Combining Semantic and Geometric Understanding for Modern Visual Recognition Tasks

Xia Chen

CMU-RI-TR-20-16
April 10, 2020

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Professor Martial Hebert, *chair*
Professor David Held
PhD Allison Del Giorno

Submitted in partial fulfillment of the requirements for the degree of Master in Robotics.

Abstract

For autonomous driving perception, visual data, such as camera image and LiDAR point cloud, consists of two aspects: semantic feature and geometric structure. While usually studied separately, these two properties can be combined and jointly used by a unified framework. In this work, we apply and validate this idea on modern visual recognition tasks. For image panoptic segmentation, we introduce position-sensitive embedding that is able to distinguish instances with similar appearance but at different locations. Such embedding allows a simple single-stage network to generate panoptic segments in a highly efficient way. For LiDAR point cloud detection, we fuse deep semantic feature extracted from pseudo range image with raw geometric information. This additional feature fusion stage significantly improves the detector's mAP on all categories, outperforming other state-of-the-art approaches. The methodology of combining semantic and geometric features gives a unique perspective of looking at the problems in modern visual recognition tasks.

Acknowledgments

I would like to thank my advisor Prof. Martial Hebert, all committee members, Peiyun Hu, Jianren Wang, and Mingyuan Zhong for all the help and meaningful discussions about this work.

Contents

When this dissertation is viewed as a PDF, the page header is a link to this Table of Contents.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Recent advances in deep learning on visual data have started a revolution in designing the perception systems of autonomous vehicles. Compared with traditional methods, deep neural networks are much better at extracting meaningful semantic information from images. For example, fully convolutional network (FCN) is a popular and effective approach to encode dense per-pixel embedding for semantic segmentation. However, the application of autonomous driving requires a better scene understanding approach that not only should be able to distinguish objects from background, but also locate each instance precisely on the map. We call these two aspects of the problem *semantic scene understanding* and *geometric scene understanding* respectively. In practice, semantic scene understanding is well-studied on 2D image data while geometric scene understanding is usually applied to 3D point cloud data. The reason behind this is simple: images have richer semantic information due to higher dimensionality and density, and point clouds are easier for localization because each point naturally contains 3D positional information.

This thesis explores an interesting idea: combining semantic scene understanding and geometric scene understanding together for modern visual recognition tasks in a unified framework. On images, we propose a method that utilizes position-sensitive embedding to generate simultaneously semantic and instance masks for panoptic segmentation. On point clouds, we show that adding deep semantic information to raw geometric point features can significantly improve the 3D detection performance. Experiments demonstrate that jointly using semantic and geometric information is highly effective on many common perception tasks including both detection and segmentation.

1.2 Thesis Overview

This thesis can be divided into two main parts as the above idea is tested on two types of input data, images and point clouds, both widely used in autonomous driving perception. I will now delineate the organization of this thesis following this division.

1.2.1 PanoNet

Chapter 3 introduces PanoNet for panoptic segmentation on images. PanoNet is a novel one-stage segmentation network using a single FCN as backbone. The network generates per-pixel embedding to be then clustered into instance-level masks. In additional to the raw RGB features, we also feed the network with spacial information so that it can distinguish objects with similar appearance but at different locations. PanoNet is conceptually simpler than the traditional two-branch design (one for semantic segmentation and one for instance segmentation) and does not require any fusion process to combine the two results together. Our key contributions are listed below:

- We propose PanoNet, a single-stage panoptic segmentation method using highly similar structure and sharable parameters for both semantic and instance segmentation branches. The design of PanoNet is much simpler than that of other state-of-art solutions.
- PanoNet utilizes position of objects on top of their feature embedding for segmentation without region proposals. We develop a unique training procedure that allows the network to learn how the position-sensitive information should be encoded, resulting in a significant performance improvement.
- PanoNet has excellent speed-accuracy trade-off in terms of both memory and time consumption. With less than 3 GB memory used, PanoNet achieves real-time inference speed (20 FPS) on high-resolution 2048×1024 images. On

the other hand, PanoNet still has decent performance both visually and on the panoptic quality metrics.

