Instance Segmentation of LiDAR Point Clouds

Instance Segmentation of LiDAR Point Clouds

Feihu Zhang[1], Chenye Guan[2], Jin Fang[2], Song Bai[1], Ruigang Yang[2], Philip H.S. Torr[1], Victor Prisacariu[1]

***Abstract*— We propose a robust baseline method for instance segmentation which are specially designed for large-scale outdoor LiDAR point clouds. Our method includes a novel dense feature encoding technique, allowing the localization and segmentation of small, far-away objects, a simple but effective solution for single-shot instance prediction and effective strategies for handling severe class imbalances. Since there is no public dataset for the study of LiDAR instance segmentation, we also build a new publicly available LiDAR point cloud dataset to include both precise 3D bounding box and point-wise labels for instance segmentation, while still being about 3∼20 times as large as other existing LiDAR datasets. The dataset will be published at https://github.com/feihuzhang/LiDARSeg.**

I. Introduction

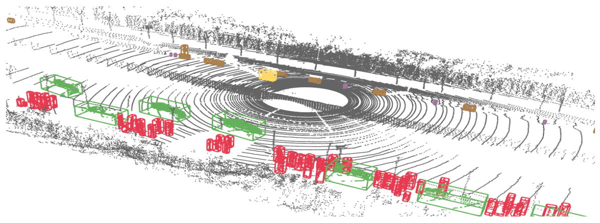
Accurately capturing the location, velocity, type, shape, pose, size *etc*. of objects (*e.g*. cars, pedestrians, cyclists *etc*.) in the outdoor scenes is a vital task for many vision and robotic applications. The LiDAR system can extract 3D information from the surrounding environment with high accuracy in various lighting conditions and has become a key component for *e.g*. autonomous driving and robotic systems.

LiDAR detection has seen much recent work, with the introduction of methods such as [7], [42], [55], [58], which estimate the locations of *e.g*. the cars, as either 3D or 2D bounding boxes. But, for detection, many outliers (*e.g*. points from the road or neighborhoods) will be mixed into the bounding boxes. Moreover, any imperfection (errors of orientation, size, center *etc*.) of the detected bounding boxes will heavily reduce the precision (illustrated in Fig. 1(b)).

A far less explored avenue of research is instance segmentation in LiDAR data. Unlike the coarse estimation of object detection, instance segmentation aims to accurately pick up every reflected point from each foreground object. It can significantly reduce the interference of outlier points for each object, better represent the irregular shapes and contribute to more accurate sensor fusions (illustrated in Fig.1(c)), motion planning and 3D HD map construction. Moreover, segmentation can better handle the occlusions or neighborhood objects where 3D bounding boxes are hard to estimate and usually ambiguous with large overlaps (as illustrated in Fig.1(b-c)).

However, LiDAR instance segmentation is more challenging and different from *2D* image or RGB-D based instance segmentation (*e.g*. [9], [16]). As illustrated in Fig. 1(a):
1) Data representation: LiDAR points are sparse and distributed irregularly in a large 3D space. Sparsity also increases considerably with distance;

[1]Authors are with the University of Oxford.
[2]Authors are with Baidu Research.


(a) Results of Instance Segmentation

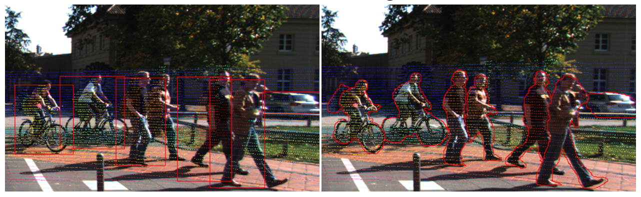

(b) Detection Results (c) Segmentation Results

Fig. 1: Problem and performance illustrations. (a) Results of instance segmentation produced by our method. (b) Results of the state-of-the-art detection methods [42] (projection on 2D image). (c) Our method (on 2D image). It can produce more accurate instance segments and help get better sensor fusions by merging image and point cloud segments/masks (zoom in to see sensor fusion results).

2) Large-scale scenes: One LiDAR frame can capture more than 120k points in a space of larger than $140m \times 100m \times 5m$;
3) Small objects: There are many small but important objects (*e.g*. pedestrian, bicyclist *etc*.). These objects are relatively tiny with few points reflected and are especially hard to be discovered in large-scale scenes. For example, one pedestrian is less than $0.5m^3$ in the $140m \times 100m \times 5m$ point cloud.
4) Class imbalances: The class imbalance problem is severe in real-world road scenes. For example, the imbalance ratio between the major class (*e.g*. car) and the minor class (*e.g*. pedestrian) can be larger than a factor of 20.

In this paper, we focus on the multi-class and multi-object instance segmentation of the LiDAR point clouds in large-scale outdoor scenes. We propose a robust baseline model which includes a new dense feature encoding scheme for point cloud representations to achieve better localization and segmentation of the far and small objects, a powerful densely connected stacked backbone and a simple but effective approach for instance prediction.

