# A Multimodal Hybrid Late-Cascade Fusion Network for Enhanced 3D Object Detection

Carlo Sgaravatti, Roberto Basla, Riccardo Pieroni, Matteo Corno, Sergio M. Savaresi, Luca Magri, and Giacomo Boracchi

Politecnico di Milano, Italy

**Abstract.** We present a new way to detect 3D objects from multimodal inputs, leveraging both LiDAR and RGB cameras in a hybrid late-cascade scheme, that combines an RGB detection network and a 3D LiDAR detector. We exploit late fusion principles to reduce LiDAR False Positives, matching LiDAR detections with RGB ones by projecting the LiDAR bounding boxes on the image. We rely on cascade fusion principles to recover LiDAR False Negatives leveraging epipolar constraints and frustums generated by RGB detections of separate views. Our solution can be plugged on top of any underlying single-modal detectors, enabling a flexible training process that can take advantage of pre-trained LiDAR and RGB detectors, or train the two branches separately. We evaluate our results on the KITTI object detection benchmark, showing significant performance improvements, especially for the detection of Pedestrians and Cyclists. Code can be downloaded from: https://github.com/CarloSgaravatti/HybridLateCascadeFusion.

**Keywords:** 3D Object Detection · Multimodal · Autonomous Driving.

## 1 Introduction

3D Object Detection is a fundamental task in Computer Vision. The goal is to locate objects in 3D starting from 3D measurements and/or RGB images. 3D Object Detection solutions are broadly applied in Autonomous Vehicles (AV) where finding the location, dimension and orientation of cars, pedestrians and cyclists is key for safe navigation and road safety [8, 9, 26]. Solutions based on Deep Neural Networks can be *single-modal* [13, 29, 41, 43, 47] or *multi-modal* [5, 17, 22, 25, 39]. Single-modal detectors usually process either RGB images or LiDAR Point Clouds, while multimodal ones improve the accuracy of 3D Object Detection thanks to complementary information sources [19, 35]. Point Clouds allow an accurate representation of the 3D scene's geometry, but their sparsity does not permit a full understanding of the semantics of the scene. Indeed, LiDAR-based detectors can accurately detect cars, but they struggle to detect occluded, small or distant objects [26]. Unlike Point Clouds, RGB images do not provide depth and single-modal 3D detectors from RGB images struggle

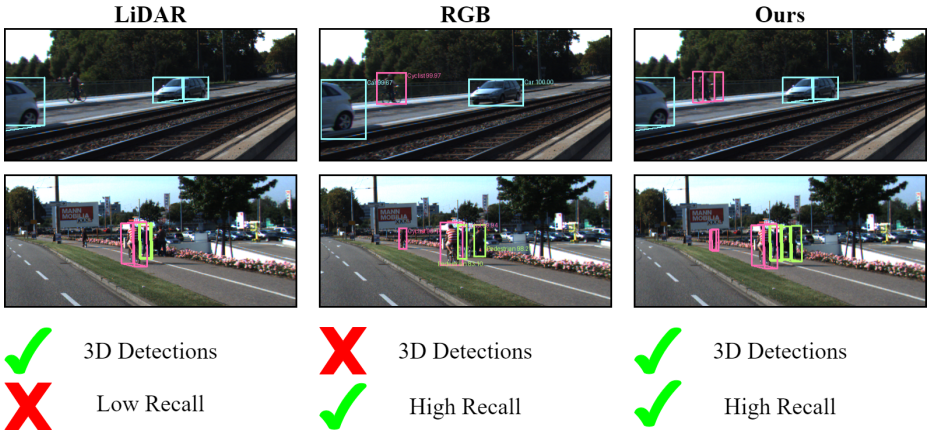



**Fig. 1: Left:** LiDAR branch struggles in detecting cyclists and pedestrians. **Center:** RGB branch correctly detects all the objects but lacks 3D information. **Right:** Our method recovers all the detections missed by the LiDAR and provides 3D information.

to accurately localize objects in 3D [26]. In contrast, RGB images do provide rich semantic information that can be used to distinguish small objects such as cyclists and pedestrians, especially when they are far from the sensor. Fig. 1 highlights the main strengths of our method. The LiDAR branch struggles with challenging objects like cyclists and pedestrians that are not detected in both examples. Our method can provide a higher recall by recovering such missed 3D objects leveraging RGB 2D information.

The main difficulty in training a multimodal object detection network is that LiDAR and RGB images have completely different data representations, namely a scattered set of 3D points and a fixed-size tensor. Based on how these representations are fused, multimodal approaches can be classified into *early* fusion, *cascade* fusion and *late* fusion [35]. Early fusion [5, 12, 39] combines the two information sources in the first stages of an end-to-end trainable network. The fusion of rich intermediate features comes at the cost of requiring paired data for training, and results in additional computational overhead, critical for real-time applications [19]. Cascade fusion [25, 36] exploits a 2D RGB detector to find region proposals in the 3D space, thus they heavily suffer limitations of RGB detectors. Finally, late fusion methods exploit two parallel RGB and LiDAR branches, focusing on filtering out False Positive LiDAR detections [18, 24].

To improve the detection accuracy for challenging classes like Cyclists and Pedestrians, we propose a hybrid late-cascade fusion approach. The proposed solution is illustrated in Fig. 2 and combines state-of-the-art methods for 3D and 2D detection in the context of a stereo camera system. Our key intuition is to recover missed detections of small or distant objects from the LiDAR branch by leveraging the geometric constraints between 3D and 2D predictions of LiDAR and RGB branches and between 2D predictions of different views in the RGB branch. We exploit late fusion principles to match the detections predicted by

the two single-modal branches and filter our LiDAR False Positives. We take advantage of both the computational efficiency of single-modal detectors and the possibilities of training the two networks separately or using pre-trained models. Moreover, we rely on cascade fusion principles [25, 36] to recover LiDAR False Negatives by extracting specific regions of the 3D space from RGB detections that are not associated with any LiDAR detection. Specifically, we first exploit geometric consistencies between 3D and 2D detections, to confirm or filter out LiDAR detections. In particular, we project 3D bounding boxes on the images and match these with 2D detections by solving an optimization problem based on the Intersection over Union (IoU). Then, to retrieve missed objects in 3D, we use epipolar constraints to pair 2D detections from the two images, and then we intersect their frustums. The point cloud at the frustums' intersection is fed to a specific 3D localization model to estimate the 3D bounding box. Our solution is based on simple rules and lightweight detection modules, that add on top of existing — possibly pre-trained — image and LiDAR detection networks. Our approach does not incur in high computational overheads since we exploit the cascade principle only for regions where the LiDAR detector fails.

Our contribution is two-fold:
*i*) We propose a novel hybrid fusion solution, combining late and cascade fusion approaches, to deal with Multimodal 3D Object Detection.
*ii*) We design a Detection Recovery module, exploiting an epipolar-based assignment procedure to assign pairs of 2D detections from different views.
Our solution, evaluated on the KITTI benchmark [9], improves single-modal LiDAR detectors, especially for cyclists and pedestrians, outperforming in some categories also multimodal detectors based on ad-hoc architectures. Extensive ablation studies demonstrate the effectiveness of our approach.

## 2 Related Work

3D Object Detection networks for AV can be divided into RGB-based, LiDAR-based and Multi-modal detectors.

