remote sensing

MDPI

Article

SPV-SSD: An Anchor-Free 3D Single-Stage Detector with Supervised-PointRendering and Visibility Representation

Lingmei Yin, Wei Tian *, Ling Wang, Zhiang Wang and Zhuoping Yu

School of Automotive Studies, Tongji University, Shanghai 201804, China
* Correspondence: tian_wei@tongji.edu.cn

Abstract: Recently, 3D object detection based on multi-modal sensor fusion has been increasingly adopted in automated driving and robotics. For example, the semantic information provided by cameras and the geometric information provided by light detection and ranging (LiDAR) are fused to perceive 3D objects, as single modal sensors are unable to capture enough information from the environment. Many state-of-the-art methods fuse the signals sequentially for simplicity. By sequentially, we mean using the image semantic signals as auxiliary input for LiDAR-based object detectors would make the overall performance heavily rely on the semantic signals. Moreover, the error introduced by these signals may lead to detection errors. To remedy this dilemma, we propose an approach coined supervised-PointRendering to correct the potential errors in the image semantic segmentation results by training auxiliary tasks with fused features of the laser point geometry feature, the image semantic feature and a novel laser visibility feature. The laser visibility feature is obtained through the raycasting algorithm and is adopted to constrain the spatial distribution of fore- and background objects. Furthermore, we build an efficient anchor-free Single Stage Detector (SSD) powered by an advanced global-optimal label assignment to achieve a better time–accuracy balance. The new detection framework is evaluated on the extensively used KITTI and nuScenes datasets, manifesting the highest inference speed and at the same time outperforming most of the existing single-stage detectors with respect to the average precision.

Keywords: 3D object detection; sequential fusion; spatial visibility of LiDAR; anchor-free detection

check for updates

Citation: Yin, L.; Tian, W.; Wang, L.; Wang, Z.; Yu, Z. SPV-SSD: An Anchor-Free 3D Single-Stage Detector with Supervised-PointRendering and Visibility Representation. *Remote Sens.* **2023**, *15*, 161. https://doi.org/10.3390/rs15010161

Academic Editor: Tamas Sziranyi

Received: 5 December 2022
Revised: 19 December 2022
Accepted: 21 December 2022
Published: 27 December 2022

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Accurate and real-time 3D object detection is an indispensable task in the perception system of intelligent vehicles. Current LiDAR-only object detection approaches [1] have achieved impressive performance, especially with multi-views to take advantage of LiDAR point spatial information as much as possible. These include VISTA [2] based on both bird's-eye view (BEV) and range view (RV) and MASS [3] using the dense top-view. However, these LiDAR-only methods still have some shortcomings. On the one hand, they rarely consider that the point cloud captured by LiDAR in the autonomous driving scene is not real "3D" data. Limited by the visibility constraints, the original point cloud cannot well perceive the information behind objects that LiDAR fails to detect. On the other hand, self-driving cars generally use multi-sensor combinations such as LiDAR and cameras: LiDAR provides accurate spatial information, while the camera provides rich context and semantic information; thus, data fusion can achieve the advantages of different types of complementary sensors.

The current fusion dilemma lies in the fact that most 3D detectors extract features on the bird's-eye view (BEV), which is difficult to align with the front view image captured by the monocular camera. The sequential fusion style at the input stage can avoid this problem. A typical method is PointPainting [4]. It first projects the point cloud into the image semantic segmentation results, then appends the corresponding semantic scores to each point. This method enables easily missed small objects to be detected due to semantic

information but decreases the precision of large objects which can be detected accurately by LiDAR-only methods (Table 1). A conclusion can be illustrated from the results reported on the KITTI benchmark [5] that PointPainting contributes to significant improvements on the average precision (AP) of pedestrians and cyclists, while contributing little to or even deteriorating the performance of car detection. The reason is that sequential fusion makes subsequent detection network rely heavily on the quality of semantic segmentation. As Figure 1 shows, many background points are misclassified as cars, inevitably leading to more false positives.

Table 1. Deterioration of car detection performance by sequentially appending image semantic segmentation results to point cloud. Test results of Painted PointPillars are obtained on the validation set using the official implementation, while the test results of Painted PointRCNN are from the KITTI server.

Method	Car (%)			Pedestrian (%)			Cyclist (%)		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
PointPillars [4]	87.22	76.95	73.52	57.75	52.29	47.91	82.29	63.26	59.82
Painted PointPillars [4]	86.26	76.77	70.25	61.50	56.15	50.03	79.12	64.18	60.79
Delta	−0.96	−0.18	−3.27	+3.75	+3.86	+2.12	−3.17	+0.92	+0.97
PointRCNN [6]	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
Painted PointRCNN [4]	82.11	71.70	67.08	50.32	40.97	37.87	77.63	63.78	55.89
Delta	−4.85	−3.94	−3.62	+2.34	+1.6	+1.86	+2.67	+4.96	+3.36

Current 3D object detectors often ignore the fact that measuring the 3D environment as (x,y,z) points destroys the hidden spatial distribution information [7,8]. According to the physical characteristics of raycasting, "visibility" ensures that there are no obstacles between the LiDAR origin and the detected object, and everything behind the detected object along its line-of-sight is occluded. The visibility constraint can estimate the free area distribution in the 3D space and can provide context information for object detection through modeling the bounding box coordinates as the Gaussian parameters (i.e., the mean and variance), and redesigning the loss function of the bounding box [9]. Moreover, the database sampling is one of the general augmentation strategies, which randomly copies and pastes virtual objects into point cloud scenes. However, such a method often inserts objects behind walls or buildings, ignoring their visibility and authenticity. In this paper, we adopt a visibility feature to assist both object detection and data augmentation.

Most voxel-based detectors prefer anchor-based detection heads. Anchor-based methods generate dense anchor boxes to guarantee the high recall. However, for complex scenes, such as in the nuScenes [10], anchor-based methods will generate dozens of anchors in consideration of the number of categories and orientation bins in each grid, resulting in cubic-level growth of parameters and slowing down the inference. In contrast, existing voxel-based anchor-free detectors directly regress bounding boxes in the foreground area of the heatmap, capable to run in real-time yet with limited performance. As shown in Table 2, the average precision of the car category achieved by the AFDet (anchor-free) on the KITTI validation dataset is 0.91% lower than the SECOND (anchor-based), and the gap between the CenterPoint and the SECOND is even larger.

Table 2. Performance comparison of existing voxel-based anchor-free/-based detectors on the KITTI dataset. Above are the validation results of car detection by the SECOND, the PointPillars and the AFDet approach. The AFDet only provides results for the KITTI validation dataset. Below are the car detection results of the SECOND, the PointPillars and the CenterPoint approach on the KITTI test dataset.

Method	Det. Head	3D AP (%)			BEV AP (%)		
		Easy	Mod.	Hard	Easy	Mod.	Hard
SECOND [11]	anchor-based	**87.43**	**76.48**	69.10	**89.96**	**87.07**	79.66
PointPillars [12]	anchor-based	83.73	76.04	69.12	89.68	86.34	**84.38**
AFDet [13]	anchor-free	85.68	75.57	**69.31**	89.42	85.45	80.56
SECOND	anchor-based	**84.65**	**75.96**	68.71	89.39	83.77	78.59
PointPillars	anchor-based	82.58	74.31	68.99	**90.07**	**96.56**	**82.81**
CenterPoint [14]	anchor-free	81.17	73.96	**69.48**	88.47	85.05	81.19

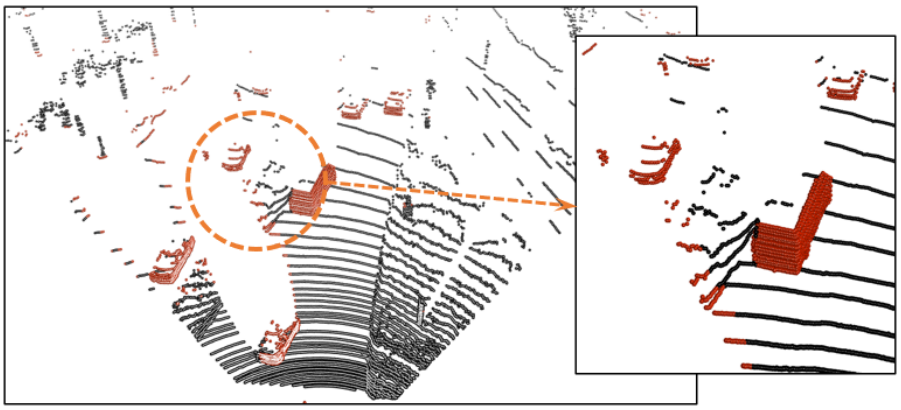
To address the above issues, this paper explores an anchor-free 3D single-stage object detector with supervised-PointRendering and visibility representation. The supervised-PointRendering decorates points by appending image semantic segmentation results to points, with the additional point-wise supervision task to rectify incorrect semantic segmentation results. We adopt a raycasting algorithm to reconstruct the spatial visibility features of laser, then fuse them with the image semantic feature and point cloud feature. We also apply the optimal transport assignment strategy [15], which was originally used in the 2D object detection to boost the confidence-based global label assignment. We adapt it to the 3D object detection like [16,17], yet with modifications, such as adding orientation to the algorithm and enlarging center prior region of small objects to make it more suitable for 3D detection framework. With such an implementation, our anchor-free model can boost the inference speed while achieving a higher precision. Compared with existing approaches, the main highlights of this paper are summarized as follows:

- We propose a novel supervised-PointRendering to eliminate the effects of incorrect image semantics on object detection, significantly improving the detection precision of all categories.
- We introduce laser visibility features into the sequential fusion-based 3D object detection, which supplements more scene context to boost the 3D object detection.
- We design an anchor-free detection head powered by the 3D optimal transport assignment (OTA-3D) in the voxel-based approaches, achieving a decent balance between precision and speed. At present, our model has achieved comparable precision with the state-of-the-art single-stage approaches on both KITTI and nuScenes datasets with an ultra-high inference speed, showing the generality and efficiency of our model in different traffic scenarios.



