Multimodal PointPillars for Efficient Object Detection in Autonomous Vehicles

Margarida Oliveira, Ricardo Cerqueira, João Ribeiro Pinto, Joaquim Fonseca, and Luís F. Teixeira, *Member, IEEE*

***Abstract*—Autonomous Vehicles aim to understand their surrounding environment by detecting relevant objects in the scene, which can be performed using a combination of sensors. The accurate prediction of pedestrians is a particularly challenging task, since the existing algorithms have more difficulty detecting small objects. This work studies and addresses this often overlooked problem by proposing Multimodal PointPillars (M-PP), a fast and effective novel fusion architecture for 3D object detection. Inspired by both MVX-Net and PointPillars, image features from a 2D CNN-based feature map are fused with the 3D point cloud in an early fusion architecture. By changing the heavy 3D convolutions of MVX-Net to a set of convolutional layers in 2D space, along with combining LiDAR and image information at an early stage, M-PP considerably improves inference time over the baseline, running at 28.49 Hz. It achieves inference speeds suitable for real-world applications while keeping the high performance of multimodal approaches. Extensive experiments show that our proposed architecture outperforms both MVX-Net and PointPillars for the pedestrian class in the KITTI 3D object detection dataset, with 62.78% in AP_{BEV} (moderate difficulty), while also outperforming MVX-Net in the nuScenes dataset. Moreover, experiments were conducted to measure the detection performance based on object distance. The performance of M-PP surpassed other methods in pedestrian detection at any distance, particularly for faraway objects (more than 30 meters). Qualitative analysis shows that M-PP visibly outperformed MVX-Net for pedestrians and cyclists, while simultaneously making accurate predictions of cars.**

***Index Terms*—Autonomous driving, deep learning, object detection.**

I. Introduction

With the advances in Artificial Intelligence, more specifically in Deep Learning, Autonomous Driving has gained an increased focus in the scientific community. These systems intend to address the safety of the passengers and other road users in various scenarios, by developing efficient and accurate systems that automate the driving activity and thus, prevent human failures. Moreover, pedestrian safety is a core requirement, since pedestrian accidents have the highest fatality rates compared with other types of road accidents [1]. With Artificial intelligence, it is expected that the number of pedestrian accidents and their severity will be diminished [2].

In this sense, the detection of pedestrians is one of the most important tasks in autonomous driving but also a specially challenging one, since for state-of-the-art algorithms, small objects are harder to detect [3], [4]. Although progress has been made recently, the development of effective and safe Autonomous Driving systems is still a very challenging task [5].

The implementation of self-driving cars needs to meet strict requirements regarding human safety. In a world conceived for humans, the deployment of such intelligent systems is a huge challenge. Especially, when these systems interact with pedestrians, a high degree of caution is required as their correct and timely detection and classification is critical.

For this reason, it is imperative to have intelligent systems designed to detect the diverse elements of the environment surrounding the self-driving vehicle. Thus, object detection applications are developed to detect and recognize the most relevant parts of the scene, as well as their position, size, orientation and class. Objects of interest usually consist of dynamic objects, including vehicles, such as cars, motorcyclists and cyclists, as well as pedestrians. This last category is often the focus when developing Autonomous Driving systems, as they are the most vulnerable road users [6].

The perception of the scene is crucial to develop self-driving vehicles, and therefore, appropriate sensor data enables the development of effective algorithms. The low resolution and texture of LiDAR (Light Detection and Ranging) point clouds make them less precise for the detection of smaller objects, such as pedestrians. These objects are clearly visible in 2D images, but are difficult to detect and distinguish from other objects in point cloud representations. On the other hand, the image does not provide depth information, however it yields fine-grained texture and color information. Therefore, in comparison with LiDAR-only or camera-only methods, fusion architectures can overcome the limitations of each data modality by combining information of LiDAR point clouds with images [7].

In this paper, we propose Multimodal PointPillars (M-PP), an object detector that combines LiDAR and RGB camera modalities. Based on MVX-Net (Multimodal VoxelNet) [8], M-PP combines the PointPillars [9] 3D detector with a 2D encoder for image feature extraction in a fusion architecture.

Manuscript received 19 February 2024; revised 17 May 2024; accepted 23 May 2024. Date of publication 4 June 2024; date of current version 16 July 2025. This work was supported by Portuguese National Funds through Project ATLAS - Trusted Autonomous Navigation, under Grant COMPETE2030, with Funding Reference COMPETE2030-FEDER-00646700. *(Corresponding author: Luís F. Teixeira.)*

Margarida Oliveira is with the University of Porto, 4099-002 Porto, Portugal, and also with the Bosch Car Multimédia, 4200-465 Braga, Portugal (e-mail: up202003050@up.pt).

Ricardo Cerqueira, João Ribeiro Pinto, and Joaquim Fonseca are with the Bosch Car Multimédia, 4705-820 Braga, Portugal.

Luís F. Teixeira is with the INESC TEC and Faculdade de Engenharia, University of Porto, 4099-002 Porto, Portugal (e-mail: luisft@fe.up.pt).

Digital Object Identifier 10.1109/TIV.2024.3409409

This match benefits from the efficient processing of point clouds in a pillar-based 2D representation as well as the additional information from the image, enabling more accurate detection of small objects, while keeping a low inference time. Relative to the prior art, the contributions of this work are:

- A new approach for multimodal object detection based on PointPillars and MVX-Net;
- State-of-the-art performance in object detection for autonomous vehicles, with especial benefits for smaller object classes;
- Considerable improvements in inference time for more viable real applications.

II. Related Work

The use of Deep Learning for scene understanding using point clouds is a growing research field that has gained an increased interest in the last years [10]. Benchmark datasets contributed to the growth of 3D point clouds in deep learning, namely KITTI [11], nuScenes [12], Waymo [13], among others. Diverse approaches have been proposed to address several problems related to point cloud perception, such as 3D shape classification, 3D object detection, and 3D point cloud segmentation. In this regard, we present a comprehensive survey of the most relevant proposed architectures for point cloud learning. The state-of-the-art proposals for 3D object detection can be categorized by the data representations they rely on. Thus, these architectures can be divided into three categories: grid-based methods, point-based methods, and graph-based methods [14].

A. Grid-Based Methods

Grid-based methods transform a point cloud into a regular grid to be fed into a neural network. VoxelNet [15] proposes an approach for extracting features from points in 3D voxels. A point cloud is divided into equally spaced 3D voxels and then, a unified feature representation is generated from the set of points within each voxel. This is done through the Voxel Feature Encoding (VFE) layer, which is a newly introduced technique to locally encode the points in a voxel. The detections are then obtained by passing the resulting volumetric representation into a Region Proposal Network (RPN). The main disadvantage of this method is the high computational cost, since the memory consumption and computational complexity of the 3D CNN grows cubically with the resolution of the voxel grid.

Moreover, due to the sparsity of the point clouds, numerous empty voxels are generated, leading to unnecessary use of computational resources. SECOND [16] proposes some modifications to the VoxelNet architecture. It implements a 3D sparse convolution method, which increases the speed of both training and inference. Also, a new form of angle loss regression is introduced to improve the performance of orientation estimation. This approach achieves lower inference time by avoiding unnecessary operations in empty voxels.