Moreover, we contribute a large dataset for the study of LiDAR point cloud based instance segmentation. Existing datasets (*e.g*. KITTI [12] and Nuscenes [4]) only label 3D bounding boxes. Our new dataset has both 3D bounding box and point-wise labels, which allows robust instance segmentation models to be trained. It has a total of 130k point cloud frames, with more than 3 millions foreground objects, so is $3 \sim 20\times$ larger than existing LiDAR datasets.

II. Related Work

In this section, we review related work on point-cloud-based detection and instance segmentation. We also explore the datasets available for training point-cloud methods.

A. 3D Detection

Several methods [7], [10], [42], [42], [46], [51], [55], [58] propose point-cloud-based detectors, that estimate object locations and produce 2D or 3D bounding boxes. Image information is also leveraged, in approaches like [5], [6], [20], [34], [53], [56], [59]. However, the accuracy of image-based 3D detection approaches is heavily dependent on the quality of the image data, so they would not be reliable in all lighting conditions (*e.g.* in the dark). Multi-sensor fusion approaches [23], [54] have also been proposed to increase the reliability of the detectors.

These detection methods rely on the generation of accurate 3D bounding box proposals. As the variances of objects' sizes and shapes increase, especially for many small objects, it becomes much more difficult to produce accurate bounding box proposals.

B. 3D Instance Segmentation

Instance segmentation is the problem of simultaneous locating and delineating each distinct object of interest appearing in a scene. Based on recent advances in object detection [13], [25], [26], [29], [37]–[40], instance segmentation [9], [32], [33] has achieved good results on 2D images. Many of the latest instance segmentation models are based on segment or mask proposals [9], [16], [32]. These methods, however, can not immediately be extended to 3D point clouds due to irregularity and sparsity of the large 3D space and the difficulty of generating proposals.

Recent approaches have shown promising performance for point clouds in smaller indoor environments. Some of them (*e.g.* [52], [28]) learn a similarity or affinity matrix for grouping the points and generating instance-level segments. Volumetric approaches, like 3D-SIS [18], uses both 3D geometry and 2D images as input for anchor based detection and mask prediction. Direct bounding box regression is used in [57], with a shape generation/reconstruction stage.

However, there are currently limited work for large-scale, outdoor point cloud instance segmentation. Existing approaches are designed for indoor scenes, which (i) are considerably smaller, (ii) have denser point clouds and (iii) require a slammer variability of object shape and size. Some also assume associated RGB data and are not applicable to the challenging outdoor scenes or produce poor performance.

C. Point Cloud Datasets

There are several popular point cloud datasets for semantic segmentation and detection. Most of them provide RGB-D data, filmed by ToF cameras. Examples are (i) [14], [24], [44], [48], [49], built for 2D object detection; (ii) LabelMe [41] and SUN RGB-D [47], providing polygons in 2D and bounding boxes in 3D; (iii) NYU v2 [45] consisting of 464 short sequences with semantic labels; (iv) Armeni *et al.* [1], [2] providing an indoor dataset with 3D meshes and semantic annotations for 265 rooms and (v) ScanNet [8], a RGB-D video dataset containing with 2.5M views in 1513 scenes for 3D semantic and instance labels.

All these datasets target (relatively) small-scale indoor scenes. There are also a few point cloud datasets for large-scale outdoor scenes. Semantic3D [15] is built for large-scale semantic segmentation. KITTI [12], Apollo [3], H3D [31] and Nuscenes [4] are all LiDAR datasets developed for object detection with only 3D bounding boxes labels.

We present in Section IV our LiDAR segmentation dataset which has both 3D bounding boxes and point-wise labels.

III. Proposed Model

Fig. 2 outlines the components of our approach for large-scale instance segmentation of LiDAR point clouds, detailed in the following sections as follows: §III-A for the feature representation, §III-B for the network backbone, §III-C for our final instance prediction and §III-D for our loss.

A. Dense Feature Representation

The feature representation is the initial but crucial step for point cloud based recognition tasks. In this section, we develop our dense feature encoding technique for the instance segmentation.

1) Challenges for Feature Representation

A popular type of irregular feature representation, used for point-level semantic segmentation, is the approach employed by PointNet [35] and PointNet++ [36], where local and global features are learned for each point. The input points are unordered, so such methods are unstable to use powerful convolutional layers to learn spatial or geometric information (*e.g.* bounding box regression). Then, Li *et al.* propose the transformed CNN for feature extraction [22], which is also for classification and semantic segmentation.

For large-scale LiDAR point clouds, regular representations, like voxels or grids, enable richer geometric information (*e.g.*, height, center, distance *etc.*) to be captured efficiently by convolutions. Some approaches then use hand-crafted features [7], [30], [55]. These yield satisfactory results when rich and detailed 3D shape information is available. However, they do not adapt well to sparse scenes and produce poor accuracies for small objects. Other approaches, *e.g.* VoxelNet [58], learn sparse point-wise features for voxels and use 3D convolutions for feature propagation, which shows good results in 3D object detection.