(a)

Figure 1. *Cont.*



(b)

Figure 1. (**a**) Semantic segmentation results by DeeplabV3+ [18]; (**b**) the point cloud is painted with the 2D segmentation results. Cars are denoted in red. The zoomed figure depicts the misclassified ground area.

2. Related Work

2.1. 3D Object Detection with Point Cloud

LiDAR-based 3D object detectors are generally divided into single-stage and two-stage detectors [19,20] depending on their architecture.

The single-stage detectors directly predict classification scores and object size and location in one stage. For instance, the VoxelNet [21] generates regularized data by voxelizing the point cloud, then feed the data into the standard 3D convolution neural network (CNN). SECOND [11] explores the sparse convolution to reduce the computation cost caused by standard 3D convolution and proposes the submanifold sparse convolution to prevent the feature diffusion. The Point-GNN [22] extracts features through a novel graph neural network and shows comparable performance with state-of-the-art two-stage detectors. The SASSD [23] employs an auxiliary network for box center regression and segmentation, aiming to learn the structure and localization information. This inspires us to exploit point-level supervision to eliminate the boundary-blurring effect in semantic segmentation.

Two-stage detectors typically refine the candidate proposals generated in the first stage. The Voxel-RCNN [24] applies a voxel-based CNN to generate regions of interest (RoIs) and refines the proposals by the devised voxel RoI pooling module. The PV-RCNN [25] takes advantage of the efficient voxel-based backbone network and the flexible PointNet-based set abstraction to extract more representative features. The Part-A^2 [26] learns intra-object part locations to enrich the RoI features, reducing the ambiguity of the bounding boxes.

It is worth noting that most voxel-based models use an anchor-based detection head. However, this mechanism has several drawbacks. First, to achieve the optimal detection performance, one needs to cluster a set of optimal anchors based on training data which requires heuristic tuning. Second, the anchor mechanism greatly increases both the number of predictions and the design complexity of detection heads. In contrast, the anchor-free mechanism reduces the parameter number of the detection head significantly, making the training and decoding phase of the detector much simpler. Some efforts have been made in voxel-based anchor-free detectors, such as the CenterPoint [14] and the AFDet [13]. However, these detectors achieve less outperforming results on the KITTI dataset than anchor-based detectors. In the 2D detection, anchor-free detectors [27–29] have developed rapidly in recent years and can exceed anchor-based detectors with suitable label assignments.

Given the high efficiency of the one-stage network and great potential of the anchor-free mechanism in 3D object detection, we focus on developing an anchor-free single-stage detection approach with improved precision, which has attained competitive performance with the top-performing single-stage detectors with a high inference speed.

2.2. Multi-Modal Fusion

Object detection methods utilizing LiDAR-camera fusion have progressed by leaps and bounds in recent years. According to [4], they can be divided into the following types: object-centric fusion, feature-level fusion, detection seeding and sequential fusion.

The representatives of object-centric fusion are the MV3D [30] and the AVOD [31]. They generate proposals from both the image and point cloud projection view, then perform deep feature fusion on the RoI features on the BEV or the front view. However, the projection from the point cloud to the image plane loses massive spatial information, which imposes an adverse effect on the detection accuracy.

The feature-level fusion, pioneered by the ContFuse [32], fuses features from the image and LiDAR backbone networks at different scales. These methods often calculate a mapping relationship to convert the point clouds to the image plane. The core problem lies in that each BEV feature vector of the point cloud can correspond to multiple pixels in the 2D image, resulting in fuzzy feature align and thus limited performance. To address this challenge, AutoAlign [33] proposes a cross-attention module to fit the mapping relationship. BEVFusion [34] accelerates the BEV pooling process during the view transformation process and achieves a decent balance between precision and speed.

The detection seeding methods such as the Frustrum PointNet [35] and the ConvNet [36] utilize 2D detection results to limit the frustum searching space to seed the 3D proposal. These detectors rely heavily on the performance of 2D detectors and thus heavily restrict the recall rate.

Sequential fusion is a simple, general yet effective strategy compared with other fusion methods. The LRPD [37] and the PointPainting [4] both use the output of an image semantic segmentation network to assist object detection. The PointPainting [4] projects LiDAR points into the semantic segmentation results of the image to decorate the points with semantic scores and then feed the painted points to the feature extraction network. However, the "boundary-blurring effect" occurs in the semantic segmentation process because the high-level feature map generally has quite low resolution. This error becomes more apparent when re-projecting them into the 3D space.

2.3. Visibility Representation

As far as we know, research focusing on spatial visibility representation is mostly carried out in robotic mapping. Buhmann et al. [38] used a 2D probabilistic occupancy map based on sonar sensor data to navigate mobile robots. Hornung et al. [39] propose a general 3D occupancy map to describe the space state, indicating the occupied, free and unknown area. The visibility representation through a raycasting algorithm is the core of constructing such occupancy maps.

Although visibility has gained popularity in the robotics area, the visibility reasoning has not received sufficient attention in 3D object detection. Richter et al. [40] integrated the occupancy grid map into the probabilistic framework to detect objects with known surface. Notably, Hu et al. [41] proposed the concept that LiDAR point clouds are not real "3D" but "2.5D". They reconstructed the spatial visibility state through a raycasting algorithm and converted 3D spatial features to 2D multi-channel feature maps, which were then integrated into the PointPillars [12]. Such visibility features can be directly concatenated with a voxelized point cloud and thus bring better data alignment.

3. Proposed Method

In this part, we devise the anchor-free 3D single-stage object detector with supervised-PointRendering and LiDAR visibility. Section 3.1 presents our detection network architecture. Section 3.2 describes the supervised-PointRendering to enhance the point cloud feature. Section 3.3 describes the extraction of the spatial visibility feature of laser to provide scene representation. Section 3.4 introduces the anchor-free detection head with an improved label assignment. Section 3.5 presents the losses used in training.

3.1. Network Architecture

Our framework is composed of data processing, a backbone network, and a detection head.

In the data processing, we adopt an off-the-shelf image segmentor to output the per pixel semantic class scores and then append the scores to the points. We reconstruct the spatial visibility feature though a raycasting algorithm and concatenate them with the voxelized point clouds. The painted point clouds with visibility features are fed into the backbone network.

Our backbone network contains two modules: a typical 3D convolution backbone network followed by a spatial semantic feature aggregation (SSFA) module [42] for feature extraction and an auxiliary network which aims to exploit point-wise supervisions. As manifested in Figure 2, the backbone network is made up of four convolution blocks, each of which contains submanifold convolutions with a kernel size of three. In the last three blocks, the features are downsampled with a stride of two. Then, we concatenate features from the backbone along the height dimension to generate the BEV feature maps. High-level semantic features and low-level spatial features are fused adaptively in the semantic aggregation. The auxiliary network scatters the convolution features back to points and performs foreground segmentation and center regression tasks.

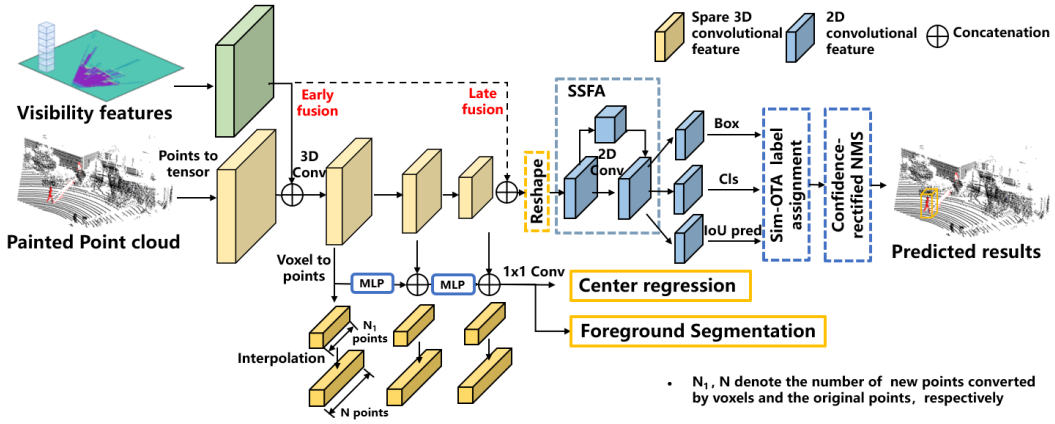


Figure 2. The pipeline of our proposed anchor-free 3D single-stage object detection network with supervised-PointRendering and spatial visibility representation.

The anchor-free detection head has three tasks, including classification, localization and intersection-over-union (IoU) prediction with a confidence function [43] to rectify the confidence used in the non-maximum-suppression (NMS) post-processing. In the end, we employ a global optimal label assignment to find the best label assigning at minimal global costs.

3.2. Supervised-PointRendering

3.2.1. PointRendering

With an input image, the image semantic segmentation network outputs semantic class scores per pixel. The high-level features of an image are represented by these scores. We transform LiDAR points into the camera coordinate system, project them into the image, and append the segmentation scores of corresponding pixels to the LiDAR point features. We choose segmentation scores as image feature encoding because scores implicitly contain uncertainty information, which requires the network to distinguish the semantic category itself, thus reducing the influence of incorrect semantic segmentation on subsequent networks.

