PanoNet3D: Combining Semantic and Geometric Understanding for LiDAR Point Cloud Detection

Xia Chen, Jianren Wang, David Held, Martial Hebert
Robotics Institute, Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, Pennsylvania
{xiac, jianrenw, dheld, hebert}@cs.cmu.edu

Abstract

Visual data in autonomous driving perception, such as camera image and LiDAR point cloud, can be interpreted as a mixture of two aspects: semantic feature and geometric structure. Semantics come from the appearance and context of objects to the sensor, while geometric structure is the actual 3D shape of point clouds. Most detectors on LiDAR point clouds focus only on analyzing the geometric structure of objects in real 3D space. Unlike previous works, we propose to learn both semantic feature and geometric structure via a unified multi-view framework. Our method exploits the nature of LiDAR scans – 2D range images, and applies well-studied 2D convolutions to extract semantic features. By fusing semantic and geometric features, our method outperforms state-of-the-art approaches in all categories by a large margin. The methodology of combining semantic and geometric features provides a unique perspective of looking at the problems in real-world 3D point cloud detection.

1. Introduction

With the recent advent of autonomous vehicles, detecting and localizing obstacles on LiDAR point clouds has become a popular research topic. While the output of LiDAR sensors is three-dimensional, it is fundamentally different than true 3D data (such as 3D mesh models). Because of the sweeping mechanics of LiDAR, the data can be densely represented in 2D format (range image). This is commonly referred to as 2.5D [5]. Many popular 3D detectors like PointPillars [11] often ignore such fact and treat the LiDAR data purely as a collection of (x, y, z, i) points (i is the point's intensity or reflectance). Though these works achieve good performance on detection tasks, they do not take advantage of the intrinsic structure of the data.

The simplest way to address this issue is to format the data as normal 2D images and to apply well-studied 2D image detectors on them. However, this solution has several drawbacks. First, spatial coordinates are fundamentally different from images' RGB values. The spatial structure of points cannot be easily extracted from 2D convolutions on the projected range image. Second, range images are not scale-invariant. That is, closer objects contain a much larger number of pixels compared to objects that are far away. Various scales of objects make it hard for the network to generalize.

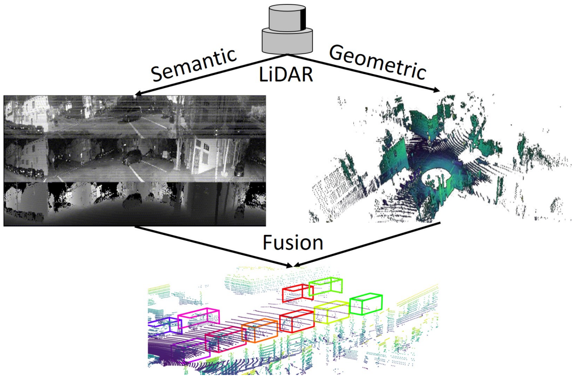

Figure 1. LiDAR can be interpreted semantically and geometrically by nature. *PanoNet3D* utilize both information for LiDAR object detection.

While 2D convolution is not efficient at understanding the 3D geometric structure of point clouds, it can still extract meaningful deep semantic features from range images just like from conventional color pictures. We argue that combining both deep semantic features from range images and raw geometric structures from 3D point clouds together can yield better detection results. More specifically, in the first step, we extract semantics from the projected range image with a fully convolutional network (FCN). The output high dimensional semantic features are then fused with low dimensional raw geometric features which are usually computed by simple geometric manipulations or shallow networks. Final predictions are generated from a main-stage detector with a 3D sparse convolutional network as the backbone. In such manner, we utilize semantics from 2.5D range images while keeping scale invariance in 3D space at the same time. Our experiment shows that additional semantic features significantly improve the detection performances on NuScenes [2] dataset, surpassing the current first-place method CBGS [26] on the official leader-board. The key contributions of this work are the following:

- We introduce PanoNet3D, a novel approach that feeds both deep semantic features and raw geometric features of point cloud data to the main detector. By doing so, the detector is exposed to both the spatial structure of point cloud as well as semantic information natural to the LiDAR sensor.
- PanoNet3D achieves significant improvements on both single-sweep input and multiple-sweep LiDAR input. Our design of temporal aggregation allows aggregating multiple scanned frames for denser input data without the redundancy of repeatedly running the same semantic feature extraction network on these frames.
- The integration of a pano-view feature extractor of PanoNet3D enables natural and simple removal of occluded points after a crop-and-paste data augmentation. Handling occlusions of augmented objects is hard in bird-eye view and is often ignored by BEV detectors.
- PanoNet3D beats state-of-the-art (SOTA) performance on 3D object detection. With several improvements on network architectures, it achieves 0.54 mAP on NuScenes dataset detection challenge, out-performing PointPillars [11] and CBGS [26].