**RGB Detectors**. Despite the lack of depth information and the presence of occlusions that characterize RGB images, it is possible to train RGB-based object detection networks to extract 3D information from images [4, 20, 33, 34]. These approaches have gained a lot of attention due to the low cost of the camera and the maturity of CNNs for extracting features from images. However, while 3D RGB detectors can successfully leverage the semantics of the image representation, they are usually characterized by poor 3D localization accuracy.

**LiDAR-based Detectors**. Object detection is more challenging on 3D Point Clouds rather than in images, due to their sparse and scattered nature. Several Deep Learning architectures have been proposed to address this challenge for LiDAR Point Clouds. In particular, Point-based methods [30, 42, 43] extract features by applying point operators to the raw point cloud, while Voxel-based approaches [31, 37, 41, 46–48] encode the point cloud into voxels and apply 3D CNNs to extract features. PointPillars [13] extracts features on vertical columns

and projects them into the Bird's Eye View (BEV) before applying 2D CNNs. PV-RCNN [29] exploits the advantages of both voxel-based and point-based approaches defining a Voxel Set Abstraction module to integrate voxel features into key points sampled from the raw point cloud. While LiDAR-based detectors accurately detect objects like Cars, the sparsity of the point cloud does not allow for the same precision in detecting smaller objects like Pedestrians and Cyclists.

**Multimodal 3D Object Detection**. Multimodal approaches for 3D Object Detection have gained a lot of popularity over the last few years. These methods can be divided into three categories, depending on the processing stage in which the RGB and LiDAR data are fused [35].

Early fusion solutions usually combine the features from two modalities in the early stage of an end-to-end trainable network. Prominent examples are: MV3D [5], which builds 3D proposals from the BEV and extracts RoI features from the images using the proposals projections, AVOD [12] and BEVFusion [17], that fuse the image features with the LiDAR ones on the BEV, PointFusion [40], that projects 3D points in the image and concatenates RGB features of the corresponding pixels to the features of the points, SFD [39] and VirConv [38], that use Depth Completion to build a dense pseudo point cloud from the image to be fused with the original point cloud. While early fusion networks have shown promising results, they are limited by the shortage of large-scale multimodal datasets [35]. Indeed, the application of Data Augmentation, usually employed to solve data scarcity issues [23, 28, 49], to multimodal data is limited by the necessity of maintaining alignment between the two modalities [35].

Cascade fusion approaches first process the RGB data to produce either bounding boxes or segmentation masks and use these to enrich or crop the raw point cloud. PointPainting [32] makes use of a 2D semantic segmentation network to enrich the point cloud with segmentation masks. Faraway-Frustum [44] combines a MaskRCNN and Frustum Network to detect objects that are far from the sensor. FrustumPointNet [25], FrustumConvNet [36] and FrustumPointPillars [21] lift 2D RGB detections to frustums to reduce the 3D search space for the LiDAR detector. However, cascade fusion solutions are limited by the performance of 2D detectors. Our approach exploits cascade fusion principles, but we first rely on the LiDAR detector to find a set of objects from the scene geometry. Then, we pair RGB detections to find missed 3D detections.

Late fusion approaches leverage two parallel object detection networks from each modality and combine their outputs in a final module. CLOCs [22] exploits Geometric and Semantic consistency between LiDAR 3D detections and RGB 2D detections and builds a fusion network to adjust the confidence scores of the 3D detections. Çaldıran *et al.* [50] filter out LiDAR False Positive detections with an asymmetric late fusion approach. Recently, Peri *et al.* [24] use a 3D RGB detector to filter the LiDAR detections that are not near any RGB detection according to the distance between the bounding box centers. Differently, Ma *et al.* [18] use a 2D RGB detector and match LiDAR detections with RGB ones on the image plane. These approaches allow removing LiDAR False Positive detections but

assume a high recall for the LiDAR detector. Our work also recovers LiDAR False Negatives by exploiting unmatched RGB detections to find new objects.

## 3 Problem Formulation

The input of our multi-modal 3D Object Detector is a set of $K$ pairs of stereo images $\mathcal{I} = \{(I_1^l, I_1^r), ..., (I_K^l, I_K^r)\}$, where $I_i^l \in \mathbb{R}^{W_i^l \times H_i^l \times 3}$ and $I_i^r \in \mathbb{R}^{W_i^r \times H_i^r \times 3}$ correspond to views of the same scene from the left ($l$) and right ($r$) cameras, and a Point Cloud containing $M$ points $\mathcal{P} = \{p_1, p_2, ..., p_M\}$, $p_j = (x_j, y_j, z_j, r_j)^T \in \mathbb{R}^4$, where $(x_j, y_j, z_j)$ is the position of the point $p_j$ and $r_j$ is the corresponding reflectance. The point cloud is expressed in LiDAR coordinates, with $T$ being the known transformation matrix from LiDAR to camera coordinates, which are in the coordinate system of a reference camera, *e.g.* $I_1^l$. We assume to know for each image $I_i^q$, with $q \in \{l, r\}$, the camera matrix $P_i^q \in \mathbb{R}^{3\times4}$, that projects 3D points in the coordinates of the reference camera to the image plane.

The 3D Object Detection algorithm computes a set of 3D bounding boxes $\mathcal{B}$ surrounding the objects in the 3D space:

$$(\mathcal{I}, \mathcal{P}) \longmapsto \mathcal{B} = \{(b_p^{3d}, s_p, \lambda_p) | b_p^{3d} \in \mathbb{R}^7, s_p \in [0, 1], \lambda_p \in \Lambda, p = 1, \ldots, P\} \quad (1)$$

where $b_p^{3d} = (x_p, y_p, z_p, l_p, h_p, w_p, \theta_p)^T$ concatenates the 3D coordinates of the center $(x_p, y_p, z_p)$, the dimensions $(l_p, h_p, w_p)$ of the bounding box, and $\theta_p$ is the yaw angle; $s_p$ is the confidence score, $\lambda_p$ is the associated label from the set of classes $\Lambda$ and $P$ is the number of detections.

## 4 Proposed Solution

At a high level, our method comprises 5 modules, as depicted in Fig. 2: RGB branch (a), LiDAR branch (b), Bbox Matching (c), Detection Recovery (d) and Semantic Fusion (e). The RGB and LiDAR branches leverage pre-trained models to predict a set of 2D and 3D detections, respectively. In the Bbox Matching module, 3D detections from the LiDAR branch are projected in every image and compared with 2D detections from the RGB branch. We compute the IoU between them to establish matches by solving an optimization problem. Unmatched RGB detections are fed to the Detection Recovery module where we compute Frustum Proposals to crop portions of Point Clouds that we further process to detect missed 3D objects. Finally, the Semantic Fusion module combines the labels in case the 2D and 3D predictions are discordant.

