layer – and a pointer matrix – to store the number of the corresponding row in the feature matrix. Based on the aforementioned work in [3], the same author adapted this concept to perform sparse convolutions on 3D space.

In the approaches presented in [4, 3], a site in the input/hidden layer is defined as active if any element in the layer that it takes as an input is not in its ground state. This leads to a rapid increase in the active sites of deeper layers, "dilating" the sparse data in every layer during a forward pass, making it impractical when implementing modern convolutional neural networks such as VGG networks, ResNets and DenseNets. To overcome these challenges, the research work in [5] offers a new approach to sparse convolutions, largely based on the mechanisms of [4, 3]. The authors of Submanifold Sparse Convolutional Networks propose two slightly different sparse convolution operations, a Sparse Convolution (SC) and Valid Sparse Convolutions (VSC). SC interprets active sites and ground states - replaced with a zero vector in this implementation - the same way as the sparse convolutions mentioned before, and since there is no padding, the output size is reduced. VSC's methodology also ignores ground states, replacing them with a zero vector, while distinctly handling the active states. First, padding is applied, so that the output is of the same size as the input. Then, instead of making a site active if any of the inputs to its receptive field is active, only its central position is considered, dealing with the dilation of the set of active sites and ensuring that the output set of active sites mirrors that of the input set. Without the problem of dilation of active sites, the networks implemented with these convolutional operators can be much deeper. On the other hand, restricting the output of the convolutions using this method can hinder the hidden layers ability to capture relevant information. Implementation of these convolutions is done using a feature matrix - containing one row for each active site – and a rule generation algorithm using a hash table to store the location and row for all active sites. A comparison between the operation

of sparse convolutions and submanifold sparse convolutions is depicted in Fig.3. The middle extractor in [12] uses a combination of the techniques mentioned above, taking advantage of submanifold sparse convolutions and sparse convolutions (to perform downsampling), improving the speed of the algorithm by using a custom GPU-based rule book algorithm.

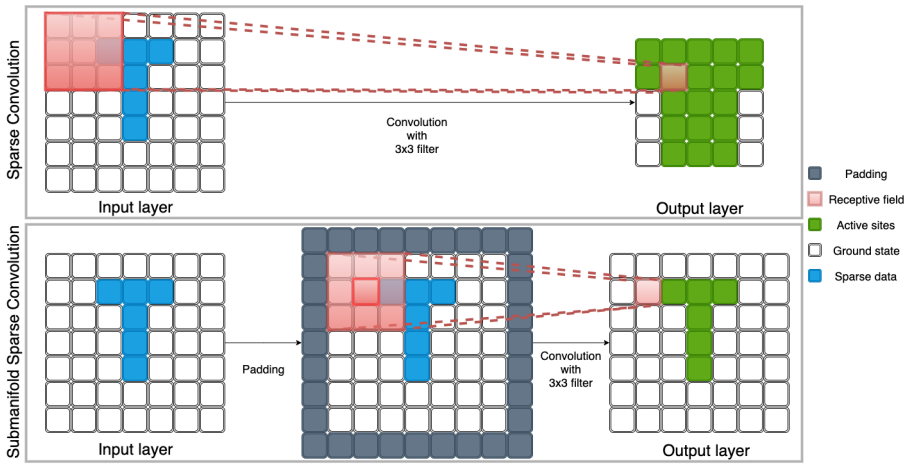

Figure 3: 3x3 Sparse convolution vs 3x3 Submanifold convolution.

PointNet [9] extract point-wise features directly from point clouds using a novel CNN-based architecture. This network encodes space features of each point within a subset of points from a euclidean space and extracts local and global structures. Then, it combines both input points and the extracted features into a global point cloud's signature. 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.

3.2 Compound methods

Figure 4 depicts an architecture of a compound feature extraction method to learn more meaningful information from point clouds. It merges two different types of feature extractors to complement each other, forming a "single-stage" end-to-end feature extractor.

The purpose of this synergy is to explore the advantages of feature extractors based on PointNet to encode local-features from regions, and a CNN-based extractor to take the role of global-feature extractor. The latter extractor leverages the local-features previously extracted to add more context to the shape description. This solution outputs a feature description in the form of a tensor, designated single feature map representation (c.f.Figure 4) to feed a Region Proposal Network (RPN) [13], [12], [6] or a Multi head [14].