2. Related Work

2.1. Point Cloud Representation

Deep learning architectures take different formats of 3D point clouds as input. The first class consumes raw point clouds directly, including PointNet [16], PointNet++ [17], and PointRCNN [19]. This type of approaches require no pre-processing of point clouds (such as voxelization or rendering), but their performance suffers when the scene is large and sparse. For common LiDAR sensors, a single sweep typically contains over 50,000 points. So these networks usually need to down-sample input, losing the resolution of raw data.

Some networks simply treat point clouds as a bird-eye-view (BEV) image, *e.g.*, AVOD [10] and Complex-YOLO [20]. BEV images work particularly well for LiDAR point clouds as we usually only care about x-y (2D) localization of objects. This formatting allows 2D image detection frameworks to be re-applied on point clouds at the cost of partly losing vertical geometric structure information.

Another type of 3D point cloud formatting is voxelization. Examples include VoxelNet [25], SECOND [21], and PIXOR [22]. Voxelized point cloud usually has a finite spatial size with pooling as the technique to convert per-point features to per-voxel features. The performance of this class of detectors is usually linked to voxel resolution.

Recently, LaserNet [15] shows that when the size of the training dataset is large enough, detectors performing on the perspective view of point clouds (range images) can achieve performance on par with BEV detectors. Similarly, MVF [24] extracts semantics from both range images and BEV images with two 2D convolutional towers. For point clouds scanned by LiDARs, the range image format is much denser and has no range limits compared to BEV-based representations.

2.2. Object Detection

Object detection has traditionally been studied on 2D images. Various Convolutional Neural Network (CNN) detectors are proposed since R-CNN [4]. These detectors can be categorized into two major classes: two-stage detectors and single-stage detectors. Two-stage detectors usually consist of a Region Proposal Network (RPN) [18] that produces candidate region proposals and a second stage network regressing the final bounding boxes. On the other hand, single-stage detectors rely on a Single Shot Detector (SSD) [13] that densely produces bounding box predictions with a single fully convolutional network (FCN). Single-stage detectors are simpler and typically faster than two-stage detectors. With focal loss [12] alleviating the problem of foreground-background class imbalance, singlnie-stage detectors can achieve similar or even better results compared to two-stage detectors.

Object detection on 3D point clouds is a more recent research topic. Many works borrow ideas from 2D image detectors as there is no fundamental difference between these two tasks. The only necessary modification of the detection head is the regression of additional parameters required to define 3D bounding boxes. Many modern point cloud detectors adopt single-stage frameworks, including SECOND [21], PointPillars [11], PIXOR [22], and LaserNet [15]. Single-stage point cloud detector is more favorable for autonomous driving applications due to its simplicity and fast inference speed.

2.3. Detection on LiDAR Point Cloud

Object detection on LiDAR point cloud data has several domain-specific problems. We discuss convolution types, temporal aggregation, and data augmentation below.

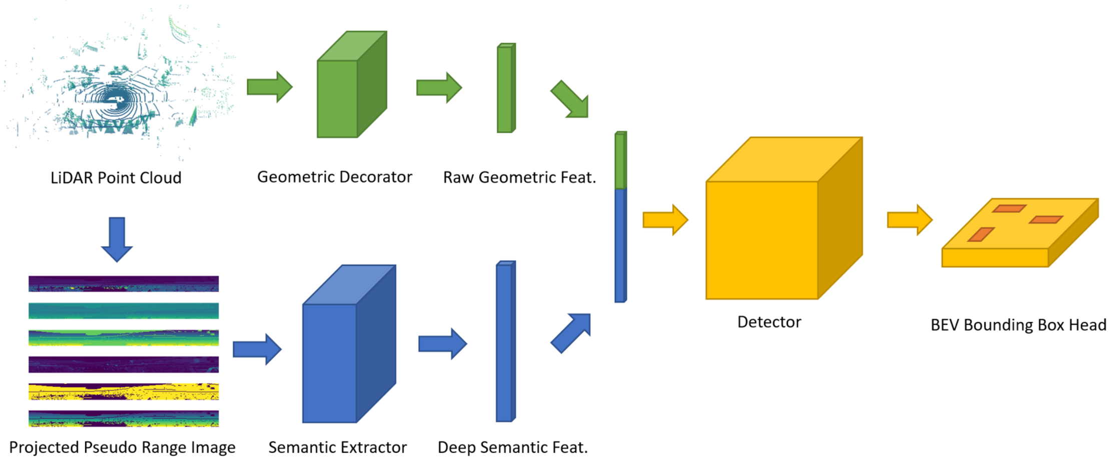

Figure 2. PanoNet3D's framework for point cloud detection. The top branch takes LiDAR point cloud as input and decorates raw point features with several simple local geometric features. The lower branch converts point cloud to pseudo range image and feeds it into a 2D FCN to get per-pixel deep semantic feature. The output features of these two branches are then aggregated and passed to the main detector. A final bounding box head generates detected proposals on the BEV plane.

