3D Object Detection from Point Cloud

Alexander Arzhanov
CS230 Deep Learning (Fall 2019)
aarz@stanford.edu

1. Introduction

The recent progress in deep learning helped to significantly improve such computer vision tasks as object detection [15, 14], as well as semantic and instance segmentation [7, 9]. Whereas much of the effort in scientific community was focused on the 2D scene understanding, an accurate 3D perception is indispensable for remote sensing in such applications as robotic object manipulation, augmented reality, and autonomous vehicles. In particular, majority of the modern autonomous driving systems heavily rely on LiDAR sensors for object tracking and collision avoidance. One of the major challenges in developing a LiDAR-based 3D object detection system stems from the fact that the point cloud data is irregular, unordered, and usually sparse, which makes direct application of typical convolution-based methods difficult [11]. Thus most of the proposed methods either first voxelize the point clouds [20], or project them into a bird's eye view [19]. These approaches, however, can be very computationally expensive, or suffer from information loss during quantization.

In this project we analyze VoteNet [10] – the recently proposed end-to-end deep learning network that leverages the Hough voting algorithm [8] to detect 3D objects directly from the raw point cloud data. The model achieved state-of-the-art results in 3D object detection tasks on two large datasets with interior 3D scans, ScanNet [5] and SUN RGB-D [18], relying solely of point cloud data. The VoteNet paper is also a Best Paper Award Nominee in ICCV 2019 [1].

To our best knowledge, VoteNet has not yet been tested on any outdoor point cloud data. To this end, we first test and then adapt the VoteNet model to KITTI [6] dataset of outdoor LiDAR point cloud data.

2. VoteNet Architecture

The VoteNet architecture, shown in Fig. 1, can be decomposed into two subnetworks: the *Vote Proposal Network* (VPN) that consumes the raw point cloud and produces the virtual vote points; and the *Object Proposal and Classification Network* (OPCN) that operates on the vote points to propose and classify objects in a 3D scene.

2.1. Vote Proposal Network

The raw point cloud comprised of N points is input to the VPN. Each point is generally characterized by 3 spatial coordinates in Euclidean space and C_0 optional features (*e.g.* height over ground plane, RGB-triplet, LiDAR reflectance). The VPN network learns the semantic signature of the cloud at different contextual scales and subsamples a number of seed points enriched with the extracted deep features. These points are then used to suggest a set of M virtual points with 3 spatial coordinates and C_L learned features that lie much closer to the centroids of objects to be detected. Then the virtual vote points are grouped and aggregated into the corresponding cluster signature vectors, which are used to generate suggestions for the 3D bounding boxes along with the class labels.

PointNet layer. The PointNet layer, first introduced in Ref. [12], takes an unordered point set $P = \{p_i\}_{i=1}^{N}$ with $p_i = [x_i; f_i] \in \mathbb{R}^{(3+C_l)}$ as an input, where x_i being point coordinates in Euclidean space with f_i any additional semantic features, and learns a symmetric set function g of the following form:

$$g(\{p_i\}) = \gamma \circ \underset{p_i \in P}{\mathrm{MAX}}(\{h(p_i)\}) \tag{1}$$

that maps P to a single vector, where $h : \mathbb{R}^{C_l} \to \mathbb{R}^{D_l}$ and $\gamma : \mathbb{R}^{D_l} \to \mathbb{R}^{C_{l+1}}$ are continuous functions, implemented with multi-layer perceptron (MLP) networks, and MAX is a channel-wise max-pooling operator that collapses N input vectors into a single one, *i.e.* $\mathbb{R}^{D_l} \times \cdots \times \mathbb{R}^{D_l} \to \mathbb{R}^{D_l}$. Effectively, the PointNet layer learns to extract a subset of the most informative keypoints and encode their semantic information as an aggregated cloud signature feature vector.

The set-abstraction module. Since a single PointNet layer aggregates information of the whole point set, it can not capture local spatial structures induced by the metric space [13]. In order to encode the fine geometric patterns, the PointNet layers are recursively applied on overlapping local regions of progressively larger scales that are partitioned by a distance metric. The idea is similar to the 2D CNN networks, that are designed to learn the hierarchical representations of inputs at different resolution scales.