1.2.2 PanoNet3D

On point cloud data, 3D object detection is the most common task. Chapter 4 proposes PanoNet3D to solve this problem. Similar to PanoNet, per-point semantic embedding is extracted from an FCN. The input of this encoding network is generated by projecting all points onto the perspective plane of the LiDAR sensor. Then, the semantic embedding is concatenated with raw point features (including 3D coordinates and reflected intensity) and fed into a voxel-based network to generate final detection results. The key contributions of this work are the following:

- We introduce PanoNet3D, a novel approach that feeds both deep semantic feature and raw geometric feature of point cloud data to main detector. By doing so, the detector is exposed to both spatial structure of point cloud and semantics natural to the LiDAR sensor.
- PanoNet3D achieves significant improvements on both single-sweep input and multiple-sweep input. Our design of temporal aggregation allows using multiple scanned frames for denser input data without the redundancy of repeatedly running the same semantic feature extraction network on these frames.
- 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 [35] and CBGS [82].

Chapter 2

Background

2.1 Autonomous Vehicle Perception

Autonomous driving systems has been proven to have many benefits compared to human drivers, including preventing accidents caused by human errors, cutting emission, and reducing driving-related stress [72]. Perception system is a vital component for autonomous vehicles as it provides understanding of the environment to the downstream parts (usually localization and mapping) [62]. A typical perception system receives images from cameras and point clouds from LiDAR sensors. These two types of information introduce different tasks. For images, panoptic segmentation extracts most useful information as it includes both instance segmentation and semantic segmentation, such that *things* like cars and pedestrians as well as *stuff* like road and vegetation can be segmented at the same time. For LiDAR point clouds, we usually cares about 3D detection as it can output estimated positions of obstacles. Next, we will discuss these two tasks separately.

2.2 Panoptic Segmentation on Images

Things and *stuff* have long been studied separately: the former formulated as tasks known as object detection or instance segmentation, the latter formulated as tasks known as semantic segmentation. To find an effective design of a unified vision system that generates rich and coherent scene segmentation, panoptic segmentation [32] was introduced, and it has become particularly important in autonomous driving and augmented reality [69].

The uniqueness of panoptic segmentation lies in two aspects: First, this task should be solved efficiently since it needs to be fast in real applications. Second, it unifies the feature presentation and network architecture for semantic segmentation and instance segmentation. However, to our best knowledge, there are no methods satisfying both requirements at the same time.

Although semantic segmentation and instance segmentation are highly relevant, very dissimilar methods has been adopted for each task. For semantic segmentation, FCNs with specialized backbones enhanced by dilated convolutions [13] dominate popular leader boards [16, 21]. For instance segmentation, region-based Mask R-CNN [28] with a Feature Pyramid Network (FPN) [41] backbone has been used as a foundation for all top entries in recent recognition challenges [7, 49, 79]. To make full use of these top-performing methods, most previous works use two parallel branches, one for instance-level recognition with RPN [56] and one for semantic-level segmentation [32, 33, 52]. However, neither inference efficiency nor the correlation between these two highly relevant tasks is considered. In these proposed methods, only the feature extraction backbone is shared. Panoptic FPN [33] predicts instance segmentation masks without using any semantic segmentation result. Similarly, BiSeg [52] runs R-FCN [18] multiple times and applies a max operation to produce per-pixel likelihood of the object category.

To alleviate the time inefficiency caused by re-segmentation, our goal is to design a one-stage panoptic segmentation algorithm using FCNs as backbone. Previous work [19] has shown that semantic segmentation frameworks can also be used to distinguish different instances by training the network with a discriminative loss function and then clustering pixel-wise feature embedding into masks of instances. However, it does not take positional information into consideration because of the translational invariance of FCNs. Thus, this approach is unable to segment instances with similar appearance and feature (e.g., two cars of the same model).

We argue that both appearance and position of objects are important when using pixel-wise embedding to cluster instance masks. Inspired by Liu et al. [43], we add spacial information as additional input and train the network to be position-sensitive.

