Joint 3D Instance Segmentation and Object Detection for Autonomous Driving

Dingfu Zhou[1,2], Jin Fang[1,2], Xibin Song[1,2]*, Liu Liu[5,6], Junbo Yin[3,1,2], Yuchao Dai[4], Hongdong Li[5,6] and Ruigang Yang[1,2,7]

[1]Baidu Research [2]National Engineering Laboratory of Deep Learning Technology and Application, Beijing, China
[3]Beijing Institute of Technology, Beijing, China [4] Northwestern Polytechnical University, Xi'an, China
[5] Australian National University, Canberra, Australia [6] Australian Centre for Robotic Vision, Australia
[7] University of Kentucky, Kentucky, USA
{zhoudingfu, songxibin}@baidu.com

Abstract

Currently, in Autonomous Driving (AD), most of the 3D object detection frameworks (either anchor- or anchor-free-based) consider the detection as a Bounding Box (BBox) regression problem. However, this compact representation is not sufficient to explore all the information of the objects. To tackle this problem, we propose a simple but practical detection framework to jointly predict the 3D BBox and instance segmentation. For instance segmentation, we propose a Spatial Embeddings (SEs) strategy to assemble all foreground points into their corresponding object centers. Base on the SE results, the object proposals can be generated based on a simple clustering strategy. For each cluster, only one proposal is generated. Therefore, the Non-Maximum Suppression (NMS) process is no longer needed here. Finally, with our proposed instance-aware ROI pooling, the BBox is refined by a second-stage network. Experimental results on the public KITTI dataset show that the proposed SEs can significantly improve the instance segmentation results compared with other feature embedding-based method. Meanwhile, it also outperforms most of the 3D object detectors on the KITTI testing benchmark.

1. Introduction

Object detection, as a fundamental task in AD and robotics, has been studied a lot recently. The performance of object detection has been significantly improved based on the huge amounts of the labeled dataset [8], [38], [39] and some super strong baselines such as proposal-based [9], [35] and anchors-based methods [26], [34]. For easy generalization, objects are usually represented as a 2D BBox or 3D cuboid with several parameters *e.g.*, Bbox's center, dimension, and orientation *etc*.

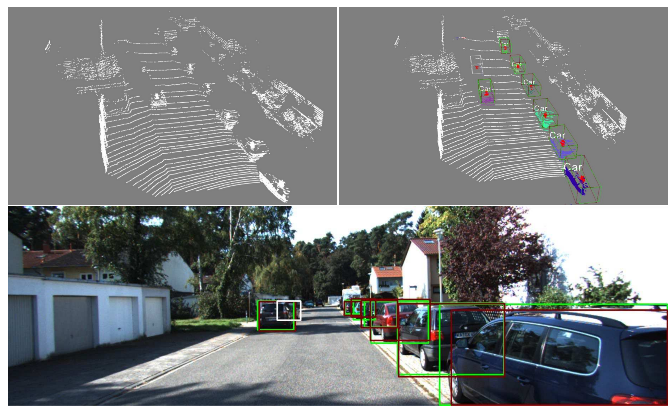

Figure 1: An example of 3D instance segmentation and object detection from LiDAR point cloud. The top sub-images illustrate the original point cloud and 3D detection results, where the ground truth and prediction results are drawn with green and other colors respectively. The red points in the top right sub-figure are predicted SEs (object centers) for foreground points. The projected 3D BBoxes in the 2D image is shown in the bottom. To be clear, the RGB image is only used for visualization here.

Many approaches have been proved that this simple representation is suitable for deep learning frameworks while it also has some limitations. For example, the shape information of the object has been discarded totally. Furthermore, for a certain BBox, some pixels from the background or other objects are inevitable to be included in it. In the case of occlusion, this situation becomes more serious. In addition, the BBox representation is not accurate enough to describe the exact location of the object. To well overcome this limitation, an additional instance mask has been employed for each BBox to eliminate the influence of other objects or background. Usually, the instance mask

*Corresponding author: Xibin Song

is binary to describe whether the pixel belongs to this object or not. With this kind of expression, each object can be clearly distinguished even they share a big overlap with each other. One straightforward idea for instance segmentation is to detect objects first and then predict the binary mask for each BBox one by one by considering it as a classification problem. Along this direction, various excellent works have been proposed and Mask-RCNN [13] is one of them.

However, Mask-RCNN is a two-stage framework and its performance highly depends on its first stage object detection results *e.g.*, Fast R-CNN [9] or Faster R-CNN [35]. Another popular branch is the proposal-free based method, which is mostly based on embedding loss functions or pixel affinity learning, such as [28]. Since these methods typically rely on dense-prediction networks, their generated instance masks can have a high resolution. In addition, proposal-free methods often report faster runtime than proposal-based ones, however, they fail to give comparable results with the two-stages based methods. Recently, with the rapid development of range sensors (*e.g.*, LiDAR, and RGB-D cameras) and also the requirement of AD, 3D point cloud-based deep learning has been mentioned frequently. Inspired by the 2D object detection framework, some one-stage or two-stages based 3D object detection frameworks have been designed, such as Frustum-Pointnet [31], VoxelNet [54], SECOND [46], PointPillars [18], Point RCNN [37], STD [48] and *etc*. Inspired by 2D instance segmentation, [41] and [17] proposed to embed the instance information in feature space and then separate them with a mean-shift clustering strategy.