These sparse regular feature representations are popular in object detectors. But, they have limited performance for segmentation models. Instance segmentation requires high-resolution feature maps for finding tiny objects (*e.g.* pedestrians and cyclists *etc.*) in large-scale point clouds. This will lead to large amount of empty/invalid voxels/grids, especially for far locations. The invalid/empty locations will make the convolutional layers unstable and discount the efficacy of feature propagation, localization and regression. This is because traditional (dense) convolutions are not specialized for the sparse data structure and the propagation can be easily stopped or affected by void elements.

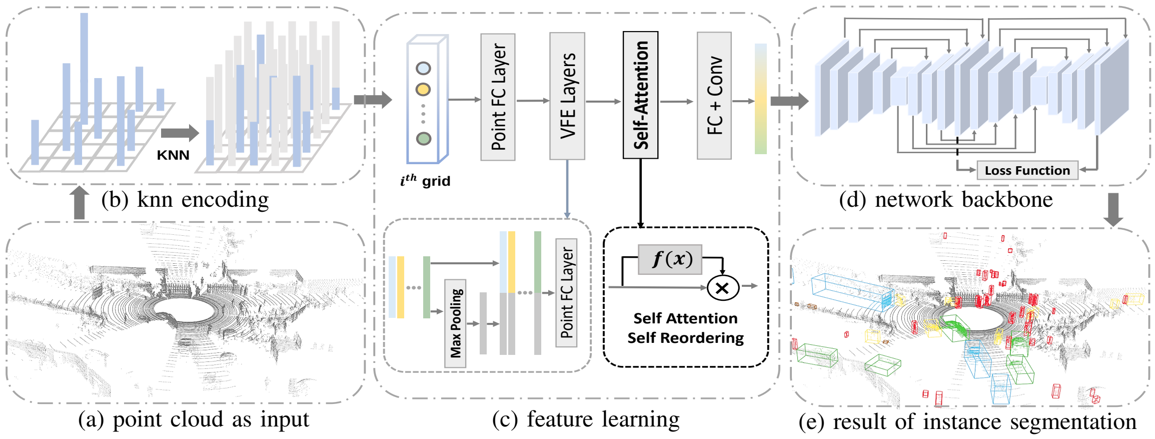


Fig. 2: Framework of the proposed method. (a) Point cloud frame as input, (b) knn to get dense representation, (c) feature learning with a self-attention block to re-organize the unordered points, (d) backbone network, (e) instance segmentation result with bounding boxes.

2) KNN Encoding as Input

It is difficult to extract enough information only from the valid points in each grid/voxel when targeting object-wise inference. We design a novel dense high-resolution bird's-eye view representation for instance segmentation. Our point cloud encoding is based on the fact that every point contains rich information not only for its own grid/location/voxel but also for its surroundings. This is especially meaningful for the locations with few (*e.g.* in the distance) or no points. The distances to valid points and the surrounding points' heights all provide rich geometric information for feature encoding.

For efficiency, we directly encode the point cloud into a 2D high-resolution bird's-eye view representation. The segmentation and localization is then based on this regular representation. Compared with more complicated 3D voxels, our approach is much faster and memory efficient, since no 3D convolutions will be involved.

Our feature encoding and learning strategies are depicted in Fig. 2(b)∼(c). All points are projected into high-resolution $(H \times W)$ 2D grids. We use KNN to augment empty or poor grid locations with their K near neighborhoods.

The KNN algorithm runs based on the horizontal (x, y) distance and can be efficiently parallelized for computation on the GPU or accelerated using K-D Tree. There are totally K points in each 2D grid. We represent each point as $P_i = \{(g_x, g_y), (\Delta x_i, \Delta y_i), (z_i, r_i)\}$, where (i) (g_x, g_y) are the coordinates of the center of the grid; (ii) $(\Delta x_i, \Delta y_i) = (x_i, y_i) - (g_x, g_y)$ is the shifts between the point coordinates and the grid center; (iii) r_i is the received reflectance and (iv) z_i is the point's height. Finally, the input becomes $\{P_0, P_1, \ldots P_{K-1}\}$ for each grid location.

3) Self-attention Block

Since the input of K points in each grid are unordered, we cannot use convolutional or fully connected layers to learn grid-level feature directly from these K points. We instead, utilize a novel self-attention and re-ordering block to learn grid-level feature representation from raw points.