3.2.2. Point-Wise Supervision Task

As Figure 2 shows, we utilize the point-wise foreground segmentation to rectify the incorrect segmentation results. Through the inverse process of voxelization, we convert the voxel feature to the world coordinate system. The feature from N_1 points are propagated to N points by interpolation, where N_1 indicates the number of new points converted by voxels, while N denotes the number of the original points. In the interpolation approach, the feature vector $f(\boldsymbol{x}_j)$ at each coordinate of original points $\boldsymbol{x}_j$ can be calculated by a weighted average on features of its k nearest neighbors $\boldsymbol{x}'_i$ among the new points (k is empirically set to three), interpreted by

$$f(\boldsymbol{x}_j) = \frac{\sum_{i=1}^{k} w_{ij}(\boldsymbol{x}'_i) f(\boldsymbol{x}'_i)}{\sum_{i=1}^{k} w(\boldsymbol{x}'_i)}, j = 1, \ldots, C, \tag{1}$$

where the weight $w_{ij}(\boldsymbol{x}'_i)$ is the inverse square of distance between the original point $\boldsymbol{x}_j$ and the new point $\boldsymbol{x}'_i$:

$$w_{ij}(\boldsymbol{x}'_i) = \frac{1}{\|\boldsymbol{x}_j - \boldsymbol{x}'_i\|^2}. \tag{2}$$

Instead of directly concatenating the obtained features with the next-stage features [23], our method applies a shared multi-layer perception (MLP) to process the features to yield point feature encoding $h(x_i)$, which is then concatenated with next-stage features, enriching the encoding for the supervision task. We apply 1×1 convolutions to generate predictions for the foreground segmentation task. Moreover, we add a center regression task to predict offsets from object points to the corresponding instance center, aiming to guide the backbone network to learn object structure information. These tasks are detachable in the inference stage, introducing no extra computational cost.

The point-wise supervision loss includes the foreground segmentation loss and the center regression loss, which are, respectively, interpreted by the focal loss and the smooth-L1 loss. The foreground segmentation label is a binary value with one to indicate that the point is within the bounding box of ground truth. The center regression predicts the offsets of foreground points to the corresponding object centers.

3.3. Visibility Representation

3.3.1. Ray Casting Algorithm

The raycasting algorithm in a horizontal plane is depicted in Figure 3. Given the center coordinates of the current voxel (x, y), (t_{mx}, t_{my}) are the time intervals from the current position to the boundary of the adjacent voxel along the ray in the x and y direction, respectively. By comparing the time intervals in two directions, one can know whether the ray reaches the horizontal face or the vertical face first. We traverse along the ray to reach the next voxel, update (x, y) and (t_{mx}, t_{my}) and mark the traversed grid. The algorithm can be easily extended to 3D space by adding corresponding variables along the z axis. Specifically, the algorithm calculates the arrival time at six faces of the current voxel to check their intersection with the exiting ray. It then proceeds to the adjacent voxel with shared face. Each ray starts at the voxel at the sensor origin and iterates the above calculation until it reaches the voxel (precomputed) occupied by the LiDAR points. Such an algorithm is very efficient because of the linear calculation.

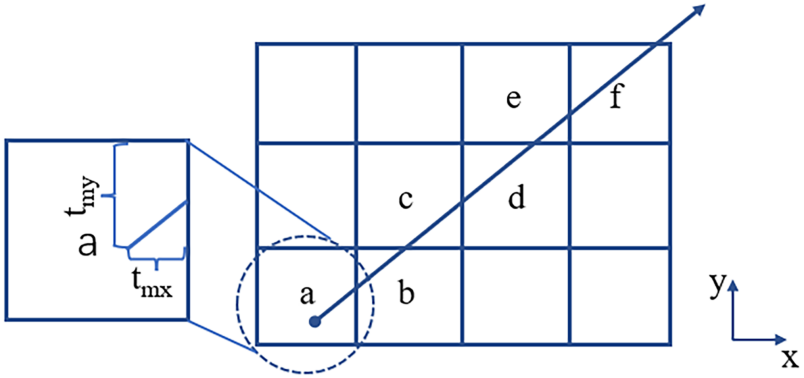
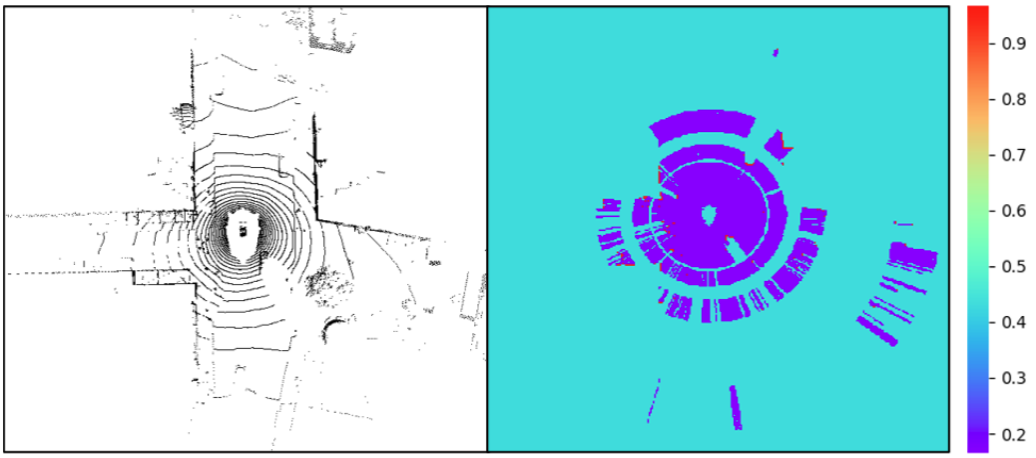


Figure 3. Ray traversal on the 2D grids. The letters a–f represent the orders of grids which the ray reaches.

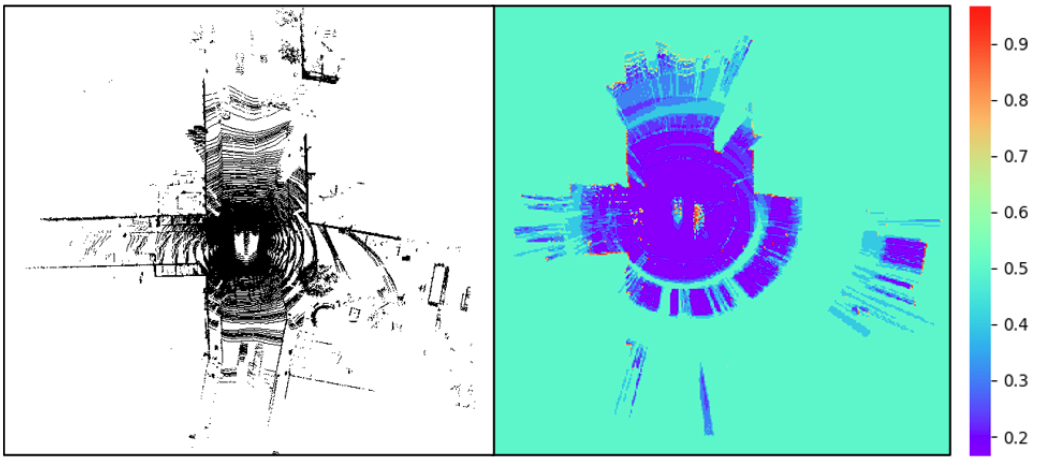
3.3.2. Spatial Visibility States

A grid can be assigned with one of the three visibility states: unknown, occupied and free space, which are represented by specific numerical values. Here, we first initialize all voxels as unknown, then execute the raycasting algorithm for each laser ray. The raycasting algorithm draws a line from the LiDAR sensor origin and the object 3D point. The voxels which the line passes are considered as free-space, and the last voxel that encloses the LiDAR point is regarded as occupied. If one grid cell is traversed by multiple rays with one indicating it is occupied and others indicating they are free, they are regarded as occupied under such circumstances. Details about this procedure can be referred to Algorithm 1. Figure 4 depicts the visibility representation of laser rays. It can be seen that the spatial visibility information is consistent with the original point cloud, providing strong support for the quantitative results obtained in the experiment.



(a)

Figure 4. *Cont.*



(b)

Figure 4. (**a**) A single sweep of LiDAR and the corresponding single-layer instantaneous visibility. The denotation of three visibility states are: occupied-space (red), unknown-space (light blue) and free-space (dark blue); (**b**) aggregated LiDAR sweeps and their single-layer superimposed temporal occupancy states. Color with increased saturation indicates a greater probability of occupancy state of the corresponding voxel.

Algorithm 1 Vanilla voxel traversal algorithm

Require: lidar origin $\boldsymbol{o}$, raw point cloud $\mathcal{P}_{raw}$, voxelized painted points $\mathcal{P}_{vox}$, ending voxel v_e.
Ensure: occupancy grids $\boldsymbol{O}$, voxelized points with visibility states $\mathcal{P}_{vis}$

$\boldsymbol{O}$[:]←UNKNOWN;
for $\boldsymbol{p}$ in $\mathcal{P}_{raw}$ **do**
  v ← $initial_voxel(\boldsymbol{p})$;
  while v ≠ v_e **do**
    v ← $next_voxel(\mathrm{v}, \boldsymbol{p} - \boldsymbol{o})$;
    if v = v_e **then**
      $\boldsymbol{O}$[v] ← OCCUPIED;
      break;
    else
      $\boldsymbol{O}$[v] ← FREE;
    end if
  end while
  $\mathcal{P}_{vis}$ ← $concat(\mathcal{P}_{vox}, \boldsymbol{O})$;
end for

3.4. Anchor-Free Detection Head with OTA-3D

3.4.1. OTA-3D

We adapt the optimal transport assignment (OTA) strategy [15] from 2D object detection to 3D detection. Different from 2D methods, we adjust the candidate area for objects of different size flexibly and introduce rotated-IoU loss to the global cost. We first regard the predictions from the grids which are within a fixed region, e.g., 0.8 m × 0.8 m around the center of car and cyclist, and 1.6 m × 1.6 m around the pedestrian, as positive candidates. Then, we finely select the global-optimal positive samples from these positive candidates. The steps are depicted in Figure 5 and introduced as follows.