### 4.1 Bounding Box Matching

Let us assume the LiDAR branch returns a collection of 3D bounding boxes $\widehat{\mathcal{B}}$, as in Eq. (1). Similarly, the RGB branch predicts a set of 2D bounding boxes $\mathcal{R}_i^q = \{(b_r^{2d}, s_r, \lambda_r)\}_{r=1}^R$, for each single image $I_i^q$. Our Bbox Matching module aims at matching every $b_p^{3d}$ in $\widehat{\mathcal{B}}$ with possibly a single $b_r^{2d}$. Once the 3D

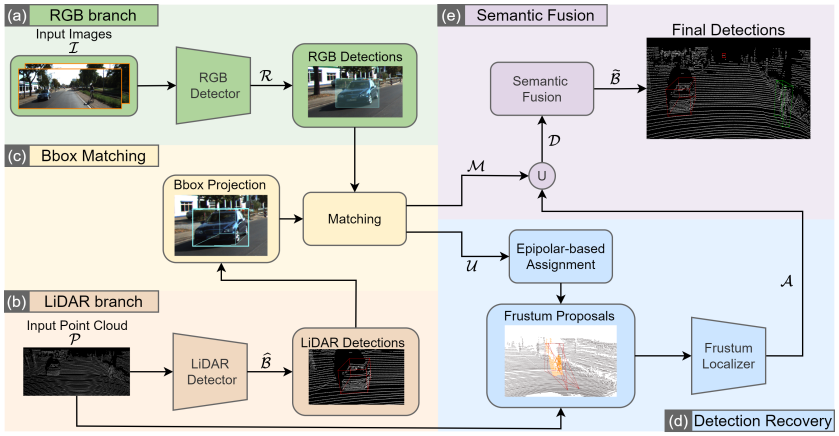



**Fig. 2:** Architecture. (a) the RGB branch outputs 2D detections $\mathcal{R}$ from each image. (b) the LiDAR branch computes 3D detections $\widehat{\mathcal{B}}$ from the input Point Cloud $\mathcal{P}$. (c) Bbox Matching projects the 3D detections and matches them with the 2D ones in each image ($\mathcal{M}$). (d) the unmatched RGB detections $\mathcal{U}$ are fed to the Detection Recovery module that matches 2D detections across stereo views, then extracts frustum proposals and uses the matched pairs to recover missed LiDAR detections ($\mathcal{A}$). (f) the Semantic Fusion module enforces semantic consistency between the LiDAR and the RGB branches.

bounding boxes are projected in the image planes, this boils down to solving an assignment problem that maximizes their IoU with $b_r^{2d}$. Specifically, as detailed in Algorithm 1, we extract corners $\{p_1, \ldots, p_8\}$ of each $b_p^{3d}$, expressed in homogeneous coordinates in the LiDAR reference system (Line 4). Then, we move the corners in the world reference system $Tp_j$ and project them to $\widetilde{p}_j = P_i^q T p_j$ using the camera matrix $P_i^q$ for each image $I_i^q$. We then extract an axis-aligned 2d bounding box $b_p^{proj}$ enclosing the projected corners (Lines 6-7). The assignment problem then becomes:

$$\max_x \sum_{p,r} IoU(b_p^{proj}, b_r^{2d}) x_{pr} \quad (2) \qquad \text{s.t.} \sum_{p,r} x_{pr} \geq \min\{P, R\} \quad (2a)$$

$$\sum_{p} x_{pr} \leq 1, \ \forall r \in \{1, \ldots, R\} \quad (2b) \qquad \sum_{r} x_{pr} \leq 1, \ \forall p \in \{1, \ldots, P\} \quad (2c)$$

where $x_{pr} \in \{0, 1\}$ denotes if $b_p^{3d}$ and $b_r^{2d}$ are matched or not. The constraints (2c) and (2b) specify that each detection in one image should be assigned with at most one detection in the other image, while (2a) enforces the assignment of all instances in a set. We solve the optimization problem with the Jonker-Volgenant algorithm [11]. We denote with $\mathcal{M}$ the set of matched bounding boxes. We consider 3D detections that have no matches in any image as False Positives (FP) of the LiDAR branch and we remove them.

Since our Bbox Matching procedure prunes out irrelevant 3D detections, we can adjust thresholds of the LiDAR branch and in particular we *i*) lower the threshold on the confidence score to include more detections in $\widehat{\mathcal{B}}$, and *ii*) relax

**Algorithm 1** Bbox Matching

**Input**:The set of RGB detections $\mathcal{R}$, the LiDAR detections $\widehat{\mathcal{B}}$, the calibration matrices $(T, \{(P_i^l, P_i^r)\}_{i=1}^K)$ and the IoU threshold $\tau_b$
**Output**: Matched pairs and unmatched RGB detections $(\mathcal{M}, \mathcal{U})$

**function** BboxMatching($\mathcal{R}$, $\widehat{\mathcal{B}}$, $T$, $\{(P_i^l, P_i^r)\}_{i=1}^K$, $\tau_b$)
  $\mathcal{M}, \mathcal{U} \leftarrow \emptyset, \emptyset$
  $C \leftarrow$ ExtractCorners($\widehat{\mathcal{B}}$)
  $C \leftarrow$ TransformCoordinates($C, T$)
  **for** $(i, q) \in \{1, \ldots, K\} \times \{l, r\}$ **do**
    $C_i^q \leftarrow$ ProjectCorners($C, P_i^q$)
    $\mathcal{B}_i^q \leftarrow$ AxisAlignedBboxes($C_i^q$)
    $(\mathcal{M}_i^q, \mathcal{U}_i^q) \leftarrow$ IouAssignment($\mathcal{B}_i^q, \mathcal{R}_i^q, \tau_b$)
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathcal{M}_i^q\}$
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{\mathcal{U}_i^q\}$
  **end for**
  **return** $(\mathcal{M}, \mathcal{U})$
**end function**

the IoU threshold in Non Maxima Suppression (NMS) to increase the number of 3D bounding boxes considered. We set both these thresholds to 0.3. As regards the threshold on the confidence score for the RGB branch, we fix it to 0.5 to guarantee the overall precision on 2D detections, so irrelevant LiDAR detections will not be matched. Finally, once the matching is performed, we remove from $\mathcal{M}$ all the matches with an IoU below $\tau_b$ and add the corresponding RGB detections to the set of unmatched RGB detections $\mathcal{U}$, which will be processed in the Detection Recovery module (Sec. 4.2). Please, note that the matching procedure is performed using only the geometry of the output detections and the 3D stereo vision constraints; we do not enforce semantic consistency at this level since LiDAR and RGB detections may not predict the same semantic class.

### 4.2 Detection Recovery

The set of unmatched RGB detections $\mathcal{U}$ typically corresponds to small and/or distant objects that the LiDAR branch has missed. Starting from $\mathcal{U}$, our Detection Recovery module aims to recover the corresponding missed 3D detections by leveraging two-view geometry of a stereo pair $(I_i^l, I_i^r)$. The output is a set $\mathcal{A} = \{\mathcal{A}_i^q | i \in \{1, \ldots, K\}, q \in \{l, r\}\}$ of pairs of RGB-LiDAR detections:

$$\mathcal{A}_i^q := \{(b_j^{3d}, b_j^{2d}, s_j^{3d}, s_j^{2d}, \lambda_j^{3d}, \lambda_j^{2d}) | (b_j^{2d}, s_j^{2d}, \lambda_j^{2d}) \in \mathcal{R}_i^q\}, \tag{3}$$

where $(b_j^{3d}, s_j^{3d}, \lambda_j^{3d})$ are the new 3D detections recovered.

At a high level, as detailed in Algorithm 2, the recovery of 3D detections is performed in 3 steps. First, each bounding box $b_l^{2d} \in \mathcal{U}_j^l$ in the left image is possibly matched to a corresponding bounding box $b_r^{2d} \in \mathcal{U}_j^r$ in the right view (Lines 3-5). Second, each pair of matched bounding boxes is backprojected to crop a 3D region (Line 7). Third, an ad-hoc Frustrum Localizer is used to detect objects in the cropped 3D region. 3D predictions are then validated by checking the geometric consistency with the input images (Lines 9-15).