3D object detection has been well studied for both indoor [30] and outdoor scenarios [52]. However, most of the 3D instance segmentation approaches are designed for indoor environment, few of them can be used directly in the outdoor AD scenario. In [19], Leibe et al proposed to obtain the object categorization and segmentation simultaneous by using a so-called Implicit Shape Model, which can integrate the two tasks into a common probabilistic framework. First, some possible local patches have been extracted and matched with an off-the-shelf Codebook. Then each activated patch casts votes for possible positions of the object center. Finally, the mean-shift clustering technique is employed for finding the correct object location over the voting space.

Inspired by [19], we propose to jointly detect and segment 3D objects from the point cloud simultaneously. Similarly, for each foreground (FG) point, the SEs have been learned from a deep neural network, which encodes the object information it belongs to, such as center, dimension, and orientation, etc. Based on the SEs, points from FG objects can be pulled into their BBoxes' center respectively. With the learned SEs, instance segmentation and ROI (region of interest) proposals can be easily generated with a clustering strategy. Fig. 2 illustrates an example of the predicted SEs for FG objects, where all the learned SE vectors start from the points and point to the object's center.

In this work, we proposed to solve the object detection and instance segmentation jointly in a unified framework to boost each other performance. By doing this, both the local instance and the global shape information can be considered. Generally, the contributions of this paper can be summarized as

- A unified end-to-end trainable framework has been designed which can obtain 3D BBox and instance segmentation jointly for the AD scenario.
- Compared with the commonly used feature embedding in a 2D image, we proposed to use SE by considering both the global BBox and local point information together.
- The experimental results on the public KITTI dataset have proved the effectiveness and efficiency compared with other state-of-the-art approaches.

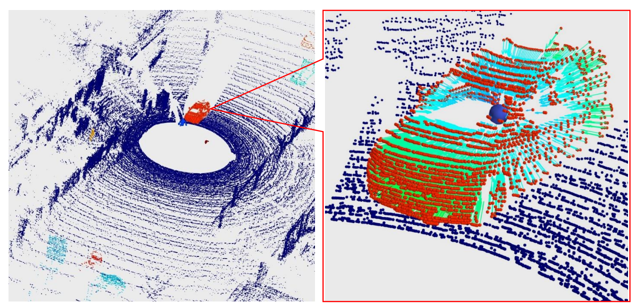

Figure 2: An illustration of FG semantic segmentation and SE for the point cloud. The right sub-fig is the SE result of a car. Colored points are semantic results and the cyan arrows are the SE vectors.

2. Related Work

Image-based Object Detection and Instance Segmentation: 2D object detection [5] and instance segmentation [15] have attracted many researchers' attention recently and leading to various top-performing methods. Both object detection and instance segmentation have achieved rapidly improvement on different public benchmarks recently based on some powerful base-line systems, such as Fast/Faster RCNN and Mask-RCNN *etc*. Due to the limitation of paper length, we only introduce the recently proposed instance segmentation frameworks here and we refer readers to the recent review paper [50] for more description of object detection.

Currently, the 2D instance segmentation performances lead mostly by two-stages based methods and Mask-RCNN is considered commonly as the pioneering work of them. This kind of approach is based on detect-and-segment in which a modern object detector is applied to detect the bounding box of the foreground object first and then a binary mask is predicted for each object one by one. Based on this superpower baseline, many variant versions [2] have been proposed successively. While this method provides good results in terms of accuracy, it generates low-resolution masks which are not always desirable (e.g. for photo-editing applications) and operates at a low frame rate, making it impractical for real-time applications such as AD.

3D Object Detection and Instance Segmentation: 3D object detection in traffic scenario [53] become more and more popular with the development of range sensor and the AD techniques [12]. Inspired by image-based object detection, the point cloud is first projected into 2D (*e.g.* bird-eye-view [3] or front-view [44]) to obtain the 2D detection result and then re-project the 2D BBox into 3D to get the final results. Another representative direction for 3D object detection is volumetric convolutional based methods due to the rapid development of the graphics processing resources. Voxel-net [54] is a pioneer work to detect the 3D objects directly with 3D convolutional by representing the LiDAR point cloud with voxels. Based on the framework of Voxelnet, two variant methods, SECOND [46] and PointPillars [18] have been proposed. Different from the two directions mentioned above, PointNet [32] is another useful technique for point cloud feature extraction. Along this direction, several state-of-the-art methods have been proposed for 3D object detection [31, 37].

SGPN [40] is the first work proposed to do the instance segmentation for a 3D point cloud in the indoor environment. In this work, a similarity matrix has been build for each point based on the extracted PointNet [32] features. Then a classifier is trained to classify whether two points belong to the same object or not. Different from SGPN, the newly proposed GSPN [49] is a generative shape proposal network, which generates the 3D model of the object based on its prior shape information and observed 3D point cloud. MASC [23] relies on the superior performance of the SparseConvNet [10] architecture and combines it with an instance affinity score that is estimated across multiple scales. Metric learning has also been employed for instance segmentation in 3D. In [41], during the feature embedding process, the author proposed to fuse both the features for semantic and instance segmentation together. While in [17], the direction information is also applied for the feature embedding process. Finally, the instances are clustered by mean-shift in the embedding features space.