- Calculating the cost matrix: Its element indicates the pair-wise matching cost between a positive candidate box and a ground truth. Different from the 2D OTA, the pair-wise cost c_{ij} is composed of the classification loss L_{ij}^{cls}, the bounding box regression loss

L_{ij}^{r} and the BEV rotated-IoU loss L_{ij}^{IoU} between a positive candidate i and the ground truth box j, which is calculated as:

$$c_{ij} = L_{ij}^{cls} + L_{ij}^{r} + \lambda L_{ij}^{IoU}. \tag{3}$$

- Calculating the dynamic k: For each ground truth, its IoU (intersection over union) score is calculated with all positive candidates. We select the top N box predictions and sum their IoU scores as k, which is further rounded by the floor operation. Here, we set $N = 10$; k is thus the number of positive samples assigned to one ground truth bounding box.
- Selecting the top k predictions with the least cost.
- Filtering the repeated predictions: In the case that the same prediction matches multiple ground truth bounding boxes, the prediction is assigned to the ground truth bounding box with the least cost.

Finally, the corresponding grids of those positive predictions are assigned as positives, while the rest of the grids are considered as negatives.

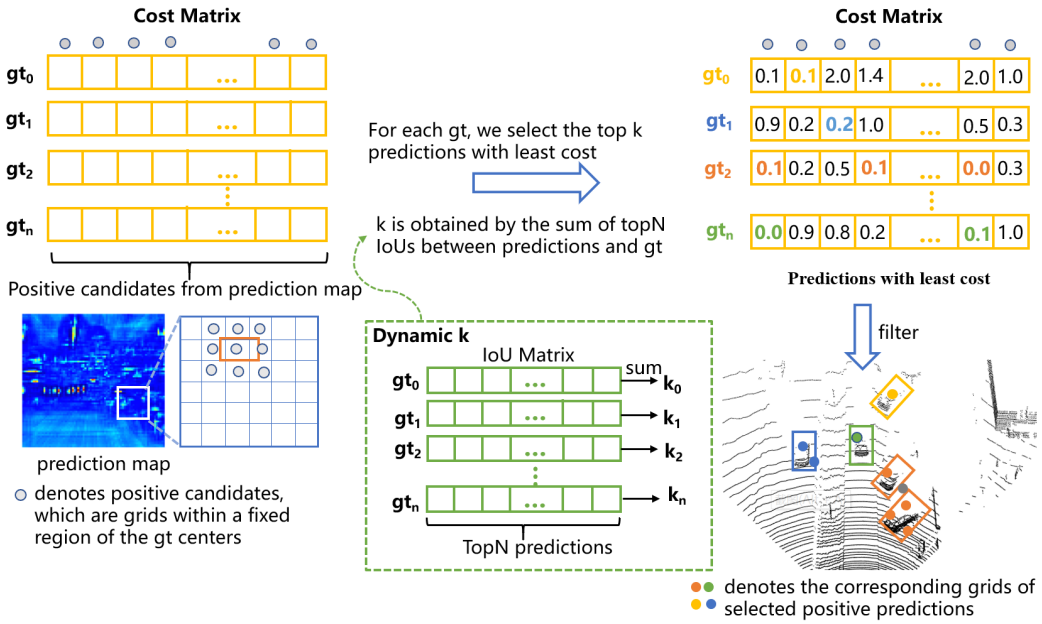


Figure 5. Optimal transport assignment in 3D object detection.

3.4.2. Anchor-Free Detection Head

To achieve real-time performance without decreasing precision, we build an anchor-free detection head with OTA-3D.

In this detection head, we reduce the predictions for each location from $N_{class} \times N_{ori}$ groups to one group. Each location directly predicts its class and offsets to the instance center, as well as the normalized distance from the mean size of the predicted class. For regression targets, we apply the following box encodings:

$$\begin{aligned} x_t = \frac{x_g - x_a}{d_a}, y_t = \frac{y_g - y_a}{d_a}, z_t = \frac{z_g - z_a}{h_a}, \\ w_t = \log\left(\frac{w_g}{w_a}\right), l_t = \log\left(\frac{l_g}{l_a}\right), h_t = \log\left(\frac{h_g}{h_a}\right), \end{aligned} \tag{4}$$

in which x, y and z indicate the center coordinates; w, l and h denote the width, length and height, respectively; the subscripts t, a and g represent the encoded value, the mean size of each class and the corresponding ground truth, respectively; and $d_a = \sqrt{(l_a)^2 + (w_a)^2}$ denotes the diagonal of the mean size box.

Given that the orientation regression is difficult without prior information, we apply a hybrid formulation of classification and regression to predict the orientation. Specifically, we split 2π into N_a bins. The network predicts both the angle bin and the residuals relative to the bin. In the experiment, we set $N_a = 12$.

In order to mitigate the misalignment between classification confidence and localization accuracy, we add a branch to predict IoU score [42] to rectify the confidence in the non-maximum-suppression (NMS) process. The confidence is estimated as $C = c \cdot i^{\beta}$, where C is the confidence in the NMS process, i denotes the predicted IoU, c denotes the classification score, and β is a hyper-parameter that controls the distinction between accurate and inaccurate predictions. It is set to four.

3.5. Loss Function

The overall loss L includes the detection head loss L_{head} and the point-wise supervision loss for foreground segmentation L_{seg} and center regression L_{ctr}:

$$L = L_{head} + \omega L_{seg} + \mu L_{ctr}. \tag{5}$$

Here we set ω = 0.9, μ = 2.0 to balance the point-wise supervision task from the main task. The detection head loss includes the classification loss L_c, the bounding box regression loss L_r and the 3D D-IoU loss L_{IoU} [44]. Among them, the 3D D-IoU is calculated by:

$$DIoU = IoU - \frac{\rho^2(b, b_{gt})}{c^2} \tag{6}$$

where b, b_{gt} represent the centers of the predicted bounding box and the ground truth, respectively, ρ represents their Euclidean distance, and c represents the diagonal distance of the smallest closed region that contains both the predicted bounding box and its ground truth. The IoU prediction loss L_{head} is calculated as follows:

$$L_{head} = \frac{1}{N_c}\sum_i L_c(s_i, u_i) + \lambda_1 \frac{1}{N_p}\sum_i [u_i > 0] L_r + \lambda_2 \frac{1}{N_p} L_{IoU} + \lambda_3 \frac{1}{N_p} L_{conf}, \tag{7}$$

where λ_i is the weight of each task; N_c and N_p, respectively, denote the number of total proposals and positive proposals selected by the OTA-3D; s_i is the classification score for each grid, while u_i denotes the corresponding class label. Here, we set $\lambda_1 = 1.0$, $\lambda_2 = 5.0$ and $\lambda_3 = 1.0$. The 3D D-IoU loss is based on [45], and L_{conf} is calculated by the smooth-L1 loss. For classification, the focal loss is chosen, which is:

$$\mathcal{L}_{cls} = -\alpha(1 - p^a)^{\gamma} \log p^a, \tag{8}$$

where p^a denotes the estimated probability, and α and γ are hyper-parameters set to 0.25 and 2, following the work [46].

The center regression loss L_{dist}, the size regression loss L_{size} and the angle regression loss L_{angle} make up the regression loss L_r . The L_{dist} regularizes offsets from positive grid location to the corresponding instance centers. The targets for L_{size} are the offsets between the real object size and the average size of its category. We adopt the smooth-L1 loss for both L_{dist} and L_{size}. Considering that the angle localization loss cannot distinguish flipped boxes, our angle loss contains the orientation classification loss L_{ori_cls} and the corresponding residual prediction loss L_{ori_reg}, which can be interpreted as:

$$L_{angle} = L_{ori_cls}(d_c^a, t_c^a) + L_{ori_reg}(d_r^a, t_r^a), \tag{9}$$

where d_c^a is the predicted orientation, d_r^a is its residual while t_c^a, and t_r^a are their ground truths.

4. Experiments and Discussion

4.1. Datasets

Our method is evaluated on both the KITTI [5] and nuScenes [10] datasets.

The KITTI detection dataset includes 7481 images for training and 7518 images for testing. Furthermorfe, the training data are split into two subsets, a training set with 3712 frames and a validation set with 3769 frames. We evaluate our method on all three classes, namely car, pedestrian and cyclist, and our evaluation criterion is chosen as the average precision (AP) with 40 recall positions. All of these positions follow the IoU threshold of 0.7 for Car and 0.5 for Pedestrian and Cyclist. In addition, according to the object size, occlusion state and truncation level, the dataset is divided into three difficulty levels: easy, moderate and hard.

The nuScenes dataset [10] contains 1000 scenes, each consisting of a video sequence. For each video, the 360° views are annotated only for the key frames (every 0.5 s). The dataset is officially divided into subsets for training, validation and testing. The training dataset includes 700 scenes (28,130 frames), while the subset for validation and testing contains 150 scenes (6019 frames) and 150 scenes (6008 frames), correspondingly. The annotations comprised ten classes. The corresponding RGB images that cover the 360° field-of-view are also provided for each key frame. We choose the mean Average Precision (mAP) and the nuScenes detection score (NDS) as the main metrics; mAP calculates the center offsets on the BEV plane as the evaluation standard instead of the intersection ratio (IoU). NDS is a weighted sum of mAP and the mean average errors on size (mASE), translation (mATE), orientation (mAOE), attribute (mAAE) and velocity (mAVE).