The attention block starts with two voxel feature encoding (VFE) layers, as proposed in [58], to learn point-wise features and attention matrix. For each grid location, we assume the K points are represented as $\mathbf{F_p}$, which is a $K \times n$ matrix (K is the number of the points and n is the length of the point-wise feature). We learn an adaptive weight matrix $\mathbf{A}$, which plays two key roles. Firstly, it helps re-organize the points into a regular and ordered representation by adapting the weights to the points, without being influenced by the order and locations of the points. Secondly, it learns a self-attention mask for the K points, to decide automatically which point should play key role and have more of our attentions.

The grid-level feature $\mathbf{F_g}$ can be learned with:

$$\mathbf{F_g} = \mathbf{A}^T \cdot \mathbf{F}_\mathrm{p} \text{ with } \mathbf{A} = \boldsymbol{f}(\mathbf{F_p}) \tag{1}$$

where, $\boldsymbol{f}$ is a mapping function to get the self-attention and self-reordering matrix $\mathbf{A}$. The output is in $K \times n$. Specifically, each (i-th) row of $\mathbf{F_g}$ is learned as:

$$\mathbf{F}^i_\mathbf{g} = a^0_i \cdot \mathbf{F}^0_\mathbf{p} \ldots + a^j_i \cdot \mathbf{F}^j_\mathbf{p} \ldots + a^{k-1}_i \cdot \mathbf{F}^{k-1}_\mathbf{p} \tag{2}$$

where $[\mathbf{F}^0_\mathbf{p}, \ldots \mathbf{F}^j_\mathbf{p}, \ldots \mathbf{F}^{k-1}_\mathbf{p}]$ are point-wise features for K points. The weight a^j_i is the j-th row and i-th column in matrix $\mathbf{A}$. Since a^j_i is adaptive and learned from $\mathbf{F}^j_\mathbf{p}$, the sequence of the K points will not influence the output $\mathbf{F}^i_\mathbf{g}$. Moreover, it helps re-weight the contributions from each point and automatically decide which point need more attentions.

After the self-attention block, the grid-level feature can be directly consumed by standard convolutional neural layers. We further use one fully connected layer to refine and reshape the $n \times n$ feature into a $48\times$ feature vector. For the mapping function $\boldsymbol{f}$, we simply use a point-wise fully connected layer along with a column-normalization to learn the self-attention and self-reordering weight matrix.

Finally, the grid-level features construct a regular dense bird's-eye view representation, which is fed to the backbone network for localization, segmentation and classification.

B. Network Backbone

We use a revised stacked hourglass block as the backbone for instance segmentation. The architecture is illustrated in Fig. 2 and detailed in Table I. The pyramid features from different layers are densely connected by concatenations which is similar to [60].

Our approach is single shot, *i.e.* no region proposal network as those in [7], [10], [42], [46], [51], [58] is used. The

TABLE I: Parameters of the backbone network.

No.	Layer Description	Output Tensor
input	feature encoding	$H\times W\times 48$
1	3×3 conv (bn, relu)	$H\times W\times 24$
2	3×3 conv (stride 2, bn, relu)	$1/2H\times 1/2W\times 48$
3	3×3 conv (bn, relu)	$1/2H\times 1/2W\times 48$
4-5	repeat 2-3	$1/4H\times 1/4W\times 64$
6-7	repeat 2-3	$1/8H\times 1/8W\times 96$
8-9	repeat 2-3	$1/16H\times 1/16W\times 128$
10-11	repeat 2-3	$1/32H\times 1/32W\times 256$
12	3×3 deconv (stride 2, bn, relu)	$1/16H\times 1/16W\times 128$
13	3×3 conv (bn, relu)	$1/16H\times 1/16W\times 128$
14-15	repeat 12-13	$1/8H\times 1/8W\times 96$
16-17	repeat 12-13	$1/4H\times 1/4W\times 64$
18-19	repeat 12-13	$1/2H\times 1/2W\times 48$
20-21	repeat 12-13	$H\times W\times 24$
Output	1×1 conv (no bn or relu)	$H\times W\times(5+C)$
Loss	Eq. (5), loss weight: 0.6	—————
22-41	repeat 2-21	$H\times W\times 24$
Output	3×3 conv (no bn or relu)	$H\times W\times(5+C)$
Loss	Eq. (5), loss weight: 1.0	—————
Concatenate	(1,20), (3,18), (5,16), (7,14), (9,12), (13,28), (15,26) (17,24), (19,22), (21,40), (23,38), (25,36), (27,34), (29,32)	

outputs of our network are of size $H\times W\times(5+C)$, and are used for (i) the foreground classification, (ii) regression of objects' center for clustering (with 2 channels), (iii) predicting objects' height limits (both upper and lower constraints for removing the false positives) and (iv) classifying objects into C classes (with the remaining C channels).

C. Instance Segmentation

Popular methods [16], [17] use anchors to predict bounding boxes and achieve the instance IDs. However, in LiDAR point clouds the sizes and poses of the objects have large diversity (from large trucks to small children), making the anchor-based bounding box regression difficult. Others, like SGPN [52], learn a similarity score for each pair of points and then merge them into groups. However, a LiDAR point cloud frame usually contains more than 100k points, making the learning of the similarity matrix not easily tractable (with a time complexity of N^2). Also, the similarity based clustering is likely to be ambiguous and might miss a lot of small objects for large-scale outdoor scenes.