Experimental results show that the position-sensitive feature embedding leads to significant improvement of performance. With light-weight backbone like ICNet [77], we achieve real-time performance and accuracy comparable to the state-of-the-art. Fig. 2.1 shows the speed-accuracy trade-off of panoptic segmentation methods. To the best of our knowledge, our method is the first approach achieving real-time performance on high-resolution images on a single GPU.

2.3 3D Detection on Point Clouds

Both industry and academia are excited about the advent of autonomous vehicles. Significant progress has been made since the vision-guided Mercedes-Benz robotic Van in 1980 that used LiDAR sensor and computer vision technology [6]. Since then, detecting and localizing obstacles on LiDAR point clouds has become a popular research topic. While LiDAR sensors output 3D point clouds, it is fundamentally different than true 3D data (such as 3D mesh models). Because of the sweeping mechanics of LiDAR, the data can be represented in 2D format (range image). This is commonly referred as 2.5D [24]. Many popular 3D detectors like PointPillars [35] 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 decent performance on detection tasks, they completely destroy the natural organization of the data.

To address this issue, the simplest way is to format the data as normal 2D images and apply 2D image detectors on them. However, this solution has several drawbacks. First, spatial coordinates are different from image's RGB channels. 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 have much larger number of pixels compared to objects that are far away. Various scales of similar objects make it hard for the network to generalize.

Inspired by previous exploration on image segmentation. We argue that combining both deep semantic features from range images and raw geometric structures from 3D point clouds together can achieve best results. At the first step, we extract semantics from 2.5D range image with FCN. The resultant high dimensional semantic features are then fused with low dimensional raw geometric features. Final predictions are generated from a 3D sparse convolutional network. In such manner, we utilize semantics from 2.5D range images and still keep scale invariance in real 3D space at the same time. Our method shows that additional semantic features significantly improve the detection performances on NuScenes [8] dataset, surpassing the current first-place method CBGS [82] on the official leader-board.