Deep Learning on Point Clouds: different from the 2D image, the point cloud is un-organized and the traditional CNN can not be applied directly for feature extraction. In order to take advantage of classic CNNs, [4, 44] proposed to project the point cloud into front-view or bird-eye-view first and then all the 2D CNNs designed for 2D images can be applied directly. Another popular representation for point cloud data is voxelized volumes [54, 27, 36]. Based on this operation, all the points are well organized in 3D coordinate, then the 3D CNNs can be employed for feature extraction. A drawback of these representations is the memory issue, due to the sparsity of point clouds. To handle this, sparse convolution has been proposed, in which the convolution only happens for the valid voxels. Base on this operation [46, 10], both the speed and memory issues have been solved. Another direction is to process the point cloud directly without any transformation. The pioneering work of this work is PointNet [32] which applied MLPs to extract point-wise features directly. Following this direction, many frameworks have been proposed for classification [33], object detection [37], semantic segmentation [14, 29] and other applications [25, 24, 7].

3. Proposed Approach

We aim at solving the 3D instance segmentation and detection problem jointly within a given single frame of the point cloud in the AD scenario. Specifically, the point cloud is scanned by a widely used 64-lines Velodyne LiDAR sensor. By the combination of the instance segmentation and detection, we can achieve the following benefits: 1) the instance mask-based representation is good at catching the local geometric information point-wisely, 2) the BBox based object representation can help to exploit the global shape information of the whole object.

3.1. Overview

An overview of our method is described in Fig. 3. Generally, the proposed approach can be divided into two parts: SE learning-based object proposal and the local BBoxes refinement. First of all, point-wise features can be obtained by employing a backbone network *e.g.*, PointNet++ [33]. With the sampling and grouping operations, both the local features and global context information has been extracted. Following the backbone network, there are two branches for semantic segmentation and instance-aware SE, which are encoded as objects' center and dimension, etc. For each point, the ground truth of semantic class and the information of BBox's it belongs to can be easily generated. Therefore, the first stage of the network can be trained by supervision signals. Based on the SE results, a deep clustering layer is employed for generating the instance segmentation. At the same time, for each cluster, a BBox is also generated. Then, for each proposal, a refine network (*e.g.* PointNet [32]) is applied for refining the 3D BBox of each proposal. Here, all proposals share the same network parameters. In order for more generations, we transform the proposals into a local normalized coordinate system. Finally, the refine network

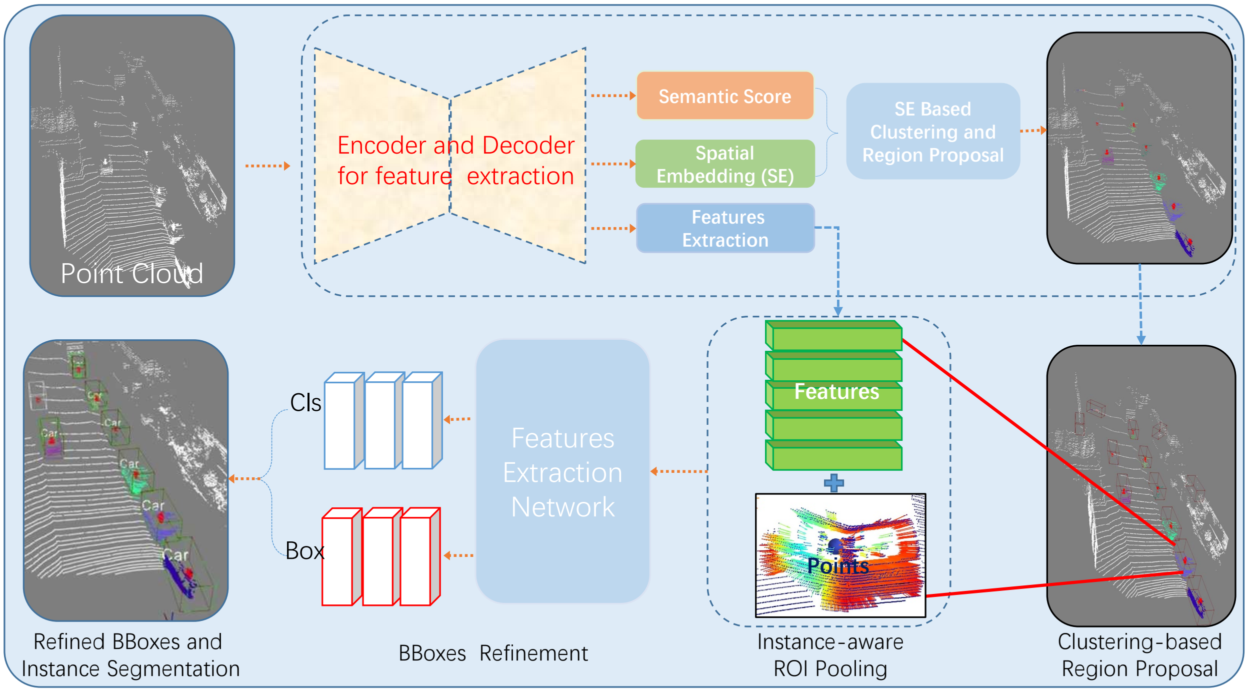

Figure 3: Joint instance segmentation and 3D BBoxes regression framework is a two-stages network that can be divided into two parts as pixel-wise semantic segmentation, SE, clustering-based region proposal and the second stage BBox refinement. Finally, 3D BBox together with an instance mask is generated for each object.

outputs the refined 3D BBoxes and instance masks.