The left-right Bounding Box matching is cast as an assignment problem similar to Eq. (2), but instead of maximizing the IoU, here we minimize a distance

**Algorithm 2** Detection Recovery

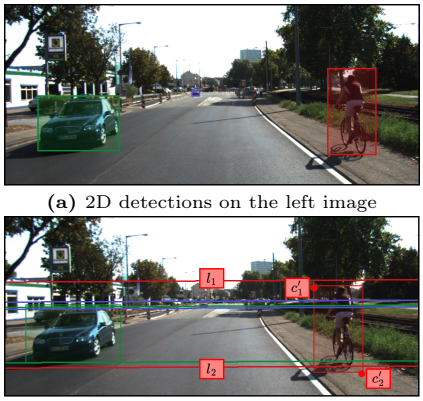
**Input**: Unmatched detections $(\mathcal{U}_i^l, \mathcal{U}_i^r)$, the Point Cloud $\mathcal{P}$, the calibration matrices $(T, P_i^l, P_i^r)$, the minimum number of points $p_{\min}$, the IoU threshold $\tau_r$ and the enlargement factor $e$
**Output**: Recovered pairs of detections $(\mathcal{A}_i^l, \mathcal{A}_i^r)$

1: **function** DetectionRecovery($\mathcal{U}_i^l$, $\mathcal{U}_i^r$, $\mathcal{P}$, $T$, $P_i^l$, $P_i^r$, $p_{\min}$, $\tau_r$, $e$)
2: $\quad \mathcal{A}_i^l, \mathcal{A}_i^r \leftarrow \emptyset, \emptyset$
3: $\quad F_{lr}^i \leftarrow$ FundamentalMatrix$(P_i^l, P_i^r)$ ▷ Epipolar-based assignment
4: $\quad \mathcal{D} \leftarrow$ ComputeDistanceMatrix$(\mathcal{U}_i^l, \mathcal{U}_i^r, F_{lr}^i)$
5: $\quad \mathcal{M}_{2d} \leftarrow$ Assign$(D, \mathcal{U}_i^l, \mathcal{U}_i^r)$
6: $\quad$ **for** $(b_l^{2d}, s_l^{2d}, \lambda_l^{2d}, b_r^{2d}, s_r^{2d}, \lambda_r^{2d}) \in \mathcal{M}_{2d}$ **do**
7: $\quad\quad \mathcal{P}_{lr} \leftarrow$ CropPointCloud$(\mathcal{P}, b_l^{2d}, b_r^{2d}, P_i^l, P_i^r, T, e)$ ▷ Frustum Proposal
8: $\quad\quad$ **if** $|\mathcal{P}_{lr}| > p_{\min}$ **then**
9: $\quad\quad\quad b^{3d} \leftarrow$ FrustumLocalizer$(\mathcal{P}_{lr})$ ▷ Localization
10: $\quad\quad\quad b_l^{proj}, b_r^{proj} \leftarrow$ BboxProjections$(b^{3d}, T, P_i^l, P_i^r)$
11: $\quad\quad\quad$ **if** $\max(IoU(b_l^{proj}, b_l^{2d}), IoU(b_r^{proj}, b_r^{2d})) > \tau_r$ **then** ▷ Geometric Consistency
12: $\quad\quad\quad\quad s_{RGB}, \lambda' \leftarrow$ MostConfidentRGB$(s_l^{2d}, \lambda_l^{2d}, s_r^{2d}, \lambda_r^{2d})$
13: $\quad\quad\quad\quad s' \leftarrow s_{RGB} \cdot IoU(b_l^{proj}, b_l^{2d}) \cdot IoU(b_r^{proj}, b_r^{2d})$
14: $\quad\quad\quad\quad \mathcal{A}_i^l \leftarrow \mathcal{A}_i^l \cup \{(b^{3d}, s', \lambda', b_l^{2d}, s_l^{2d}, \lambda_l^{2d})\}$
15: $\quad\quad\quad\quad \mathcal{A}_i^r \leftarrow \mathcal{A}_i^r \cup \{(b^{3d}, s', \lambda', b_r^{2d}, s_r^{2d}, \lambda_r^{2d})\}$
16: $\quad\quad\quad$ **end if**
17: $\quad\quad$ **end if**
18: $\quad$ **end for**
19: $\quad$ **return** $(\mathcal{A}_i^l, \mathcal{A}_i^r)$
20: **end function**

between bounding boxes based on epipolar geometry. Ideally, the corners of a bounding box in the right image should belong to the epipolar lines defined by the corners of the corresponding bounding box in the left image. When the stereo pair is rectified as in Fig. 3b, the epipolar lines are horizontal. However, the predictions of the RGB branch may have small inconsistencies, as shown in Figs. 3a and 3b, but still the corners $(c_1', c_2')$ are expected to be close to the epipolar lines $(l_1, l_2)$ defined by the bounding box in the other image. This is illustrated for the bounding box of the cyclist in Fig. 3b. We thus define the cost $d(\cdot,\cdot)$ for matching $b_l^{2d}$ and $b_r^{2d}$ as the sum of the Euclidean distances $\tilde{d}(\cdot,\cdot)$ between each corner of $b_r^{2d}$ and the epipolar lines of the corresponding corner of $b_l^{2d}$ (Line 5):

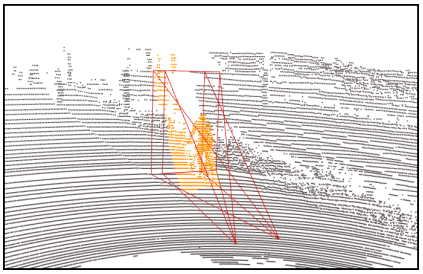
$$d(b_l^{2d}, b_r^{2d}) = \tilde{d}(l_1, c_1') + \tilde{d}(l_2, c_2'). \tag{4}$$

We solve the corresponding assignment problem to get matches $\mathcal{M}_{2d}$ using the Jonker-Volgenant algorithm. We then back-project and intersect in the 3D space each pair of bounding boxes in $\mathcal{M}_{2d}$, and we obtain Frustum Proposals (Fig. 3c) containing portions of LiDAR point cloud (Line 7). In practice, 2D detections may be slightly geometrically inaccurate, thus their Frustum Proposal might cut away useful points. Therefore, we enlarge the 2D bounding boxes by an enlargement factor $e$ for width and height, keeping the centers of the bounding boxes fixed. Frustum Proposals containing fewer points than a certain threshold $p_{\min}$ are ignored; otherwise, each proposal is fed to the Frustum Localizer, which localizes the object in the 3D space. We adopt as Frustum Localizer FrustumPointNet [25], and we enrich the Frustum Proposals input by a Gaussian mask proposed in FrustumPointPillars [21], added as an additional channel to




**(a)** 2D detections on the left image

**(b)** 2D detections and epipolar lines on the right image



**(c)** Frustum point cloud obtained by selecting the points inside the pair of frustums given by the two assigned detections.

**Fig. 3:** Illustration of the frustum proposals, obtained from the Detection Recovery module assignment procedure between two stereo images $I_i^l$ and $I_i^r$.

the Point Cloud. We assign the estimated label and the score of the most confident RGB detection to the localized 3D object returned by the FrustumPointNet. However, since RGB detections are typically more confident than those in 3D, we down-weight the confidence score by the IoU with the 2D detections as (Line 13):

$$s^{3d} = s_{RGB} \cdot IoU(b_l^{proj}, b_l^{2d}) \cdot IoU(b_r^{proj}, b_r^{2d}) \tag{5}$$

where $s_{RGB}$ is the score extracted by the two stereo RGB detections, and $(b_l^{proj}, b_r^{proj})$ are the projections in the two image planes of the bounding box predicted by the Frustum Localizer. We also discard 3D objects having a projection on 2D bounding boxes with an IoU lower than a threshold $\tau_r$ (Lines 9-10).