4.2. Setup of Supervised-PointRendering

4.2.1. Setup on KITTI Dataset

As for the semantic segmentation network, we pre-trained the DeeplabV3+ [18] on the Cityscapes [47] dataset. Note that in the Cityscapes semantic segmentation benchmark, the "rider" (considered as "pedestrian") and the "bicycle" are two different categories, while in the KITTI object detection they are grouped into one category "cyclist". We solve this problem by mapping a "bicycle" and a "pedestrian", which is within a distance of 1m to the bike, into a "cyclist", and classifying the "bicycle" without people around it as the "background". After PointRendering, the input dimension of the point cloud has changed from four to eight, namely $(x, y, z, r, s_{bg}, s_{car}, s_{pede}, s_{cyc})$, where r is the intensity, and s represents the segmentation score of each class.

4.2.2. Setup on nuScenes Dataset

We choose the HTCNet [48] as the image semantic segmentation network due to its outstanding performance and pretrain it on the nuImage [49] dataset. Remarkably, on the nuScenes dataset, the LiDAR and camera work at different frequencies. Therefore, we adopt the transform procedure from [10]. During the image projection, there are laser points projecting on two images simultaneously in the case that the field of view of two cameras are overlapped. We adopt the average segmentation score of the overlapped images to decorate the points. For sweeps that do not have synchronous images, we assign them with images which are adjacent to the capture time of these sweeps.

4.3. Setup of Visibility Representation

4.3.1. Visibility Consistency in Data Augmentation

To keep the consistency of the spatial visibility state after the database sampling of data augmentation, we adopt the"drilling" strategy [41], which allows the rays between the scene origin and added targets to traverse "occupied" voxels in the original scene. As illustrated in Figure 6, for the authenticity of the scene, part of the wall is "drilled" by removing the orange points of the wall.

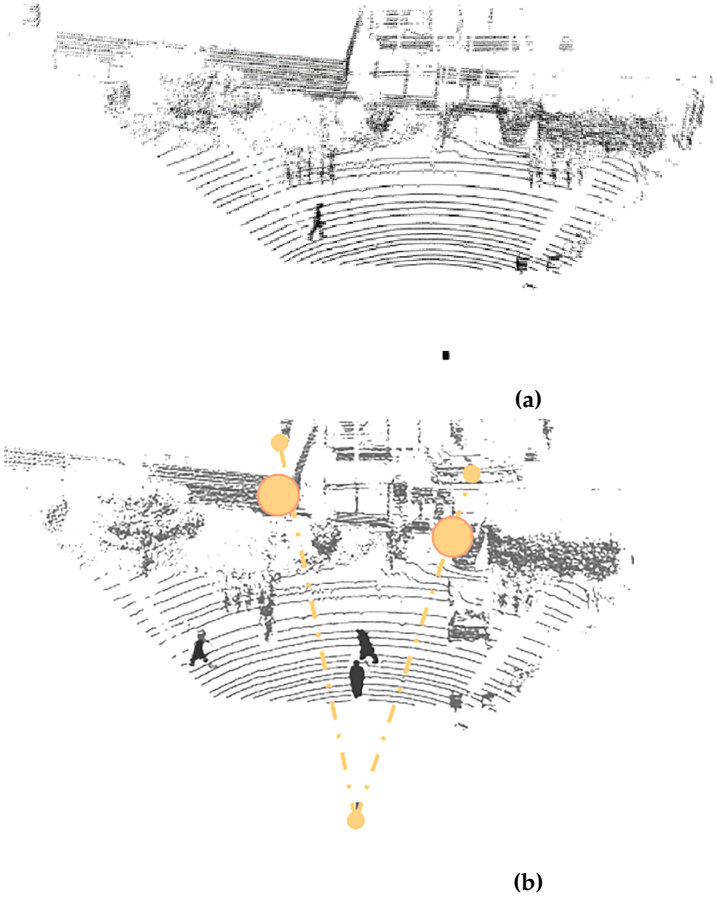


Figure 6. The "drilling" strategy in data augmentation through visibility reasoning: (**a**) the raw point cloud; (**b**) the point cloud after ground truth database sampling. The orange part will be removed after "drilling".

4.3.2. Spatial Visibility Feature Extraction

Numerical values of the three occupancy states are empirically defined as: unknown (0.5), occupied (0.7) and free space (0.4). Given one point cloud frame as input, the LiDAR origin is set to (0, 0, 0). If the input consists of aggregated LiDAR sweeps, the LiDAR origin is changed according to the position of the current frame relative to the previous frame, and Bayesian filtering is applied to accumulate these temporal visibility states as a 3D probability occupancy map.

The visibility computation is implemented in C++ and integrated into PyTorch training. We use the multi-threading and SIMD instructions to speed up the parallel execution, making the computation much faster. On two Intel(R) Xeon(R) Platinum 8163 CPU, it takes 18.2 ± 2.1 ms on average to compute the visibility features.

4.4. Choice of Key Parameter

We give an explanation about the choice of two typical parameters.

One is about the choice of visibility encoding. The reconstructed space visibility is represented by three different states, namely, unknown, occupied and free. For these three states, we use three numerical values to encode them. In this paper, we consider the following two groups of parameter settings to represent these states.

- Setting 1: The unknown state is set to 0, the occupied state is set to 1 and the free state is set to -1.
- Setting 2: According to the formula provided by Octomap [39], the unknown state is set to 0.5, the occupied state is set to 0.7 and the free state is set to 0.4.

We tested these two settings on the KITTI validation set, and the results of pedestrian detection are shown in Table 3. Notably, here we only demonstrate the experiment results on pedestrians because the visibility feature improves the precision of pedestrians more obviously. According to the final performance, [0.7, 0.5, 0.4] is finally selected as the visibility encoding parameters in our model.

Another is about the choice of the value of β in the confidence function. We follow the same setting in the CIA-SSD [42] which obtains the best result with $\beta = 4$.

For the KITTI dataset, we, respectively, set [0, 70.4] m, [−40, 40] m and [−3, 1] m as the detection range on the x, y and z axes. The size of the input voxel is set as $0.05 \times 0.05 \times 0.1$ m^3, and we use the ADAM optimizer to train the network with an initial learning rate of 0.003. The network is trained with a batch size of 56 on 8 RTX 2080 Ti GPUs, and the learning rate is decayed by 10 at 35 epochs and 40 epochs.

For the nuScenes dataset, the detection range on the x, y and z axes is set to [−54, 54] m, [−54, 54] m, [−5, 3] m, respectively. We set the input voxel size as $0.75 \times 0.75 \times 0.2$ m^3 and train the entire network with a batch size 12 for 20 epochs on 4 NVIDIA GeForce RTX 3090. We follow the same training schedule as [14], and two test-time augmentations are adopted, including double flip testing and yaw rotations.

Here, we decouple the semantic segmentation network and the spatial visibility calculation from the detection framework for a better storage management and inference speed measurement.

Table 3. Performance comparison of visibility state encoding for pedestrian detection on the KITTI validation dataset. Here, the baseline is tested with the integration of visibility features.

Visibility Encoding [U, O, F]	Pedestrian AP (BEV) (%)			mAP (BEV) (%)	Pedestrian AP (3D) (%)			mAP (3D) (%)
	Easy	Mod.	Hard		Easy	Mod.	Hard	
[1, 0, −1]	62.86	58.08	52.88	57.94	58.76	55.90	47.74	54.13
[0.7, 0.5, 0.4]	64.57	60.15	54.50	59.74	60.84	57.98	49.77	56.20
Delta	+1.71	+2.07	+1.62	+1.80	+2.08	+2.08	+2.03	+2.07

4.5. Comparison with State of the Art

4.5.1. Results on the KITTI dataset

As revealed in Table 4, our model outperforms other advanced strategies on the difficulty levels of "moderate" and "hard" and on the classes of car and cyclist by a large margin. Specifically in the car category, our method outperforms PointPillars, SECOND and VoxelNet, respectively, by 6.03%, 4.38% and 15.23% on the moderate AP. What is worth mentioning is that our model surpasses the existing anchor-free model, the CenterPoint, by 6.38% on the moderate AP. Compared to the very recent methods, i.e., the SA-SSD [23] and the CIA-SSD [42], our model achieves a gain of about 1.24% and 2.53% on the hard AP, showing the effectiveness of our strategy.

The results demonstrate that our model deals fairly well with difficult objects, which have more occlusions or sparser points. Moreover, the accurate image semantic segmentation results provided by the supervised-PointRendering effectively eliminate the false negative predictions caused by sparse points and occlusions. In addition, the inference speed of our model surpasses all state-of-the-art voxel-based single stage detectors (with preprocessed image semantic segmentation). The FPS results in Table 4 are from the official KITTI leader board. Compared with other one-stage models, the high efficiency of our model is mainly attributed to the anchor-free detection head, which reduces a large number of parameters and computation cost.

However, our mAP on pedestrians underperforms the state-of-the-art methods on the test set. One of the reasons is that the number of pedestrians in KITTI is much less than that of cars. Even with database sampling in data augmentation, the variety of pedestrians is still lacking. Additionally, the voxel resolution of a pedestrian on the BEV is much worse than that of a cyclist or a car (see Figure 7). Due to the above issues, it is difficult to learn a

pedestrian detector well with voxelized features on the KITTI dataset. A similar problem also appears in the VoxelNet [24].

Table 5 shows the comparison between our method and other advanced single-stage methods. It can be seen that the direct PointPainting decreases the precision of car detection, while our supervised-PointRendering successfully solves this problem, with gains of (1.73%, 6.49% and 10.39%) on three difficulty levels in the car category compared with the SECOND baseline. Remarkably, our method also surpasses the SA-SSD and CIA-SSD by 3.06% and 3.16% on the "moderate" difficulty level of car.

Table 4. The 3D detection AP of compared methods on the KITTI test data. The modalities include lidar (L) and images (I). The CLOCs_SecCas is short for the approach of SECOND+Cascaded R-CNN.