3.2. Instance-aware SE

Inspired by the 2D instance segmentation [28], many works [41] have been proposed to segment objects in the feature space (rather than the spatial space directly) by using a discriminative loss function [17]. By using this kind of loss, features belong to the same instance are pulled closer and those belonging to different instances are pushed far away. However, the instance label information can not be explicitly integrated into the loss function directly and this kind of loss is encoded in feature space by using several hyper-parameters [6].

Although this kind method achieved impressive performance for the indoor environment, few methods have been proposed for the AD scenario. Before the introduction of our approach, we analyze the difference of instance segmentation between the 2D and 3D. Scale [51], spatial layout ambiguity and occlusion are three main problems in 2D image space. They have seriously effected the performances of object detection and instance segmentation. While these problems don't exist anymore in the 3D point cloud. On the contrary, objects become separable in the spatial space. However, the direct use of the clustering method from the point cloud yields unsatisfied results. Therefore, for easy clustering or segmentation, a well designed intermediate procedure is required to explore the point's latent properties such as semantic class, instance label, and the object's information that the point belongs to.

Point cloud feature extraction: for extracting point-wise features for point cloud, we employ the commonly used PointNet++ network with multi-scale sampling and grouping operations as our backbone networks. Particularly, the designed framework is backbone independent and it can be replaced by other structures such as PointConv [45], EdgeConv [42] or sparse convolution network [11] *etc*. Based on the extracted features, we would like to predict the object information as below.

Semantic information: with the point-wise features as input, one segmentation branch is designed for semantic classes prediction. Thanks to the multi-scale sampling and grouping strategies, both the local structure and global context information has been encoded in each point-wise feature vector. And this is useful to handle objects with different sizes. To well tackle the classes imbalance problem in the classification, focal loss [21] is employed here as

$$\mathbf{L}_{\text{cls}} = -\sum_{i=1}^{C}(y_i \log(p_i)(1-p_i)^{\gamma}\alpha_i + (1-y_i)\log(1-p_i)(p_i)^{\gamma}(1-\alpha_i)), \tag{1}$$

where C denotes the number of classes; y_i equals 1 if the ground-truth belongs to the i_{th} class and 0 otherwise; p_i is the predicted probability for the i_{th} class; $\gamma \in (0, +\infty)$ is a focusing parameter; $\alpha_i \in [0, 1]$ is a weighting parameter for the i_{th} class.

Object information: in our intuition, as long as all the points belong to the same object pulled to its physical center, then they can be separated into different instances directly. Therefore, we take the object center (c_x, c_y, c_z) as one important information of the SEs. Instead of regressing the center value directly, we define the offset between individual point and object center as our regression target label. For each FG point $\mathbf{p}^i = (p_x^i, p_y^i, p_z^i)$, the ground truth label is defined as

$$c_{\text{offset}}^i = (p_x^i - c_x^k, p_y^i - c_y^k, p_z^i - c_z^k)^T, \tag{2}$$

where (c_x^k, c_y^k, c_z^k) represent is the object center of instance k. Traditionally, the embedding of object center is enough for 3D instance segmentation. For the task of object detection, other information such as the BBox dimension (l, w, h) and orientation angle θ (head direction of the object) are also required. For these parameters, we directly assign the ground truth box information to the corresponding points. During the training, all the parameters are predicted point-wisely from the network, however, only the FG points are contributed for the final loss computation.

3.3. Clustering-based Proposal Generation

Based on the predicted SEs results, all the FG points are aggregated to the centroids of their corresponding objects. We show an example of predicted SE in the top right corner of Fig. 3, where we represent the pulled points (the original location plus the predicted offset) with red color. From this example, we can obviously find that these red points can be separated via a simple clustering algorithm (*i.e*. K-means [1]) easily. An example of the instance segmentation results is also shown in the bottom right corner of Fig. 3, where each instance has been displayed with different colors. After the clustering, a mean BBox is also generated for each instance by averaging the top k predictions (e.g., $k = 5$). In addition, we will keep the clustering ids of points and BBox for the next stage Region of Interesting (ROI) pooling.

3.4. BBox Refinement

Although the BBox prediction from the first-stage is very precise, there still has some space for improvement. Similar to other two-stage based methods, we directly perform PointNet++ network based on interior points inside the object proposal. Furthermore, an instance-aware ROI polling strategy is proposed to compensate for the inaccuracy of BBox in the proposal stage. Specifically, two things have been done in this strategy: first, points belong to one cluster will be used for the second stage refinement even some of them is not inside of the BBox. Second, some FG points even they are inside the BBox will be removed out if they share different cluster-ids with the BBox. To well utilize the local information, we transform the proposal to a local normalized coordinate system. For each ROI, M points together with features extracted in the first stage are randomly selected as the inputs for the refinement network.