PointPillars [9], proposed in 2019, is motivated by the previously described works and aims to achieve fast processing time by dividing a 3D point cloud into vertical columns or pillars. A new encoder is proposed, which employs PointNets to learn a local representation of the points in each pillar, enabling the use of 2D convolutions. Instead of feeding voxels into 3D convolutions, this method applies 2D convolutions to the projected pillars, reducing the computation time.

The point cloud is first divided into pillars, where the height of each pillar covers the entire point cloud. Each point in each pillar is augmented with the x, y and z deltas between the point and the mean of the points in the pillar, as well as its offset to the center of the pillar in x and y. The resulting features of each pillar are passed to a simplified version of PointNet, outputting a tensor of size (C, P), where C is the number of channels per each non-empty pillar and P is the maximum number of pillars. After being encoded, the features are scattered back to their respective locations to create a pseudo-image of size (C, H, W), where H represents the height and W the width of the pseudo-image. Following this methodology, a 2D CNN backbone is employed, allowing efficient processing. A detection head is finally applied to perform object detection, similar to Single Shot Detector (SSD) [17], modified to output 3D bounding boxes with orientation.

Point-Voxel Feature Set Abstraction for 3D Object Detection (PV-RCNN) [18] takes advantage of the integration of both 3D voxel CNN and PointNet abstraction, thereby learning more discriminative point cloud features. It is a two-stage 3D detection framework, which achieves accurate 3D object detection from point clouds. The previous works in 3D detection are based on either 3D voxel CNN with sparse convolutions, which generate high-quality 3D object proposals, or PointNet-based architectures, which capture more accurate contextual information.

This proposal benefits from both methods, by applying a 3D voxel CNN with sparse convolution as the backbone. Two novel operations are implemented to perform the correspondence of the 3D object proposal features from the scene. First, the voxels of the overall scene feature volumes are transformed into a small number of feature keypoints, by employing voxel-to-keypoint scene encoding. Then, the scene keypoint features are aggregated into RoI (Region of Interest) grids to perform the confidence prediction and location refinement.

B. Sensor Fusion

The data acquired through a LiDAR sensor provides accurate depth information of the surrounding environment in the form of 3D point clouds. These sensors are robust to illumination changes and less affected by variations in weather conditions than visual cameras [26], [27]. However, LiDAR technologies produce sparse representations of distant objects and are unable to capture the fine textures of objects, unlike RGB cameras, which provide detailed texture information [26]. Moreover, the generated points are not uniformly distributed in 3D space, with points becoming more sparse for distant objects. For these reasons, object detection would benefit from a combination of the two types of data modalities, providing more robust perception systems. Therefore, the information acquired by the different data modalities is merged, following a sensor fusion strategy.

There are three main data fusion stages: early, which consists in merging raw data from each sensor modality and then feeding

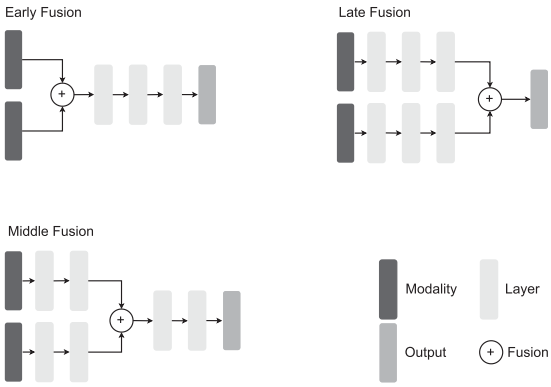


Fig. 1. Overview of the different types of data fusion methods.

that to a network; late, which involves having different networks for each type of data and then merging the outputs; and middle, which consists in fusing features from each modality at intermediate layers [26]. Fig. 1 provides an overview of each of these fusion methods.

1) Early Fusion: merges the raw or pre-processed data obtained from the different sensors before feeding them into a neural network. This strategy benefits from low memory costs, since it jointly processes the multiple sensor modalities and thus, only one neural network is employed. The network learns the combination of features from both modalities at an early stage, which allows exploring the information fully from the raw data. However, this technique has limitations in terms of model flexibility. Replacing an input sensor modality or changing input channels sizes, implies the complete retraining of the network. Moreover, early fusion suffers from spatial-temporal data misalignment, which implies that when calibration errors occur, data re-alignment is necessary. In addition, different sensor modalities commonly have different sampling rates, however, early fusion networks require their inputs to be synchronized, which may involve discarding information from modalities with faster sampling rates or repeating data from slower modalities.

MVX-Net [8] is a single-shot object detection architecture, published in 2019. It combines RGB and LiDAR data by implementing two early fusion approaches: PointFusion and VoxelFusion. The PointFusion method employs a 2D object detection network to extract high-level image features which encode semantic information. Then, by projecting each LiDAR point onto the image and collecting the features at their corresponding image coordinates, this algorithm is capable of gathering information about the presence of objects, as well as their class and localization from both 2D images and 3D point clouds.

On the other hand, VoxelFusion aggregates features from the RGB image at the voxel level. First, the 3D space is divided into a set of equally spaced voxels and the points are grouped into these voxels. By projecting each non-empty voxel onto the image plane, a 2D ROI is produced. A pre-trained detector network is employed to output feature maps, and by pooling the features within the ROI, a feature vector is produced for each non-empty voxel. These features are then appended to the voxel features obtained from the LiDAR points.

2) Late Fusion: uses separate networks to learn relevant features in each data modality, and then combines their outputs into a single predictor. These techniques allow including new sensor modalities without interfering with the existing networks, with only the respective network needing to be trained. However, having multiple models leads to higher computational cost.

3) Middle Fusion: Fuses data or relevant features from the different modalities in the middle of the architecture, one or multiple times. One approach that implements middle fusion is Multi-View 3D Object Detection (MV3D) [22], a sensor-fusion framework that uses LiDAR point clouds and RGB images as input and predicts oriented 3D bounding boxes. First, 3D object proposals are generated from the Bird's Eye View (BEV) projection of the LiDAR points, and then, multi-view features are deeply fused through region-based representation. Another approach is the Aggregate View Object Detection (AVOD) [23], which proposes a middle fusion based method. This proposal relies on feature extractors to generate feature maps from the BEV map and from the RGB image. An RPN is employed to generate non-oriented region proposals, taking as input the feature maps. Finally, the region proposals are fed to the detection network for the final classification.

C. Overview and Comparison

Table I presents results reported in [10] for several state-of-the-art architectures, using the test set of the KITTI 3D detection benchmark. In summary, PointPillars achieves the highest inference speed, with 62 Hz, however, performance on the pedestrian class is low relative to other methods. The results for MVX-Net were only reported for the class car, which achieves competitive performance compared with the other multimodal approaches. Moreover, by fusing information from LiDAR and RGB camera modalities at an early stage, its performance and run-time heavily depend on its 3D detection component, as a result MVX-Net is a good candidate for improvement. Therefore, we propose a new architecture that joins the advantages of these methods, by using a single-shot architecture and combining sensor data at an early stage. Our work proposes a fusion approach focused on pedestrian detection with low inference time, which can be a suitable prototype for a real-time environment.

III. Methodology

The proposed approach, Multimodal PointPillars (M-PP), modifies the MVX-Net framework to use PointPillars as a 3D detector. The MVX-Net framework enables the mixing and matching of different object detection networks for both image and LiDAR modalities. It is an early fusion methodology, and therefore, several architectures can integrate its pipeline. Although more recent and sophisticated base architectures could be used, we find that MVX-Net and PointPillars offer the optimal combination of simplicity and performance to allow us to obtain an efficient and flexible framework.

