PointSee: Image Enhances Point Cloud

Lipeng Gu, Xuefeng Yan, Peng Cui, Lina Gong, Haoran Xie, Senior Member, IEEE, Fu Lee Wang, Senior Member, IEEE, Jing Qin, Senior Member, IEEE, and Mingqiang Wei, Senior Member, IEEE

Abstract—There is a prevailing trend towards fusing multi-modal information for 3D object detection (3OD). However, challenges related to computational efficiency, plug-and-play capabilities, and accurate feature alignment have not been adequately addressed in the design of multi-modal fusion networks. In this paper, we present PointSee, a lightweight, flexible, and effective multi-modal fusion solution to facilitate various 3OD networks by semantic feature enhancement of point clouds (e.g., LiDAR or RGB-D data) assembled with scene images. Beyond the existing wisdom of 3OD, PointSee consists of a hidden module (HM) and a seen module (SM): HM decorates point clouds using 2D image information in an offline fusion manner, leading to minimal or even no adaptations of existing 3OD networks; SM further enriches the point clouds by acquiring point-wise representative semantic features, leading to enhanced performance of existing 3OD networks. Besides the new architecture of PointSee, we propose a simple yet efficient training strategy, to ease the potential inaccurate regressions of 2D object detection networks. Extensive experiments on the popular outdoor/indoor benchmarks show quantitative and qualitative improvements of our PointSee over thirty-five state-of-the-art methods.

Index Terms—3D object detection, feature enhancement, multi-modal fusion, PointSee.

I. INTRODUCTION

LiDARS, RGB, and depth cameras are prevalent yet distinct types of sensors mounted on autonomous driving vehicles and robotics for environmental perception. For example, the well-known KITTI self-driving cars possess one LiDAR and one RGB camera [1] for 3D object detection (3OD). Among the three types of sensors, LiDARs and depth cameras capture the spatial geometry information of 3D scenes, which are often represented by point clouds [2], [3]; in contrast, RGB cameras are used to take photos (RGB images) of 3D scenes, which contain rich semantic information. Recent years have witnessed considerable attempts to fuse the two-modal information (point clouds and RGB images) for 3D, since the spatial structural features from point clouds and the semantic features from RGB images have the potential to complement each other to enhance the performance of cutting-edge 3OD networks [4], [5], [6], [7], [8], [9], [10].

Currently, there are three types of multi-modal fusion solutions: point-level fusion, proposal-level fusion, and result-level fusion [11], [12]. First, point-level fusion methods, such as MVX-Net [13], EPNet [14], PointAugmenting [15] and Point-Painting [16], utilize the mapping relationship between points of point clouds and pixels of RGB images to enhance each point with additional point-wise features (e.g., CNN features, segmentation scores) from the images. However, these additional features are difficult to acquire or align with point clouds. Second, proposal-level fusion methods, such as MV3D [17] and AVOD-FPN [18], fuse the region proposals generated from both images and point clouds to produce higher-quality proposals, leading to heavy computations. Third, result-level fusion methods, such as F-PointNet [19] and F-ConvNet [20], utilize the outputs of 2D object detection (2OD) networks to generate a series of frustum point clouds as the region proposals and perform 3D in individual region proposals. This type of fusion regresses at most one object of interest in each 3D bounding frustum independently and is hard to apply to the off-the-shelf 3D networks. Thus, these methods may fail once occlusion (i.e., two or more objects of interest in a frustum point cloud) exists. Moreover, most existing fusion solutions have complex data processing pipelines or require large changes to the structure of off-the-shelf 3D networks.

We propose PointSee, a lightweight, flexible, and effective multi-modal fusion solution that enhances various off-the-shelf 3D networks by semantic feature enhancement of point clouds assembled with scene images. PointSee primarily consists of a hidden module and a seen module (see Fig. 1).
We propose a **light-weight** and **plug-and-play** module PointSee. PointSee can be plugged into cutting-edge 3D networks, particularly point-based networks, **without concern for any data augmentation** to improve their performance with marginal computational cost.

- We propose a hidden module (HM) for re-encoding additional label information **offline** into point clouds using 2D image information without changing their 3D network structures.
- We propose a seen module (SM) for enriching the point cloud with additional representative semantic features to “brighten the eyes” of 3D networks and hence obtain better detection results.

## II. RELATED WORK

We first introduce single-modal based 2D and 3D object detection methods, followed by cross-modal based 3D object detection methods from point-, proposal- and result-level perspectives respectively.

### A. Single-Modal Based 2D Object Detection

After the development of the last decade, 2D networks have achieved great success, especially in terms of high precision and real-time. 2D networks are mainly divided into three categories: two-stage, single-stage, and anchor-free networks. Two-stage networks [30], [31], [32], [33], [34], also dubbed region proposal-based networks, first generate a large number of coarse region proposals and then further refine 2D bounding boxes on the generated region proposals, which causes the problem of high computational load. To improve detection efficiency, single-stage networks [35], [36], [37], [38] remove the process of region proposal generation and directly generate the category probabilities and location coordinates of objects of interest based on a series of anchors. Early single-stage networks trade detection accuracy for higher inference speed, while modern single-stage networks are surprising in inference speed and detection accuracy. Unlike single-stage networks, anchor-free networks [39], [40], [41], [42] directly perceive the region of objects of interest on the image and then yield the category probabilities and location coordinates. In our PointSee, we use the latest 2D network Yolov5x to generate high-quality 2D bounding boxes for frustum point clouds.

### B. Single-Modal Based 3D Object Detection

Existing LiDAR-based methods only input the raw point cloud, and can be generally divided into three categories: voxel-based, point-based, and PV-based. Voxel-based methods [43], [44], [45], [46] divide the point cloud into regular voxels and then use 3D CNNs to extract voxel features for 3D. Point-based methods take the whole raw point cloud as input, extract a small number of key points and their features via some SA layers, and then carry out 3D detection. Point-based methods [23], [47], [48], [49], [50] are represented by the one-stage 3DSSD [23] and the two-stage PointRCNN [24]. Different from PointRCNN,
3DSSD disposes of all upsampling layers and refinement modules for computational efficiency. PV-based methods combine the advantages of point-based and voxel-based methods, but have large computation loads. In this work, 3DSSD and PointRCNN are used as baselines for evaluating our PointSee.

C. Cross-Modal Based 3D Object Detection

Many cross-modal based 3D networks have made considerable achievements in recent years. Intuitively, images and point clouds can provide complementary information to enhance the 3D detection performance. Existing cross-modal based methods are generally classified into point-, proposal- and result-level fusion methods.

Point-level fusion: Point-level fusion methods point-wisely decorate point clouds using the features (e.g., CNN features, segmentation scores) extracted from images. These methods project point clouds onto images for geometry-related data augmentations. However, it often introduces a poor point-pixel alignment problem. Differently, our PointSee can simply yet effectively omit the projection procedure, and directly obtain semantic features of point clouds by the well-designed Seen Module. Therefore, PointSee has no point-pixel alignment problem.