We use a simple while effective way to predict the instance objects. We predict the horizontal center (similar to [50]) and the height limits of the object. We use them as constraints to group points into each candidate object and further remove outliers. Namely, for each foreground grid, we predict an object center and the height limits. If the predicted centers of different grids/points are close enough ($< 0.3m$), we merge them into a single object. This is very effective for small objects since points are very close to each other. Also, the learning target is simple and easy to learn.

D. Loss Function

We use the class-balanced focal loss [26] to find foreground points and for classification:

$$L_f(x,y) = -\sum_{i=1}^{C}(1-p_i)^{\gamma}\,\log(p_i)$$
$$p_i = \begin{cases} \frac{1}{1+\exp(-x_i)}, & \text{if} \quad i=y. \\ \frac{1}{1+\exp(x_i)}, & \text{otherwise.} \end{cases} \tag{3}$$

where x is a $C-$channel output for C binary classification tasks. p_i is the probability after *sigmoid* function. We use $\gamma=2$ which is the same as that in [26].

For the instance prediction, the learning target is set to $(\Delta x, \Delta y, h)$, where Δx and Δy are the shifts from the current location to the object's center and h is the upper and lower height limits of the object. The center of object is set as the center of the bounding box. We employ the smooth L_1 loss for regression. It is defined as:

$$L_s(x) = \begin{cases} 0.5x^2, & \text{if } |x|<1. \\ |x|-0.5, & \text{otherwise.} \end{cases} \tag{4}$$

The final multi-task loss for classification and regression is:

$$\begin{aligned} L \;=\; & \sum_{x\in I} L_f(x_{seg}, y_{seg}) + \sum_{x\in I'} \alpha_1 \cdot L_f(x_{cls}, y_{cls}) \\ & + \sum_{x\in I'} \alpha_2 \cdot L_s(x_{ct}-y_{ct}) + \alpha_3 \cdot L_s(x_h - y_h). \end{aligned} \tag{5}$$

where, x_{seg} and y_{seg} are the predictions and labels for categorizing foreground grids, x_{ct} and y_{ct} are the $2-$channel predictions and ground truths of the object centers, x_h and y_h are the predictions and ground truths of the objects' height limits, and x_{cls} and y_{cls} are the predictions and labels for classification. The weights of $(\alpha_1, \alpha_2, \alpha_3)$ for balancing different tasks are set to $(0.1, 20, 0.1)$.

The loss for classification and clustering only takes effects on the foreground regions I'. We use the "one point one vote" strategy to predict the objects' categories. This is to reduce the influence of the extreme values of unexpected noises.

IV. Dataset

A big limitation in developing a robust instance segmentation algorithm for LiDAR data is the lack of a publicly available dataset. We build a new large LiDAR point cloud dataset, which contains both 3D bounding boxes and point-wise labels for instance segmentation.

Table II shows comparisons between our dataset and the existing LiDAR datasets. Our new dataset contains 130k LiDAR frames with around three millions foreground objects, which is $3\sim 20$ times larger than existing datasets. Mostover, Our dataset contains point-wise labels for instance segmentation.

Our dataset consists of two parts, real data and simulation data, both are larger than either KITTI or Apollo. It is much more expensive and difficult to label instance segments than 3D bounding boxes, To guarantee the diversity of the dataset, we augment the 30k real dataset with 100k simulation samples. The domain gaps between our simulation and the real data are very small. In our experiments, simulation data can improve the AP by 2∼3.5% in training.

A. Real Data

The data was acquired using Apollo cars [3] with a Velodyne HDL-64E S3 laser scanner. The sensor is positioned on the top and center of the car. The range of the vertical scanning angle is $[-24.9°, 2.0°]$, and the scanning distance range is 120m. During the data acquisition, the scanner speed is set to 10Hz. The point clouds were also motion-compensated using a high-precision GPS/IMU installed on the vehicle.

TABLE II: Comparisons of different LiDAR point cloud datasets.

Dataset	trainig data (simulation + real)	test data (real)	number of objects (total)	classes of objects	labeled region (angle)	label types: 3D bounding box	label types: point label
Kitti [12]	7,481	7,518	80,256	4	45°	√	×
Apollo [3]	10k	10k	475k	4	360°	√	×
H3D [31]	27k	-	1.1m	2	360°	√	×
Nuscenes [4]	34k	6k	1.4m	23	360°	√	×
Ours	100k+20k	10k	3.2m	7	360°	√	√

Data was labeled manually on point cloud with the help and guidance of color images, first by setting the bounding box for each individual object, and second by removing false positives (mainly the points from the ground and the neighborhood objects) from the 3D bounding box.