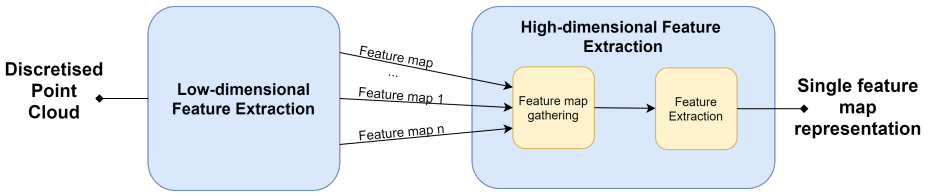

Figure 4: Architecture of a compound-based feature extractor.

Projects VoxelNet [13], SECOND [12], MEGVII [14] and PointPillars [6] are examples of recent and novel research works that followed up the above described method. The former three research works converts point cloud data into equally spaced 3D voxels that feed the feature extractor based in the architecture depicted in Figure 2. These projects differ from each other in the choices made for each stage as shown in table 1. Regarding the local-feature extractor, both VoxelNet and SECOND implement an solution called Voxel Feature extractor that follows a voxel-wise approach, whereas the MEGVII rely on a extractor called 3D feature extractor. The Voxel Feature extractor applies a simplified version of PointNet to take all the points within a voxel as input and takes 20 ms to extract point wises features from it. The 3D feature extractor addressed in MEGVII adopts the solution previously detailed that combines regular 3D sparse convolutions for features extraction and a submanifold sparse convolution to downsample the feature map. On the other hand, the PointPillars project divides the point cloud data in pillars and applies a local-feature extractor called Pillar Feature Net with a runtime of 1.3 ms [6]. The pillars points are subject to data augmentation as this model introduces a variable regard to its distance to the arithmetic mean of all points of respective pillar. To convert a point cloud to a sparse pseudo-image, the encoder first learns a set of features from pillars, then scatter them back to the original pillar locations to create the 2D pseudo-image. This image is forwarded to a 2D CNN that follows the same architecture as the one analysed in subsection 3.1. This solution relies in a simplified version of PointNet to extract local features from pillars.

Regarding the global-feature extractor, it consists in the element responsible for grouping all local feature into larger units and processing it to produce higher level features. Literature has shown that different solutions can be used to perform these tasks. The research project VoxelNet opted by implementing a 3D CNN, however it reduces the processing speed of features extraction. The 3D CNN runtime, 170 ms, is much higher than the required inference time of the whole network. For this reason, SECOND, MEGVII and PointPillars adopted faster run time local-features extractors. SECOND replaced 3D CNN by a Submanifold with Sparse Downsampling, reducing runtime to 20 ms. However, more recent works, such as MEGVII and PointPillars, have already outperformed SECOND in terms of inference time. PointPillars resorts to a 2D CNN with a runtime of 7.7 ms to output the final features as result of

the concatenation of all features originated from different strides. The project MEGVII sets a RPN like the one adopted in VoxelNet but to operate only as a global-feature encoder, i.e. does not perform the object detection as in [13]. Therefore, this RPN concatenates all features to construct the high resolution feature map need by further blocks to detect and classify objects. According to authors this project outperformed previous works in terms of accuracy, nevertheless, this model was tested against a manipulated dataset. Data augmentation techniques has been applied to generate a more balanced data distribution - this technique allow trained models to achieve better performing results. This project does not provide runtime analysis.

Table 1: Architecture design choices and run time of feature extractors and Benchmark KITTI accuracy performance on moderate level

Projects	**Local Extractor**	**Global Extractor**	**Time (ms)**	**AP(%)**
VoxelNet	Voxel-Feature	3D CNN	190	65%
SECOND	Voxel-Feature	Submanifold CNN	22	74%
MEGVII	Submanifold CNN	RPN	-	-%
PointPillars	Pillar-Feature	2D CNN	9	75%

3.3 Fusion-based methods

Several methods, such as PointsNet, MV3D and PointFusion, propose a combination of images and LIDAR data to improve performance and detection accuracy. These models succeeded in difficult scenarios, such as classifying small objects.

Frustum PointNet [8] uses a feature extraction based on an object-wise approach. It utilises 2D CNN detectors to propose 2D regions from the image and classify them. These 2D regions are then lifted to a 3D point cloud, becoming frustum proposals also called Frustum-cloud. Afterwards, they apply PointNet++ to the regions to further estimate the location, size and orientation of 3D objects.

Another relevant fusion-based work is MV3D [1], which fuses data from RGB images and LiDAR sensors, to extract high-dimensional features. It produces a multi-view representation of 3D point clouds and extracts feature maps from each view. They combine features from the front view, bird's eye view (BEV) and camera view to alleviate the information loss. Instead of using object-wise features, such as in F-PointNet, it uses region-wise features extracted from BEV. RPNs are applied to generate 3D object proposals using the bird's eye view map as input. Then, a region-based fusion network combines features from the multiple views and provides object proposal's class predictions and oriented 3D bounding box regression.