Proposal-level fusion: Proposal-level methods first generate extensive region proposals on both the image and point cloud, then fuse these region proposals, and further refine 3D bounding boxes based on the fused region proposals. These methods are hard to balance the inference speed and detection performance, due to their complex network structures.

Result-level fusion: Result-level methods first employ 2D bounding boxes output by 2D networks to generate a series of frustum point clouds, and then perform 3D detection on individual frustum point clouds. These methods are of poor flexibility and difficult to play as a plug-and-play module to serve various 3D networks. Different from them, e.g., F-PointNets and F-ConvNet, PointSee can be embedded in any off-the-shelf point-based 3D networks, and use additional representative semantic features to decorate raw point clouds for better 3D detection results (see Table I).

III. METHODOLOGY

We first introduce the motivation and architecture of PointSee in Section III-A. Then, we introduce the hidden module (HM), and the seen module (SM) in Sections III-B and III-C, respectively, followed by the proposed training strategy for 3D object detection in Section III-D. Finally, we introduce the difference between PointSee and other result-based methods in Section III-E.

A. Overview

Motivation: While there is a trend to fuse multi-modal information for 3D, existing fusion efforts often fail to meet the requirements of lightweightness, flexibility of plug-and-play and accurate alignment of features. For instance, they may have complex data processing and fusion pipelines, leading to low lightweightness; they may introduce image features for certain 3D networks and are difficult to transfer to other 3D networks, leading to poor flexibility of plug-and-play; also, multiple points in a point cloud may be projected to the same pixel in the image, leading to an inaccurate alignment of point-to-point features. Therefore, a lightweight, flexible, and effective multi-modal fusion solution is required to facilitate various 3D networks by representative semantic feature enhancement of point clouds assembled with scene images.

We introduce PointSee which is powerful and flexibly embedded in various 3D networks. PointSee takes point clouds and images as input to focus 3D networks on the regions where objects may exist, as well as to enhance point semantic features for better 3D detection performance. As shown in Fig. 2, PointSee consists of a hidden module (HM) and a seen module (SM). HM is an offline processing module, which is not required to embed in 3D networks. Only SM is embedded in the head of 3D networks for end-to-end training and inference.

B. Hidden Module

Data augmentations are commonly applied in the training process of various tasks (e.g., 2D and 3D). We surprisingly find that most of the existing data augmentations, e.g., Random-Rotation and GT-Paste, are only suitable for single-modal based methods. Thus, for cross-modal fusion based 3D networks, data augmentations are either discarded or significantly modified to ensure proper alignment between images and point clouds. Facing this problem, PointPainting is the first to process point clouds offline, so as to neither modify the structure of 3D networks nor be affected by data augmentations.

To skillfully avoid the problem of data augmentations that cannot adapt to cross-modal architectures, HM is designed based on the strategy: each point is appended with additional label information so that the re-encoded point clouds can be fed to the 3D network without concerning the data processing and data augmentation pipelines. Specifically, HM contains three parts: a 2D network, a 3D-to-2D projection module, and a re-encoding module.

We use Yolov5x as the image-based 2D network to predict objects of interest in the image and output their 2D bounding boxes \( Det_{2D} = \{(x_1, y_1, x_2, y_2, c)\}_{i=0}^{M-1} \), where \( x_1, y_1, x_2, y_2 \) represent the coordinates of the upper-left and lower-right corners of the 2D bounding box, \( c \) represents the category and \( M \) represents the number for detected objects in a frame. Based on the known Homogenous transformation matrix \( T \) and camera matrix \( C \), the 3D-to-2D projecting module projects the point cloud onto the image plane, and then captures all points inside the 2D bounding boxes to form multiple frustum point clouds. As shown in Fig. 2(c), there is at least one object in each frustum point cloud.

The re-encoding module point-wisely appends frustum point clouds with \( \text{seg} \_\text{label} \), \( \text{cls} \_\text{label} \) and \( \text{index} \_\text{label} \) in each frame. \( \text{seg} \_\text{label} \) is the mask information (1 or 0 for foreground or background points), which is obtained by determining whether the point is inside ground-truth 3D bounding boxes \( GT_{3D} = \{(x, y, z, h, w, l, ry)\}_{i=0}^{K-1} \) or not, where \( x, y, z, h, w, l, ry \) represent the center coordinates, length, width, height and
Fig. 2. Overview of PointSee. PointSee includes two parts: (1) Hidden Module (HM); and (2) Seen Module (SM). HM extracts frustum point clouds based on 2D bounding boxes output by the 2OD network, and then expands three dimensions (semantic segmentation label, category label and object number) for retained points. SM extracts additional point-wise semantic features to decorate these retained points for better detection results.

Algorithm 1: Hidden Module.

Input:
Points: \( P \in \mathbb{R}^{N,D} \), with \( N \) points and \( D \geq 3 \)
Detections: \( \text{Det}_{2D} \in \mathbb{R}^{M,5} \), with \( M \) detections
3D bounding boxes: \( \text{GT}_{3D} \in \mathbb{R}^{K,7} \), with \( K \) labels
Homogenous transformation matrix: \( T \in \mathbb{R}^{4,4} \)
Camera matrix: \( C \in \mathbb{R}^{3,4} \)

Output: Re-encoded points: \( P' \in \mathbb{R}^{N,D+3} \)

1. \( P' \leftarrow \text{Zero}(0, D + 3) \)
2. \( \text{num} \leftarrow 0 \)
3. for \( \text{det}_{2D} \in \text{Det}_{2D} \) do
4.  for \( p \in P \) do
5.    \( p_{\text{image}} \leftarrow \text{PROJECT}(C, T, p) \)
6.    if \( p_{\text{image}} \) inside \( \text{det}_{2D} \) then
7.      \( p \leftarrow \text{Concatenate}(p, 0) \)
8.      for \( \text{gt}_{3D} \in \text{GT}_{3D} \) do
9.        if \( p \) inside \( \text{gt}_{3D} \) then
10.         \( p[-1] \leftarrow 1 \)
11.     end
12.    end
13.    \( p \leftarrow \text{Concatenate}(p, \text{det}_{2D}[4]) \)
14.    \( p \leftarrow \text{Concatenate}(p, \text{num}) \)
15.    add \( p \) to \( P' \)
16. end
17. num \leftarrow \text{num} + 1
18. end
19. return \( P' \)

heading angle of the 3D bounding box, respectively and \( K \) represents the number for ground-truth 3D bounding boxes in a frame. \( \text{cls}_\text{label} \) is the category of 2D bounding box to which each point belongs. For the KITTI dataset, \( \text{cls}_\text{label} \) uses 0, 1, 2 and -1 to represent Car, Pedestrian, Cyclist and background respectively. \( \text{index}_\text{label} \) is the number of 2D bounding boxes to which each point belongs in the current frame. The above label information is used for the subsequently seen module to perform the 3D instance segmentation (3IS) task.