We remark that all cascade fusion approaches based on frustums [25,36] have been designed for single-view settings. Our solution, leveraging stereo pairs, analyzes intersections of frustums from multiple views thus feeds the 3D localization network with selected points that most likely refer to the target object. Therefore, we expect the Detection Recovery module to better find challenging objects, *i.e.* smaller or sparse objects.

### 4.3 Semantic Fusion

The Semantic Fusion module, detailed in Algorithm 3, enforces semantic consistency on all the 3D detection since matching RGB and LiDAR detections can refer to different predicted classes. In particular, the Semantic Fusion module replaces the LiDAR label and confidence score with the RGB ones [18], as we assume that RGB images contain better semantic information. The input of the semantic module contains the set of matched detections $\mathcal{M}$ from the Bbox Matching module and the set of recovered detections $\mathcal{A}$ from the Detection Recovery module, which we define as $\mathcal{D} = \{\mathcal{D}_i^q | i \in \{1, \dots, K\}, q \in \{l, r\}\}$,

**Algorithm 3** Semantic Fusion

**Input**: Matching detections in each view $\mathcal{D}$
**Output**: Final detection output $\tilde{\mathcal{B}}$

1: **function** SemanticFusion($\mathcal{D}$)
2: $\tilde{\mathcal{B}} \leftarrow \emptyset$
3: **for** $i = 1, \dots, K$ **do**
4: $\widehat{\mathcal{B}}^i \leftarrow$ GetUniqueLidarDetections($\mathcal{D}_i^l, \mathcal{D}_i^r$)
5: **for** $(b_j^{3d}, s_j^{3d}, \lambda_j^{3d}) \in \widehat{\mathcal{B}}^i$ **do**
6: **if** BothMatched($\mathcal{D}_i^l, \mathcal{D}_i^r, b_j^{3d}$) **then**
7: $(s_l^{2d}, \lambda_l^{2d}, s_r^{2d}, \lambda_r^{2d}) \leftarrow$ GetMatchedSemantics($\mathcal{D}_i^l, \mathcal{D}_i^r, b_j^{3d}$)
8: $q_{max} \leftarrow \text{argmax}\{s_l^{2d}, s_r^{2d}\}$
9: $\lambda_j' \leftarrow \lambda_{q_{max}}^{2d}$
10: $s_j' \leftarrow$ ProbabilisticEnsemble($s_j^{3d}, \lambda_j^{3d}, s_l^{2d}, \lambda_l^{2d}, s_r^{2d}, \lambda_r^{2d}$)
11: **else**
12: $(s^{2d}, \lambda_j') \leftarrow$ GetSingleMatchedSemantic($\mathcal{D}_i^l, \mathcal{D}_i^r, b_j^{3d}$)
13: $s_j' \leftarrow$ ProbabilisticEnsemble($s_j^{3d}, \lambda_j^{3d}, s^{2d}, \lambda_j'$)
14: **end if**
15: $\tilde{\mathcal{B}} \leftarrow \tilde{\mathcal{B}} \cup (b_j^{3d}, s_j', \lambda_j')$
16: **end for**
17: **end for**
18: **return** $\tilde{\mathcal{B}}$
19: **end function**

where $\mathcal{D}_i^q = \mathcal{M}_i^q \cup \mathcal{A}_i^q$. When multiple RGB views having different predicted classes are matched to the same 3D detection, we propagate the label from the most confident RGB detection (Lines 8-9). When all matching detections have the same predicted class, we adjust the confidence score of the LiDAR detections through the RGB confidence (Lines 10 and 13). We follow the probabilistic ensemble framework in [6], which assumes conditional independence between different modalities, obtaining the following formulation of the final detection confidence score for class $y \in \Lambda$ with $L$ matching modalities[1]:

$$p(y|\{x_i\}_{i=1}^L) \propto \frac{\prod_{i=1}^L p(y|x_i)}{p(y)^{L-1}} \tag{6}$$

where $p(y|x_i)$ is the confidence score for the $i$-th matching modality, and $p(y)$ is the class prior, which can be obtained by computing the per-class frequencies or treated as a uniform prior. In this work, we follow this second approach.

## 5 Experiments

We evaluate our proposed solution on the KITTI object detection dataset [9] and compare it against state-of-the-art single-modal (LiDAR only) and multi-modal detectors. Finally, we extensively ablate the components of our approach to demonstrate their effectiveness.

[1] In our case, between LiDAR and one or two images depending on whether there is a match on both the stereo images or on only one of the two.

### 5.1 Experimental Setup

**Dataset**. The KITTI object detection [9] dataset provides 7481 training samples and 7518 testing samples, with both LiDAR Point Clouds and RGB camera images. We follow the evaluation protocol defined in [5] to split the training dataset into 3712 training samples and 3769 validation samples and the KITTI evaluation protocol, which defines three classes of difficulties: easy, moderate and hard. Further details are in [9]. We evaluate our approach using the 3D Average Precision (AP) and the BEV AP.

**LiDAR/RGB Detectors**. We test our method using different LiDAR detectors: SECOND [41], PointPillars [13], PV-RCNN [29] and PartA2 [31], from the MMDetection3D [7] framework. We use the pre-trained PointPillars, PV-RCNN and PartA2 models freely available from MMDetection3D. Differently, we train SECOND on the Point Clouds of the KITTI training set, using the parameters suggested by [7], applying object noise, random flip on the BEV and ground-truth sampling as data augmentation procedures, and selecting the model associated with the highest 3D AP on the validation set at the 80th epoch, with 10 epochs as patience. As a 2D detector, we use a FasterRCNN [27] using MMDetection's [3] implementation, using ResNet101 [10] as the backbone and a Feature Pyramid Network (FPN) [15] as the neck to detect objects at different scales. We train the Faster RCNN model on the left images of the KITTI training set, applying data augmentation techniques from [2] to add Gaussian noise, motion blur and several transformations to simulate different climate conditions such as rain or sun flares. We use the 2D AP on the validation set to select the best model at the 200th epoch. To increase the performance on hard cases in all 2D detections and to prevent filtering out overlapped bounding boxes (due to occluded objects), we exploit Soft-NMS [1]. For the Frustum Localizer, we re-implement Frustum PointNet [25] and train it to localize the objects on the cropped Point Clouds extracted by the KITTI training dataset RGB ground truths. As suggested in [25], we add noise to the ground truth 2D bounding boxes to simulate inconsistencies. All the experiments were conducted on a cluster with multi-GPU nodes equipped with 8 A100.