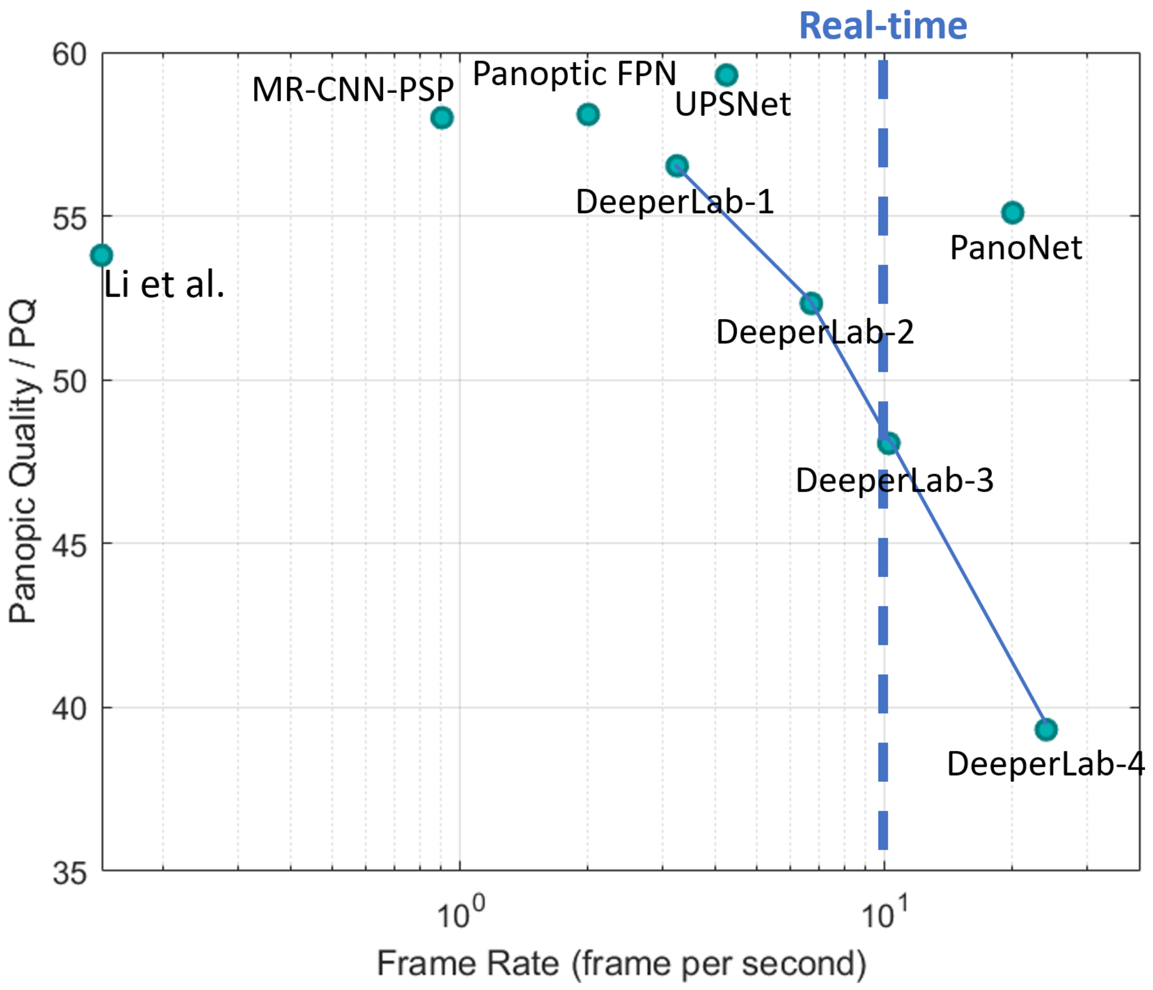

Figure 2.1: Panoptic quality (PQ) vs inference speed on Cityscapes validation set (full scale 1024 $\times$ 2048 high-resolution image). The listed methods: Li et al.[36], Mask R-CNN and PSPNet (MR-CNN-PSP) [33, 64], Panoptic FPN [33], UPSNet [64], DeeperLab [67] variants and our PanoNet. Our model runs significantly faster than other methods with comparable performance.

Chapter 3

PanoNet

3.1 Related Work

3.1.1 Semantic Segmentation:

Semantic segmentation is a classical and well-studied computer vision problem. Convnets have been long used to exploit the contextual information for segmentation [17, 22, 26, 53, 68, 71]. Recently, a prevalent family of approaches based on Fully Convolutional Networks (FCNs) [46] have demonstrated state-of-art performance on several benchmarks [7, 16, 21, 79]. Four great ideas have been proposed among these methods. The first idea is fusing multi-scale feature [10, 25, 63, 76], since higher-layer feature contains more semantic meaning but less local information, and combining multi-scale features can improve the performance. The second idea is using dilated convolution to increase local information and enlarge the receptive field at the same time [9, 12, 13, 14]. The third idea is to adopt probabilistic graphical models (e.g. CRFs) to refine the segmentation result [9, 13, 39, 78]. However, this post-processing is time-consuming and breaks the end-to-end modeling. Fourth idea is the usage of Encoder-decoder networks [3, 5, 40, 51, 73]. Previous works [14] show that it can help to obtain sharper segmentation.

3.1.2 Instance Segmentation

Instance segmentation task is to predict the boundary of each object in the scene. We categorize current instance segmentation methods into two categories: detection-based methods and segmentation-based methods.

For detection-based methods, most models adopt RPN [56] to generate instance proposals. FCIS [37] and MaskLab [11] utilize the 'position-sensitive score map' idea proposed by R-FCN [18]. Mask R-CNN [28] extends Faster R-CNN by adding a branch for predicting object masks. PANet [44] adds a bottom-up path to the FPN backbone aiming at enhancing the localization capability of the entire feature hierarchy and demonstrates outstanding performance.

The other approach is segmentation-based. The methods that fall into this category always need to learn spatial transformation for instance clustering. Many methods attempt to predict link relationship between each pixel with its neighbors [20, 38, 57] through graphical models [2, 75], embedding vectors [50] or discriminative loss [19]. InstanceCut [31] predicts object boundary while Watershed [4] predicts the watershed energy through an end-to-end convolutional neural network and applies multi-cut. Instances are also separated by exploiting depth ordering within an image patch [60, 74]. However, this method requires ground-truth depth maps during training which we do not assume that we have.