Similar to PointPainting [16], the offline operation has some advantages. First, appending additional label information at each point not only does not require changing the complex data loading and processing pipeline of the 3OD network, but also does not add some extra computational load to the 3OD network. More importantly, this offline operation is not affected by any data augmentation of both point clouds and images. And there is no problem that pixels or convolution features of images are difficult to align with point clouds due to some data augmentations, such as folding, clipping, and scaling of the image, rotation of the point cloud, and GT-Paste [43].

Remark: In PointSee, HM takes the 2OD results as input. Intuitively, it raises the concern of the entire 3OD pipeline which may depend heavily on the 2OD performance. However, we find that PointSee is predominantly affected by the recall rate rather than the detection accuracy. To validate it, we conduct experiments to dig into the dependencies between PointSee and various 2OD networks in Fig. 9. It is clearly observed that the 2OD network with a higher recall can exploit the power of PointSee to improve cutting-edge 3OD networks with marginal computational cost.

Option: Naturally, if an object is not detected by the 2OD network, it will not be detected by the 3OD network either. To reduce the effect of missing 2D detections and further improve the robustness of the 3OD network, we propose an optional module dubbed “RZ-based Frustum Fusion”. This module can compensate for potential failures in 2OD by integrating 3OD into HM of PointSee.

RZ-based Frustum Fusion relies on the Rotation angle around Z-axis in the LiDAR coordinate to fuse frustum point clouds from both the 2D and 3D domains, improving the overall
reliability of the entire 3OD pipeline. Prior to frustum fusion, we generate two sets of frustum point clouds for 2D/3D domains by inputting 2D/3D detections into HM. These sets are represented as

\[ F^k = \{ f^k \}^N_k, \quad k \in \{ 2d, 3d \} \]  
\[ f = (r_{z1}, r_{z2}, s, c) \]  

Herein, 2d and 3d respectively represent that frustum point clouds are obtained based on detections of the 2OD and 3OD (to be enhanced by PointSee networks), and \( N \) denotes the number of frustum point clouds. \( r_{z1} \) and \( r_{z2} \) respectively represent the minimum and maximum rotation angles around the Z-axis in the LiDAR coordinate for all points within the frustum point clouds. \( s \) and \( c \) represent the confidence score and category associated with the corresponding detection. Notably, before 3D detections are fed into HM, they are first projected onto the image plane to obtain 2D bounding boxes.

We illustrate an example of how to fuse two frustum point clouds \((f^{2d}, f^{3d})\) of the same category into a new set of frustum point clouds \((\hat{F})\). The fusion can be carried out using (3) as

\[
\hat{F} \leftarrow \begin{cases} 
 f^{2d} \cup f^{3d}, & RZ_{IoU}(f^{2d}, f^{3d}) > \theta_1 \\
 f^{2d}, & RZ_{IoU}(f^{2d}, f^{3d}) < \theta_1 \land s^{2d} > \theta_2 \\
 f^{3d}, & RZ_{IoU}(f^{2d}, f^{3d}) < \theta_1 \land s^{3d} > \theta_3 
\end{cases}
\]

\[ RZ_{IoU}(f^{2d}, f^{3d}) = \frac{IA}{(r_{z2}^{2d} - r_{z1}^{2d}) + (r_{z2}^{3d} - r_{z1}^{3d}) - IA} \]  
\[ IA(f^{2d}, f^{3d}) = \max(0, \min(r_{z2}^{3d}, r_{z2}^{2d}) - \max(r_{z1}^{3d}, r_{z1}^{2d})) \]

where \( \theta_1 \) is the threshold for determining whether two frustum point clouds are the same object, and \( \theta_2 \) and \( \theta_3 \) are the thresholds for the confidence score of the corresponding detection. The experimental results in the 5th and 6th rows of Table IV demonstrate that 2D detections are more reliable than 3D detections for PointSee. Thus, on KITTI, we empirically set \( \theta_1 \) to 0.3, \( \theta_2 \) to 0.8, and for specific categories, we set \( \theta_2 \) to 0.3 for the Car category and 0.1 for the Pedestrian and Cyclist categories.

Please note the fact that outdoor scenes tend to have sparse objects with long distances between them, while indoor scenes generally host a larger number of closely spaced and even overlapped objects. We discover the effectiveness of the optional module “RZ-based Frustum Fusion” in the expansive outdoor scenes during the experiments. But within the confined indoor scenes, it often leads to erroneous fusion. Consequently, this optional module, as a plug-and-play component, is only implemented within the outdoor KITTI dataset to enhance the robustness of the whole 3D detection pipeline, and we avoid it in the indoor SUN-RGBD and ScanNetV2 datasets. Besides, considering the density of objects within the indoor scenes, we do not drop background points outside the frustum point clouds in HM for the solidness of the entire 3OD pipeline.

C. Seen Module

We find that the 3D detection performance will be clearly improved if only foreground points in the point cloud are fed into the 3OD network. Further, suppose the point cloud has additional mask information (0 or 1 for the foreground or background point). In that case, the performance of the 3D network may also achieve an improvement without adding additional computational overhead. To verify it, we use ground-truth 3D bounding boxes to generate the point-wise mask information for the point cloud, which is fed into 3DSSD [23]. As illustrated in the 1st and 2nd rows of Table VI, the mask information is beneficial for the 3D network. However, the segmentation results of existing lightweight 3IS networks are not remarkably accurate, especially for small objects. In addition, semantic segmentation features are critical clues that can provide the discriminativeness for subsequent object detection. Therefore, for the detection task, the high-dimensional segmentation features output from the 3IS network are more suitable as additional local segmentation features for point clouds than the mask information of each point.

Similar to existing point-level fusion methods, our seen module (SM) is designed to obtain point-wise representative semantic features to decorate the point cloud for better performance of any 3OD network. Compared to other multi-modal fusion solutions, e.g., PointPainting [16], EP-Net++ [51], the distinct advantage of SM is that it directly acquires point-wise features without the point-to-pixel matching process, thus avoiding the inaccurate alignment of features.

The fundamental idea behind SM relies on local segmentation features, which are then fused with global geometric features and intrinsic point cloud features. Such fusion generates point-wise representative semantic features, enhancing the performance of the 3OD network with minimal computational cost. SM comprises six components: data preprocessing, a local segmentation feature branch, a global geometric feature branch, an intrinsic point cloud feature branch, and a feature fusion module, all of which are built upon a full-MLPs design (see Fig. 3). In the data preprocessing stage, the input re-encoded point cloud is first geometrically augmented to obtain the augmented re-encoded point cloud. This augmentation involves rotational and foliated transformations, which modify the spatial arrangement of the points within the frustum point cloud. Then, \( ACT1 \) divides the augmented and unaugmented point cloud into multiple frustum point clouds based on \( index_label \). Within each of these frustum point clouds, the points are resampled to 2048. After that, for \( ACT2 \), the one hot vector is generated by \( cls_label \) for the 3IS task in the local segmentation feature branch. For the KITTI dataset, \( one hot vector \) represents Car, Pedestrian, Cyclist by 100, 010 and 001. Additionally, the \( seg_label \) is also extracted to compute Cross Entropy Loss for 3IS in the local segmentation feature branch. The subsequent use of \( ACT3 \) involves separating the augmented and unaugmented frustum point clouds.