TABLE I
COMPARISON OF THE RESULTS OBTAINED FOR 3D OBJECT DETECTION ON THE KITTI TEST 3D DETECTION BENCHMARK, ADAPTED FROM [10]

Method	Architecture	Modality	Inference Rate (FPS)	Cars			Pedestrians			Cyclists		
				E	M	H	E	M	H	E	M	H
3DSSD [19]	Single Shot	L	25.0	88.36	79.57	74.55	**54.64**	**44.27**	40.23	**82.48**	64.10	56.90
PointRCNN [20]	Region Proposal	L	10.0	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
VoxelNet [15]	Single Shot	L	2.0	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
SECOND [16]	Single Shot	L	26.3	83.34	72.55	65.82	48.96	38.78	34.91	71.33	52.08	45.83
PointPillars [9]	Single Shot	L	**62.0**	82.58	74.31	68.99	51.45	41.42	38.89	77.10	58.65	51.92
PV-RCNN [18]	Region Proposal	L	12.5	**90.25**	**81.43**	**76.82**	52.17	43.29	**40.29**	78.60	63.71	**57.65**
PointGNN [14]	-	L	1.7	88.33	79.47	72.29	51.92	43.77	40.14	78.60	63.48	57.08
MVX-Net [8]	Single Shot	L & I	**16.7**	84.99	71.95	64.88	-	-	-	-	-	-
F-PointNets [21]	Region Proposal	L & I	5.9	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01
PointPainting [7]	Region Proposal	L & I	2.5	82.11	71.70	**77.08**	50.32	40.97	37.87	77.63	63.78	55.89
MV3D [22]	Region Proposal	L & I	2.8	74.97	63.63	54.00	-	-	-	-	-	-
AVOD [23]	Region Proposal	L & I	12.5	76.39	66.47	60.23	36.10	27.86	25.76	57.19	42.08	38.29
F-ConvNet [24]	Region Proposal	L & I	2.1	**87.36**	**73.39**	66.69	52.16	43.38	38.80	**81.98**	**65.07**	**56.54**
IPOD [25]	Region Proposal	L & I	5.0	80.30	73.04	68.73	**55.07**	**44.37**	**40.05**	71.99	52.23	46.50

The 3D bounding box IoU threshold is 0.7 for cars and 0.5 for pedestrians and cyclists. In the modality column, L refers to the LiDAR modality and I refers to the image modality. Also, the letters E, M, and H represent easy, moderate, and hard difficulties, respectively. The results are presented in percentages (%).

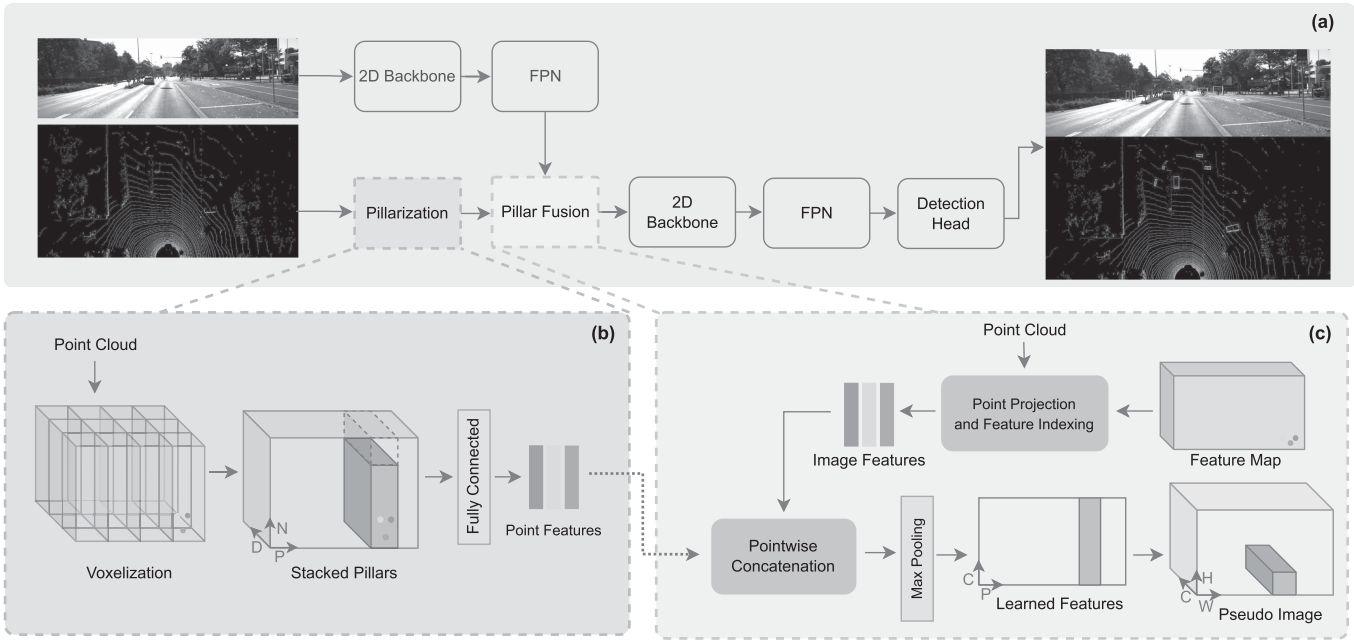


Fig. 2. Overview of Multimodal PointPillars architecture: (a) presents all modules that compose the fusion network; (b) denotes the preprocessing of the point cloud, by dividing it into pillars, stacking them, and learning a set of features (this process is hereby designated as pillarization); and (c) represents the fusion of image and point features, by first projecting the point cloud into the image and indexing the image features. Point and image features are fused through summation and a set of features is learned for each pillar, which are scattered back to a 2D pseudo-image.

A. Overall Architecture

The image modality aims to complement the information obtained by the LiDAR point clouds, performing as an auxiliary module for the object detection task. For this reason, M-PP uses a 2D convolutional network to learn image features. Fig. 2 shows our proposed architecture. In Fig. 2(a) the point cloud is passed to a module responsible for extracting the point features, here named pillarization, detailed in Fig. 2(b). These point features, as well as image features from the 2D network are passed to the fusion module, Pillar Fusion, which processes the whole fusion part, detailed in Fig. 2(c).

The processed point cloud is first equally divided into vertical columns or pillars, in Fig. 2(b). In addition to x, y, z, and intensity, the points are augmented by adding the arithmetic mean of the points in the pillar, as well as the offset in x and y, between each point and the center of the pillar, forming a stacked pillar representation of (D, P, N), which are the number of input features per point, maximum number of pillars, and maximum number of points per pillar, respectively. A set of

features is obtained for each point in the pillar by applying a fully connected layer, followed by BatchNorm [28] and ReLU [29]. The output tensor has size (C, P, N), which refers, respectively, to the number of point features, the maximum number of pillars, and the maximum number of points per pillar.

Simultaneously, the RGB image is fed to a ResNet backbone followed by an FPN, obtaining a feature map. Unlike MVX-Net, the dimensionality of these features is first reduced by a convolutional layer. Applying the calibration matrix, the raw point cloud is projected into the image plane, in Fig. 2(c). The image features are then indexed by their corresponding projected points in the point cloud. The image and point features are then combined in the Pillar Fusion module via a sum operation, in Fig. 2(c). Through a max pooling layer, a set of features are learned for each pillar, forming a tensor of (C, P). These features are then scattered back to the original pillar locations, creating a pseudo-image in a 2D representation of size (C, H, W), where H and W are the height and width of the generated feature map. A 2D backbone composed of convolutional layers is used followed by a single-shot detection head [17].