Other works focus on fast prediction speed to achieve real-time performance. Box2Pix [61] relies on a single FCN [46] to predict object bounding boxes, pixel-wise semantic object classes and offset vectors toward object centers. However, because of the single architecture and delicate offset vector prediction, the final result is not competitive.

3.1.3 Panoptic Segmentation

Normally, instance segmentation only focuses several semantic classes (such as humans and cars, usually referred as *things*) and ignores the other (such as road and sky, usually referred as *stuff*). On the other hand, semantic segmentation cannot provide masks of each individual objects. The idea of combining instance and semantic segmentation together is first proposed by Kirillov et al. [32]. The new task, called panoptic segmentation, requires the output to contain both pixel-wise semantic

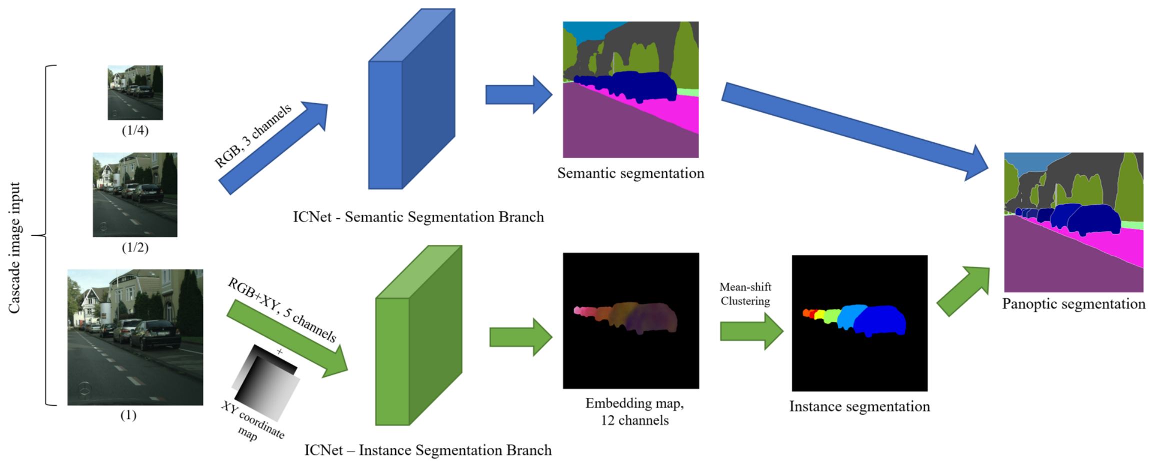

Figure 3.1: The PanoNet framework for panoptic segmentation. The top branch takes normal images as input and outputs semantic segmentation results. The lower branch takes original images along with additional XY coordinate map and outputs instance segmentation results. The outputs of the two branches can be merged into panoptic segmentation directly without requiring any additional procedures

prediction and object boundaries for *things* classes. A simple solution to this problem is proposed in [32] by heuristically combining the instance segmentation results from a Mask R-CNN and semantic segmentation results from a PSPNet. Recently, many works are aimed to find a unified network for the two sub-tasks. Panoptic FPN [33] slightly modifies Mask-RCNN by enabling it to also generate pixel-wise semantic segmentation prediction. UPSNet [64] designs a panoptic head with a single network as backbone. However, these methods all rely on a region-proposal based object detector and fail to run in real-time.

In contrast to those methods, we proposed a framework that exploits the strong correlation between detection and segmentation tasks. We use information in both spacial and feature embedding domain for instance grouping to alleviate the inefficiency caused by detection-based segmentation. Our design is simple yet effective. With ICNet [77] as backbone, we achieve ~20-fps inference speed on Cityscapes' full scale images (resolution: 2048×1024) with decent prediction accuracy in both semantic and instance segmentation tasks.

3.2 Method

3.2.1 Network Structure

We choose ICNet [77] (originally designed for semantic segmentation) as the backbone of PanoNet. ICNet uses cascade feature fusion unit and cascade label guidance to speed up low-speed segmentation network (PSPNet50) while still maintaining decent accuracy. We modify ICNet by adding an extra branch that takes additional coordinate input channels and outputs an pixel-wise embedding map for instance segmentation. The whole structure of PanoNet is shown in Fig. 3.1.