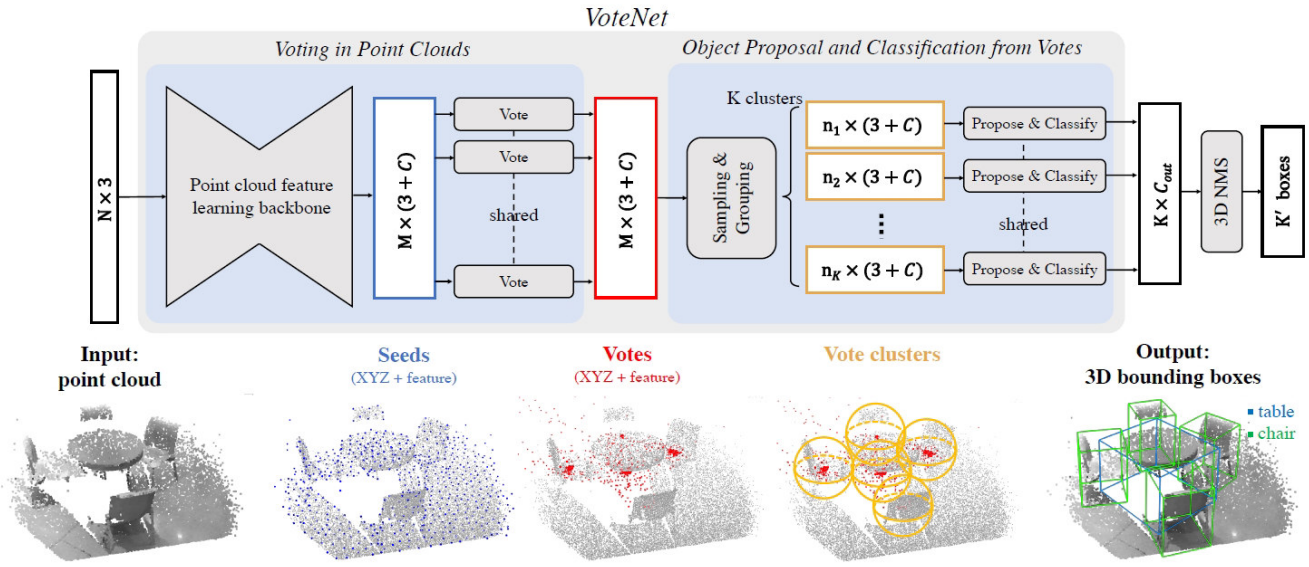

Figure 1. Overview of the VoteNet architecture. Taken from Ref. [10].

In particular, the hierarchical structure is implemented by repetitive application of three steps: *sampling*, *grouping*, and *feature aggregation* with PointNet layer. These three steps together compose a *set-abstraction module*. The sampling step relies on farthest point sampling (FPS) algorithm to iteratively choose a subset of K points that effectively define the spatial coordinates of neighborhood centroids. The grouping step assigns to each centroid a maximum number of N' closest points that lie within a neighborhood sphere defined by Euclidean radius r, so that there are a total of K local groups each having at most N' points with $(3 + C_l)$ features. Next, the PointNet layer is applied to each point group yielding a total of K new abstracted points each with a $(3 + C_{l+1})$ aggregated feature vector.

The feature extraction network. Starting with the raw point cloud (PC) as an input, several set-abstraction (SA) modules are stacked in a hierarchical fashion with sequentially increasing contextual scale [13]. Similar to a 2D CNN segmentation network structure [16, 3], features from lower layers are upsampled and concatenated with skip connections from higher layers in feature propagation (FP) layers in order to pass both global and local semantic information to the total of M output points each with it own $(3 + C_L)$ feature vector.

The Hough voting module. Having extracted a subset $S = \{s_i\}_{i=1}^{M}$ comprised of points enhanced with deep semantic features that take into account local geometric structures, the subset S is passed into a shared deep *Hough voting module* [10]. Based on each feature vector s_i, the voting module, that is implemented as a combination of MLP, ReLU and batch normalization, generates a feature correction Δs_i, such that $v_i = s_i + \Delta s_i$. The spatially corrected vote points v_i are no longer confined to object surfaces, but are generally found to be closer to the centroids of the objects. The last step is important for generating accurate 3D bounding box predictions in the subsequent OPCN network, as a 3D object centroids are otherwise likely to be found in an empty space far from the scanned surface point.