Method	Modality	Car (%)			Pedestrian (%)			Cyclist (%)			FPS
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
Voxelnet [21]	L	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37	4.4
SECOND [11]	L	84.65	75.96	68.71	45.31	35.52	33.14	75.83	60.82	53.67	20
PointPillars [12]	L	82.58	74.31	68.99	**51.45**	41.92	38.89	77.10	58.65	51.92	42
PointRCNN [6]	L	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53	-
CenterPoint [14]	L	81.17	73.96	69.48	47.25	39.28	36.78	73.04	56.67	50.60	-
CLOCs_SecCas [50]	L	86.38	78.45	72.45	-	-	-	-	-	-	-
SA-SSD [23]	L	88.75	79.79	74.16	-	-	-	-	-	-	25
CIA-SSD [42]	L	**89.59**	80.28	72.87	-	-	-	-	-	-	32
MV3D [30]	L & I	74.97	63.63	54.00	-	-	-	-	-	-	2.8
AVOD-FPN [31]	L & I	83.07	71.76	65.73	50.46	**42.27**	**39.04**	63.76	50.55	44.93	10
F-PointNet [35]	L & I	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01	5.9
Ours	L & I	87.22	**80.34**	**75.40**	45.83	38.45	36.03	**78.36**	**64.40**	**56.92**	**33**

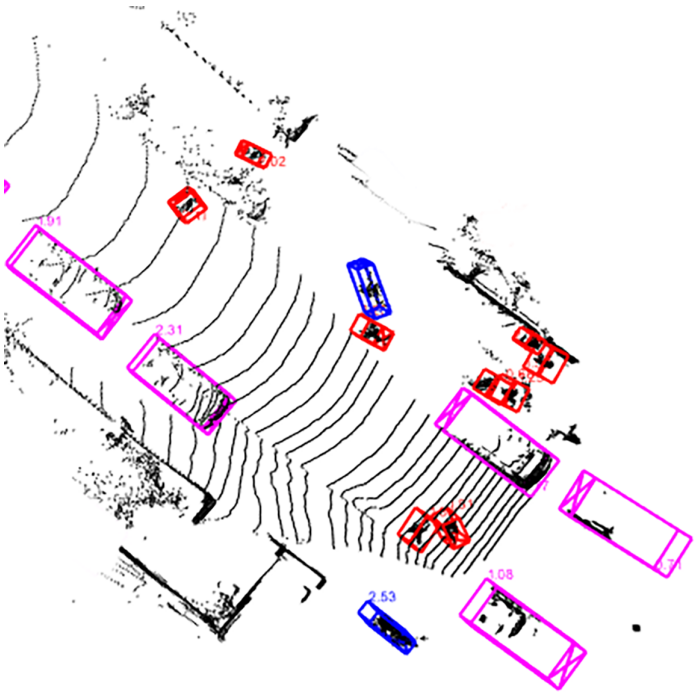


Figure 7. Qualitative results on the KITTI dataset. The pink, blue and red boxes represent cars, cyclists and pedestrians, respectively.

4.5.2. Results on nuScenes Dataset

Given the fact that the nuScenes dataset provides more classes in different scales and its point clouds are sparser, a modified version of our method is presented. In this version,

we predict the centers of the objects by adopting a class-specific center heatmap head, and we utilize regression heads to estimate other location information such as size, rotation and velocity following [14]. The results (Table 6) on the test set show that our network achieves remarkable performance, validating the generality of our model on the different driving platforms. Our model achieves an improvement of 4.1% and 7.3% on the NDS and mAP metrics over the CenterPoint, showing the effectiveness of our fusion strategy and extra visibilty feature. Specifically, it improves the AP on traffic cone (Tr. Cone), Trailer and Motorcycle (Motor.) over 10%. Our results on the validation set (Table 7) show that our model surpasses the FUTR3D [51] by 0.5% and 0.7% on Car and Bus. Notably, our model demonstrates strong competitiveness in the detection of difficult object classes such as construction vehicle (Cons. Veh.) and pedestrian.

Table 5. The 3D detection AP of compared single-stage methods on the KITTI validation set. The AP is calculated with 11 recall points.

Method	Car (%)			Pedestrian (%)			Cyclist (%)		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
PointPillars [12]	87.22	76.95	73.52	57.75	52.29	47.91	82.29	63.26	59.82
Painted PointPillars [4]	86.26	76.77	70.25	**61.50**	**56.15**	50.03	79.12	64.18	60.79
Delta	−0.96	−0.18	−3.27	+3.75	+3.86	+2.12	−3.17	+0.92	+0.97
VoxelNet [24]	81.97	65.46	62.85	57.86	53.42	48.87	67.17	47.65	45.11
SECOND [11]	87.43	76.48	69.10	-	-	-	-	-	-
SA-SSD [23]	**90.15**	79.91	78.78	-	-	-	-	-	-
CIA-SSD [42]	90.04	79.81	78.80	-	-	-	-	-	-
AFDet [13]	85.68	75.57	69.31	-	-	-	-	-	-
Ours	89.16	**82.97**	**79.49**	60.75	55.67	**50.20**	**83.67**	**69.59**	**64.21**

Table 6. Comparison with state-of-the-art methods on the nuScenes test set. The table is mainly sorted by nuScenes detection score (NDS).

Methods	Stages	NDS	mAP	Car	Truck	Bus	Trailer	Cons. Veh.	Ped.	Motor.	Bicycle	Tr.Cone	Barrier
WYSIWYG [41]	One	41.9	35.0	79.1	30.4	46.6	40.1	7.1	65.0	18.2	0.1	28.8	34.7
PointPillars [12]	One	45.3	30.5	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
3DVID [52]	One	53.1	45.4	79.7	33.6	47.1	43.1	18.1	76.5	40.7	7.9	58.8	48.8
3DSSD [53]	One	56.4	42.6	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
Cylinder3D [54]	One	61.6	50.6	-	-	-	-	-	-	-	-	-	-
CenterPoint [14]	Two	65.5	58.0	84.6	51.0	60.2	53.2	17.5	83.4	53.7	28.7	76.7	70.9
CGBS [55]	One	63.3	52.8	81.1	48.5	54.9	42.9	10.5	80.1	51.5	22.3	70.9	65.7
CVCNet [56]	One	64.2	55.8	82.6	49.5	59.4	51.1	16.2	83.0	61.8	38.8	69.7	69.7
Ours	One	**69.6**	**65.3**	**86.6**	**56.5**	**65.8**	**60.4**	**29.7**	**86.5**	**68.3**	**42.9**	**80.6**	**75.5**

Table 7. Comparison with state-of-the-art methods on the nuScenes validation dataset. The results of PointPillars, SECOND and FUTR3D are reproduced with their official implementation.

Methods	Stages	NDS	mAP	Car	Truck	Bus	Trailer	Cons. Veh.	Ped.	Motor.	Bicycle	Tr. Cone	Barrier
PointPillars [12]	One	45.3	29.6	70.5	25.0	34.5	20.0	4.5	59.9	16.8	1.7	29.6	33.2
SECOND [11]	One	48.4	27.1	75.5	21.9	29.0	13.0	0.4	59.9	16.9	0.0	22.5	32.2
MEGVII [55]	One	62.5	50.7	81.6	51.7	67.2	37.5	14.8	77.7	42.6	17.4	57.4	59.2
FUTR3D [51]	One	68.0	64.2	86.3	**61.5**	71.9	42.1	26.0	82.6	**73.6**	**63.3**	70.1	64.4
Ours	One	**69.2**	**64.7**	**86.8**	61.0	**72.6**	**42.1**	**27.1**	**86.5**	71.8	56.2	**73.3**	**69.7**

4.6. Ablation Study

In this part, we first study the effectiveness of the anchor-free detection head with OTA-3D. Then, we analyse the effect of supervised-PointRendering and the spatial visibility feature comprehensively. We conduct the experiments on the KITTI validation set.

4.6.1. Study on Anchor-Free Detection Head with OTA-3D

We adopt the backbone from the SECOND network [11] combined with the 3D D-IoU loss and an IoU prediction branch. We re-implement the 3D IoU loss based on [45] and update it to the 3D D-IoU loss. Then, we add an IoU prediction branch to further improve the AP of each category. The results show that these strategies benefit large objects such as cars.

Table 8 shows that our anchor-free head with OTA-3D boosts the moderate AP of car by 3.29%. The combination of anchor-free head and suitable label assignment strategy not only reduces the parameter amount to $\frac{1}{N_{class} \times N_{ori}}$, but also yields better performance compared to anchor-based approaches such as SA-SSD [23] and CIA-SSD [42]. Our baseline for the subsequent ablation experiments is the model with the above strategies.

Table 8. Comparison of different strategies for the detection head. OTA-3D Head denotes the anchor-free head with the OTA-3D strategy. The AP is calculated with 11 recall points.

OTA-3D Head	3D D-IoU	IoU Pred.	Car (%)	Pedestrian (%)	Cyclist (%)
			Mod.	Mod.	Mod.
			76.48	51.14	66.74
		✓	77.37	52.68	67.23
	✓	✓	78.56	**54.55**	67.65
✓	✓	✓	**81.85**	53.70	**68.13**

4.6.2. Study on Supervised-PointRendering

This experiment compares different semantic encodings, such as the object category ID, segmentation score, one-hot encoding and VoxelRendering. The VoxelRendering is to voxelize the segmentation scores, extract semantic features through 3D sparse convolution and fuse them with point cloud features before downsampling. Among them, the segmentation score encoding attains the largest gain in AP, which is 0.9% higher than that with numerical number encoding in the car category (Table 9). We argue that the segmentation score implies the classification confidence information, guiding the model to distinguish the category itself.

It can be seen in Table 9 that the AP on car drops after pure PointRendering. After utilizing point-wise supervision, our model boosts the moderate AP by 1.86%, indicating that the foreground segmentation task corrects the segmentation errors successfully. For pedestrian and cyclist, supervised-PointRendering even improves their performance by 3.94% and 0.04%, showing the importance of accurate image semantics in small object detection.