3.5. Multi-task Loss

A multi-task loss is employed for training our network. Three kinds of loss have been used here including semantic segmentation loss, SE loss, and the 3D BBox regression loss. In addition, some hype-parameters have been used here to balance their contributions. For the first

$$\mathbf{L} = \mathbf{L}_{\text{sem-cls}} + \mathbf{L}_{\text{SE}} + \mathbf{L}_{\text{reg}},$$

where the semantic segmentation loss is defined as in Eq. (1) and the others will be described detailedly as below.

SE loss: during the training, supervision signal is generated directly for each FG point and the loss function is formulated as

$$\mathbf{L}_{\text{SE}} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{N_c}\sum_{i \in \text{ins}_c}^{N_c} \mathbf{l}_{\text{offset}}^i + \mathbf{l}_{\text{size}}^i + \mathbf{l}_{\theta}^i,$$

where $\mathbf{l}_{\text{offset}}$, $\mathbf{l}_{\text{size}}$ and $\mathbf{l}_{\theta}$ are the smooth-l_1 losses for offset, BBox dimension and orientation angle respectively. In addition, the loss is also normalized by instance number N and points number N_c inside the instance c individually.

BBox regression loss: each proposal is encoded as a 7-dimension vector as object center (c_x, c_y, c_z), object dimension (h, w, l) and head direction angle θ. The rotated 3D intersection-over-union [52] loss is employed here as

$$\mathbf{L}_{\text{reg}} = 1 - \mathbf{IoU}(\mathbf{B}_{\text{g}}, \mathbf{B}_{\text{d}}) = \frac{\mathbf{B}_{\text{g}} \cap \mathbf{B}_{\text{d}}}{\mathbf{B}_{\text{g}} \cup \mathbf{B}_{\text{d}}}, \tag{3}$$

where the $\mathbf{B}_{\text{d}}$ and $\mathbf{B}_{\text{g}}$ represent the predicted and ground truth BBoxes respectively.

4. Experimental Results

In this section, we describe the details of our experimental results, including the implementation settings, instance segmentation and 3D object detection on the public KITTI dataset.

4.1. Implementation Details

Input Data: for KITTI, we randomly select 16K points per frame. In particularly, only points within a constrained range are considered *e.g*., [-40, 40], [-1, 3], [0, 70.4] for x, y and z respectively. For these frames whose points are less than 16K, we just randomly select the existing points repeatedly.

Network architecture: to build the backbone network, the multi-scale grouping is employed four times for point feature extraction. In each scale, we randomly sample $(4096, 1024, 256, 64)$ point and PointNet is applied for extracting features of each scale. For the grouping layer, a ball query search is applied for finding the neighbor points within a certain radius. We set different radius as $(0.1, 0.4, 0.8, 1.6)$ for four different scales. After the backbone feature extraction, a 512 dimension feature vector is assigned for each point. The semantic segmentation branch is realized through a multi-layer perceptron with full-connected layers output sizes of 256, 128, C (class probability). Similarly, the SE branch is also realized with full-connected layers output sizes of 256, 128 and 7.

For the BBox refine network, the encoder part of PointNet++ is employed here too. For each proposal, 512 points are randomly selected for feature extraction. Different from the backbone network, we employ only three scales for sampling and grouping with the up-sampling operation. After extracted the features for each proposal, two branches are followed to give the refined BBox parameters and an "Objectness" score to classify the proposal as a positive object or just background points.

4.2. Dataset

To our knowledge, there is not public 3D instance segmentation dataset in the AD scenario. Therefore, we evaluate our framework for both 3D instance segmentation and object detection on the public KITTI dataset. *KITTI 3D Object Detection Data:* the whole data has been divided into training and testing two subsets, which consist of 7481 and 7518 frames respectively. Since the ground truth for the testing set is not available, we subdivide the training data into a train and val set as described in [54, 46]. Finally, we obtained 3712 data samples for training and 3769 frames for validation. On the KITTI benchmark, the objects have been categorized into "easy", "moderate" and "hard" based on their height in the image and occlusion ratio, etc. For each frame, both the left and right camera images and the LiDAR point cloud have been provided, while only the point cloud has been used for our object detection here and the RGB image is only used for visualization purposes.

4.3. 3D Instance Segmentation

To verify the effectiveness of our proposed SE strategy, we compare it with another state-of-the-art feature embedding based method [17]. To be clear, we have not implemented their methods here and we just replace the SE loss with feature embedding and directional losses based on our framework and keep other modules unchanged. Finally, a 7-dimension feature is taken for the next stage clustering, such as the commonly used mean-shift technique.

Instance Segmentation Data: in KITTI, 3D BBox annotations have been provided for three categories objects *e.g.*, car, pedestrian and cyclist. We simply generate the instance mask for each object by extracting the points inside each BBox. An example of 3D instance ground truth has been shown in Fig .4, where different colors represent different objects at the bottom of this image.