PointFusion [11] provides 3D object detection by fusing images and 3D point cloud's information. Firstly, they supply the RGB image to an RPN that proposes 2D object crops. Then, they combine point-wise features provided by

PointNet in 3D point clouds, image geometry features using ResNet architecture, and combine both outputs in a fusion network. This fusion network provides 3D bounding box regression for the object in the crops.

Fusion-based methods present some drawbacks that must be considered. First, the need for synchronisation time and calibration with the LIDAR sensor is a limiting factor. This time synchronisation and calibration task makes the solution more sensitive to sensor failures. Then, there is an increase in costs associated with the use of an additional sensor. For the three before-mentioned projects, we compare the results achieved in KITTI benchmark for 3D AP on KITTI test set for moderate difficult in three categories, namely pedestrian, car and cyclist. PointFusion states that their model in car class achieved 63 AP, while MV3D and F-PointNet achieved 62.68 and 70.39 respectively. For the pedestrian class, PointFusion achieved 28.03 AP, and F-PointNet 44.89 AP. For the cyclist class, PointFusion achieved 29.42 AP, and F-PointNet 56.77 AP. Regarding inference time, F-PointNet performs detection at 8.3 Hz, MV3D at 2.7Hz and PointFusion at 0.77 Hz. Due to the poorer accuracy on the detection of smaller objects, data-fusion is stated to be the bets approach, with research work [10] suggesting the use of a Ultrasonic Rangefinder.

4 Conclusions

Object detection research works have either prefer to preserve all geometric information and perform object detection with 3D ConvNets, or compact all information and perform 2D convolutions. The former approach requires expensive computations, which hinder the inference time (3D CNN takes 170 ms in VoxelNet [13] against the 7.7 ms runtime of the 2CNN implemented in [6]), which may turn the deployed model impractical for real-time requirements with no sign of accuracy performance improvement, according to the reported results. On the other hand, compacting the information in 2D projections and performing 2D convolutions is less computationally heavy, but introduces information losses. For this reason, some research works rely on compound methods that take advantage of the sparse nature of point clouds or fusion-based methods, suggesting that the direct application of a 2D CNN-based extractor leads to a poorer performance. In addition, the fusion-based performance results above presented show that excluding the third dimension of the point cloud leads to poorer accuracy, while the response time is still higher than compound-based solutions even though the number points are lower. Compound-based solutions are only based in LiDAR and have achieved the best performance so far, showing that splitting feature extractor into two stages conduct to better accuracy, while the achieved response time is satisfactory in some cases thanks to the exclusion of points with no representative value and application of 2D CNNs to process local-features instead of the whole point cloud. Summing up, in the present SoA, feature extractors solutions have achieved low runtimes and techniques to manage the trade-off between metrics have emerged, but accuracy of such solutions seems still far from the performance needed. For this reason,

projects have explored the fusion of technologies, showing promising results but also raising some new challenges. Furthermore, MEGVII project has shown that data augmentation strategies lead to models with better accuracy results.

Acknowledgements

This work is supported by European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project n^{o} 037902; Funding Reference: POCI-01-0247-FEDER-037902]

References

[1] Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1907–1915 (2017)

[2] Forbes: Velodyne rolling out 128-laser beam lidar. https://www.forbes.com/sites/alanohnsman/2017/11/29/velodyne-rolling-out-128-laser-beam-lidar-to-maintain-driverless-car-vision-lead/ (November 2017), accessed on 2019-11-22

[3] Graham, B.: Sparse 3d convolutional neural networks. CoRR abs/1505.02890 (2015), http://arxiv.org/abs/1505.02890

[4] Graham, B.: Spatially-sparse convolutional neural networks. CoRR abs/1409.6070 (2014), http://arxiv.org/abs/1409.6070

[5] Graham, B., van der Maaten, L.: Submanifold sparse convolutional networks. CoRR abs/1706.01307 (2017), http://arxiv.org/abs/1706.01307

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

[7] Prime, A.: Velodyne alpha puck. https://velodynelidar.com (Dec 2017), accessed on 2019-11-22

[8] Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 918–927 (2018)

[9] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660 (2017)

[10] Wiseman, Y.: Ancillary ultrasonic rangefinder for autonomous vehicles. International Journal of Security and Its Applications 12(5), 49–58 (2018)

[11] Xu, D., Anguelov, D., Jain, A.: Pointfusion: Deep sensor fusion for 3d bounding box estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 244–253 (2018)

[12] Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors 18(10), 3337 (2018)

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

[14] Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-balanced grouping and sampling for point cloud 3d object detection (2019)