2.3.1 Convolution Types

Intuitively, voxelized point cloud data is a 3D tensor and thus the detector should consist of 3D convolution layers. Because of the sparsity of LiDAR data, GPU-accelerated sparse implementation of 3D convolution is usually applied [21] in order to significantly reduce time and memory consumption. PointPillars [11] converts 3D inputs to 2D by using a pillar feature encoder that outputs per grid feature embedding on the x-y (BEV) plane. This allows the detector to use regular 2D convolutional layers that are highly optimized on GPUs by many deep learning libraries.

2.3.2 Temporal Aggregation

Some detectors aggregate multiple consecutive LiDAR sweeps and show that temporal information can improve detection performance. FaF [14] treats temporal information as an additional dimension of the input tensor, i.e., multiple frames are appended along a new dimension to create a 4D tensor. SECOND [21] proposes a simpler solution that adds relative temporal stamp to each point as an extra input channel (the input tensor remains 3D). We need to pay special attention to ego motion during temporal aggregation as the reference coordinate system shifts with the ego vehicle's movement.

2.3.3 Multi-view Aggregation

MV3D [3] proposed a multi-view detection network which has two detection branches, one for BEV and one for range view (RV). The results of the two branches are fused afterwards. While MV3D explores the possibility of jointly using both BEV and RV for point cloud detection, the paper does not give the justification of why the two views should be used jointly. RV is what the sensor sees in raw, from which we can effectively extract semantic features just like from RGB camera images. On the other hand, BEV is scale-invariant regardless of the distance to the sensor, so actual geometric structures are preserved in BEV. The need of using both semantic and geometric information leads to the combination of RV and BEV.

2.3.4 Data Augmentation

Data augmentation is extremely important for training LiDAR detection networks in autonomous driving scenarios, as real-world datasets usually have severe problem of class imbalance. For example, about half of labeled instances in NuScenes [2] dataset are cars. A copy-and-paste augmentation schematic are used in many popular detectors including SECOND [21], PointPillars [11], and CBGS [26]. This method crops ground truth bounding boxes from other frames and pastes them onto the current frame's ground plane. Hu et al. [7] argue that maintaining correct visibility during augmentation makes significant improvements in detection results. The visibility information can be calculated and explicitly expressed. However, with projection on range images, visibility is naturally encoded and requires less computation.

3. Method

The structure of PanoNet3D is illustrated in Fig. 2. This framework can be divided into two stages. 1) **Feature extraction stage**: A 2D FCN generates deep semantic feature maps from projected pseudo range images. Meanwhile, a geometric decorator generates each point's raw geometric

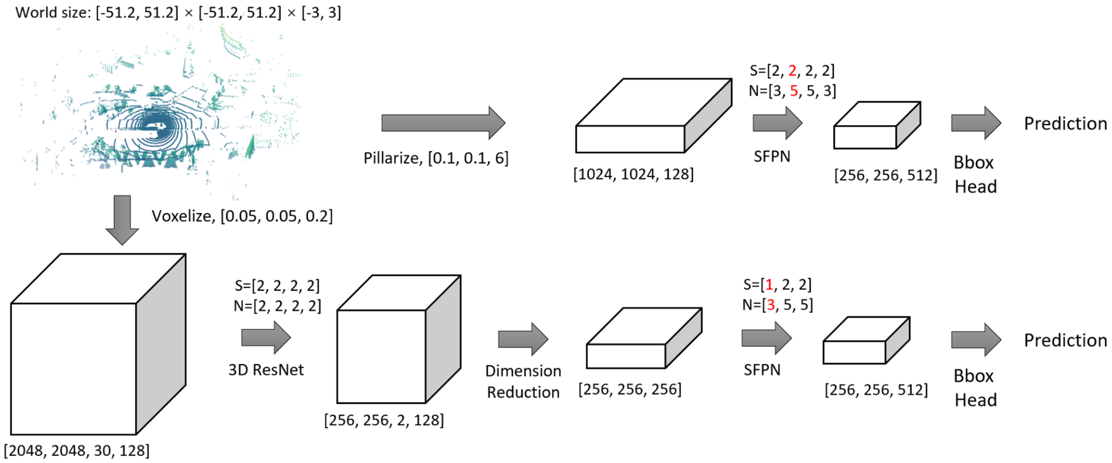

Figure 3. Structure of a detector with 2D pillars or 3D voxels as input. The initial feature is 128 dimensional. We limit the size of the whole scene to $[-51.2, 51.2] \times [-51.2, 51.2] \times [-3, 3]$ meters in x, y, z direction. The networks consist of a few layers of ResNet basic blocks. S denotes the stride of each layer and N denotes the number of blocks. The generated feature map of SFPN has the same resolution as the layer marked in red.

features, including its global position and local displacement relative to the center of its residing voxel. The semantic features and geometric features are then aggregated and passed to the next stage. 2) **Detection stage**: Per-point features are converted to per-voxel features by a simple symmetric operation such as max and average pooling. A single-stage detector then predicts oriented 3D boxes and their confidence score based on pre-defined anchors. We describe the details of each component of the network in the following sections. First, we will introduce pseudo-range-image semantic extractor and voxel geometric decorator, the combination of which generates a feature vector for each point. Then we will discuss how to temporally aggregates features from multiple frames. Last, we will describe how the main detector gives prediction as well as training and implementation details.