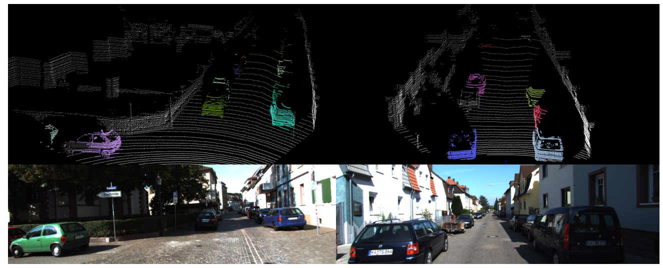

Figure 4: A generated instance segmentation ground truth based on KITTI 3D BBox annotation. Different instances have been drawn with different colors. The RGB image is only used for visualization purpose

Methods	$\mathbf{AP}_{50}$	$\mathbf{AP}_{75}$	$\mathbf{AP}_{90}$	**Mean AP**
Feature Embedding	64.75	43.25	8.53	40.78
Spatial Embedding	**74.83**	**49.40**	**12.93**	**47.15**

Table 1: Evaluation for instance segmentation on KITTI 3D object detection dataset.

For instance segmentation, we compute the mask AP at different thresholds *e.g.* ($\mathbf{AP}_{50}$, $\mathbf{AP}_{75}$, $\mathbf{AP}_{90}$). Finally, we also compute the mean mask AP which proposed in coco challenges [22]. Specifically, the thresholds are set as $[0.5 : .05 : 0.95]$. The evaluation results for 3D instance segmentation are given in Tab. 1. From the table, we can clearly see that the proposed SEs method significantly outperforms the features embedding based approach.

4.4. 3D Object Detection On KITTI

Evaluation Protocol: we employ evaluation metrics on KITTI [8] to report our results here. In [8], all the objects have been divided into "Easy", "Moderate" and "Hard" categories based on their distances and occlusion ratios.

4.4.1 Evaluation on test split

In this subsection, we compare our proposed approach on the public 3D object detection benchmark. Tab. 2 gives the evaluation results on the KITTI testing subset. We achieved the results of testing split by submitting predictions on KITTI's on-line evaluation server and the performance of other methods are also obtained from the benchmark respectively. Compared to other methods with publications, the proposed method achieved comparable results with other state-of-the-art methods on both 3D object detection and Bird-eye-View (BEV) evaluation. From the ta-

Methods	Modality	AP_{3D}**70(%)**			AP_{BEV}**70(%)**		
		Mod	Easy	Hard	Mod	Easy	Hard
AVOD-FPN [16]	LiDAR+Mono	71.76	83.07	65.73	84.82	91.17	79.62
F-PointNet [31]	LiDAR+Mono	69.79	82.19	60.59	84.67	91.17	74.77
F-ConvNet [43]	LiDAR+Mono	76.39	87.36	66.69	85.84	91.51	76.11
UberATG-MMF [20]	LiDAR+Mono	77.43	**88.40**	70.22	88.21	93.67	81.99
VoxelNet [54]	LiDAR	65.11	77.47	57.73	79.26	89.35	77.39
PointPillars [18]	LiDAR	74.31	82.58	68.99	86.56	90.07	82.81
SECOND [46]	LiDAR	75.96	84.65	68.71	86.37	91.81	81.04
3D IoU Loss [52]	LiDAR	76.50	86.16	71.39	86.22	91.36	81.20
PointRCNN [48]	LiDAR	75.64	86.96	70.70	87.39	92.13	82.72
STD [48]	LiDAR	**79.71**	87.95	**75.09**	**89.19**	**94.74**	**86.42**
Proposed Method	LiDAR	78.96	87.74	74.30	88.10	94.11	83.43

Table 2: Comparison with other public methods on the KITTI testing sever for 3D "Car" detection. For easy understanding, we have highlighted the top two numbers in bold and italic for each column and the second best is shown in blue. All the numbers are the higher the better.

Methods	Modality	**AP70**		
		Easy	Mod	Hard
MV3D [4]	LiDAR+Mono	71.29	62.68	56.56
F-PointNet [31]	LiDAR+Mono	83.76	70.92	63.65
AVOD-FPN [16]	LiDAR+Mono	84.41	74.44	68.65
IPOD [47]	LiDAR+Mono	84.10	76.40	75.30
ContFusion [20]	LiDAR+Mono	86.33	73.25	67.81
F-ConvNet [43]	LiDAR+Mono	89.02	78.80	77.09
VoxelNet [54]	LiDAR	81.97	65.46	62.85
PointPillars [18]	LiDAR	87.29	76.99	70.84
PointRCNN [37]	LiDAR	88.88	78.63	77.38
SECOND [46]	LiDAR	88.15	78.33	77.25
3D IoU Loss [52]	LiDAR	89.16	78.99	77.78
STD [48]	LiDAR	**89.70**	**79.80**	**79.30**
Proposed Method	LiDAR	89.50	79.21	78.16

Table 3: Comparison with other public methods on the KITTI validation dataset for 3D "Car" detection. For easy understanding, we have highlighted the top two numbers in bold and italic for each column and the second best is shown in blue. All the numbers are the higher the better.

ble, we can find that the proposed framework outperforms all the pure point cloud-based and most of the fusion-based method (camera and lidar fusion) for both 2D BEV and 3D among all the categories (*e.g.* easy, moderate and hard). In particular, we outperform other methods on 3D object detection for both moderate and hard categories with a big margin.