B. Objective Functions

This methodology uses the loss functions proposed in SECOND [16]. Each ground truth or anchor box can be represented as $(x, y, z, w, l, h, \theta)$ where x, y, and z are the position coordinates of the bounding box; w, l, h correspond to its width, length, and height; and θ is its orientation or yaw angle.

At first, the bounding box regression requires computing the differences between the ground-truth and anchor bounding box, relative to the size of the anchor. These relative differences are given by:

$$\Delta_x = \frac{x^{gt} - x^a}{d^a}, \ \Delta_y = \frac{y^{gt} - y^a}{d^a}, \ \Delta_z = \frac{z^{gt} - z^a}{h^a}, \tag{1}$$

$$\Delta_w = \log\left(\frac{w^{gt}}{w^a}\right), \ \Delta_l = \log\left(\frac{l^{gt}}{l^a}\right), \ \Delta_h = \log\left(\frac{h^{gt}}{h^a}\right), \tag{2}$$

$$\Delta_\theta = \sin(\theta^{gt} - \theta^a), \tag{3}$$

where $(x^{gt}, y^{gt}, z^{gt}, w^{gt}, l^{gt}, h^{gt}, \theta^{gt})$ and $(x^a, y^a, z^a, w^a, l^a, h^a, \theta^a)$ are the parameters of the ground truth and the anchor bounding box respectively, and $d^a = \sqrt{(l^a)^2 + (w^a)^2}$ is the diagonal of the base of the anchor box.

The bounding box localization is supervised via a smooth $L1$ loss of these relative differences:

$$L_{loc} = \sum_{\Delta b \in (x,y,z,w,l,h)} \text{Smooth}L1(\Delta b). \tag{4}$$

As the angle localization loss can not differentiate boxes with opposite orientations, the model is also tasked with predicting which side is the front of the object. This heading is learned by adding a direction classifier, using a cross-entropy loss function, as follows:

$$L_{dir} = -\sum_{d \in D} y_d \cdot \log(\hat{y}_d), \tag{5}$$

where D is the set of directions, y is the ground-truth direction and $\hat{y}$ is the model output direction. Following SECOND [16] and PointPillars [9], we consider only two opposing directions.

The classification training uses the focal loss [30]:

$$L_{cls} = -\alpha_t (1 - p_t)^\gamma \log(p_t), \tag{6}$$

where p_t is the model's confidence in the ground truth class. As recommended in [16], the parameters were set to $\alpha = 0.25$ and $\gamma = 2$.

The complete loss function is then given by a weighted sum of these components:

$$L_{total} = \frac{1}{N_{pos}} (\beta_{loc} L_{loc} + \beta_{cls} L_{cls} + \beta_{dir} L_{dir}), \tag{7}$$

where N_{pos} refers to the number of positive anchors. Following the original SECOND implementation, in this work the loss weighting coefficients are set as $\beta_{loc} = 2$, $\beta_{cls} = 1$ and $\beta_{dir} = 0.2$.

IV. Experimental Setup

In this section, we present an overview of the KITTI dataset and the metrics used to evaluate 3D object detection methods. We detail the hyperparameters and settings used to train our method, as well as the baselines, including data augmentation strategies.

A. Data Processing

To train and test the proposed framework, KITTI 3D object detection benchmark was used. This is one of the most widely used datasets in the literature for 3D object detection in Autonomous Driving. The dataset was recorded while driving around a mid-size city, in rural areas, and on highways in Karlsruhe, Germany. The state-of-the-art methods present their benchmark results using this dataset and therefore, the employment of KITTI becomes an obvious choice to perform a direct comparison with the published results. Typically, three classes are used in this dataset: car, pedestrian, and cyclist. A difficulty is assigned to each object regarding the minimum bounding box height, the occlusion level, and the maximum truncation, consisting of Easy (E), Moderate (M), and Hard (H) difficulties. The proposed architecture is evaluated in the validation set. The dataset was split into train and validation sets, using a standard division available in the literature. The 7481 training examples were divided into 3712 samples for the training set and 3769 samples for the validation set.

Average Precision (AP) is a widely used metric for object detection. AP consists of the average of the maximum precision achieved s, with at least a recall value r for a defined range of recall values. These values were computed by the authors of KITTI for 40 recall positions, rather than the previously used 11 recall positions [31]. Equation (8) shows this formulation:

$$AP = \frac{1}{40} \sum_{r \in \{\frac{1}{40}, \frac{2}{40}, \ldots, 1\}} \max_{\widetilde{r}: \widetilde{r} > r} s(\widetilde{r}). \tag{8}$$

In the KITTI benchmark, this metric is used to evaluate the model performance in 3D and Bird's Eye View (BEV) projection, along with the Average Orientation Similarity (AOS) [31]. Under our experiments, the AP in BEV (AP_{BEV}) was used to evaluate and benchmark the models.

B. Network Configuration and Training

Each point cloud is equally divided into pillars over the x and y axes. Under the conducted experiments, we defined 0.16 m as the pillar size for x and y. The maximum number of pillars (P) is 16000 and the maximum number of points per pillar (N) is 64. To generate the anchor boxes, we defined different settings for each class. The range used for cars was (0, 70.4), $(-39.68, 39.68)$, $(-1.78, -1.78)$ for x, y, and z, respectively, and for pedestrians and cyclists (0, 70.4), $(-39.68, 39.68)$, $(-0.6, -0.6)$. Cars have a width, length, and height of (0.6, 0.8, 1.73), pedestrians of (0.6, 1.76, 1.73), and cyclists of (1.6, 3.9, 1.56). The positive and negative thresholds were set to 0.6 and 0.45 for cars, and 0.5 and 0.35 for pedestrians and cyclists, respectively.

To benchmark the proposed methodology, PointPillars, and MVX-Net were end-to-end trained, using ADAM optimizer [32]. PointPillars with an initial learning rate of 0.0018, using batch size 6 for 160 epochs. MVX-Net was trained with an initial learning rate of 0.003, batch size 2 for 120 epochs.

To train the proposed architecture, M-PP, the ADAM optimizer was used with an initial learning rate of 0.0018. A weight decay of 0.01 was applied to reduce the learning rate during the training, and a batch size of 2 was used for 120 epochs. Moreover, an additional architecture was implemented, by using SECOND as the 3D detector in a fusion approach based on MVX-Net (M-SECOND). M-SECOND was end-to-end trained with the same setup as MVX-Net.

C. Data Augmentation

Data augmentation techniques were applied to train M-PP and PointPillars. Several architectures apply augmentation in order to improve performance on KITTI dataset [9], [16], [33]. This strategy is widely used to increase the diversity of the data, which leads to a better generalization of the model. The transformations are applied to the raw point clouds. They consist of random world flips along the x axis, random world rotations around the z axis with an angle sampled from $[-\frac{\pi}{4}, \frac{\pi}{4}]$ radians, a random world scaling with a scaling factor sampled from $[0.95, 1.05]$ and a global translation with (x, y, z) drawn from a $[0, 0.2]$ interval. To the input images, no type of augmentation is applied, other than simple resize and normalization procedures.