### 5.2 Performance Comparison with Existing Solutions

We evaluate the performance of our late-cascade fusion module on the KITTI validation set, comparing it against single-modal LiDAR detectors and multi-modal frameworks. Tabs. 1 and 2 show, respectively, the 3D AP and BEV AP for Pedestrians, Cyclists and Cars. There are only marginal improvements for cars, for which the performance of LiDAR detectors is known to be good. In contrast, our method significantly increases the performance of single-modal detectors for pedestrians and cyclists. Specifically, since LiDAR-based detectors struggle to detect cyclists in the moderate and hard cases, in these two cases our method significantly improves the Cyclists' performance. As regards pedestrians, our method provides big improvements in all scenarios. Tab. 3 compares the results of our method with multi-modal solutions on the KITTI validation

**Table 1:** Comparison with single modal detectors (3D AP) on the KITTI val set.

| Detector | Car $AP_{3d}$ | | | Pedestrian $AP_{3d}$ | | | Cyclist $AP_{3d}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard |
| SECOND | 87.83 | 78.46 | 73.75 | 59.12 | 52.78 | 47.41 | 75.58 | 61.73 | 58.18 |
| SECOND+FasterRCNN | 87.98 | 79.27 | 74.37 | 65.98 | 59.73 | 53.47 | 85.24 | 72.77 | 68.27 |
| Improvement | +0.16 | +0.81 | +0.62 | +6.86 | +6.95 | +6.06 | +9.66 | +11.04 | +10.09 |
| PointPillars | 88.52 | 79.29 | 76.34 | 57.27 | 51.00 | 46.44 | 83.88 | 62.77 | 59.50 |
| PointPillars+FasterRCNN | 89.52 | 80.11 | 77.14 | 70.38 | 63.98 | 58.13 | 88.07 | 73.88 | 69.07 |
| Improvement | +1.00 | +0.82 | +0.80 | +13.11 | +12.98 | +11.69 | +4.19 | +11.11 | +9.57 |
| PartA2 | 92.45 | 82.88 | 80.64 | 60.61 | 53.59 | 48.86 | 90.45 | 70.17 | 65.52 |
| PartA2+FasterRCNN | 92.98 | 83.80 | 81.37 | 72.44 | 65.52 | 58.98 | 94.01 | 79.39 | 74.28 |
| Improvement | +0.53 | +0.92 | +0.73 | +11.83 | +11.93 | +10.12 | +3.56 | +9.22 | +8.76 |
| PV-RCNN | 91.82 | 84.53 | 82.42 | 66.72 | 59.27 | 54.31 | 90.36 | 73.26 | 69.36 |
| PV-RCNN+FasterRCNN | 92.95 | 86.09 | 83.32 | 73.87 | 67.40 | 62.67 | 91.01 | 77.25 | 72.01 |
| Improvement | +1.13 | +1.56 | +0.90 | +7.15 | +8.13 | +8.36 | +0.65 | +3.99 | +2.65 |

**Table 2:** Comparison with single modal detectors (BEV AP) on the KITTI val set.

| Detector | Car $AP_{BEV}$ | | | Pedestrian $AP_{BEV}$ | | | Cyclist $AP_{BEV}$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard |
| SECOND | 94.79 | 88.47 | 85.83 | 64.73 | 58.89 | 53.06 | 81.28 | 67.30 | 63.69 |
| SECOND+FasterRCNN | 95.75 | 89.69 | 87.05 | 73.01 | 66.93 | 60.49 | 91.33 | 80.30 | 75.37 |
| Improvement | +0.96 | +1.22 | +1.22 | +8.28 | +8.04 | +7.43 | +10.05 | +13.00 | +11.68 |
| PointPillars | 92.58 | 88.50 | 85.76 | 61.43 | 55.60 | 51.19 | 87.74 | 66.58 | 62.70 |
| PointPillars+FasterRCNN | 95.64 | 89.64 | 86.96 | 76.08 | 70.59 | 64.70 | 92.72 | 78.79 | 73.90 |
| Improvement | +3.06 | +1.14 | +1.20 | +14.65 | +14.99 | +13.51 | +4.98 | +12.21 | +11.20 |
| PartA2 | 93.55 | 89.38 | 87.13 | 64.19 | 58.05 | 52.22 | 93.87 | 73.46 | 68.83 |
| PartA2+FasterRCNN | 93.96 | 90.51 | 89.76 | 78.41 | 71.48 | 64.86 | 98.18 | 83.82 | 78.67 |
| Improvement | +0.41 | +1.13 | +2.63 | +14.22 | +13.43 | +12.64 | +4.31 | +10.36 | +9.84 |
| PV-RCNN | 94.43 | 90.78 | 88.67 | 69.53 | 62.12 | 57.18 | 92.81 | 75.55 | 70.88 |
| PV-RCNN+FasterRCNN | 95.92 | 92.63 | 90.07 | 77.65 | 72.70 | 68.03 | 94.93 | 80.30 | 75.43 |
| Improvement | +1.49 | +1.85 | +1.40 | +8.12 | +10.58 | +10.85 | +2.12 | +4.75 | +4.55 |

set, showing how plugging PV-RCNN and Faster RCNN in our hybrid late-cascade framework permits reaching state-of-the-art results on pedestrian and cyclists. Moreover, by using PointPillars, we can provide competitive results with a lower computational time compared to current multi-modal solutions. Note that Frames Per Second (FPS) are taken from the corresponding original publications, thus the comparison of our solution is not carried out on identical computing architectures. However, our experiments indicate that the results are in line with our implementations.

### 5.3 Ablation Study

We evaluate the contribution of each component of our module using the single-modal detector PointPillars as a baseline, a mainstream LiDAR detector in real-time applications. Tab. 4 summarizes the results, where the overall AP is reported for both 3D and BEV, aggregated w.r.t. the difficulty of the detections. The advantages of incorporating RGB information can be seen already from the Bbox Matching module, which improves significantly the metrics by reducing the False Positive detections. Moreover, the Detection Recovery module provides further improvements, especially for moderate and hard cases, characterized by

**Table 3:** Performance comparison with multi-modal solutions on the KITTI val set.

| Detector | Speed (FPS*) | Car $AP_{3d}$ | | | Pedestrian $AP_{3d}$ | | | Cyclist $AP_{3d}$ | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Easy | Mod. | Hard | Easy | Mod. | Hard | Easy | Mod. | Hard |
| CLOCs-PVCas [22] | - | 89.49 | 79.31 | 77.36 | 62.88 | 56.20 | 50.10 | 87.57 | 67.92 | 63.67 |
| Frustum PointNet [25] | 5.9 | 83.76 | 70.92 | 63.65 | 70.00 | 61.32 | 53.59 | 77.15 | 56.49 | 53.37 |
| Frustum PointPillars [21] | 14.3 | 88.90 | 79.28 | 78.07 | 66.11 | 61.89 | 56.91 | 87.54 | 72.78 | 66.07 |
| PointPainting [32] | - | 88.38 | 77.74 | 76.76 | 69.38 | 61.67 | 54.58 | 85.21 | 71.62 | 66.98 |
| PointFusion [40] | - | 77.92 | 63.00 | 53.27 | 33.36 | 28.04 | 23.38 | 49.34 | 29.42 | 26.98 |
| AVOD-FPN [12] | 10 | 84.41 | 74.44 | 68.65 | - | 58.80 | - | - | 49.70 | - |
| CAT-Det [45] | 10.2 | 90.12 | 81.46 | 79.15 | **74.08** | 66.35 | 58.92 | 87.64 | 72.82 | 68.20 |
| VirConv-T [38] | 10.2 | **94.98** | **89.96** | **88.13** | 73.32 | 66.93 | 60.38 | 90.04 | 73.90 | 69.06 |
| LoGoNet [14] | - | 92.04 | 85.04 | 84.31 | 70.20 | 63.72 | 59.46 | **91.74** | 75.35 | **72.42** |
| MLF-DET-V [16] | 10.8 | 89.70 | 87.31 | 79.34 | 71.15 | **68.50** | 61.72 | 86.05 | 72.14 | 65.42 |
| Ours (PointPillars+FasterRCNN) | 29.7 | 89.52 | 80.11 | 77.14 | 70.38 | 63.98 | 58.13 | 88.07 | 73.88 | 69.07 |
| Ours (PV-RCNN+FasterRCNN) | 10.1 | 92.95 | 86.09 | 83.32 | 73.87 | 67.40 | **62.67** | 91.01 | **77.25** | 72.01 |