B. Simulation Data

We build our LiDAR simulation system [11] using real backgrounds and hundreds of foreground 3D object models.We capture real world background with a professional 3D scanner Riegl VMX-1HA which can provide the dense and high-quality point cloud from the real traffic scenarios. To obtain a clean background, moving objects can be removed by repeatedly scanning the traffic scene for several rounds (*e.g.*, 5 rounds). While, to remove the static movable obstacles, we use Pointnet++ [36] to get the initial semantic masks and then correct these wrong parts manually.

The synthetic models of various objects, such as vehicles and pedestrians, can be inserted to create different traffic scenes. The obstacle placement are based on the probability maps. We use a well-trained object detector [58] to calculate the initial location distributions of different kinds of objects. With the guidance of these distribution probabilities, we are able to place the objects as real as possible.

V. Experiments

In this section, we describe our experiments, including the training settings and strategies, ablation study, comparisons and the analysis of the results. *More details are available in the supplementary materials*[1]. We use the evaluation metrics of points based APs in the experiments. They are similar to those in the COCO instance segmentation challenges [27].

A. Training Settings and Strategies

We consider point clouds within the range of $[-64, 64] \times [-64, 64]$ meters along Z, Y, X axis respectively which leads to a resolution of 640×640 for the bird's-eye view representation.We use data resampling strategy[1] to increase the frequency of appearance for minor classes during training.

B. Ablation Study

We conduct experiments with several settings to evaluate the effects of different architectures, loss functions and features *etc*. As listed in Table III, our dense feature encoding improves the AP by 3.4%, which plays a key role in the performance improvements. The densely connected stacked backbone contributes another 1.0% improvements. Focal loss and data re-sampling can produce better classification accuracy which help achieve $1 \sim 2\%$ improvements in APs.

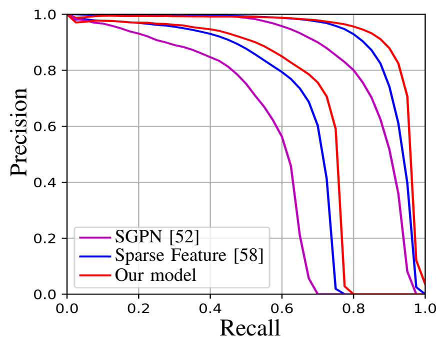


Fig. 4: Precision-Recall curves with two IoU thresholds of 0.5 and 0.95. SGPN, our method and our model with sparse feature [58] are compared in this figure.

TABLE V: Evaluations and comparisons in different ranges (%).

Range / Methods	0-30m $AP^{.50}$	0-30m $AP^{.75}$	0-30m AP	30-60m $AP^{.50}$	30-60m $AP^{.75}$	30-60m AP
SGPN [52]	50.2	48.1	52.1	48.3	41.8	39.1
Feature 1 [7]	65.9	62.4	60.9	62.0	55.5	53.9
Feature 2 [55]	66.9	63.1	62.3	62.6	55.8	54.1
Feature 3 [58]	68.7	65.9	64.1	64.2	59.4	56.4
Ours	**70.1**	**67.6**	**65.7**	**67.4**	**63.1**	**59.5**

C. Results and Comparisons

Since there is no existing instance segmentation method for LiDAR point as the baseline. We compare our method with the recent proposed SGPN [52] which is designed for small-scale indoor scenes. As shown in Table IV and Fig. 3, SGPN is not robust for road scenes when points are very sparse in the distance and there are many small objects with only a few points. As a comparison, our method far outperforms the SGPN (by 17% in AP). It can pick up more small objects and avoid most of the false positives.

We also compare our proposed dense feature encoding with existing sparse feature representations [7], [55], [58] by implementing them into our models (other settings are all the same). As shown in Table IV, we find that for large objects (*e.g.* car), existing features also produce good performances for instance segmentation. This is because large objects have rich information and are not sensitive to the feature encoding techniques. But, our dense features produce far better recognition accuracy (with $2 \sim 5\%$ improvements in AP) for small objects (pedestrian, bicyclist and traffic cone). This means that the proposed feature encoding can capture more reliable information from the original point cloud for small objects with fewer points. As shown in Table V, for objects within $0 \sim 30m$ from the LiDAR's center, our dense feature outperforms the VoxelNet's feature by about 1%. While, for objects in the range of $30 \sim 60m$, where points are much more sparse, our dense feature has $3 \sim 3.5\%$ improvements in APs.

[1]Supplementary materials: http://www.feihuzhang.com

TABLE III: Evaluations of models with different settings. $AP^{0.50}$ and AP (%) are used for evaluations.