In the local segmentation feature branch, to reduce the additional computational overhead, multiple unaugmented frustum point clouds are stacked together along the batch dimension to form a new tensor. This tensor is fed into a 3IS network to obtain high-dimensional local segmentation features \((F_{1 \alpha})\), which is
Fig. 3. Seen module. The lightweight design of the module is based on a full-MLPs architecture. It inputs disordered points with three additional dimensions inside re-encoded frustum point clouds and outputs decorated points with 16-dimensional semantic features. These semantic features are obtained by fusing global geometric features (G.G. feat.), local segmentation features (L.S. feat.), and intrinsic point cloud features (P.C. feat.), thereby enhancing the representation of the point cloud. Geo. Aug. refers to the geometric spatial augmentation applied to each frustum point cloud. ACT1 divides the disordered point cloud into multiple frustum point clouds using an index (idx), ensuring that each object belongs to a specific frustum. ACT2 transforms the category of each frustum point cloud based on the class (cls) value, resulting in a one-hot vector representation. ACT3 separates the augmented and unaugmented frustum point clouds.

In the global geometric feature branch, Mean Squared Error (MSE) is utilized as a spatial geometric invariant loss to acquire global geometric features (F_{gg}). Furthermore, to enrich the point cloud with more representative semantic features, we make use of intrinsic point cloud features (F_{pc}) through two MLPs. Finally, as shown by Module D in Fig. 4, the semantic features (F) output by SM are derived from the fusion of local segmentation features (F_{ls}), global geometric features (F_{gg}), and intrinsic point cloud features (F_{pc}). The fusion process is formulated as

\[ F = MLPs(Concat(MLPs(F_{ls}), F_{gg}, F_{pc})) \] (6)

where MLPs represents multiple MLP layers, Concat represents the concatenation operation, F are point-wise representative semantic features for decorating point clouds.

Additionally, we introduce four different fusion modules in Fig. 4 to thoroughly investigate the effect of mask information, local segmentation features, global geometric features, and intrinsic point cloud features on the 3OD network. Our experiments, as shown in Table VI, reveal that the fusion of these three features (F_{ls}, F_{gg}, and F_{pc}) in SM significantly “make the eyes of the 3D network brighter”. Moreover, we also conduct experiments to empirically demonstrate how the integration of these semantic features enhances 3OD networks at the micro level, as illustrated in Table VIII.

The point cloud in Fig. 3 is an example of the KITTI dataset, in which each point is represented by x, y, z coordinates and reflection intensity r respectively; n, m, c, n_obj and n_cls are respectively expressed as the number of points in the point cloud.
the number of sampled points for each object, the number of intrinsic point cloud features, the number of objects in the point cloud and the number of categories (without background). \( m \) is 2048 for both the KITTI, SUN-RGBD, and ScanNetV2 datasets.

D. Training Strategy for 3D Object Detection

2D networks often suffer from inaccurate 2D bounding box regression, and PointSee is also sensitive to the output of any 2D network. Inspired by the data enhancement strategy of 2D networks, a simple yet effective training strategy is designed to solve this problem in the training and inference stages of the 3D network. In the training stage, the length and width of the 2D bounding box are randomly increased by 0% - 10% to accommodate the excessive size of the 2D bounding box. In the inference stage, to mitigate the problem that the size of 2D bounding boxes is too small, the length and width of the 2D bounding box are also increased by 5% to enlarge the receptive field of the frustum point cloud. This strategy is proved to be effective in the following experiments (see Table IV).

The decorated point cloud can be fed to various 3D networks with the point-based encoder since PointSee just changes the input dimension of point clouds and adds SM (see Fig. 3). To validate the effectiveness of PointSee on various point-based 3D networks, we employ 3DSSD [23], PointRCNN [24], VoxSet [55], ImVoteNet [25], GroupFree3D [27], F-PointNet [19] as baselines to conduct experiments on the outdoor KITTI dataset [1], the indoor SUN-RGBD [26] and ScanNetV2 [28] datasets.

E. Differences From Other Result-Based Methods

PointSee and other result-based methods [19], [20], [54] share the same overall architecture which contains two parts, i.e., a proposal generator and a 3D box estimator. As shown in Fig. 5, the proposal generator obtains the regions where the objects of interest are located using off-the-shelf 2D networks, and then projects the positions of these objects into the point cloud space using camera parameters to obtain proposals in the point cloud. The 3D box estimator is designed based on PointNet and/or FCN to predict 3D bounding boxes.

We describe three representative result-based methods, i.e., F-PointNet [19], F-ConvNet [20], and RoarNet [54], and introduce their difference against our PointSee. F-PointNet and F-ConvNet first employ 2D bounding boxes output by the 2D network to generate a series of frustum point clouds, and then perform 3D based on the individual frustum point clouds; RoarNet uses a monocular pose network to acquire the pose information of objects in the image to eliminate background points. Our PointSee has two different aspects with them. First, PointSee owns the plug-and-play adaptability to arbitrary cutting-edge 3D networks. Second, PointSee exploits additional representative semantic features of point clouds for better detection results.

IV. Experiments

A. Dataset

We validate PointSee on both the outdoor KITTI [1] and indoor SUN-RGBD [26] and ScanNetV2 [28] datasets.

KITTI is a popular benchmark for 3D in outdoor scenes. It contains 7481 samples for training and 7518 samples for testing. Each sample consists of a point cloud and an RGB image.
with nine categories (Car, van, Truck, Pedestrian, Cyclist, Person_sitting, Tram, Misc, DontCare). Following the commonly applied setting, we divide all training samples into 3712 samples and 3769 samples for train split and val split. For submission to the KITTI test server, we follow the training strategy in SASA [21], where 80% of training samples are used for training and the rest 20% of training samples are used for validation.

**SUN-RGBD** is a 3D dataset for indoor scenes. It contains 5285 RGB-D images for training and 5050 RGB-D images for testing.

**ScanNetV2** is an indoor dataset that can be used to evaluate 3D tasks. ScanNetV2 contains 1513 scenarios, of which 1201 scenarios are for training and 312 for testing.

**B. Experiment Settings**

To widely verify PointSee, we select six different and representative baselines 3DSSD [23], PointRCNN [24], VoxSet [55], ImVoteNet [25], GroupFree3D [27] and F-PointNet [19] for evaluation. Our experimental models run on a single GTX3090. Furthermore, for KITTI, we use YOLOv5x for 2D, and for SUN-RGBD, we employ Faster R-CNN [30] for 2D, similar to ImVoteNet. For ScanNetV2, we straightforwardly utilize 3D detections output by GroupFree3D to wrap point cloud regions as frustum point clouds.