2.2. Object Proposal and Classification Network

The virtual vote points cluster up near object centroids thereby effectively concentrating the encoded semantic information from different surface points of the object in its geometric center. The vote points are grouped into regional subsets by their spatial proximity in a similar manner to the sampling and grouping steps in the *set-abstraction module*. In that way, sampling N' different vote points with FPS algorithm determines the spatial coordinates of vote group centers. Then the vote points that lie inside the Euclidean spheres with radii r around the established vote group centers are grouped together in N' groups. Finally, each of the groups are fed into a PointNet layer, having a similar structure as Eq. (1). This time, however, the γ function

Module (Input)	Output	Parameters
SA1 (PC)	(2048, 3 + 128)	2048, 0.2, 64/64/128
SA2 (SA1)	(1024, 3 + 256)	1024, 0.4, 128/128/256
SA3 (SA2)	(512, 3 + 256)	512, 0.8, 128/128/256
SA4 (SA3)	(256, 3 + 256)	256, 1.2, 128/128/256
FP1 (SA3, SA4)	(512, 3 + 256)	256/256
FP2 (SA2, SA3)	(1024, 3 + 256)	256/256

Table 1. Architecture of VPN. Parameters are # of clusters K, receptive field radius r (meters), MLP layer dimensions $n_1/n_2/n_3$.

	bed	table	sofa	chair	toilet	desk	dresser	nightstand	bookshelf	bathtub
Published [10]	83.0	47.3	64.0	75.3	90.1	22.0	29.8	62.2	28.8	74.4
Ours	84.9	50.8	64.2	75.1	87.0	24.5	28.1	62.2	32.0	78.0

Table 2. Average presision at 0.25 3D-IoU on SUN RGB-D test set. Comparison with results published in Ref. [10].

maps $\mathbb{R}^D \rightarrow \mathbb{R}^{2+3+2H+4S+T}$, where the resulting vector consists of 2 objectness scores, 3 center regression values, H numbers for heading bins with H corresponding regression corrections, S values for box size anchors with the $3S$ corresponding box size regression corrections, as well as T values for semantic classification [10].

Finally, the entire VoteNet network is optimized end-to-end by the multi-task hybrid regression-classification loss function of the following structure:

$$\mathcal{L} = \lambda_0 \mathcal{L}_{\text{vote}} + \lambda_1 \mathcal{L}_{\text{obj}} + \lambda_2 \mathcal{L}_{\text{cls}} + \lambda_3 \mathcal{L}_{\text{c-reg}} + \lambda_4 \mathcal{L}_{\text{a-cls}} + \lambda_5 \mathcal{L}_{\text{a-reg}} + \lambda_6 \mathcal{L}_{\text{s-cls}} + \lambda_7 \mathcal{L}_{\text{s-reg}},$$

where λ_i are weights of vote, objectness, semantic classification, center regression, heading angle class and regression, size class and regression losses, respectively. See Ref. [10] for details.

3. Results comparison: SUN RGB-D

The first part of the project involved replicating the original VoteNet paper results on the SUN RGB-D dataset, which is a single-view RGB-D dataset of ∼10k interior scans with 3D bounding box ground truth (GT) annotations for 37 object categories. Thanks to authors publicly sharing their VoteNet source code in PyTorch implementation on GitHub [2], replicating the published results [10] was rather uncomplicated. We use the same ∼50%/50% train/test split and the same model parameters for our benchmark as the authors of the VoteNet in Ref. [10]. We take 10 object classes to compare our calculations (see Tab. 2). The parameters of the backbone feature extraction network are specified in Tab. 1. The loss weights are scaled with $\lambda_{0,3,5,7} = 1.0$, $\lambda_1 = 0.5$, and $\lambda_{2,4,6} = 0.1$. The number or size templates and classes $T = S = 10$, the heading angle is binned equally in $H = 12$ classes. Training is performed using Adam optimizer with scheduled learning rate (LR) and batch norm (BN) momentum policies for 250 epochs with the mini-batch size of 8, where each sample is randomly subsampled on-the-fly to a total of $20k$ points that are then also randomly augmented with uniform rotations of $\pm 5^\circ$ and random uniform scaling of $\pm 10\%$. One training on a Nvidia K80 GPU took ∼40 hours (training on V100 reduces the training time by a factor of ∼3). Our results are tabulated in Tab. 2, where they are also compared to the published values. One can see that we have successfully managed to reproduce the reported results.

4. Adapting VoteNet to outdoor scenes

Up to now VoteNet has only been tuned and tested on RGB-D datasets with indoor 3D scenes: ScanNet and SUN RGB-D. In order to adapt the model to a different dataset, we have first implemented a custom input pipeline to optimally preprocess and feed the KITTI point cloud data into the VoteNet network. We have also partially re-implemented and adjusted the VoteNet architecture in order to properly account for the characteristics of the KITTI outdoor point clouds.