Dense Feature Encoding	Backbone		Loss Function	Classification	Performance	
	hourglass	dense connection	focal loss	data resampling	$AP^{0.50}$	AP
					63.1	56.5
√					65.8	59.9
√	√				66.4	60.3
√	√	√			67.1	60.9
√	√	√	√		68.0	61.7
√	√	√	√	√	**68.9**	**62.4**

TABLE IV: Evaluations and comparisons of different models for instance segmentation (AP %).

methods/features	car	large vehicle	pedestrian	bicyclist	motorcyclist	traffic cone	others	average
SGPN [52]	83.5	62.8	33.2	24.0	42.7	34.7	30.9	44.5
VoxelNet* [58]	87.1	70.2	47.2	29.0	45.8	46.5	35.1	51.5
PointRCNN* [43]	88.1	71.2	49.4	32.1	49.2	45.2	37.0	53.2
PointPillar* [21]	87.7	71.9	51.2	33.7	50.1	44.7	37.9	53.9
feature 1 [7]	86.1	69.6	55.5	38.8	50.0	58.3	41.4	57.1
feature 2 [55]	88.0	69.3	54.9	38.9	51.1	56.9	44.5	57.6
feature 3 [58]	**89.2**	72.2	60.2	39.8	54.8	58.7	45.7	60.1
Ours	89.0	**74.9**	**62.4**	**45.1**	**55.0**	**61.8**	**48.2**	**62.4**

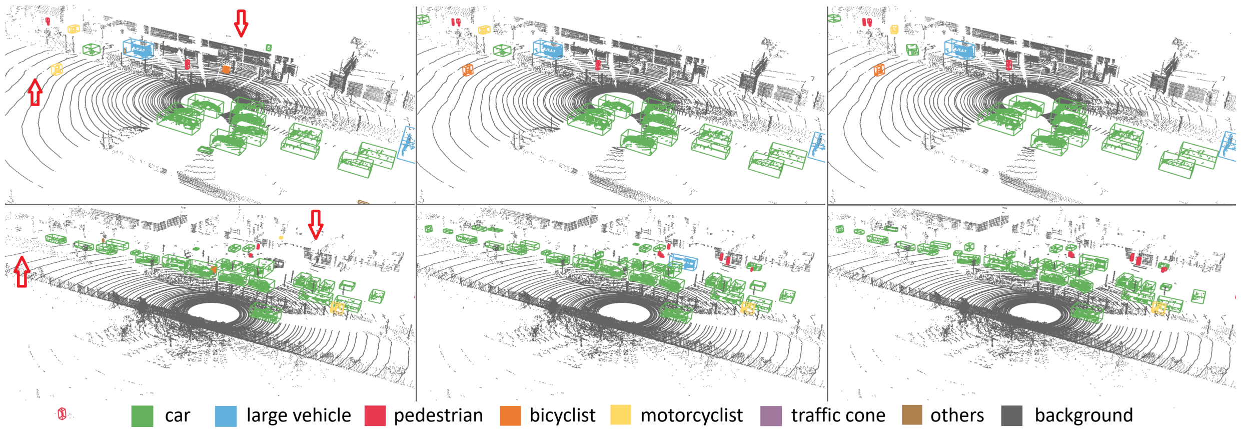


Fig. 3: Visual comparisons between SGPN [52] and our model in complex and challenging scenes. ***Left:*** results of SGPN [52]. ***Middle:*** results of our model. ***Right:*** ground truths. For visual pleasure, bounding boxes are added which is estimated directly from the instance segments. As pointed by ***red arrows***, many small objects are missed in SGPN. It also picks up many false positives.

D. Instance Segmentation vs. Detection

In this section, state-of-the-art detection methods [21], [43], [58] are compared. Points in the detected bounding boxes are counted as segments for evaluations. As shown in Table IV, our instance segmentation model outperforms the state-of-the-art 3D detectors by $8 \sim 10\%$ in AP evaluations. This is because these detectors rely on the generation of accurate 3D bounding box proposals. As both the range and the variance of objects' sizes and shapes increase, especially for many small objects (*e.g.* pedestrian, cyclists *etc.*), the large diversity of the possible bounding boxes makes it difficult to produce accurate estimations for each object. Moreover, many false positives are mixed into the bounding boxes. (Even the 100% accurate ground truth bounding boxes have 12-16% outliers). Therefore, detectors usually produce lower APs for small objects which have fewer points. Any imperfection (errors of orientation, size, center, height *etc.*) of the detected bounding boxes will heavily reduce the precision (as illustrated in Fig. 1(b-c)).

E. Application in Autonomous Driving

For application in self-driving car, picking up as many as obstacles to avoid possible collisions is the most important thing. Here, we evaluate the models' abilities for accurately picking up obstacles. All kinds of objects are counted as a single class, namely, we calculate the precision and recall without the influence of the classification. We plot the AP curves with IoU thresholds of 0.5 and 0.95 in Fig. 4, which shows that our method can achieve a higher recall with better precision. It achieves a ~90% precision at a recall of 90% in $AP^{0.5}$, which is the best among all these models.