3.1. Pseudo Range Image and Semantic Extractor

The outputs of a common LiDAR sensor are range images by nature. However, since many LiDARs' rings are not evenly spaced (sometimes the ring information is not even available), we manually project 3D point clouds back to 2D range images with evenly spaced projection angles. For the NuScenes [2] dataset, we choose the horizontal projection angle range and resolution to be $[x_{min}, x_{max}, x_{step}] = [-180°, 180°, 0.3125°]$ and vertical counterparts to be $[y_{min}, y_{max}, y_{step}] = [-30°, 10°, 1.25°]$. It is possible that more than one LiDAR point is mapped to the same pixel on range image. In this case, we simply keep the closest point and discard the rest. In addition to point's range r, we also encode height h, elevation angle ϕ and reflectance i in separate channels. Similar to LaserNet [15], the last channel of the image is a flag indicating whether a pixel contains a projected point. We call this multi-channel tensor (an example is shown in Fig. 4) **pseudo range image** of LiDAR.

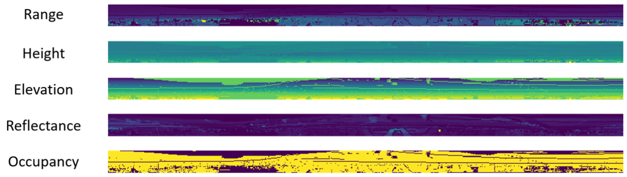

Figure 4. An example of projected pseudo range image with five channels. From top to bottom: range r, height h, elevation angle ϕ, reflectance i, and occupancy mask m.

To extract semantic features from the pseudo range image, we adopt the Semantic FPN (SFPN) design from [9]. It aggregates the features from all levels of FPN layers into a single output with per-pixel semantic embedding. The SFPN's backbone is a ResNet34 [6] without the first layer (conv1). For each projected LiDAR points, the SFPN generates a 64-dimensional semantic feature vector. The feature extractor is not trained with direct supervision. Instead, the feature vectors are passed to the main detector where they receive supervision from the final classification and localization loss.

3.2. Voxelization and Geometric Decorator

The input 3D point cloud is voxelized before being passed into the detector. We experiment with two types of voxelization: (1) regular 3D voxelization and (2) pillarization, where points are organized in vertical columns similar to PointPillars [11]. Pillarization can be seen as a special type of 3D voxelization with only one layer of voxels vertically. We annotate each point's global position $[x, y, z]$ with its distance to the LiDAR origin r and its position rel-

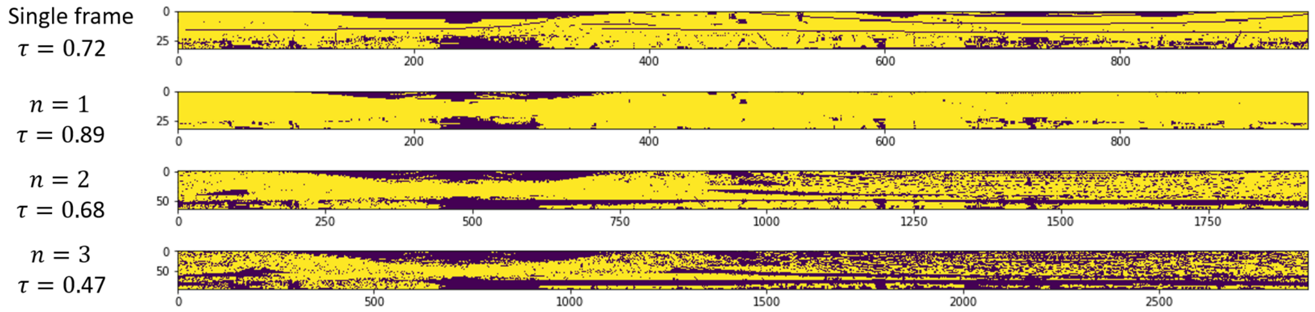

Figure 5. Occupancy maps showing the results of spatial multi-frame fusion. Yellow color indicates the pixel is occupied by a projected LiDAR point. τ is the occupancy rate (the number of occupied pixels over the number of total pixels). From top to bottom: original single frame, 10-frame aggregation with $n = 1$, $n = 2$, $n = 3$.

ative to the voxel's center at $[x_c, y_c, z_c]$. The resultant geometric feature can be expressed as a 7-dimensional vector: $[x, y, z, r, x - x_c, y - y_c, z - z_c]$. Optionally, a simple one-layer fully connected network (similar to the design of like VoxelNet [25]) can be applied to each voxel to extract more local features.