4.4.2 Evaluation on validation split

We also evaluate the proposed framework on the KITTI validation dataset. For this split, all the ground truth labels have been provided. First of all, Tab. 3 and Tab. 4 give the comparison results on validation dataset for 2D BEV and 3D object detection. We have listed nearly all the top results with publications here including: multi-modalities fusion-based [4, 31, 16, 20, 47, 43], one-stage- [46, 54, 18] and two-stage-based [37] approaches. Among all methods, the improved method achieved the second best results for all three categories on both 2D and 3D. Furthermore, it even performs much better than other fusion-based and two-stage-based methods.

In addition, we illustrate some qualitative detection re-

Methods	Modality	AP_{BEV}**70**		
		Easy	Mod	Hard
MV3D [4]	LiDAR+Mono	86.55	78.10	76.67
F-PointNet [31]	LiDAR+Mono	88.16	84.02	76.44
ContFusion [20]	LiDAR+Mono	95.44	87.34	82.42
VoxelNet [54]	LiDAR	89.60	84.81	78.57
SECOND [46]	LiDAR	89.96	87.07	79.66
PointPillars [18]	LiDAR	90.07	87.06	83.81
Proposed Method	LiDAR	**90.23**	**87.53**	**86.45**

Table 4: Comparison with other methods on the KITTI validation dataset for Bird-Eye-View (BEV) detection.For easy understanding, we have highlighted the top two numbers in bold and italic for each column and the second best is shown in blue. All the numbers are the higher the better.

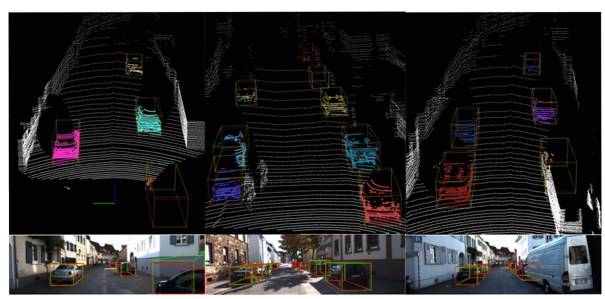

Figure 5: Three examples of joint instance segmentation and 3D object detection on the KITTI benchmark. The BBoxes in green is ground truth and these in other colors are prediction results. The foreground points in different colors represent different instances. The bottom images are only used for visualization.

sults on the validation split in Fig. 5. In this figure, different instances have been randomly highlighted with different colors on the point cloud. In addition, the predicted 3D BBoxes are drawn on both 2D image and 3D point cloud, where green and red represent the ground truth and predictions respectively.

4.5. Ablation Study

In this section, we give the ablation study for the proposed approach. We conduct all the evaluations on the "val" dataset for the Car category because the training data for "Car" is relatively large.

4.5.1 Spatial Embedding

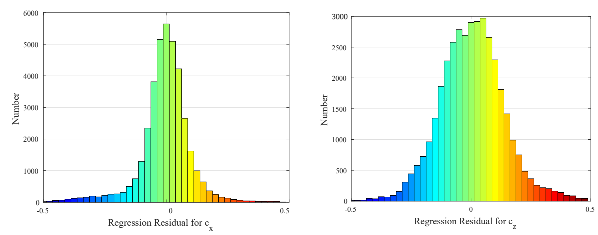

Figure 6: Spatial embedding error distribution for different elements. X-axis represents the prediction error and Y-axis represents the number of foreground points.

Spatial embedding is a very crucial step in our method. During the training, this embedding process has been supervised with ground truth labels. We calculate the error distribution between the prediction value and ground truth on the validation dataset. Fig. 6 illustrates the error distribution for c_x and c_z, which represent object center in x and z axis respectively. From this figure, we can find that all prediction error is close to 0 and nearly follows a Gaussian distribution with small variance values *e.g.*, $\sigma_{c_x} = 0.11m$ and $\sigma_{c_z} = 0.14m$. This indicates that the proposed spatial embedding can effectively pull all the foreground points into the object center.

4.5.2 Region Proposal

Usually, in order to increase detection performance, more than one RoI (Region of Interest) has been generated for each object. PointRCNN [37] generates 100 proposals each frame for KITTI dataset during testing. However, most of these proposals are redundant BBoxes because the average object number is only about 10 on KITTI. In addition, the recall of proposals is loosely related to the final 3D object detection performance. Furthermore, the inference time will increase rapidly with the increase of RoI number. The comparison results in Tab. 5 show that with only a few numbers of ROIs, our proposed approach can obtain a very high recall rate.

RoIs	Recall(IoU = 0.5)		Recall (IoU = 0.7)	
	PointRCNN	Ours	PointRCNN	Ours
10	61.02	**84.11**	29.87	**67.27**
20	77.89	**87.86**	32.55	**69.11**
30	85.89	**88.04**	32.76	**69.14**
40	**95.55**	92.09	40.04	**69.14**
50	**96.01**	94.12	40.28	**69.14**

Table 5: Comparison the recall of proposal generation (with different RoIs) to PointRCNN with 3D IoU threshold of 0.5 and 0.7 for Car on val split.

4.5.3 Inference Time

Compared with other proposal based methods, such as PointRCNN [37] which generates 100 proposals each frame for KITTI data during the inference. Taking KITTI dataset as an example, the average number of objects in each frame is about 8. For our proposed framework, 20 proposals are sufficient to recall most of the objects as shown in Tab. 5. Our experimental result shows that the proposed framework runs 4 times faster than PointRCNN in the BBox refinement stage. Currently, the proposed approach can achieve almost real-time on a single NVIDIA Tesla P40 GPU on the KITTI point cloud with only 90^o field of view.