In Figure 8, qualitative examples are also presented to illustrate the benefit of our supervised-PointRendering. It can be seen that the SECOND with PointRendering alone misses more cars, while our method avoids this phenomenon and achieves the best detection performance.

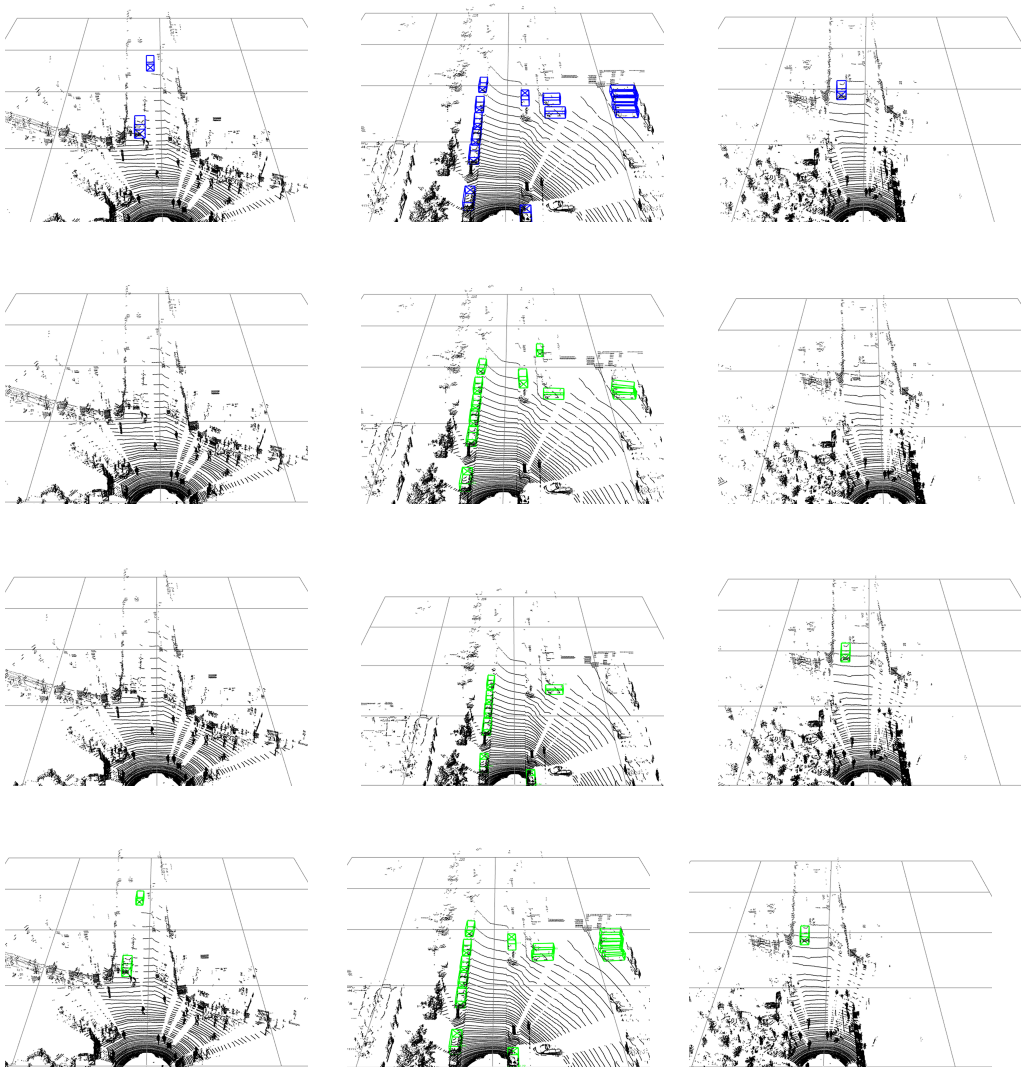


Figure 8. From top to bottom are ground truths, detection results of the SECOND, the painted SECOND and our SPV-SSD on the KITTI dataset. The ground truths and predictions are marked in blue and green respectively.

Table 9. Comparison of different semantic encodings. The class_id, one-hot, VR, seg. score and SUPV denote the object category ID, one-hot encoding, VoxelRendering, segmentation score and supervised-PointRendering, respectively. The AP is calculated with 11 recall points.

Class_ID	One-Hot	VR	Seg. Score	SUPV	Car (%)	Ped. (%)	Cyc. (%)
					Mod.	Mod.	Mod.
					81.85	53.70	68.13
✓					78.45	57.36	64.48
	✓				77.92	55.59	65.12
		✓			79.10	54.19	64.25
			✓		79.35	55.10	67.20
			✓	✓	**83.21**	**57.64**	**68.17**

4.6.3. Study on Spatial Visibility Fusion

Here we compare fusion of spatial visibility feature at different stages, as previously shown in Figure 2. Results in Table 10 show that the early fusion outperforms the late fusion,

indicating that the early fusion is more conducive to data alignment. After supplementing the spatial visibility information, the performance of detecting small objects improves significantly. The moderate APs of pedestrian and cyclist increase by 3.07% and 1.41%, respectively. Since the "Easy" and "Moderate" difficulty levels in KITTI represent the situation that can be completely observed or less occluded, it is consistent with the meaning of "visibility" in principle.

However, by employing both spatial visibility and supervised-PointRendering, the AP on pedestrian drops a little. Since the spatial visibility is calculated based on voxels, we assume that it is a similar problem to that in Section 4.5.1. Therefore, the learning of spatial visibility negatively impacts the performance of supervised point-pointing on pedestrian detection.

Table 10. Comparison of spatial visibility fusion strategies; vis.(early) indicates visibility feature in early fusion while vis.(late) is for late fusion. SUPV is short for the supervised-PointRendering. The AP is calculated with 11 recall points.

Vis.(Early)	Vis.(Late)	SUPV	Car (%)	Ped (%)	Cyc. (%)
			Mod.	Mod.	Mod.
			81.85	53.70	68.13
✓			79.33	**57.98**	69.54
	✓		78.26	55.91	69.03
✓		✓	**82.97**	55.67	**69.59**

4.6.4. Run Time and Computation Efficiency

In this section, we analyze the inference speed and time consumption of each component. We first compare our model with five popular single-stage detectors, and the result is shown in Table 11. It can be seen that our model is the fastest among them, with an average inference time of only 30.33 ms. Then, we split the pipeline into the following components: (1) point cloud pre-processing, (2) point cloud voxelization, (3) network forward processing, and (4) post-processing by NMS. As shown in Table 12, compared with the baseline method SECOND, Our anchor-free head has greatly decreased the inference time of the network, specifically, saving ∼9.4ms in model forward step. Moreover, due to the inference time of semantic segmentation network, our data reading step is a little bit slow (0.1 ms slower than SECOND).

Table 11. Comparison of our model with other single-stage detectors on run time performance. Results are in milliseconds.

1-Stage	Point-GNN	Associate-3Ddet	SASSD	3DSSD	TANet	Ours (1-Stage)
time (ms)	643	60	40.1	38	34.75	30.33

Table 12. Run time analysis (in milliseconds) of different steps during inference.

Methods	Pre-Proc.	Voxel.	Net Forward	NMS	Overall
SECOND [11]	1.5	6.6	37.5	0.7	46.3
SA-SSD [23]	1.5	<0.01	37.9	0.7	40.1
Ours	1.6	<0.01	28.1	0.6	30.3

We also analyze the computation efficiency of our model. We compare the FLOPs (floating point operations) and parameters in Table 13. The results show that the parameter amount and calculation cost of our model are rather small, demonstrating the high efficiency of our model.

Table 13. Network parameters (M) and FLOPs (G) of our model and other models.

Methods	Parameters (M)	FLOPs (G)
SECOND [11]	5.33	76.84
PV-RCNN [25]	13.13	178.50
Ours	3.76	66.45

5. Conclusions and Future Work

In this paper, we address the limitations of current LiDAR-only methods by presenting an anchor-free detection framework with a novel sensor fusion method and the LiDAR visibility representation. We propose the approach of supervised-PointRendering, which uses point-wise supervision to eliminate the influence of erroneous boundary segmentation on large object detection, improving the precision of all categories by a large margin. Current 3D object detectors process 3D point data with their coordinates yet ignore their hidden information about the free space. Here, we introduce the spatial visibility features of LiDAR to provide more spatial information for object detection. Through combining an anchor-free head with suitable label assignment, our detector solves the bottleneck of the inference time. By the experiment results on the KITTI and the nuScenes datasets, we demonstrate that our model achieves performance comparable to state-of-the-art single-stage methods with an ultra-high inference speed. The test results also demonstrate that our detector has strong robustness against different traffic scenarios.

Since this paper focuses on the improvement of 3D object detection, the adopted semantic segmentation approach is less optimized. In future work, the inference speed of the semantic segmentation network will be further promoted. Moreover, we will focus on exploiting the potential of a multi-level camera-LiDAR fusion model to boost the performance of 3D object detection.

Author Contributions: Conceptualization, W.T.; methodology, L.Y. and W.T.; software, L.Y. and L.W.; validation, L.Y.; formal analysis, L.Y.; data curation, L.W.; writing—original draft preparation, L.Y. and L.W.; writing—review and editing, W.T. and Z.W.; visualization, L.W.; supervision, W.T. and Z.Y.; project administration, W.T.; funding acquisition, W.T. All authors have read and agreed to the published version of the manuscript.

Funding: Project supported by the National Natural Science Foundation of China (No.52002285), the Shanghai Science and Technology Commission (No.21ZR1467400), the original research project of Tongji University (No.22120220593), and the National Key R&D Program of China (No.2021YFB2501104).