**3DSSD and PointRCNN**: For the KITTI dataset, we train 3DSSD and PointRCNN in an end-to-end manner with the same parameters as SASA [21]. Specifically, for 3DSSD and PointRCNN, the optimizer is Adam, one cycle. The number of epochs is 80, the learning rate is 0.01 for 3DSSD and 0.005 for PointRCNN, the weight decay is 0.01, and the number of sampled points fed to backbone networks is 16384. In addition, since HM is offline, there is no need to care about data augmentations of the original 3D network. We only change the number of channels dedicated to the input point cloud and add the code of SM.

**YOLOv5x**: For the KITTI dataset, we use the largest model YOLOv5x to train in an end-to-end manner. In addition, we merge Car, Truck and Van into the Car category, and only retain the Car, Pedestrian and Cyclist categories for training and inference. During training, the optimizer is SGD, the initial learning rate is 0.001, the final OneCycleLR learning rate is 0.01, the weight decay is 0.0005, the number of epochs is 50, and the image size is 1280 × 1280 in both the training and inference stages.

**ImVoteNet and Faster R-CNN**: For the SUN-RGBD dataset, we train ImVoteNet and Faster R-CNN without changing parameters and use the output of Faster R-CNN as 2D detections. Specifically, for ImVoteNet, the optimizer is Adam, the number of epochs is 80, the learning rate is 0.008, and the weight decay is 0.0005.

**GroupFree3D**: For the ScanNetV2 dataset, we train GroupFree3D without changing parameters and directly use its 3D detection outputs to enclose point cloud regions as frustum point clouds. Specifically, the baseline GroupFree3D (L12, O256) employs a 6-layer decoder and 256 object candidates, the optimizer is AdamW, the number of epochs is 80, the learning rate is 0.006, and the weight decay is 0.0005.

**F-PointNet**: It is used for comparison experiments in crowded scenarios (see Table IX and Fig. 8). We train F-PointNet without modifying parameters using yolov5x and the officially provided 2D detection as inputs, respectively. Specifically, the optimizer is Adam, the number of epochs is 200, the learning rate is 0.001, and the weight decay is 0.0005.

**C. Comparison and Evaluation**

We evaluate the effectiveness of our PointSee on the KITTI, SUN-RGBD, and ScanNetV2 datasets. For the KITTI dataset, our evaluation is performed on the point-based 3D detection networks represented by the one-stage 3DSSD and two-stage PointRCNN for outdoor scenes. For SUN-RGBD and ScanNet datasets, we respectively adopt ImVoteNet and GroupFree3D as baselines to evaluate the effectiveness of PointSee for indoor scenes.

**Results on the KITTI Dataset**: Table I shows the 3D object detection performance on the KITTI test set. For the most competitive Car category, 3DSSD with PointSee surpasses existing uni-modal and multi-modal 3D detection networks for three levels, and obtains comparable results to PV-based models (e.g., PV-RCNN [29]) and multi-models (e.g., CAT-Det [60], HMFI [61], and DVF-PV [62]). Furthermore, similar to the latest SASA [21], PointSee improves the performance of existing 3D networks by facilitating SA layers to sample more positive points (see Table VIII). Yet, with the same baseline model 3DSSD, PointSee outperforms SASA, especially for the easy level of the Car category, by more than 2%. For the moderate and hard levels of the Car category, PointSee outperforms SASA by more than 0.6%. Compared with the baseline model 3DSSD, PointSee improves by 2.15%, 1.65%, 1.5% for the Car category, 6.75%, 4.69%, and 3.8% for the Pedestrian category, and 0.11%, 0.64%, and 0.3% for the Cyclist category, respectively, at the three difficulty levels. These experimental results show that PointSee can enhance the uni-modal method 3DSSD with marginal computational cost across three categories, especially the Car and Pedestrian categories. This enhancement empowers 3DSSD to surpass multi-modal methods (e.g., EPNet [14], EPNet++ [51], CAT-Det [60], HMFI [61] and DVF-PV [62]) for the most competitive Car category.

**Results on the SUN-RGBD Dataset**: As shown in Table II, we use ImVoteNet as the baseline to evaluate PointSee on the SUN-RGBD dataset. PointSee enables ImVoteNet to achieve better performance in six categories and boosts the mAP by 0.9%, as well as surpassing the other ten SOTA methods. This further validates the flexibility and scalability of our proposed PointSee.

**Results on the ScanNetV2 Dataset**: As known, both the KITTI and SUN RGBD datasets are somewhat noisy. For example, KITTI has low-quality calibration of images and point clouds; SUN-RGBD has inaccurate depths. These factors may affect the performance of cutting-edge 3D methods. Therefore, we additionally utilize GroupFree3D as the baseline to validate the effectiveness of PointSee on the ScanNetV2 dataset (see
Table III). As shown in Table III, the experimental results further demonstrate the effectiveness and generalization of PointSee on ScanNetV2. Specifically, PointSee enables GroupFree3D to achieve better performance in twelve categories and boosts the mAP by 0.7%.

D. Ablation Study

We employ 3DSSD as the baseline to conduct ablation studies to validate each part of PointSee. The results provided in Tables IV and VI are trained on the KITTI train set and evaluated on the KITTI val set.
Effects of Hidden Module: The 1st and 2nd rows of Table IV verify the effectiveness of frustum point clouds, which helps the 3OD network to focus more on the regions of point clouds where objects of interest may exist. Note that the 2nd column (“GT”) in Table IV uses ground-truth 2D bounding boxes, while the 3rd and 4th columns (“2D” and “3D”) in Table IV uses outputs of the 2OD network Yolov5x and the 3OD network 3DSSD. From the experimental results of the 1st and 3rd rows of Table IV, using 2D bounding boxes output from Yolov5x for SM will degrade the performance of 3DSSD for the Car category. This is because 2D bounding boxes of the Car category output from Yolov5x may be too small or too large, resulting in some foreground points of objects being erroneously discarded or some background points being mistakenly retained. However, there are some improvements in the Pedestrian and Cyclist categories, due to the fact that 3D detection failures in these two categories are often caused by false-positive detections, and the elimination of numerous perplexing background points reduces false-positive detections of the 3OD network.

Effects of Seen Module: The 3rd and 4th rows of Table IV compare the 3D performance of 3DSSD with and without the seen module. This module significantly boosts the 3D AP for the easy, moderate, and hard levels of the three categories. The results validate the effectiveness of the module in helping point clouds extend more representative semantic features, which is very helpful in improving the performance of 3OD networks.