VI. Conclusion

In this paper, we propose a robust baseline instance segmentation solution for large-scale LiDAR point cloud. The proposed model extracts more reliable dense features for discovering far and small objects which have only a few points. We also contribute a large LiDAR point cloud dataset. The new dataset has both bounding box and point-wise labels for instance segmentation and is $3 \sim 20$ times as large as the existing LiDAR dataset.

Acknowledgement

Research is mainly supported by Baidu's Robotics and Auto-driving Lab, in part by the ERC grant ERC-2012-AdG 321162-HELIOS, EPSRC grant Seebibyte EP/M013774/1 and EPSRC/MURI grant EP/N019474/1. We would also like to acknowledge the Royal Academy of Engineering. Victor Adrian Prisacariu would like to thank the European Commission Project Multiple-actOrs Virtual Empathic CARegiver for the Elder (MoveCare).

REFERENCES

[1] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 2

[2] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 2

[3] Baidu. Apollo dataset, `http://data.apollo.auto/?locale=en-us&lang=en`. 2, 4, 5

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1, 2, 5

[5] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016. 2

[6] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015. 2

[7] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1907–1915, 2017. 1, 2, 3, 5, 6

[8] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. 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. 2

[9] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 1, 2

[10] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1355–1361. IEEE, 2017. 2, 3

[11] J. Fang, D. Zhou, F. Yan, T. Zhao, F. Zhang, R. Yang, Y. Ma, and L. Wang. Augmented LiDAR Simulator for Autonomous Driving *IEEE Robotics and Automation Letters*, 2020. 5

[12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition*, 2012. 1, 2, 5

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

[14] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014. 2

[15] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98, 2017. 2

[16] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1, 2, 4

[17] J. Hou, A. Dai, and M. Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. *arXiv preprint arXiv:1812.07003*, 2018. 4

[18] J. Hou, A. Dai, and M. Nießner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. 2019. 2

[19] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[20] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. 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. 2

[21] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12697–12705, 2019. 6

[22] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 828–838, 2018. 2

[23] M. Liang, B. Yang, S. Wang, and R. Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018. 2

[24] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1417–1424, 2013. 2

[25] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017. 2

[26] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2, 4

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[28] C. Liu and Y. Furukawa. Masc: Multi-scale affinity with sparse convolution for 3d instance segmentation. *arXiv preprint arXiv:1902.04478*, 2019. 2

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2

[30] D. Maturana and S. 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. 2

[31] A. Patil, S. Malla, H. Gang, and Y.-T. Chen. The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes. *arXiv preprint arXiv:1903.01568*, 2019. 2, 5

[32] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015. 2

[33] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016. 2

[34] C. R. Qi, W. Liu, C. Wu, H. Su, and L. 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

[35] C. R. Qi, H. Su, K. Mo, and L. 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. 2

[36] C. R. Qi, L. Yi, H. Su, and L. 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. 2, 5

[37] J. Redmon, S. Divvala, R. Girshick, and A. 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. 2

[38] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017. 2

[39] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[40] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2

[41] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008. 2

[42] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *arXiv preprint arXiv:1812.04244*, 2018. 1, 2, 3

[43] S. Shi, X. Wang, and H. Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019. 6

[44] A. Shrivastava and A. Gupta. Building part-based object detectors via

3d geometry. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1745–1752, 2013. 2

[45] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *2011 IEEE international conference on computer vision workshops (ICCV workshops)*, pages 601–608. IEEE, 2011. 2

[46] M. Simon, S. Milz, K. Amende, and H.-M. Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. In *European Conference on Computer Vision*, pages 197–209. Springer, 2018. 2, 3

[47] S. Song, S. P. Lichtenberg, and J. 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. 2

[48] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *European conference on computer vision*, pages 634–651. Springer, 2014. 2

[49] S. Song and J. Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 808–816, 2016. 2

[50] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25. Springer, 2016. 4

[51] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, volume 1, pages 10–15607, 2015. 2, 3

[52] W. Wang, R. Yu, Q. Huang, and U. 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. 2, 4, 5, 6

[53] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. Data-driven 3d voxel patterns for object category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1903–1911, 2015. 2

[54] D. Xu, D. Anguelov, and A. Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018. 2

[55] B. Yang, W. Luo, and R. 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. 1, 2, 5, 6

[56] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia. Ipod: Intensive point-based object detector for point cloud. *arXiv preprint arXiv:1812.05276*, 2018. 2

[57] L. Yi, W. Zhao, H. Wang, M. Sung, and L. Guibas. Gspn: Generative shape proposal network for 3d instance segmentation in point cloud. *arXiv preprint arXiv:1812.03320*, 2018. 2

[58] Y. Zhou and O. 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 (CVPR)*, pages 4490–4499, 2018. 1, 2, 3, 5, 6

[59] M. Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3d representations for object recognition and modeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2608–2623, 2013. 2

[60] F. Zhang, V. Prisacariu, R. Yang, and P. Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 185–194, 2019. 3