Data Availability Statement: The datasets generated and analysed during the current study are available in the [KITTI] repository and the [nuScenes] repository. [http://www.cvlibs.net/datasets/kitti (accessed on 4 December 2022)] and [https://www.nuscenes.org/nuscenes (accessed on 4 December 2022)].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Wang, B.; Lan, J.; Gao, J. LiDAR filtering in 3D object detection based on improved RANSAC. *Remote Sens.* **2022**, *14*, 2110. [CrossRef]
2. Deng, S.; Liang, Z.; Sun, L.; Jia, K. VISTA: Boosting 3D Object Detection via Dual Cross-VIew SpaTial Attention. In Proceedings of the Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 19–24 June 2022; pp. 8448–8457.
3. Peng, K.; Fei, J.; Yang, K.; Roitberg, A.; Zhang, J.; Bieder, F.; Heidenreich, P.; Stiller, C.; Stiefelhagen, R. MASS: Multi-attentional semantic segmentation of LiDAR data for dense top-view understanding. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 15824–15840. [CrossRef]
4. Vora, S.; Lang, A.H.; Helou, B.; Beijbom, O. PointPainting: Sequential Fusion for 3D Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 4603–4611.
5. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The KITTI vision benchmark suite. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.

6. Shi, S.; Wang, X.; Li, H. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 770–779.
7. Kamal, A.; Dhakal, P.; Javaid, A.Y.; Devabhaktuni, V.K.; Kaur, D.; Zaientz, J.; Marinier, R. Recent advances and challenges in uncertainty visualization: A survey. *J. Vis.* **2021**, *24*, 861–890. [CrossRef]
8. Yang, L.; Hyde, D.; Grujic, O.; Scheidt, C.; Caers, J. Assessing and visualizing uncertainty of 3D geological surfaces using level sets with stochastic motion. *Comput. Geosci.* **2019**, *122*, 54–67. [CrossRef]
9. Choi, J.; Chun, D.; Kim, H.; Lee, H.J. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 502–511.
10. Caesar, H.; Bankiti, V.; Lang, A.H.; Vora, S.; Liong, V.E.; Xu, Q.; Krishnan, A.; Pan, Y.; Baldan, G.; Beijbom, O. nuScenes: A Multimodal Dataset for Autonomous Driving. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11618–11628.
11. Yan, Y.; Mao, Y.; Li, B. SECOND: Sparsely Embedded Convolutional Detection. *Sensors* **2018**, *18*, 3337. [CrossRef]
12. Lang, A.H.; Vora, S.; Caesar, H.; Zhou, L.; Yang, J.; Beijbom, O. PointPillars: Fast Encoders for Object Detection From Point Clouds. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 16–20 June 2019; pp. 12689–12697.
13. Ge, R.; Ding, Z.; Hu, Y.; Wang, Y.; Chen, S.; Huang, L.; Li, Y. AFDet: Anchor Free One Stage 3D Object Detection. *arXiv* **2020**, arXiv:2006.12671.
14. Yin, T.; Zhou, X.; Krähenbühl, P. Center-based 3D Object Detection and Tracking. In Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 11779–11788.
15. Ge, Z.; Liu, S.; Li, Z.; Yoshie, O.; Sun, J. OTA: Optimal Transport Assignment for Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 303–312.
16. Yang, L.; Hou, W.; Cui, C.; Cui, J. GOSIM: A multi-scale iterative multiple-point statistics algorithm with global optimization. *Comput. Geosci.* **2016**, *89*, 57–70. [CrossRef]
17. Chen, Y.; Tai, L.; Sun, K.; Li, M. Monopair: Monocular 3d object detection using pairwise spatial relationships. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 12093–12102.
18. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. *arXiv* **2018**, arXiv:1802.02611.
19. Zhao, Z.Q.; Zheng, P.; Xu, S.t.; Wu, X. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.* **2019**, *30*, 3212–3232. [CrossRef]
20. Zou, Z.; Shi, Z.; Guo, Y.; Ye, J. Object detection in 20 years: A survey. *arXiv* **2019**, arXiv:1905.05055.
21. Zhou, Y.; Tuzel, O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4490–4499.
22. Shi, W.; Rajkumar, R.R. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 1708–1716.
23. He, C.H.; Zeng, H.; Huang, J.; Hua, X.; Zhang, L. Structure Aware Single-Stage 3D Object Detection From Point Cloud. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11870–11879.
24. Deng, J.; Shi, S.; Li, P.; gang Zhou, W.; Zhang, Y.; Li, H. Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection. In Proceedings of the AAAI, Virtual, 2–9 February 2021.
25. 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. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10526–10535.
26. Shi, S.; Wang, Z.; Wang, X.; Li, H. Part-A2 Net: 3D Part-Aware and Aggregation Neural Network for Object Detection from Point Cloud. *arXiv* **2019**, arXiv:1907.03670.
27. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
28. Law, H.; Deng, J. CornerNet: Detecting Objects as Paired Keypoints. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
29. Zhou, X.; Wang, D.; Krähenbühl, P. Objects as Points. *arXiv* **2019**, arXiv:1904.07850.
30. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3D Object Detection Network for Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6526–6534.
31. Ku, J.; Mozifian, M.; Lee, J.; Harakeh, A.; Waslander, S.L. Joint 3D Proposal Generation and Object Detection from View Aggregation. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–8.
32. Liang, M.; Yang, B.; Wang, S.; Urtasun, R. Deep Continuous Fusion for Multi-sensor 3D Object Detection. In Proceedings of the ECCV, Munich, Germany, 8–14 September 2018.
33. Chen, Z.; Li, Z.; Zhang, S.; Fang, L.; Jiang, Q.; Zhao, F.; Zhou, B.; Zhao, H. AutoAlign: Pixel-Instance Feature Aggregation for Multi-Modal 3D Object Detection. *arXiv* **2022**, arXiv:2201.06493.

34. Liu, Z.; Tang, H.; Amini, A.; Yang, X.; Mao, H.; Rus, D.; Han, S. BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. *arXiv* **2022**, arXiv:2205.13542.
35. Qi, C.; Liu, W.; Wu, C.; Su, H.; Guibas, L.J. Frustum PointNets for 3D Object Detection from RGB-D Data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 918–927.
36. Wang, Z.; Jia, K. Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Macau, China, 3–8 November 2019; pp. 1742–1749.
37. Fürst, M.; Wasenmüller, O.; Stricker, D. LRPD: Long Range 3D Pedestrian Detection Leveraging Specific Strengths of LiDAR and RGB. In Proceedings of the IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 20–23 September 2020; pp. 1–7.
38. Buhmann, J.M.; Burgard, W.; Cremers, A.B.; Fox, D.; Hofmann, T.; Schneider, F.E.; Strikos, J.; Thrun, S. The Mobile Robot Rhino. In Proceedings of the SNN Symposium on Neural Networks, Nijmegen, The Netherlands, 14–15 September 1995.
39. Hornung, A.; Wurm, K.M.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Auton. Robot.* **2013**, *34*, 189–206. [CrossRef]
40. Richter, S.; Wirges, S.; Königshof, H.; Stiller, C. Fusion of range measurements and semantic estimates in an evidential framework/Fusion von Distanzmessungen und semantischen Größen im Rahmen der Evidenztheorie. *TM-Tech. Mess.* **2019**, *86*, 102–106. [CrossRef]
41. Hu, P.; Ziglar, J.; Held, D.; Ramanan, D. What You See is What You Get: Exploiting Visibility for 3D Object Detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 10998–11006.
42. Zheng, W.; Tang, W.; Chen, S.; Jiang, L.; Fu, C.W. CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud. In Proceedings of the AAAI, Virtual, 2–9 February 2021.
43. Bodla, N.; Singh, B.; Chellappa, R.; Davis, L.S. Soft-NMS — Improving Object Detection with One Line of Code. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5562–5570.
44. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. In Proceedings of the AAAI, New York, NY, USA, 7–12 February 2020.
45. Zhou, D.; Fang, J.; Song, X.; Guan, C.; Yin, J.; Dai, Y.; Yang, R. IoU Loss for 2D/3D Object Detection. In Proceedings of the International Conference on 3D Vision (3DV), Quebec City, QC, Canada, 16–19 September 2019; pp. 85–94.
46. Lin, T.Y.; Goyal, P.; Girshick, R.B.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *42*, 318–327. [CrossRef]
47. 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 (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 3213–3223.
48. Chen, K.; Pang, J.; Wang, J.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Shi, J.; Ouyang, W.; et al. Hybrid Task Cascade for Instance Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 4969–4978.
49. nuImages. 2020. Available online: https://www.nuscenes.org/nuimages (accessed on 4 December 2022).
50. Pang, S.; Morris, D.D.; Radha, H. CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 10386–10393.
51. Chen, X.; Zhang, T.; Wang, Y.; Wang, Y.; Zhao, H. Futr3d: A unified sensor fusion framework for 3d detection. *arXiv* **2022**, arXiv:2203.10642.
52. Yin, J.; Shen, J.; Guan, C.; Zhou, D.; Yang, R. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 11495–11504.
53. Yang, Z.; Sun, Y.; Liu, S.; Jia, J. 3DSSD: Point-Based 3D Single Stage Object Detector. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 14–19 June 2020; pp. 11037–11045.
54. Zhu, X.; Zhou, H.; Wang, T.; Hong, F.; Li, W.; Ma, Y.; Li, H.; Yang, R.; Lin, D. Cylindrical and asymmetrical 3d convolution networks for lidar-based perception. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 6807–6822. [CrossRef]
55. Zhu, B.; Jiang, Z.; Zhou, X.; Li, Z.; Yu, G. Class-balanced grouping and sampling for point cloud 3D object detection. *arXiv* **2019**, arXiv:1908.09492.
56. Chen, Q.; Sun, L.; Cheung, E.; Yuille, A.L. Every view counts: Cross-view consistency in 3D object detection with hybrid-cylindrical-spherical voxelization. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 21224–21235.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.