**Table 4:** Ablation studies on the KITTI val set.

| Detector | Overall $AP_{3d}$ | | | Overall $AP_{BEV}$ | | | Speed (FPS) |
|---|---|---|---|---|---|---|---|
| | Easy | Mod. | Hard | Easy | Mod. | Hard | |
| PointPillars | 76.56 | 64.35 | 60.77 | 80.59 | 70.23 | 66.55 | 62.5 |
| FasterRCNN | - | - | - | - | - | - | 37.1 |
| + Bbox Matching | 80.87 | 69.94 | 65.42 | 86.32 | 76.43 | 72.48 | 35.4 |
| + Detection Recovery | 81.05 | 71.92 | 67.57 | 86.57 | 78.45 | 74.50 | 29.7 |
| + Semantic Fusion | **82.65** | **72.66** | **68.12** | **88.15** | **79.68** | **75.19** | 29.7 |

more False Negatives. This means that the method successfully recovers missed detections. Thus, the RGB detector finds objects that the LiDAR detector cannot detect. Finally, the Semantic Fusion module also contributes to the overall performance improvement, which confirms our hypothesis that the RGB branch is more reliable in providing semantic information.

**Inference Speed**. We measure the inference speed of the proposed solution for a real-time application on one A100 GPU. While PointPillars is known to have a fast point cloud encoder, the Faster RCNN that we used provides a lower computational speed, around 37 FPS. As reported in Tab. 4, the computational overhead given by the three modules is low, allowing us to match real-time requirements, being modern LiDAR sensors' frame rates usually between 10 and 20 FPS. We measure the inference speed of each module separately, considering that, in a real-time application, the LiDAR and RGB branches can be parallelized. Thus, we do not sum the computational time of the LiDAR and RGB branches, but we take the slowest one.

**Effect of Frustum Proposals enlargement**. Tab. 5 shows the performance of PointPillars in the LiDAR branch for several enlargement factors for Cyclists and Pedestrians, which are the main categories interested by the Detection Recovery module. It can be noticed how slightly enlarging the 2D bounding boxes is beneficial, especially for Cyclists. As the Frustum Localizer is trained to localize objects whose center is near the back-projection of the 2D bounding box center, increasing the enlargement factor too much does not result in significant performance degradation. However, it has to be noted that the number of points increases, and so does the computational complexity. Thus, we set the enlargement factor to 5%.

**Table 5:** Effect of the enlargement factor.

| Enlarge % | Cyclist $AP_{3d}$ | | | Pedestrian $AP_{3d}$ | | |
|---|---|---|---|---|---|---|
| | Easy | Mod. | Hard | Easy | Mod. | Hard |
| 0% | 86.36 | 72.52 | 67.76 | 70.37 | 63.90 | 58.02 |
| 5% | **88.07** | **73.88** | **69.07** | 70.38 | 63.98 | 58.13 |
| 10% | 86.29 | 73.82 | 69.05 | 70.50 | 64.02 | 58.13 |
| 20% | 86.28 | 73.79 | 67.71 | **70.67** | **64.09** | 58.12 |
| 30% | 86.30 | 72.42 | 67.65 | **70.67** | 64.06 | **58.24** |
| 50% | 86.24 | 72.49 | 67.64 | 70.60 | 63.99 | 58.08 |

**Table 6:** Effect of the detector on the Frustum Proposals.

| Detector | Cyclist $AP_{3d}$ | | | Pedestrian $AP_{3d}$ | | |
|---|---|---|---|---|---|---|
| | Easy | Mod. | Hard | Easy | Mod. | Hard |
| RGB Baseline | 86.35 | 70.19 | 65.45 | 67.83 | 60.83 | 54.62 |
| Frustum PointNet | **88.07** | **73.88** | **69.07** | **70.38** | **63.98** | **58.13** |

**Effect of the Frustum Localizer**. In Tab. 6 we compare the performance of using the Frustum Localizer on Frustum Proposals with a simple baseline, based only on geometry and RGB information, showing how the Frustum Localizer performs better for both Cyclists and Pedestrians. The baseline sets the BEV center of the bounding boxes as the mean BEV coordinates between the two intersections of the two bottom lines of each frustum (projected onto the BEV). Predefined anchor sizes are used as dimensions of the bounding box. As it is not possible to provide an accurate estimation of the yaw angle, we greedily compare the width and the length of the anchor with the distribution of the BEV points between the maximum and minimum depth of the two intersections: we set it to 0 if the difference between the maximum and minimum coordinate in the x-axis is higher than the one on the y-axis, and to $\frac{\pi}{2}$ if it is lower. The center height coordinate is set as the mean of the points inside the vertical column corresponding to the same BEV bounding box. The orientation estimation is the main issue with the simple baseline together with the center estimation if the 2D bounding boxes are not accurate, justifying the additional computational effort for the Frustum Localizer.

## 6 Conclusions

In this paper, we have proposed a hybrid late-cascade fusion approach that exploits a 3D LiDAR detector, a 2D RGB detector and the geometrical constraints of a stereo camera system. Our Detection Recovery module leverages RGB information to recover missed LiDAR detections. Our solution increases the performance of single-modal LiDAR detectors, especially for more challenging classes like Cyclists and Pedestrians. Moreover, our solution can combine any state-of-the-art detector (potentially without the need of re-training), without incurring in a prohibitive computational overhead.

**Acknowledgement.** This paper is supported by the FAIR (Future Artificial Intelligence Research) project, funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence) and by GEOPRIDE ID: 2022245ZYB, CUP: D53D23008370001 (PRIN 2022 M4.C2.1.1 Investment). Model training and testing were possible thanks to the HPC grant from by the Ministry of Education, Youth and Sports of the Czech Republic through the e-INFRA CZ (ID:90254).