4.1. Preprocessing and training scheme

An official KITTI benchmark for 3D object detection [6] contains 7481 annotated training scenes and 7518 testing scenes without GT annotations. We utilize the frequently used train/validation split (e.g. [4, 17]) to divide the annotated part of the dataset into a train split of 3712 scenes and an validations split of 3769 scenes. Each scene includes a Velodyne point cloud, an RGB image of front-facing camera, a set of calibration matrices for various plane projections, and (in case of training scenes) GT bounding box annotations and class labels. See Fig. 4 (b) and (c).

As the 3D bounding box GT annotations are available only in the view field region of the front facing camera, we first project the 360-degree LiDAR data onto the image plane and extract the point cloud that lie inside the resulting frustum, see Fig. 4 (a). This step reduces the *average* number on points per scene from ∼110k to 16,384. For each scene we randomly subsample 16,384 points, and for scenes with fewer points, we randomly copy the points up to a total of 16,384. The scenes are furthermore augmented on-the-fly with random flips in horizontal plane, random uniform rotations along up-axis in $\pm 5^\circ$ range, and random uniform scaling of $\pm 10\%$. In addition to three Euclidean coordinates for each point, we also include up to two additional features: the provided laser reflection intensities, and a height estimate for each point. The latter is estimated as a 1% percentile of all point positions along the up-axis.

We perform training on three most abundant classes of the KITTI dataset: *car*, *pedestrian*, and *cyclist* with 14357, 2207, and 734 bounding boxes across all scenes in train split, respectively. We choose one size template per class, and 12 bins for the heading angle. We train with a batch size of 16 and incorporate a scheduled learning policy with a starting LR of 0.001 and LR-decays by 0.1 at 60, 90, and 120 epochs, as well as an exponential BN momentum decay. We use same loss function as in SUN RGB-D case.

Module (Input)	Output dimensions	Clusters K	MSG Radii	MLP layers
SA1 (PC)	(4096, 3 + 96)	2048	0.1, 0.5	16/16/32, 32/32/64
SA2 (SA1)	(1024, 3 + 256)	1024	0.5, 1.0	64/64/128, 64/96/128
SA3 (SA2)	(512, 3 + 512)	512	1.0, 2.0	128/196/256, 128/196/256
SA4 (SA3)	(64, 3 + 512)	256	2.0, 4.0	256/256/512, 256/384/512
FP1 (SA3, SA4)	(512, 3 + 512)	—	—	512/512
FP2 (SA2, SA3)	(1024, 3 + 512)	—	—	512/512

Table 3. Enhanced architecure of VPN with two MSG radii of receptive fields $r_{1,2}$ (meters), and MLP parameters for each MSG group.

4.2. Network Architecture

Since the VoteNet architecture was previously tuned only for indoor point cloud data, we adapt the network to the distinct characteristics of the outdoor LiDAR scenes. For example, the typical scales of the KITTI scenes are significantly larger than those of indoor scans, with depth fields reaching beyond 70 meters along forward-axis. In order to adequately reflect this in the network architecture, we adapt the receptive field radii and increase the number of clusters for feature aggregation. Moreover, the LiDAR point cloud has strongly varying density and is generally more sparse than the point clouds produced from the RGB-D imagery. Point features extracted from sparse regions may generalize poorly to dense regions, and vice versa. In order to capture fine details of point cloud and, at the same time, mitigate the corruption of local patterns due to sampling deficiency in sparse cloud regions, we enhance the set-abstraction modules of the backbone network with multi-scale grouping (MSG) layers, which, as illustrated on Fig. 2, concatenate features at different scales before feeding them into the feature aggregation layer. This results in a robust feature learning under non-uniform sampling density.

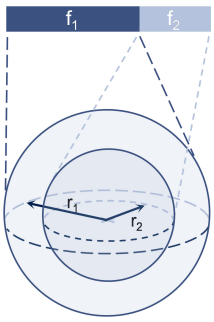

Figure 2. Illustration of multi-scale grouping (MSG) layer.

Since the original VoteNet implementation was quite inflexible with many hard-coded network parameters, we have partially re-implemented the backbone feature extractor network in order to easily configure all network parameters in a separate config-file to allow iterating rapidly through various network hyper-parameters. The best found set of backbone network parameters are specified in Tab. 3, where four MSG-based SA layers are used to subsample the inputs to 4096, 1024, 256, and 64 points, respectively, whereupon the last two FP layers upsample the points back to 1024 points, each enhanced with additional 512 deep features. See Tab. 3 for details. Finally, we also enlarge the receptive field radius of the vote aggregation layer to 0.7 meters.