3.3. Semantic and Geometric Feature Aggregation

We use concatenation as the method of aggregating semantic features and geometric features together. The embeddings of those points that are not assigned with semantic features (discarded during range image projection) are padded with zeros. For each voxel, a locally aggregated feature vector is generated from point-wise embeddings via symmetric pooling operations: we apply element-wise max pooling on high-dimensional semantic features and average pooling on low-dimensional geometric features. Now, we are able to obtain a voxel-wise feature vector that encodes both semantic and geometric features of the set of points inside the voxel.

3.4. Temporal Aggregation

When multi-frame data is available, we add a timestamp t as an additional channel to each point. Such temporal aggregation requires a new design of range image projection for semantic feature extractor. For example, when we aggregate 10 consecutive sweeps, the point cloud is now 10 times denser and thus a large portion of points will be discarded by the pseudo range image projection process. To prevent such loss of information, we propose two solutions: temporal multi-frame fusion and spatial multi-frame fusion. The ablation study of these two aggregation methods is discussed in Section 4.4.

3.4.1 Temporal Multi-Frame Fusion

Temporal multi-frame fusion retrieves the pseudo range image at each frame respectively, and then concatenates them along a new dimension to form a batch of images as input. This is equivalent to running the same feature extractor on each individual frame.

3.4.2 Spatial Multi-Frame Fusion

In spatial multi-frame fusion, we transform all frames' points to the keyframe's coordinate system and increase the resolution of the pseudo range image to allow more points to be projected. The main design choice required here is the multiple n between the new linear resolution and the single-frame resolution. Fig. 5 shows the occupancy map of different n. When n is too large, the range image becomes sparse and inefficient for dense feature extraction. Ideally, we want the occupancy rate τ to be as close as possible to the original one. For NuScenes dataset (20 Hz frame rate), we choose $n = 2$ for 10-frame aggregation. Notice that in this setting, the range image has only $4\times$ pixels while the 3D point cloud has $10\times$ points. When multiple points are projected to the same pixel, we prioritize those with timestamps closer to the key-frame. Spatial multi-frame fusion allows us to enhance the resolution of input range image efficiently without too much redundancy caused by close or repeating points.

3.5. Detector

As discussed in Section 3.2, the input of the detector can have two types of formats: 2D pillars or 3D voxels. The detector is designed accordingly. For 2D pillar input, we can directly apply an SFPN as the backbone to get the final feature map. On the other hand, for 3D voxel input with the shape of $[H, W, D, C]$, we first adopt a sparse 3D ResNet to downscale the tensor to $[H/s_H, W/s_W, D/s_D, C]$, where s_H, s_W, s_D are the downscale factors. Then we lower the dimension of the tensor by reshaping it to $[H/s_H, W/s_W, D \times C/s_D]$, so that it can be similarly fed into a 2D RPN to generate the BEV feature map. The bounding box regression head is attached to the feature map. We follow the multi-group head design as in CBGS [26]. The detailed detector structure is illustrated

	car	truck	bus	trailer	cons.	pedes.	mcycle	bicycle	cone	barrier	mAP
Point Pillars [11]	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9	30.5
SARPNET [23]	59.9	18.7	19.4	18.0	11.6	69.4	29.8	14.2	44.6	38.3	32.4
CBGS [26]	**81.1**	**48.5**	**54.9**	42.9	10.5	**80.1**	51.5	22.3	70.9	**65.7**	52.8
Ours	80.1	45.4	54.0	**51.6**	**15.1**	79.1	**53.1**	**31.3**	**71.8**	62.9	**54.5**

Table 1. Detection mAP by categories compared on NuScenes test set.

	car	truck	bus	trailer	cons.	pedes.	mcycle	bicycle	cone	barrier	mAP
CBGS* [26]	79.8	45.8	58.6	31.1	11.7	74.8	38.3	14.2	55.0	56.6	46.6
Ours w/o sem. feat	80.1	44.2	59.1	32.2	10.9	74.5	40.2	20.2	57.8	55.6	47.5
Ours	**82.6**	**49.9**	**62.4**	**36.3**	**11.8**	**80.6**	**53.8**	**33.8**	**67.2**	**64.5**	**54.3**

Table 2. Detection mAP by categories compared on NuScenes validation set. *: reproduced with officially released code and our experimental setup. The second line shows the result of our model without aggregation of range-image-based semantic features (row i. in Tab. 3).

in Fig. 3.