Effects of Training Strategy: The 4th and 5th rows of Table IV verify that our proposed training strategy can effectively increase the robustness of the 3OD network and reduce their sensitivity to 2D bounding box regression (too large or too small) of the 2OD network. In addition, it is worth noting that the 5th row of Table IV obtains comparable results to the 2nd row of Table IV. Remarkably, the performance for the Pedestrian category even surpasses that achieved using ground-truth (GT) annotations (see 2nd rows of Table IV), which further demonstrates the superiority of our PointSee.

Effects of RZ-based Frustum Fusion: Table V shows the performance of 3DSSD with PointSee and PointRCNN with RZ-based frustum fusion in the KITTI Val set. The experiment demonstrates that for the Car category, there is not a significant difference between using only 2D detections or 3D detections. However, there is a noticeable performance difference between the Pedestrian and Cyclist categories. Moreover, for both 3DSSD and PointRCNN, the fusion of 2D and 3D detections through this module significantly enhances the detection accuracy for all three categories (see 4th and 8th rows of Table V). Fig. 7 also demonstrates the effectiveness of this module in compensating for 2D detection failures using 3D detections output by the 3OD network itself (e.g., 3DSSD and PointRCNN). Specifically, the 1st and 3rd rows of Fig. 7 illustrate how failed 2D missing detections are corrected by 3D detection, while the 2nd and 4th rows show how failed 2D and 3D false detections are mutually corrected, resulting in further performance improvements for PointSee. These results demonstrate that our RZ-based Frustum Fusion can further enhance the reliability of the entire 3OD pipeline.

Effects of Feature Fusion Module: Fig. 4 shows four different feature fusion modules, and the experimental results are shown in Table VI. Module A (GT mask) verifies our observation that the additional mask information for the point cloud can help the 3OD network obtain a performance improvement. Further, Module A (Pred. mask) decorates the point cloud with the

### Table IV

| Method          | Input of Hidden Module | Car     | Pedestrian | Cyclist |
|-----------------|------------------------|---------|------------|---------|
|                 | w/o 2D 3D              | Easy    | Mod.      | Hard    | Easy    | Mod.      | Hard    | Easy    | Mod.      | Hard    |
| 3DSSD           | ✓                       | ✓       | ✓         | ✓       | 91.84   | 84.52     | 82.10   | 59.46   | 54.69     | 50.46   | 89.82   | 69.73     | 65.49   |
|                 | ✓                       | ✓       | ✓         | ✓       | 93.50   | 86.66     | 84.05   | 64.29   | 58.82     | 53.99   | 94.52   | 78.81     | 75.86   |
|                 | ✓                       | ✓       | ✓         | ✓       | 91.28   | 82.09     | 79.57   | 63.41   | 56.99     | 51.98   | 91.80   | 68.65     | 65.58   |
|                 | ✓                       | ✓       | ✓         | ✓       | 91.41   | 83.55     | 81.74   | 65.68   | 58.52     | 53.07   | 92.08   | 70.66     | 67.13   |
|                 | ✓                       | ✓       | ✓         | ✓       | 92.19   | 85.39     | 82.62   | 67.02   | 61.83     | 55.46   | 90.19   | 73.14     | 68.38   |
|                 | ✓                       | ✓       | ✓         | ✓       | 92.48   | 85.39     | 82.52   | 59.41   | 53.44     | 48.07   | 90.13   | 71.08     | 66.24   |
|                 | ✓                       | ✓       | ✓         | ✓       | 92.58   | 86.03     | 83.22   | 67.15   | 62.29     | 55.87   | 91.76   | 74.32     | 69.12   |

### Table V

| Method          | Input of Hidden Module | Car     | Pedestrian | Cyclist |
|-----------------|------------------------|---------|------------|---------|
|                 | w/o 2D 3D              | Easy    | Mod.      | Hard    | Easy    | Mod.      | Hard    | Easy    | Mod.      | Hard    |
| 3DSSD           | ✓                       | ✓       | ✓         | ✓       | 91.84   | 84.52     | 82.10   | 59.46   | 54.69     | 50.46   | 89.82   | 69.73     | 65.49   |
|                 | ✓                       | ✓       | ✓         | ✓       | 92.19   | 85.39     | 82.62   | 67.02   | 61.83     | 55.46   | 90.19   | 73.14     | 68.38   |
|                 | ✓                       | ✓       | ✓         | ✓       | 92.48   | 85.39     | 82.52   | 59.41   | 53.44     | 48.07   | 90.13   | 71.08     | 66.24   |
|                 | ✓                       | ✓       | ✓         | ✓       | 92.58   | 86.03     | 83.22   | 67.15   | 62.29     | 55.87   | 91.76   | 74.32     | 69.12   |
| PointRCNN       | ✓                       | ✓       | ✓         | ✓       | 91.28   | 80.70     | 78.37   | 65.12   | 56.43     | 49.24   | 90.02   | 72.42     | 67.44   |
|                 | ✓                       | ✓       | ✓         | ✓       | 91.33   | 81.86     | 79.11   | 63.19   | 56.21     | 49.09   | 85.23   | 68.36     | 65.22   |
|                 | ✓                       | ✓       | ✓         | ✓       | 89.72   | 81.87     | 79.27   | 79.27   | 53.50     | 49.09   | 85.23   | 68.36     | 65.22   |
|                 | ✓                       | ✓       | ✓         | ✓       | 91.58   | 81.98     | 79.22   | 65.74   | 56.85     | 50.97   | 92.50   | 72.74     | 67.86   |

The evaluation metric 3D average precision is calculated on 40 recall points. "W/O" denotes the absence of hm, while “2D” and “3D” denote that hm inputs 2D detections and its own 3D detections, respectively.
predicted mask information, and the results are much different than those of Module A (GT mask). It is caused by two main reasons: i) the lightweight 3IS network is not very effective for small objects, and ii) the 2OD network may have the problem of missing detection. To solve the problem, we use the local segmentation features output from the 3IS network instead of the mask information. In Table VI, the comparison between the 4th (Module B) and 3rd (Module A) rows highlights the effectiveness of local segmentation features, especially for the Pedestrian category. The comparison between the 5th (Module C) and 4th (Module B) rows in Table VI also indicates that intrinsic point cloud features contribute to the performance improvement of 3D detection. The 6th row (Module D) in Table VI demonstrates that global geometric features facilitate the 2OD network to achieve the best performance across all three categories. From the results above, we can observe that features obtained through a full-MLPs architecture, including local segmentation features, global geometric features, and intrinsic point cloud features, all contribute to the improvement of 3D detection performance.

E. Compatibility Study

PointSee is an easy-to-plug-in design and can be applied to various point-based 3OD networks. As shown in Table VII, PointSee obtains notable enhancement on the one-stage model 3DSSD and the two-stage model PointRCNN.