5. Results

The evolution of average precision (AP) metrics for different variant of VoteNet architecture evaluated on KITTI validation split is shown on Fig. 3. All variants were trained for 150 epochs. The *original* VoteNet model has essentially the same network parametrization as used for SUN RGB-D dataset previously. The *tuned* VoteNet network has the model parameters adjusted specifically for the characteristics of KITTI outdoor scenes. Finally, as can be seen on the figure, the best performance was archived with the *tuned* VoteNet network that has been enhanced with *MSG layers*.

A representative output of the network is visualized in Fig. 4 (d), where the bounding box and corresponding class predictions are shown in red. We see that in cases when objects are relatively close to the LiDAR, we generally get better results since the number of foreground points is relatively large. The model also manages to predict well the oriented amodal 3D boxes of partially occluded objects, e.g. parked cars. The common mode of failure is, however, related to the distant objects, where the number of foreground points is substantially smaller (sometimes resulting in less

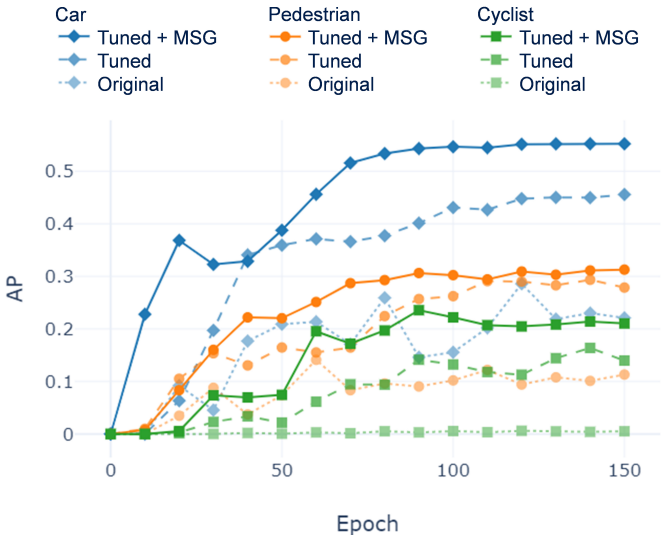

Figure 3. The average precision (AP) with 3D IoU threshold of 0.25 evaluated on *original*, *tuned*, as well as both *tuned* and enhanced with *MSG layers* VoteNet architectures.