5. Conclusion and Future Works

In this paper, we proposed a unified framework for joint 3D object detection and instance segmentation. In particular, we proposed a spatial embedding module to pull all the points which belong to the same object together and it works well in the real autonomous driving scenario. The proposed framework can obtain state-of-the-art performance with only a few region proposals. This is very important for real-time perception in real-world applications. Currently, we use the PointNet++ as be backbone network which is the bottleneck for the real-time detection rate. In the future, we would like to design a more efficient backbone network to make the system run in real-time for object detection in 360 degrees view-point.

Acknowledgement Yuchao Dai's research was supported in part by the Natural Science Foundation of China grants (61871325, 61420106007, 61671387), and the National Key Research and Development Program of China under Grant 2018AAA0102803. Hongdong Li's research is funded in part by the ARC Centre of Excellence for Robotics Vision (CE140100016), ARC-Discovery (DP 190102261) and ARC-LIEF (190100080) grants, as well as a research grant from Baidu Research, Robotics and Autonomous Driving Laboratory (RAL). The authors from ANU gratefully acknowledge the GPUs donated by NVIDIA Corporation. We thank all anonymous reviewers and ACs for their constructive comments.

References

[1] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 132–149, 2018. 5

[2] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2018. 3

[3] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016. 3

[4] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017. 3, 7

[5] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2

[6] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 7–9, 2017. 4

[7] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented lidar simulator for autonomous driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938, 2020. 3

[8] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 1, 6

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 2

[10] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018. 3

[11] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9224–9232, 2018. 4

[12] Richard Hartley and Hongdong Li. An efficient hidden variable approach to minimal-case camera motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 34(12):2303–2314, 2012. 3

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2

[14] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018. 3

[15] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The apolloscape dataset for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 954–960, 2018. 2

[16] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018. 7

[17] Jean Lahoud, Bernard Ghanem, Marc Pollefeys, and Martin R Oswald. 3d instance segmentation via multi-task metric learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9256–9266, 2019. 2, 3, 4, 6

[18] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 2, 3, 7

[19] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an implicit shape model. In *Workshop on statistical learning in computer vision, ECCV*, volume 2, page 7, 2004. 2

[20] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018. 7

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 4

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 6

[23] Chen Liu and Yasutaka Furukawa. Masc: Multi-scale affinity with sparse convolution for 3d instance segmentation. *arXiv preprint arXiv:1902.04478*, 2019. 3

[24] Liu Liu, Dylan Campbell, Hongdong Li, Dingfu Zhou, Xibin Song, and Ruigang Yang. Learning 2d-3d correspondences to solve the blind perspective-n-point problem. *arXiv preprint arXiv:2003.06752*, 2020. 3

[25] Liu Liu, Hongdong Li, and Yuchao Dai. Efficient global 2d-3d matching for camera localization in a large-scale 3d map. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2372–2381, 2017. 3

[26] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C

Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[27] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928. IEEE, 2015. 3

[28] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019. 2, 4

[29] Anh Viet Phan, Minh Le Nguyen, Yen Lam Hoang Nguyen, and Lam Thu Bui. Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, 108:533–543, 2018. 3

[30] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9277–9286, 2019. 2

[31] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 2, 3, 7

[32] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017. 3

[33] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 3

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 2

[36] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017. 3

[37] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *CVPR*, 2019. 2, 3, 7, 8

[38] Xibin Song, Peng Wang, Dingfu Zhou, Rui Zhu, Chenye Guan, Yuchao Dai, Hao Su, Hongdong Li, and Ruigang Yang. Apollocar3d: A large 3d car instance understanding benchmark for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5452–5462, 2019. 1

[39] Peng Wang, Xinyu Huang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 1

[40] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2569–2578, 2018. 3

[41] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. Associatively segmenting instances and semantics in point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4096–4105, 2019. 2, 3, 4

[42] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):146, 2019. 4

[43] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1742–1749. IEEE, 2019. 7

[44] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 3

[45] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. 4

[46] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2, 3, 6, 7

[47] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Ipod: Intensive point-based object detector for point cloud. *arXiv preprint arXiv:1812.05276*, 2018. 7

[48] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1951–1960, 2019. 2, 7

[49] Li Yi, Wang Zhao, He Wang, Minhyuk Sung, and Leonidas J Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3947–3956, 2019. 3

[50] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 2019. 2

[51] Dingfu Zhou, Yuchao Dai, and Hongdong Li. Reliable scale estimation and correction for monocular visual odometry. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 490–495. IEEE, 2016. 4

[52] Dingfu Zhou, Jin Fang, Xibin Song, Chenye Guan, Junbo Yin, Yuchao Dai, and Ruigang Yang. Iou loss for 2d/3d object detection. In *2019 International Conference on 3D Vision (3DV)*, pages 85–94. IEEE, 2019. 2, 5, 7

[53] Dingfu Zhou, Vincent Frémont, Benjamin Quost, Yuchao Dai, and Hongdong Li. Moving object detection and segmentation in urban environments from a moving platform. *Image and Vision Computing*, 68:76–87, 2017. 3

[54] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2, 3, 6, 7