Results on the KITTI Dataset: The backbone of 3DSSD and PointRCNN primarily comprises multiple SA (Set Abstraction) layers that are responsible for extracting key points for object localization and 3D bounding box regression. Within these SA layers, some points of small or occluded objects are easily discarded by mistake. It leads to the problem of missing detection. In contrast, PointSee is helpful in avoiding this problem. As shown in Table VII, PointSee improves the detection performance of 3DSSD and PointRCNN for the three categories. In the case of more challenging instances with fewer points, PointSee can better focus on these points, and acquire more representative semantic features to aid the subsequent 3OD task.

Although PointSee enhances the performance of point-based encoders clearly, it behaves not well for voxel-based encoders. For example, PointSee has slight decrements for VoxSeT [55] for the Car and Cyclist categories. This is because, i) the 3OD performance of point-based encoders is limited by the quality of the key points extracted during the downsampling of SA layers; as shown in Table VIII, PointSee essentially helps SA layers to sample more key points and facilitates the extraction of more discriminative features for better 3OD performance. ii) different from point-based encoders, there is no SA layer for downsampling in VoxSeT, but rather the set-to-set transformation used to learn features of point clouds. Therefore, PointSee does not behave well for voxel-based encoders. For the Pedestrian category, PointSee shows a significant improvement over VoxSeT. This is due to the fact that VoxSeT inherently lacks strong detection capabilities for the Pedestrian category, and the Hidden Module (HM) of PointSee eliminates a considerable number of distracting background points, thereby enhancing the detection accuracy in the Pedestrian category.

Inference Speed: As shown in the 6th column of Table VII, SM of PointSee adds 6 ms, 3 ms, and 4 ms of additional time consumption per frame for 3DSSD, PointRCNN, and VoxSeT, respectively. PointSee brings obvious performance improvement to the 3OD network with a little more but tolerable time. To better illustrate the overall detection pipeline’s efficiency, we...
Fig. 6. Visualization of 2D and 3D detections (including output by Yolov5x and 3DSSD, the 1st column), detections output by 3DSSD with PointSee (the 2nd column) and detections output by 3DSSD (the 3rd column) on the KITTI val set. The predicted boxes and ground-truth boxes are labeled in blue and red, respectively. The 3D detections in the 1st column are projected onto the image plane and subsequently fed into HM of PointSee. These visualizations show the capability of PointSee to reduce false and missing detections respectively. Particularly, the 2nd and 3rd rows illustrate the capability of PointSee to facilitate the detection of these hard instances and generate more accurate 3D bounding boxes, respectively.

### TABLE VIII

**Effects of PointSee on the SA Layer and the Candidate Point Generation Layer of 3DSSD on the KITTI Val Set**

| Method               | 256 (fp)  | 256 (dp)  | 256 (candidate) |
|----------------------|-----------|-----------|-----------------|
|                      | Point Recall | Positive Points | Point Recall | Positive Points | Point Recall | Positive Points |
| 3SSSD                | 94.42      | 16        | 87.84          | 7             | 94.42        | 16             |
| 3SSSD + only Hidden Module | 96.17      | 51        | 95.48          | 41            | 96.17        | 51             |
| 3SSSD + PointSee     | 96.24      | 53        | 95.55          | 40            | 96.24        | 53             |

The evaluation metric point recall and positive points is the recall rate of sampled positive points and the average number of sampled positive points per object. “256 (fp)” and “256 (dp)” represent the key points sampled at the last sa layer with the F-PS and D-PS sampling strategies, respectively. “256 (candidate)” represents the key points sampled at the candidate generation layer.
Additionally include the runtime of HM (including the runtime of 2OD, 3OD, and HM itself) in Table VII. Specifically, HM of PointSee adds 37 ms, 59 ms, and 29 ms of time consumption per frame for 3DSSD, PointRCNN, and VoxSeT, respectively. Besides, PointSee only adds 0.9 M parameters to the 3OD network.

Quantitative Analysis: In Fig. 6, we visualize representative detection results on the KITTI val set. The 1st column shows 2D and 3D detections (including output by Yolov5x and 3DSSD). The 2nd column shows detections output by 3DSSD with PointSee. The 3rd column shows detections output by 3DSSD. PointSee assists the 3OD network in reducing false detections significantly (see the 1st and 4th rows of Fig. 6) and missing detections (see the 2nd row of Fig. 6), which benefits from the fact that the frustum point clouds reduce the field of view of the 3OD network and only focus on the area where there may be objects. In addition, as illustrated in the 3rd row of Fig. 6, PointSee also enables the 3OD network to obtain more representative semantic features for harder instances (e.g., small, distant, or occluded objects), which can facilitate the detection of these harder instances and generate more accurate 3D bounding boxes.

Comparison for Occlusion Cases: We compare PointSee to the result-level fusion methods F-PointNet [19] and 3DSSD [20] under the occlusion scene. For a fair comparison, the 2D detection inputs for both F-PointNet and 3DSSD are provided by Yolov5x. As shown in Fig. 8, F-PointNet fails for the occluded distant objects, regardless of whether the 2OD network detects them or not. Our PointSee behaves well at sensing objects that are occluded at a distance, even if they are not detected by the 2OD network (see the 1st row) or not even labeled as ground-truth boxes (see the 2nd and 3rd rows). Table IX further verifies that PointSee outperforms F-PointNet on the occlusion scene. Specifically, PointSee is superior to F-PointNet for all three occlusion levels, i.e., Easy, Moderate, and Hard levels. Especially for both the Moderate and Hard levels, PointSee outperforms F-PointNet by even more than 10% in terms of $R_{11}$ and $R_{40}$, no matter whether it uses official 2D detection results or Yolov5x-based detection results.
Fig. 8. Qualitative comparisons of F-PointNet and PointSee for occlusion cases. The 1st column shows 2D detections output by Yolov5x. The 2nd column shows the performance of F-PointNet using 2D detections output by Yolov5x. The 3rd column shows the performance of 3DSSD with PointSee using 2D detections output by Yolov5x. The predicted boxes and ground-truth boxes are labeled in blue and red, respectively. For occluded distant objects, F-PointNet fails regardless of whether they are detected by the 2OD network, whereas our PointSee can even sense them when they are not detected by the 2OD network (see the 1st row) or not even labeled as ground-truth boxes (see the 2nd and 3rd rows).

Fig. 9. The performance of PointSee in terms of various 2OD networks on Car category of the KITTI val set. The performance of 3DSSD with PointSee is measured by 3D Average Precision calculated on 40 recall points. The performance of the 2OD network is measured by AP@.5 and Recall Rate respectively.