PointPillars applies a ground-truth sampling strategy, proposed in [16], that randomly introduces different ground-truth objects into the point clouds during training [34]. We extend this type of augmentation to multimodal detection by sampling image features only for the points in the original point cloud. We conduct multiple experiments using different configuration parameters or Sampling Strategies (SS) to study their effect on our proposed architecture: namely, SS1 applies sampling to pedestrians and cyclists, SS2 applies it only for cyclists, SS3 applies sampling to all three object classes (car, pedestrian, and cyclist), and SS4 applies sampling to all objects except pedestrians.

V. Experiments

In this section, we present the results obtained in the experiments performed. The results are assessed regarding performance and inference time. We present a comparison between the proposed architectures, highlighting the best trade-offs in this context. Additionally, we perform an in-depth analysis of the performance impact of multimodality in these architectures, considering the most relevant aspects for object detection.

A. Sampling Strategy

The M-PP baseline uses the transformations described in the previous chapter, without any ground-truth sampling. However, in order to optimize the model and achieve better results, a study was conducted by performing a set of experiments regarding ground-truth sampling. Hahner et al. [35] developed an extensive study focused on data augmentation in PointPillars. In particular, this strategy relies on two parameters: GT-Filtering and GT-Sampling.

GT-Filtering consists in excluding the annotations that contain less than a certain amount of points. GT-Sampling is the operation that samples the additional objects into the scene. During a pre-processing phase, the annotations and their corresponding points in the train set are saved, by iterating once through the entire dataset. During training, ground-truth objects are randomly added to the scene, provided that the additional objects do not collide with any of the original objects in the scene. The number of objects of each class to be added is pre-defined. Hahner et al. [35] only conduct experiments to assess the impact of data augmentation for the class Car, which is the most over-represented class in KITTI. Under this work, a simpler study is developed, applied for all the three classes in the dataset, particularly focusing in cyclists and pedestrians classes, which are underrepresented in the KITTI dataset.

Table II presents the experiments described, as well as the M-PP baseline (without ground-truth sampling). The configuration parameters were applied with the same values for all the difficulties. M-PP SS1 applies the same values to the parameters for sampling pedestrians and cyclists. The results show that the cyclists are highly improved, however, the performance on pedestrians decreases. In M-PP SS2 the same sampling configuration is applied for cyclists, but not employing any sampling to the other objects. This strategy shows considerable improvement for the pedestrians class, by only sampling the cyclists in the scene. An in-depth analysis of this behavior is further developed.

M-PP SS3 applies more ground-truths to all the three object classes and decreases the number of minimal points required for each sample object. The results show improvements in both cars and cyclists classes, however lower results in pedestrians class. Finally, M-PP SS4 configuration values achieved comparable results with M-PP SS1 and M-PP SS3 for cyclists. However, it does not employ any sampling to the pedestrians since from the previous experiments we could conclude that oversampling this class deteriorates its performance. Overall, the results show

TABLE II
COMPARISON OF THE RESULTS OBTAINED ON THE KITTI VALIDATION SET, USING THE SAMPLING STRATEGY PROPOSED IN [16]

Model	Sampling Strategy	Car			Pedestrian			Cyclist		
		E	M	H	E	M	H	E	M	H
M-PP Baseline	Results	90.18	87.42	86.22	67.29	61.34	57.92	78.56	56.70	52.97
SS1	GT-Filtering	5			10			10		
	Oversampling	15			10			10		
	Results	89.78	87.21	84.96	62.61	56.27	51.65	83.01	67.25	63.49
SS2	GT-Filtering	0			0			10		
	Oversampling	0			0			10		
	Results	89.11	87.03	79.51	**69.29**	**62.78**	**60.28**	80.03	60.93	57.69
SS3	GT-Filtering	5			5			5		
	Oversampling	20			20			15		
	Results	90.82	90.02	89.41	61.84	55.10	50.73	83.10	66.99	63.59
SS4	Points Filter Threshold	5			0			5		
	Oversampling	20			0			15		
	Results	89.22	86.68	84.59	66.67	60.40	58.24	83.61	64.78	61.10

The values of each configuration parameter are detailed and are the same for all the difficulties in each class. The AP_{BEV} with an IoU threshold of 0.7 for cars and 0.5 for pedestrians and cyclists was used for evaluation, presented in percentage (%).

TABLE III
COMPARISON OF THE RESULTS OBTAINED ON THE KITTI VALIDATION SET

Method	Cars			Pedestrians			Cyclists		
	E	M	H	E	M	H	E	M	H
PointPillars	**90.04**	**86.99**	79.74	57.33	52.60	47.33	**81.46**	**62.90**	**60.96**
MVX-Net	89.67	80.03	71.26	66.24	58.67	55.92	53.30	41.77	40.72
M-PP (proposed)	89.11	87.03	79.51	69.29	62.78	60.28	80.03	60.93	57.69
M-SECOND (proposed)	89.42	86.86	**85.07**	**70.91**	**64.04**	**61.89**	74.92	60.26	55.94

The AP_{BEV} IoU threshold is 0.7 for cars and 0.5 for pedestrians and cyclists. The results are presented in percentage (%). SS1 and SS3 refer to the sampling strategy applied to the data.

notable improvements in pedestrians and cyclists classes, highlighting M-PP SS2 with the best results in pedestrians class. Therefore, SS2 will be hereby referred as M-PP.

Table III shows the results obtained for each model architecture. Additionally, the more relevant experiments conducted by applying oversampling are also presented. The models were evaluated using the KITTI validation set. The AP_{BEV} metric is used with IoU thresholds of 0.7 for cars and 0.5 for pedestrians and cyclists.

MVX-Net (baseline) obtained the least favorable performance in the cyclists class. Cyclists are the least represented class in the KITTI dataset, and thus, as the baseline architecture does not apply any sampling strategy, the performance degrades. Moreover, the results demonstrate that PointPillars benefits from multimodality, achieving better results for the three classes with our approach M-PP. In the fusion approaches: MVX-Net, M-PP, and M-SECOND, the results achieved for the pedestrians class have minimal differences between the Moderate to Hard difficulties, varying less than 5%. This can be justified by the improvement that fusion strategies have regarding small objects and at longer distances, Fig. 3. Although the results obtained by M-SECOND are comparable with M-PP, the model shows decreased performance for the cyclists class.

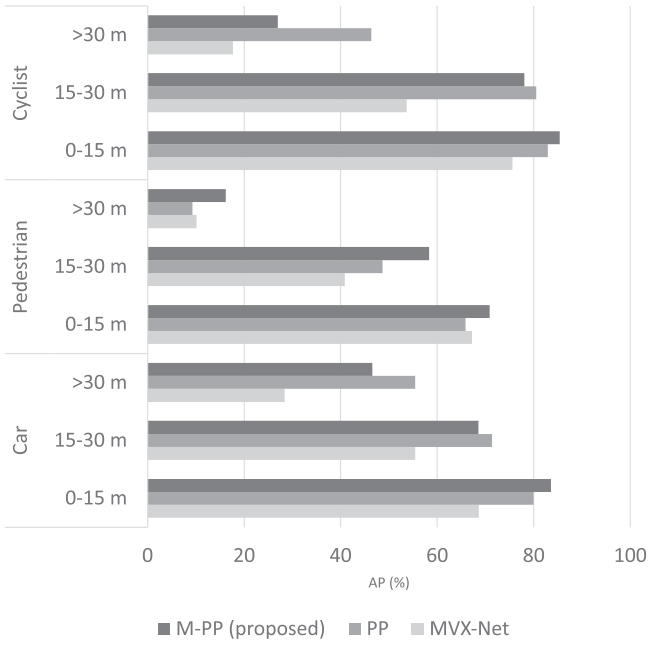


Fig. 3. AP_{BEV} performance of object detection based on distance (KITTI validation set).