3.6. Data Augmentation

We make several improvements on data augmentation schematics used in SECOND [21]. Ground truth boxes are cropped and saved offline, and then pasted onto the current frame's ground plane. Additionally, we allow the augmented object to randomly rotate around the LiDAR center within 45 degrees (its distance to the center of the frame remains unchanged). We also perform global augmentations that randomly transform the whole point cloud, including translation (within [-0.2m, 0,2m]), rotation (within [-45°, 45°]) and scaling (within [0.95x, 1.05x]).

Newly pasted objects may occlude with other objects. Traditional methods often ignore such occlusions, and keep all augmented objects even they are not detectable by a real LiDAR sensor. However, our method naturally solves this problem by removing all annotations that have less than 3 points projected on the pseudo range image. As a result, the objects that should not be visible to the LiDAR are easily filtered out.

3.7. Implementation Details

Our implementation is based on CBGS's [26] official code base[1]. All object classes share the detection backbone except an exclusive two-layer regression head for each category group. The experiments are conducted on 4 NVIDIA 1080 Ti with PyTorch's official implementation of multi-GPU synchronized batch normalization. We train the network with Adam optimizer [8], one-cycle policy [1] (max learning rate: 0.0001, division factor: 5), and the batch size of 4 for 20 epochs. The IoU threshold of the non-maximum suppression is 0.2 and the maximum number of final predicted bounding boxes is 100. The anchors selected as the mean values of all labels. On 1080 Ti, our model runs at 20 fps during inference.

4. Results

We first compare the quantitative performance of our method against other SOTA methods on the NuScenes dataset, while the results on the KITTI dataset are presented in the supplemental materials. Qualitative results (visualization of predictions) are shown in Fig. 4.2.4. Next, we conduct ablation studies to explain how we make the decisions during network design and show where the performance improvements come from.

4.1. Main Results

We submitted the results of our method to the NuScenes test server. In Tab. 1, we compare PanoNet3D against other methods on the NuScenes detection leaderboard. Our overall mAP surpasses the current first-place method CBGS [26] by 1.7%. For fairness, we also compare our method against CBGS's reproducible performance on NuScenes validation set in Tab. 2. The results of CBGS are reproduced with its official code and under the same experimental setup as ours. Our model improves mAP on all categories including 2.5% on car. Higher performance gains are observed on 'tall-and-thin' object categories such as bicycle and cone. These objects have larger projection sizes on depth image rather than on the BEV plane, so our model can achieve better overall understandings compared to traditional detectors. We also re-trained our model and baseline (CBGS) from scratch for 4 more times with different random seeds. The performance errors of all trails are within 0.4% mAP.

4.2. Ablation Study

Tab. 3 shows a series of ablation studies. Based on these results, we can make the following key observations. Each

[1] https://github.com/poodarchu/Det3D

	Method	Range image feat.	# Input frames	Voxelization	BEV resolution(m)	mAP
a.	Point Pillars [11]	-	1	Pillar	0.25	24.0
b.	Point Pillars [11]	-	10	Pillar	0.25	29.5
c.	Ours	✗	1	Pillar	0.25	31.5
d.	CGBS [26]	-	1	Voxel	0.1	39.2
e.	Ours	✓	1	Voxel	0.1	43.1
f.	Ours	✓	1	Pillar	0.25	45.2
g.	Ours	✗	10	Voxel	0.1	46.3
h.	CGBS [26]	-	10	Voxel	0.1	46.6
i.	Ours	✗	10	Voxel	0.05	47.5
j.	Ours	✓	10	Pillar	0.25	47.9
k.	Ours	✓	10	Pillar	0.1	48.0
l.	Ours	✓	10	Voxel	0.1	52.9
m.	Ours	✓	10	Voxel	0.05	54.3

Table 3. Ablation studies on NuScenes validation set. 'range image feat.' means whether the detector uses perspective-view-based semantic feature extractor. 'BEV resolution' means the x-y resolution when the point cloud is voxelized.

line of the result is represented with small letters. Across all factors, we find that the pano-view semantic feature extractor contributes the most to the performance gain.

4.2.1 Baseline Comparison

The major difference between PanoNet3D and other detectors is its range-image-based semantic feature extractor. Without the aggregation of extracted semantic features, our pillar-based detector should have a similar framework to PointPillar's [11] except for the backbone design. Our pillar-based baseline model achieves better performance than PointPillars (a.-c.). The most likely reason is our SFPN backbone is able to utilize multi-level features more efficiently. On the other hand, for voxel-based detectors, CBGS has similar performance to our baseline model (without the semantic feature extractor), showing that our improvements against CBGS do not come from the different detector backbones used by PanoNet3D and CBGS (g.-h.).