Macro-level and Micro-level Analysis: To better grasp how PointSee facilitates point-based 3OD networks, we perform in-depth analysis from both the macro and micro aspects. From the macro level, similar to previous result-level fusion 3OD methods, PointSee relies on 2OD networks to initially locate the area of objects in the image to help 3OD networks focus on the areas in point clouds where objects may exist to meet the purpose of reducing false detections and missing detections. From the micro level, the performance of point-based 3OD networks is largely limited by the quality of key points extraction (see “Point Recall” and “Positive Points” in Table VIII) during the downsampling of SA layers. As shown in Table VIII, using 3DSSD as a baseline, we validate the impact of “only hidden module” and PointSee on the quality of key points extraction at the last SA layer and the candidate generation layer. We find that the hidden module can help 3DSSD dramatically improve the quality of key points extraction, which also validates the idea that “PointSee improves detection performance via helping 3OD networks focus on detection areas” from a macro perspective. In addition, the seen module can further improve the quality of key...
point extraction based on the hidden module. This is because the 3D semantic segmentation sub-task in the seen module can better facilitate 3DSSD to learn the surface information of objects and extract more discriminative semantic features, which also contributes to the extraction of key points and the regression of 3D bounding boxes.

Dependency on the 2OD Network: We utilize five kinds of Yolov5 models and other three 2OD networks, i.e., YoloF [71], FCOS [41], RetinaNet [38], to evaluate the effect of different 2OD networks on PointSee. As illustrated in Fig. 9, AP@.50 ranges from 90.4% to 95.3% and Recall Rate ranges from 85.5% to 91.1%. We find that the main factor affecting the performance of PointSee is Recall Rate of the 2OD network rather than AP@.50. For example, Yolov5x provides the best 3OD results (86.03%), since it has the highest Recall Rate (91.1%) while not the highest AP@.50 (93.5%); RetinaNet leads to the worst 3OD results, since it has low Recall Rate (86.5%) but with the highest AP@.50 (95.3%). The above experimental results verify that PointSee performs better if assembled by 2OD networks with a higher Recall Rate.

F. Limitations

PointSee has some limitations: i) For objects incorrectly detected by both the 2OD and 3OD networks, PointSee will enhance the features of that region, possibly leading to false detection (see the 1st row of Fig. 10). ii) For objects that are not detected by both the 2OD and 3OD networks, PointSee may “blind” the 3OD network (see the 2nd row of Fig. 10). iii) PointSee has no clear improvements or even slight decrements for voxel-based encoders (see comparisons of VoxSeT [55] and its variant “VoxSeT+PointSee” in Table VII). In light of these limitations, our ongoing work will continually focus on the fusion of 2D and 3D detections, and extend the applicability of PointSee to voxel-based 3OD networks, such as BevFusion [72].

V. CONCLUSION

LiDARs, RGB, and depth cameras, as three different sensors, are widely equipped in autonomous driving vehicles and robotics. Images from the RGB camera can help understand point clouds from the LiDAR or depth camera for better 3D applications. In this paper, we propose an effective point semantic feature enhancement module, called PointSee. PointSee is lightweight and can be flexibly embedded into arbitrary 3D networks to improve their 3D detection performance. Experimental results on the outdoor KITTI and indoor SUN-RGBD and ScanNetV2 datasets validate that PointSee assists 3D networks to focus more on the regions where objects may be present in the point cloud, and the obtained representative semantic features can guide the point-based networks to better model potential objects. In the future, we will explore the lightweight 2D network trained jointly with arbitrary 3D networks to solve the occasional blindness of 3D networks.
Lipeng Gu received the MSc degree from Donghua University, in 2021. He is currently working toward the PhD degree with the Computer Science and Technology School, Nanjing University of Aeronautics and Astronautics. His research interests include deep learning, computer vision, and computer graphics.

Xuefeng Yan received the PhD degree from the Beijing Institute of Technology, in 2005. He is a professor with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA), China. He was the visiting scholar with Georgia State University, in 2008 and 2012. His research interests include intelligent computing, MBSE/complex system modeling, simulation and evaluation.

Peng Cui is a researcher with the Institute of Software and Simulation, Dalian Naval Academy, Dalian, China. His research interests include MBSE/complex system modeling, simulation and evaluation.

Lina Gong received the PhD degree in computer software and theory from the China University of Mining and Technology, Beijing, China, in 2020. She is currently a lecturer with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. She also studied as a visitor one year with the Software Analysis and Intelligence Lab (SAIL), School of Computing, Queen’s University, Kingston, ON, Canada. Her research interests include deep learning, software analysis, software testing, and mining software repositories.

Haoran Xie (Senior Member, IEEE) received the PhD degree in computer science from the City University of Hong Kong, Hong Kong SAR, and the EdD degree in digital learning from the University of Bristol, U.K. He is currently an associate professor with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong SAR. His research interests include artificial intelligence, Big Data, and educational technology. He has published 300 research publications, including 159 journal articles such as IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Affective Computing, IEEE Transactions on Circuits and Systems for Video Technology, and so on. He is the editor-in-chief of Natural Language Processing Journal, Computers & Education: Artificial Intelligence and Computers & Education: X Reality. He has been selected as the world top 2% scientists by Stanford University.

Fu Lee Wang (Senior Member, IEEE) received the BEng degree in computer engineering, the MPhil degree in computer science and information systems from the University of Hong Kong, Hong Kong, and the PhD degree in systems engineering and engineering management from the Chinese University of Hong Kong, Hong Kong. He is the dean with the School of Science and Technology, Hong Kong Metropolitan University, Hong Kong. He has more than 250 publications in international journals and conferences and led more than 20 competitive grants with a total greater than HK$20 million. His current research interests include educational technology, information retrieval, computer graphics, and bioinformatics. He is a fellow of BCS and HKIE and a senior member of ACM. He was the chair of the IEEE Hong Kong Section Computer Chapter and ACM Hong Kong Chapter.
Jing Qin (Senior Member, IEEE) is currently an associate professor with the School of Nursing, The Hong Kong Polytechnic University and is a key member with the Centre for Smart Health. His research focuses on creatively leveraging advanced virtual reality (VR) and artificial intelligence (AI) techniques in healthcare and medicine applications. He won the Hong Kong Medical and Health Device Industries Association Student Research Award for his PhD study on VR-based simulation systems for surgical training and planning. He won 3 best paper awards for his research on AI-driven medical image analysis and computer-assisted surgery, including one of the most prestigious awards in this field: MIA-MICCAI best paper award in 2017.

Mingqiang Wei (Senior Member, IEEE) received the PhD degree in computer science and engineering from the Chinese University of Hong Kong (CUHK), in 2014. He is a full professor with the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics (NUAA). Before joining NUAA, he served as an assistant professor with the Hefei University of Technology, and a postdoctoral fellow with CUHK. He was a recipient of the CUHK Young Scholar Thesis Awards in 2014. He is now an associate editor for ACM Transactions on Multimedia Computing, Communications, and Applications, The Visual Computer (TVC), Journal of Electronic Imaging, and a leading guest editor for IEEE Transactions on Multimedia, and TVC. He has published 140 research publications, including IEEE Transactions on Pattern Analysis and Machine Intelligence, SIGGRAPH, IEEE Transactions on Visualization and Computer Graphics, CVPR, ICCV, et al. His research interests focus on 3D vision, computer graphics, and deep learning.