B. Object Distance Analysis

In order to understand the improvements of our model compared to the baseline, an analysis of detection performance based on object distance was conducted. For each class, we split ground truth objects and detections according to their distance into three buckets, 0 to 15 m, 15 to 30 m, and greater than 30 m. For

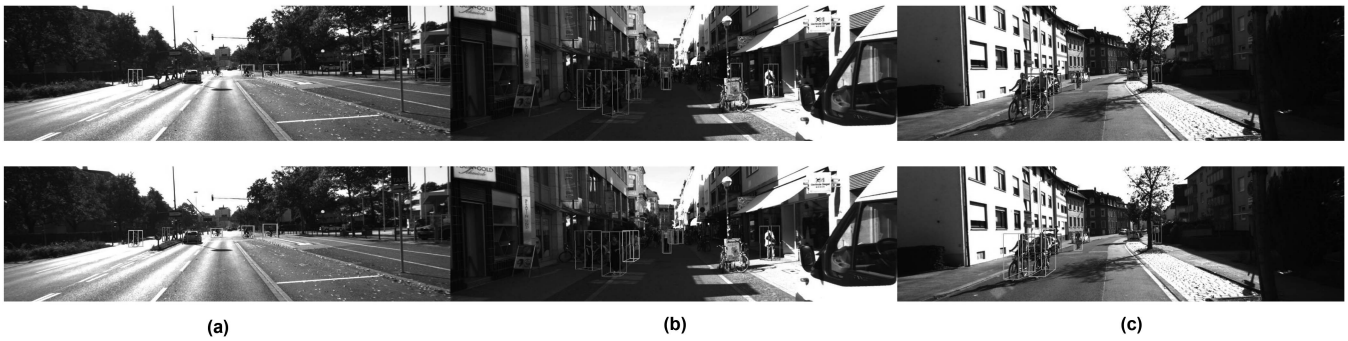


Fig. 4. Predictions from MVX-Net (above) and M-PP (below) in the KITTI validation set. Car detections in blue, pedestrian detections in red, and cyclist detections in orange.

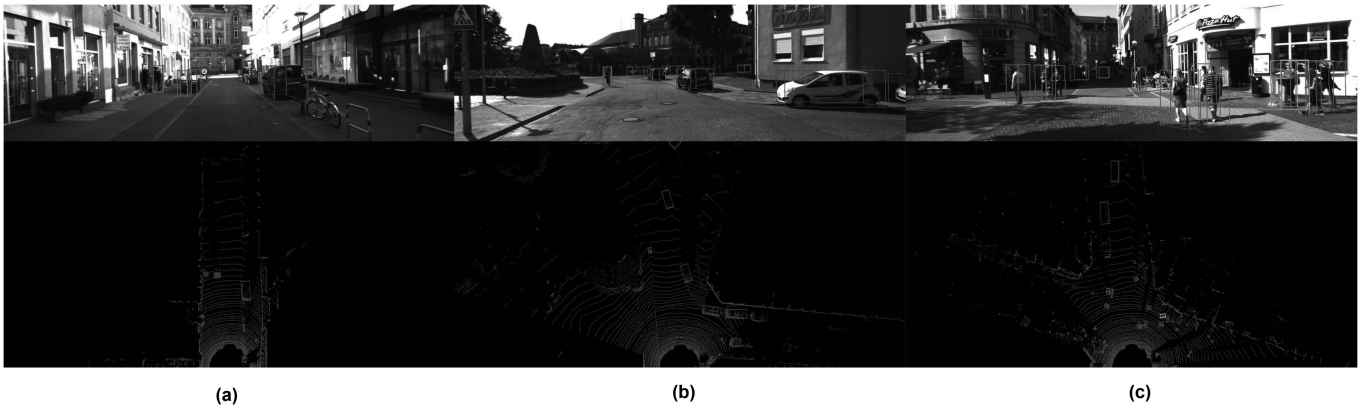


Fig. 5. Misclassification examples of M-PP in the KITTI validation set. Car detections in blue, pedestrian detections in red, and cyclist detections in orange.

each bin, we compute AP_{BEV}, ignoring both ground truth and detected objects that fall outside the bin. The results are shown in Fig. 3. Our detector outperforms both MVX-Net and PointPillars for closer objects ($0 - 15$ m), although PointPillars is superior for faraway cars and cyclists. M-PP improves pedestrian detection at every distance, achieving a performance approximately 10% higher than PointPillars for distances between 15 and 30 meters and 6% higher than MVX-Net for distances higher than 30 meters. For smaller objects, the sparseness of the point cloud can be compensated by the semantic information of the RGB image, enabling a more complete representation of the object than a point cloud-only approach. This behavior is evidenced by the higher performance of M-PP in pedestrians.

As for the better results of M-PP in close-range versus the benefits of PP in more distant cars and cyclists, this may be explained by the applied sampling strategy in each methodology. While PP applies sampling for all three classes (as in [16]), M-PP results refer to the use of SS2, which only applies sampling to the cyclists class. As visible in Table II, this results in improved overall performance in detecting pedestrians at all ranges. However, this improvement appears to come at the expense of detection performance on cars and cyclists at a distance greater than 15 meters, where PP outperforms the proposed method. According to these results, M-PP performance for more distant cars and cyclists could, if needed, likely be improved by applying a similar sampling strategy to the one used by PP, naturally at the expense of pedestrian detection performance.

C. Qualitative Analysis

Some qualitative results are presented in Figs. 4 and 5. The predictions are shown as 3D bounding boxes projected onto the RGB image and the point cloud in BEV projection. Different colors are used to distinguish the detections by class. Blue, red, and orange boxes for cars, pedestrians, and cyclists, respectively.

In Figure 4 predictions from MVX-Net (above) and M-PP (below) are presented in samples (a) 1069, (b) 1106, and (c) 2418 in the KITTI validation set. The examples provided show various scenes, which include diverse objects. For both models, predictions of cars are generally accurate, typically failing with false positives in objects of neighboring classes, such as vans or trucks. Pedestrians and cyclists are the most challenging classes and sometimes are misclassified as each other. Sample (b) shows an example of this behavior in M-PP results, which misclassifies a pedestrian as a cyclist. In this particular sample, the object is labeled as a pedestrian, although the person is holding a bicycle, which confuses the model, resulting in ambiguous behavior. This sample includes pedestrians at a distance of more than 25 meters who are not visible by MVX-Net but are accurately detected with M-PP. Moreover, MVX-Net predicts more false positives for the

TABLE IV
COMPARISON OF THE INFERENCE SPEED RESULTS OBTAINED WITH EACH ARCHITECTURE

Model	Inference (Hz)
PointPillars	**45.44**
MVX-Net	2.4
M-PP (proposed)	28.49
M-SECOND (proposed)	14.34

pedestrian class, as shown in example (a), where M-PP detects better pedestrians and cyclists.