## References

1. Bodla, N., Singh, B., Chellappa, R., Davis, L.S.: Improving object detection with one line of code. CoRR **abs/1704.04503** (2017), http://arxiv.org/abs/1704.04503
2. Buslaev, A., Iglovikov, V.I., Khvedchenya, E., Parinov, A., Druzhinin, M., Kalinin, A.A.: Albumentations: Fast and flexible image augmentations. Information **11**(2) (2020). https://doi.org/10.3390/info11020125, https://www.mdpi.com/2078-2489/11/2/125
3. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. arXiv preprint arXiv:1906.07155 (2019)
4. Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., Urtasun, R.: Monocular 3d object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016)
5. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. CoRR **abs/1611.07759** (2016), http://arxiv.org/abs/1611.07759
6. Chen, Y.T., Shi, J., Ye, Z., Mertz, C., Ramanan, D., Kong, S.: Multimodal object detection via probabilistic ensembling. In: European Conference on Computer Vision. pp. 139–158. Springer (2022)
7. Contributors, M.: MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d (2020)
8. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3213–3223 (2016)
9. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3354–3361 (2012). https://doi.org/10.1109/CVPR.2012.6248074
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
11. Jonker, R., Volgenant, A.: A shortest augmenting path algorithm for dense and sparse linear assignment problems. Computing **38**(4), 325–340 (1987). https://doi.org/10.1007/BF02278710, https://doi.org/10.1007/BF02278710

12. Ku, J., Mozifian, M., Lee, J., Harakeh, A., Waslander, S.L.: Joint 3d proposal generation and object detection from view aggregation. CoRR **abs/1712.02294** (2017), http://arxiv.org/abs/1712.02294
13. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. CoRR **abs/1812.05784** (2018), http://arxiv.org/abs/1812.05784
14. Li, X., Ma, T., Hou, Y., Shi, B., Yang, Y., Liu, Y., Wu, X., Chen, Q., Li, Y., Qiao, Y., et al.: Logonet: Towards accurate 3d object detection with local-to-global cross-modal fusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17524–17534 (2023)
15. Lin, T., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. CoRR **abs/1612.03144** (2016), http://arxiv.org/abs/1612.03144
16. Lin, Z., Shen, Y., Zhou, S., Chen, S., Zheng, N.: Mlf-det: Multi-level fusion for cross-modal 3d object detection. In: International Conference on Artificial Neural Networks. pp. 136–149. Springer (2023)
17. Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D.L., Han, S.: Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In: 2023 IEEE international conference on robotics and automation (ICRA). pp. 2774–2781. IEEE (2023)
18. Ma, Y., Peri, N., Wei, S., Hua, W., Ramanan, D., Li, Y., Kong, S.: Long-tailed 3d detection via 2d late fusion. arXiv preprint arXiv:2312.10986 (2023)
19. Mao, J., Shi, S., Wang, X., Li, H.: 3d object detection for autonomous driving: A comprehensive survey. International Journal of Computer Vision **131**(8), 1909–1963 (2023)
20. Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. CoRR **abs/1612.00496** (2016), http://arxiv.org/abs/1612.00496
21. Paigwar, A., Sierra-Gonzalez, D., Erkent, O., Laugier, C.: Frustum-pointpillars: A multi-stage approach for 3d object detection using rgb camera and lidar. In: 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). pp. 2926–2933 (2021). https://doi.org/10.1109/ICCVW54120.2021.00327
22. Pang, S., Morris, D., Radha, H.: Clocs: Camera-lidar object candidates fusion for 3d object detection. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 10386–10393. IEEE (2020)
23. Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621 (2017)
24. Peri, N., Dave, A., Ramanan, D., Kong, S.: Towards long-tailed 3d detection. In: Conference on Robot Learning. pp. 1904–1915. PMLR (2023)
25. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from RGB-D data. CoRR **abs/1711.08488** (2017), http://arxiv.org/abs/1711.08488
26. Qian, R., Lai, X., Li, X.: 3d object detection for autonomous driving: A survey. Pattern Recognition **130**, 108796 (2022)
27. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. CoRR **abs/1506.01497** (2015), http://arxiv.org/abs/1506.01497
28. Reuse, M., Simon, M., Sick, B.: About the ambiguity of data augmentation for 3d object detection in autonomous driving. In: 2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW). pp. 979–987 (2021). https://doi.org/10.1109/ICCVW54120.2021.00114

29. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: PV-RCNN: point-voxel feature set abstraction for 3d object detection. CoRR **abs/1912.13192** (2019), http://arxiv.org/abs/1912.13192
30. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. CoRR **abs/1812.04244** (2018), http://arxiv.org/abs/1812.04244
31. Shi, S., Wang, Z., Shi, J., Wang, X., Li, H.: From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. IEEE transactions on pattern analysis and machine intelligence **43**(8), 2647–2664 (2020)
32. Vora, S., Lang, A.H., Helou, B., Beijbom, O.: Pointpainting: Sequential fusion for 3d object detection. CoRR **abs/1911.10150** (2019), http://arxiv.org/abs/1911.10150
33. Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 913–922 (2021)
34. Wang, Y., Chao, W., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. CoRR **abs/1812.07179** (2018), http://arxiv.org/abs/1812.07179
35. Wang, Y., Mao, Q., Zhu, H., Zhang, Y., Ji, J., Zhang, Y.: Multi-modal 3d object detection in autonomous driving: a survey. CoRR **abs/2106.12735** (2021), https://arxiv.org/abs/2106.12735
36. Wang, Z., Jia, K.: Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. CoRR **abs/1903.01864** (2019), http://arxiv.org/abs/1903.01864
37. Wu, H., Wen, C., Li, W., Li, X., Yang, R., Wang, C.: Transformation-equivariant 3d object detection for autonomous driving. Proceedings of the AAAI Conference on Artificial Intelligence **37**(3), 2795–2802 (Jun 2023). https://doi.org/10.1609/aaai.v37i3.25380, https://ojs.aaai.org/index.php/AAAI/article/view/25380
38. Wu, H., Wen, C., Shi, S., Li, X., Wang, C.: Virtual sparse convolution for multimodal 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21653–21662 (2023)
39. Wu, X., Peng, L., Yang, H., Xie, L., Huang, C., Deng, C., Liu, H., Cai, D.: Sparse fuse dense: Towards high quality 3d detection with depth completion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5418–5427 (2022)
40. Xu, D., Anguelov, D., Jain, A.: Pointfusion: Deep sensor fusion for 3d bounding box estimation. CoRR **abs/1711.10871** (2017), http://arxiv.org/abs/1711.10871
41. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10) (2018). https://doi.org/10.3390/s18103337, https://www.mdpi.com/1424-8220/18/10/3337
42. Yang, Z., Sun, Y., Liu, S., Jia, J.: 3dssd: Point-based 3d single stage object detector. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 11037–11045 (2020), https://api.semanticscholar.org/CorpusID:211259226
43. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: STD: sparse-to-dense 3d object detector for point cloud. CoRR **abs/1907.10471** (2019), http://arxiv.org/abs/1907.10471

44. Zhang, H., Yang, D., Yurtsever, E., Redmill, K.A., Özgüner, U.: Faraway-frustum: Dealing with lidar sparsity for 3d object detection using fusion. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 2646–2652 (2021). https://doi.org/10.1109/ITSC48978.2021.9564990
45. Zhang, Y., Chen, J., Huang, D.: Cat-det: Contrastively augmented transformer for multi-modal 3d object detection (2022), https://arxiv.org/abs/2204.00325
46. Zheng, W., Tang, W., Chen, S., Jiang, L., Fu, C.: CIA-SSD: confident iou-aware single-stage object detector from point cloud. CoRR **abs/2012.03015** (2020), https://arxiv.org/abs/2012.03015
47. Zheng, W., Tang, W., Jiang, L., Fu, C.W.: Se-ssd: Self-ensembling single-stage object detector from point cloud. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 14494–14503 (June 2021)
48. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. CoRR **abs/1711.06396** (2017), http://arxiv.org/abs/1711.06396
49. Zhou, Y., Guo, C., Wang, X., Chang, Y., Wu, Y.: A survey on data augmentation in large model era. ArXiv **abs/2401.15422** (2024), https://api.semanticscholar.org/CorpusID:267311830
50. Çaldiran, B.E., Acarman, T.: A late asymmetric fusion approach to eliminate false positives. In: 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC). pp. 2080–2085 (2022). https://doi.org/10.1109/ITSC55140.2022.9922182