4.2.2 Range Image Semantic Feature Extractor

For single-frame pillar-based detectors, semantic features extracted from range images significantly improve the average mAP by 13.7% (c.-f.). With the help of a semantic feature extractor, our single-frame model is able to achieve comparable results against other multi-frame models. For multi-frame voxel-based detectors, semantic features also improve the average mAP by over 6% (g.-l., i.-m.). From Tab. 2, we can further observe that combining deep semantic features with raw geometric features leads to improvements across all 10 categories. The perspective view is natural to LiDAR sensors and contains semantics that cannot be extracted from real-world Euclidean space, which helps the detector to achieve a better overall understanding of the scene.

4.2.3 Pillar or Voxel

For single-frame input, the pillar-based detector performs slightly better (e.-f.), while for multi-frame input, the voxel-based detector is more favorable (k.-l.). One possible explanation is that the pillar-based detector is sufficient for the density of a single-frame point cloud and can prevent overfitting caused by more complex 3D convolutions. Multi-frame input has much denser point clouds whose features can not be well extracted by the simpler pillar feature extractor.

4.2.4 BEV Resolution

Finer grids of voxelization usually lead to better detection performance. However, its impact is less dominant than other factors. Increasing BEV resolution from 0.25m to 0.1m improves mAP of 10-frame pillar-based detector by 0.1% (j,-k.), and increasing BEV resolution from 0.1m to 0.05m improves mAP of 10-frame voxel-based detector by 1.4% (l.-m.).

4.3. Voxel Feature Pooling Methods

We test all pooling methods during aggregating point-wise features to voxel-wise features. With all combinations shown in Tab. 4, we conclude that max pooling on higher-dimensional semantic features with average pooling on lower-dimensional geometric features yields the best results.

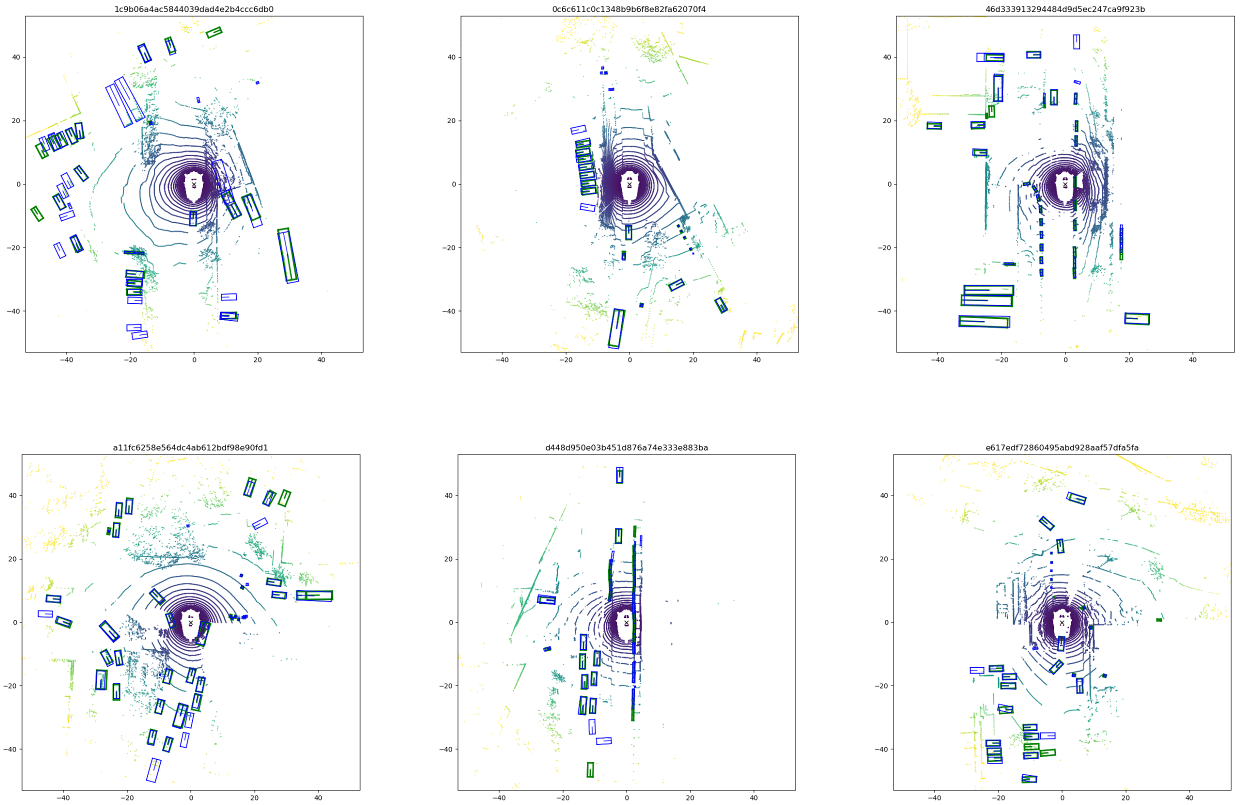

Figure 6. Detection examples of PanoNet3D on NuScenes dataset. Ground truths are annotated in green boxes and detection results are annotated in blue boxes.