Fig. 5 presents some misclassification examples of M-PP in samples (a) 147, (b) 615, and (c) 1693 in the KITTI validation set. Sample (a) shows people sitting on an outdoor coffee shop, detected by M-PP as pedestrians. Although the model correctly identifies these objects, there are no annotations in the dataset, contributing to the deterioration of the model performance under the metrics used. In Fig. 5(b), M-PP misclassifies a cyclist as a pedestrian, and in (c) it does the opposite. Moreover, in some samples, there are unlabeled objects that are accurately detected by M-PP, however for the evaluation pipeline they consist of false positives.

D. Inference Rate

To develop an effective application for object detection in the context of Autonomous Driving, inference speed is a core component for running in real-time. Therefore, the inference rate was computed for each architecture. The training and evaluation of the proposed framework, as well as the selected state-of-the-art architectures were conducted using a system with a Xeon Gold 6150 CPU and a Tesla V100-SXM2 32 GB GPU.

Table IV presents the average inference (in Hz) over the whole KITTI validation dataset (3769 samples). We find M-PP to be the fastest approach, running at 28.49 Hz. M-SECOND decreases more than 14 Hz in inference when compared with M-PP. In a real-world context, the developed product or application has to accomplish a set of requirements depending on the target hardware used. The resources to run the model would be much lower than the ones used to perform these tests, and thus, the model inference would automatically drop. Considering this, the results obtained with M-PP compared with M-SECOND reveal a relevant difference for real-time inference.

Fig. 6 presents a graphic of the trade-off between performance (in %) for AP_{BEV} in the pedestrians class and inference rate (in Hz). As in the KITTI benchmark, this trade-off was computed in moderate difficulty [11].

Both M-PP and M-SECOND achieved the best results, reaching a balance between performance and inference speed. M-SECOND shows a slightly higher performance, however, the gain in the inference speed with M-PP compensates for this. As LiDAR typically scans at a rate of 10 Hz [36], a solution intended to run in real-time should be able to execute at least this rate (< 100 ms). In this sense, our approach addresses real-world run-time requirements and therefore, it is suitable for deployment.

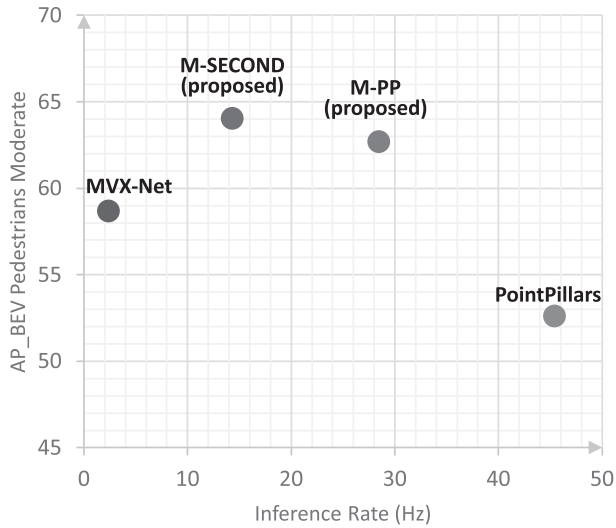


Fig. 6. Trade-off between performance (in %) for AP_{BEV} in moderate difficulty of pedestrians class and inference rate (in Hz). M-PP runs faster than the other fusion approaches, at 28.49 Hz, while achieving a competitive performance.

E. Experiments on Other Data

Additionally, in order to assess the behavior of the proposed methodology in different scenarios and conditions, experiments were conducted on the nuScenes object detection dataset [12]. The proposed methodology, M-PP, was trained, evaluated, and benchmarked against the state-of-the-art approach, MVX-Net, on the nuScenes validation set.

In these experiments, only the front camera was used for fusion. Additionally, models trained on nuScenes are typically given multiple sequential point clouds accumulated and ego-motion corrected. This means that some dynamic objects may be "stretched" in the input grid due to their movement. To compensate for this, an adaptation to the proposed architecture was explored using an FPN module similar to [37], denoted as M-PP FPN. By processing features at multiple scales, this FPN enables the model to deal with these variable "size" objects. For both MVX-Net and M-PP the procedure in [12] is followed, and 10 sequential LiDAR sweeps are concatenated into a single point cloud before providing them to the models.

The results of these experiments with M-PP and MVX-Net on nuScenes are presented in Table V. As one can see, even in the different conditions and scenarios offered by the nuScenes dataset, the M-PP approach offers improved performance over MVX-Net. In agreement with the discussions presented above, this is also true for classes such as 'Pedestrian', 'Motorcycle', or 'Traffic cone', which may once again denote the higher capabilities of the proposed method for detecting small objects. Moreover, the greatest improvements were found when using the alternative M-PP FPN architecture, which has shown considerable promise for small object detection and should be further explored in future research endeavors.

TABLE V
COMPARISON OF THE RESULTS OBTAINED ON THE NUSCENES VALIDATION SET

Method	mAP	NDS	Car	Truck	Bus	Trailer	Constr. Vehicle	Pedestrian	Motorcycle	Bicycle	Traffic Cone	Barrier
MVX-Net	15.39	27.97	56.7	16.5	11.6	0.2	0.1	52.4	6.4	0.1	1.5	8.5
M-PP (proposed)	31.0	37.3	76.7	**35.7**	**45.9**	21.8	3.4	57.5	15.8	0.03	16.8	36.0
M-PP FPN (proposed)	**36.79**	**41.94**	**78.3**	32.0	41.4	**23.6**	**3.8**	**70.5**	**33.2**	**7.7**	**30.5**	**47.0**

The results are presented in percentage (%).

VI. CONCLUSIONS AND FUTURE WORK

In this work, we tackle the overlooked problem of underperforming small object detection by proposing a novel fusion architecture for 3D object detection. We show that a pillar-based approach to 3D Object Detection benefits from the integration of image based data. An MVX-Net-like approach was used to extract relevant image features from the 2D detector. This architecture allows joint processing of image and point cloud information, allying the advantages of image and LiDAR modalities in an early stage. The results obtained with M-PP show an improvement on both performance and inference speed when compared with the baseline MVX-Net.

Additionally, experiments were conducted applying different parameters for ground-truth sampling. This technique showed to noticeably improve the performance for the cyclist class, which are the most underrepresented in the KITTI dataset, by adding these objects to the scene. Moreover, by oversampling the cyclist class, the performance in pedestrians improves. Through a qualitative analysis, we showed that M-PP is better at detecting pedestrians and cyclists, while maintaining accurate predictions of cars, when compared with MVX-Net. Our method achieved a compromise between performance and inference time, significantly accelerating the inference speed of MVX-Net with 28.49 Hz, as well as outperforming both MVX-Net and PointPillars in pedestrian detection, achieving 62.78% for pedestrians Moderate difficulty in the KITTI validation set. These conclusions are also confirmed by the results obtained on the nuScenes dataset.

In order to bring the proposed methodology closer to applicability in real vehicles, it would be useful to explore the use of a more compelling real-world dataset such as Waymo [13]. Additionally, although MVX-Net and PointPillars offered, in our opinion, an optimal combination of simplicity and performance, it would be important to explore similar multimodal 3D frameworks inspired on more recent base architectures. On the other hand, like MVX-Net, our proposal faces limitations arising from the availability, synchronization, and calibration of the LiDAR sensor. These approaches are only able to detect objects where there are LiDAR points in the point cloud. As an early fusion approach, they require alignment of sampling rates and temporal synchronization among sensors. The robustness of our method to calibration errors and other sensor failures is also yet to be studied. Future work on efficient multimodal object detection should also involve improving the robustness of these methods to sensor failures and miscalibrated data, since future safe autonomous driving systems will require efficient, effective and robust perception modules.