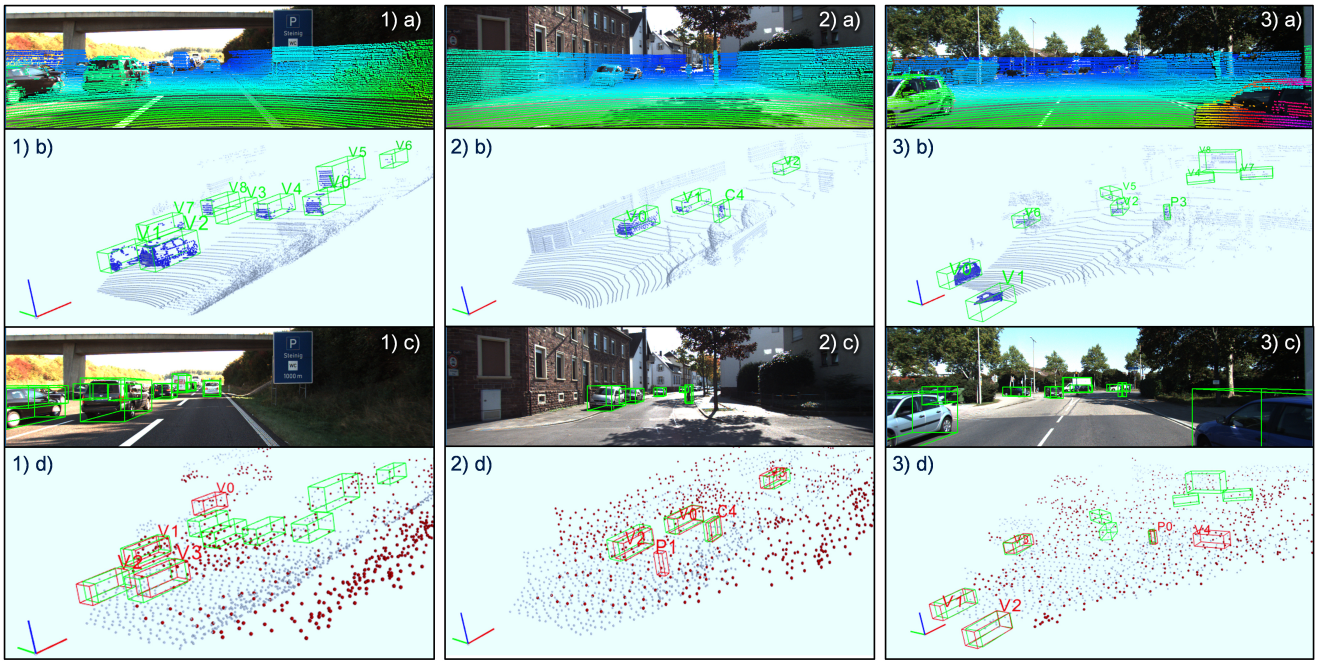

Figure 4. Visualization of KITTI scenes (validation split) and the corresponding VoteNet results: (1) Velodyne point clouds and their projections onto the image plane; (2) the subsampled LiDAR point cloud and the annotated 3D bounding boxes (green), labels: *V* - car/vehicle, *P* - person, *C* - cyclist; (3) the GT bounding boxes projected onto image plane; (4) VoteNet results in form of 3D bounding boxes and the corresponding class label (red); the seed PC (blue) and the corresponding votes (red) are also shown.

than a few vote points). Furthermore, as seen on Fig. 4 (2-d), the model sometimes wrongly labels such narrow tall objects like trees as *pedestrian*, but gets the *cyclist* correctly predicted on the same scene. It also appears challenging for the model to correctly labels multiple instances of the same object packed close to each other, e.g. Fig. 4 (1-d).

We also note that vote points, shown as red dots on Fig. 4 (d), do not generally tend to cluster up near centroids of the objects, as expected. We do observe clustering in some KITTI scenes (e.g. Fig. 4 (1-d)), but that is not the case in the majority of cases. This is most likely due to the fact that, whereas the point clouds extracted from the depth images (RGB-D) are rather dense, the much more sparse LiDAR point cloud data might pose a complication for the OPCN network, since the signal-to-noise ratio (rate of positive votes to negative votes) is much lower. So, for example, the ratio of positive seeds to all seed point is about 0.5%.

The final average precision results are shows in Tab. 4. We see that tuning the model architecture parameters to the characteristics of the dataset resulted in a significant boost to the performance of the model, elevating the evaluation metrics for *car* by **10.2 AP**, for *pedestrian* by **16.6 AP**, and *cyclist* by **13.6 AP** points. Another huge improvement came from enhancing SA module with MSG layers, which brought a total improvement of **34.2 AP**, **19.9 AP**, and **20.5 AP** for *car*, *pedestrian*, and *cyclist* categories, respectively. The recall score for the three considered classes are **66.4**, **47.6**, and **42.8**, respectively. Although that the performance gain, relative to the original model, is quite substantial, the overall AP score, if compared to the current state-of-the-art results of KITTI 3D object detection benchmark, is still relatively modest.

	Car	Pedestrian	Cyclist
Original	21.0	11.3	0.5
Tuned	31.2	27.9	14.1
Tuned + MSG	55.2	31.2	21.0

Table 4. Final AP scores on KITTI validation split, IoU at 0.25.

6. Outlook

Despite the shown performance gains brought by improving VoteNet model to optimally capture the characteristics of LiDAR outdoor point cloud, we have identified a major issue that still needs to be addressed in order to achieve results comparable with current state-of-the-art in the KITTI 3D object detection benchmark. Namely, in order to mitigate the poor signal-to-noise ratio in such large-scale outdoor scenes as KITTI, one could implement a kind of point filtering or foreground pre-segmentation step. In the latter case, a variation of the VoteNet model would be to predict foreground scores for each point and weight point features by the predicted scores, thereby weighting the foreground point with larger contributions to the votes.

References

[1] ICCV 2019 official website. http://iccv2019.thecvf.com/program/main_conference. Accessed: 2019-11-30.

[2] VoteNet official github repo. https://github.com/facebookresearch/votenet. Accessed: 2019-11-08.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[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.

[5] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017.

[6] 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.

[7] 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.

[8] 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.

[9] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3496–3504, 2017.

[10] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. *arXiv preprint arXiv:1904.09664*, 2019.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[17] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[18] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[19] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7652–7660, 2018.

[20] 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.