Deep sem. feat. aggr.	Raw geo. feat. aggr.	mAP
Max	Max	44.6
Max	Average	45.2
Average	Max	44.3
Average	Average	44.9

Table 4. Study on pooling methods during voxel-wise feature aggregation. The experiment is done with a single-frame pillar detector as baseline.

4.4. Temporal Aggregation Methods

We also experiment with different temporal aggregation approaches, the results of which are shown in Tab. 5. Spatial multi-frame fusion with $n = 2$ is the best among them, showing that our previous analysis is correct. However, notice that the optimal n is not a fixed value. If the number of aggregated frames or the input dataset changes, we might need to change n accordingly to accommodate the data.

Aggregation method	n	mAP
Temporal 10-frame fusion	-	52.9
Spatial 10-frame fusion	1	53.1
Spatial 10-frame fusion	2	54.3
Spatial 10-frame fusion	3	52.2

Table 5. Results of different multi-frame temporal aggregation approaches.

5. Conclusion

We explore the possibility of combining both semantic and geometric understanding of 3D LiDAR point clouds. Experiments show that both objects' appearance to the sensor and their actual shapes in 3D space are important for detection networks. By enhancing each point's raw geometric coordinates with deep semantic features extracted from pseudo range images, we are able to achieve a better understanding of the scene and better overall detection performance.

References

[1] Another data science student's blog – The 1cycle policy. 4326

[2] 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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3 2019. 4322, 4323, 4324

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3D Object Detection Network for Autonomous Driving. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:6526–6534, 11 2016. 4323

[4] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. 4322

[5] R. M. Goldstein, H. A. Zebker, and C. L. Werner. Two-Dimensional Phase Unwrapping. *Radio Sci.*, 23:713–720, 1988. 4321

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 4324

[7] P. Hu, J. Ziglar, D. Held, and D. Ramanan. What you see is what you get: exploiting visibility for 3D object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12 2019. 4323

[8] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 12 2015. 4326

[9] A. Kirillov, R. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. *arXiv preprint arXiv:1901.02446*, 2019. 4324

[10] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. Joint 3D proposal generation and object detection from view aggregation. In *IEEE International Conference on Intelligent Robots and Systems*, pages 5750–5757. Institute of Electrical and Electronics Engineers Inc., 12 2018. 4322

[11] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. PointPillars: Fast encoders for object detection from point clouds. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 12689–12697, 2018. 4321, 4322, 4323, 4324, 4326, 4327

[12] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017. 4322

[13] 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, 2016. 4322

[14] W. Luo, B. Yang, and R. Urtasun. Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3569–3577. IEEE Computer Society, 12 2018. 4323

[15] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington. LaserNet: An efficient probabilistic 3D object detector for autonomous driving. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:12669–12678, 3 2019. 4322, 4324

[16] 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 IEEE Conference on Computer Vision and Pattern Recognition*, volume 2017-January, pages 77–85, 11 2017. 4322

[17] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 2017-December:5100–5109, 6 2017. 4322

[18] 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. 4322

[19] S. Shi, X. Wang, and H. Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 770–779. IEEE Computer Society, 12 2018. 4322

[20] M. Simon, S. Milz, K. Amende, and H.-M. Gross. Complex-YOLO: Real-time 3D object detection on point clouds. *arXiv preprint arXiv:1803.06199*, 3 2018. 4322

[21] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. *Sensors (Switzerland)*, 18(10), 10 2018. 4322, 4323, 4326

[22] B. Yang, W. Luo, and R. Urtasun. PIXOR: Real-time 3D object detection from point clouds. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 7652–7660. IEEE Computer Society, 12 2018. 4322

[23] Y. Ye, H. Chen, C. Zhang, X. Hao, and Z. Zhang. SARPNET: Shape attention regional proposal network for LiDAR-based 3D object detection. *Neurocomputing*, 379:53–63, 2 2020. 4326

[24] Y. Zhou, P. Sun, Y. Zhang, D. Anguelov, J. Gao, T. Ouyang, J. Guo, J. Ngiam, and V. Vasudevan. End-to-end multi-view fusion for 3D object detection in lidar point clouds. In *Conference on Robot Learning*, pages 923–932, 2019. 4322

[25] Y. Zhou and O. Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4490–4499. IEEE Computer Society, 12 2018. 4322, 4325

[26] B. Zhu, Z. Jiang, X. Zhou, Z. Li, and G. Yu. Class-balanced grouping and sampling for point cloud 3D object detection. *arXiv preprint arXiv:1908.09492*, 8 2019. 4322, 4323, 4325, 4326, 4327