ACKNOWLEDGMENT

The authors also wish to acknowledge the creators of the KITTI 3D object detection dataset, the nuScenes dataset, and the MMDetection3D repository [38], which proved essential to this work.

REFERENCES

[1] Z. Wang, O. Zheng, L. Li, M. Abdel-Aty, C. Cruz-Neira, and Z. Islam, "Towards next generation of pedestrian and connected vehicle in-the-loop research: A digital twin co-simulation framework," *IEEE Trans. Intell. Veh.*, vol. 8, no. 4, pp. 2674–2683, Apr. 2023.

[2] B. Zhao, X. Zhang, H. Chen, and W. Zhu, "A novel prediction algorithm of pedestrian activity region for intelligent vehicle collision avoidance system," *IEEE Trans. Intell. Veh.*, vol. 8, no. 3, pp. 2173–2183, Mar. 2023.

[3] S. Iftikhar, Z. Zhang, M. Asim, A. Muthanna, A. Koucheryavy, and A. A. Abd El-Latif, "Deep learning-based pedestrian detection in autonomous vehicles: Substantial issues and challenges," *Electronics*, vol. 11, no. 21, 2022, Art. no. 3551.

[4] L. G. Galvão, M. Abbod, T. Kalganova, V. Palade, and M. N. Huda, "Pedestrian and vehicle detection in autonomous vehicle perception systems - A review," *Sensors*, vol. 21, no. 21, 2021, Art. no. 7267.

[5] M. Martínez-Díaz and F. Soriguera, "Autonomous vehicles: Theoretical and practical challenges," *Transp. Res. Procedia*, vol. 33, pp. 275–282, 2018.

[6] A. Rasouli and J. K. Tsotsos, "Autonomous vehicles that interact with pedestrians: A survey of theory and practice," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 3, pp. 900–918, Mar. 2020.

[7] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "PointPainting: Sequential fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4604–4612.

[8] V. A. Sindagi, Y. Zhou, and O. Tuzel, "MVX-Net: Multimodal VoxelNet for 3D object detection," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 7276–7282.

[9] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12697–12705.

[10] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[11] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[12] H. Caesar et al., "nuScenes: A. multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11618–11628.

[13] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2443–2451.

[14] W. Shi and R. Rajkumar, "Point-GNN: Graph neural network for 3D object detection in a point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1711–1719.

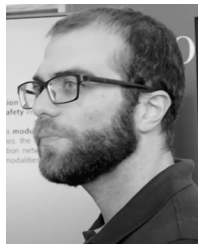
[15] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2018, pp. 4490–4499.

[16] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337.

[17] W. Liu et al., "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.

[18] S. Shi et al., "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10529–10538.
[19] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3DSSD: Point-based 3D single stage object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11040–11048.
[20] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.
[21] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 918–927.
[22] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1907–1915.
[23] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–8.
[24] Z. Wang and K. Jia, "Frustum ConvNet: Sliding frustums to aggregate local point-wise features for amodal 3D object detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 1742–1749.
[25] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive point-based object detector for point cloud," 2018, *arXiv:1812.05276*.
[26] D. Feng et al., "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 3, pp. 1341–1360, Mar. 2021.
[27] M. P. Da Silva, D. Carneiro, J. Fernandes, and L. F. Teixeira, "MobileWeatherNet for LiDAR-only weather estimation," in *Proc. Int. Joint Conf. Neural Netw.*, 2023, pp. 1–8.
[28] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[30] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2980–2988.
[31] J. R. Van Der Sluis, E. A. I. Pool, and D. M. Gavrila, "An experimental study on 3D person localization in traffic scenes," in *Proc. IEEE Intell. Veh. Symp. (IV)*, 2020, pp. 1813–1818.
[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015.
[33] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7652–7660.
[34] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, pp. 1–48, 2019.
[35] M. Hahner, D. Dai, A. Liniger, and L. Van Gool, "Quantifying data augmentation for LiDAR based 3D object detection," 2020, *arXiv:2004.01643*.
[36] C. Qu, S. S. Shivakumar, W. Liu, and C. J. Taylor, "LLOL: Low-latency odometry for spinning lidars," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 4149–4155.
[37] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2017, pp. 2117–2125.
[38] MMDetection3D Contributors, "MMDetection3D: OpenMMLab next-generation platform for general 3D object detection," 2020. [Online]. Available: https://github.com/open-mmlab/mmdetection3d

Margarida Oliveira received the M.S. degree in computer vision from the University of Porto, Porto, Portugal, in 2022. Since 2020, she has been a Deep Learning Researcher with Bosch Portugal. Her research interests include the development of interior vehicle sensor monitoring technologies for action and emotion detection and recognition, the study of perception algorithms for autonomous driving, including scene semantic segmentation using LiDAR technologies, and 3D object detection with multi-sensor data.



Ricardo Cerqueira received the M.S. degree in informatics from the Faculty of Engineering, University of Porto, Porto, Portugal, in 2018. Since 2019, he has been a Senior Deep Learning Researcher with Bosch Portugal. His research interests include uncertainty quantification and calibrationand safety and explainability of deep learning models, in the context of autonomous vehicles.

João Ribeiro Pinto received the M.S. degree in bioengineering and the Ph.D. degree in electrical and computers engineering from the University of Porto, Porto, Portugal, in 2017 and 2022, respectively. During 2022–2023, he was a Senior Deep Learning Researcher with Bosch Portugal. He has authored or coauthored more than 40 publications, supervised 16 M.Sc. students, and contributed to six projects. His research focuses on biometrics and wellbeing monitoring, especially within the context of intelligent vehicles. He was the recipient of 2022 Max Snijder Award by the European Association for Biometrics.

Joaquim Fonseca received the M.S. degree in industrial electronics and computers engineering from the University of Minho, Braga, Portugal, in 2015. Since 2016, he has been an AI Technical Lead with L4 Automated Driving Innovation, Bosch Portugal. His research interests include deep learning and computer vision, especially the development of advanced sensing technologies for interior environments that accurately capture and interpret human activities, emotions, and intentions within the vehicle's cabin.

Luís F. Teixeira (Member, IEEE) received the B.A. degree in electrical and computer engineering, the M.S. degree in computer networks and services, and the Ph.D. degree in electrical and computer engineering from the University of Porto, Porto, Portugal, in 2001, 2004, and 2009, respectively. From 2001 to 2008, he was a Researcher with INESC Porto. From 2008 to 2013, he was a Senior Scientist with Fraunhofer AICOS. From 2019 to 2021, he was the President of APRP, the Portuguese Association for Pattern Recognition. He is currently an Associate Professor with the Faculdade de Engenharia, Universidade do Porto, Porto, and Senior Researcher with INESC TEC. He has authored or coauthored more than 50 publications, with more than 800 citations, and supervised more than 100 M.Sc. students and seven Ph.D. students. His research interests include computer vision, machine learning, and human-centred computing. He has participated in numerous projects and is currently P.I. of the CAGING project, funded by the Portuguese Foundation for Science and Technology.