For semantic segmentation task, we simply adopt the same design as the original ICNet. For instance segmentation, the input and the final output layers of the ICNet framework are modified. Two more layers of normalized horizontal and vertical coordinates are appended to the RGB color channels as the input of this branch, and the final output is a 12-channel position-sensitive instance embedding map. In post-processing, a fast version of mean-shift clustering algorithm is applied class-wise to generate the instance segmentation map. More details of the PanoNet structure are discussed in the following sections.

3.2.2 Loss Function

We design the overall loss function by dividing it into two sub-terms, one for each branch of the PanoNet. For the semantic segmentation branch, we follow the ICNet's weighted softmax cross entropy loss design. Because there are $\mathcal{T} = 3$ scales of reference guidance, the loss can be defined as

$$\mathcal{L}_{sem} = \sum_{t=1}^{\mathcal{T}} \lambda_t \mathcal{L}_{sce,t} \tag{3.1}$$

$\mathcal{L}_{sce,t}$ denotes the softmax cross entropy loss of each scale, and λ_t are the weighting factors.

For instance segmentation, we adopts the idea of discriminative loss [19] proposed by Kirillov et al. The loss pulls the pixel embeddings closer to the mean of their cluster and pushes different clusters away from each other. We define that C is the number of clusters, N_c, μ_c is the number of pixels and mean of cluster c, x_i is the embedding vector, and $\delta_v, \delta_d, \delta_r$ are the margins of the variance, distance and regularization loss terms. $[x]_+ = \max(0, x)$ denotes the hinge function and $\|\cdot\|$ denotes the L2 distance. Then our modified instance segmentation loss can be expressed as

$$\mathcal{L}_{var} = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}\sum_{i=1}^{N_c}[\|\mu_c - x_i\| - \delta_v]_+^2 \tag{3.2}$$

$$\mathcal{L}_{dist} = \frac{1}{C(C-1)}\sum_{\substack{c_a=1 \\ c_a\neq c_b}}^{C}\sum_{c_b=1}^{C}[2\delta_d - \|\mu_{c_a} - \mu_{c_b}\|]_+^2 \tag{3.3}$$

$$\mathcal{L}_{reg} = \frac{1}{C}\sum_{c=1}^{C}\frac{1}{N_c}[\|\mu_c\| - \delta_r]_+^2 \tag{3.4}$$

$$\mathcal{L}_{inst,t} = \alpha\mathcal{L}_{var,t} + \beta\mathcal{L}_{dist,t} + \gamma\mathcal{L}_{reg,t} \tag{3.5}$$

$$\mathcal{L}_{inst} = \sum_{t=1}^{\mathcal{T}}\lambda_t\mathcal{L}_{inst,t} \tag{3.6}$$

method	PQ	SQ	RQ	PQ^{Th}	SQ^{Th}	RQ^{Th}	PQ^{St}	SQ^{St}	RQ^{St}	mIoU	AP	runtime (ms)
Li et al. [36]	53.8	-	-	42.5	-	-	62.1	-	-	71.6	28.6	7000
MR-CNN-PSP [32]	58.0	79.2	71.8	52.3	77.9	66.9	62.2	**80.1**	75.4	75.2	32.8	1105
Panoptic FPN [33]	58.1	-	-	52.0	-	-	62.5	-	-	**75.7**	33.0	500
UPSNet [64]	**59.3**	**79.7**	**73.0**	**54.6**	**79.3**	**68.7**	**62.7**	**80.1**	**76.2**	75.2	**33.3**	236
PanoNet	55.1	77.5	67.9	46.8	74.6	60.9	60.5	79.6	72.9	74.6	23.1	**50**

Table 3.1: Panoptic segmentation results on Cityscapes validation set. Superscripts Th and St stand for things and stuff. Runtimes, if not stated in the original paper, are guessed in favor of the method. Metrics that are not reported are left as '-'.

3.2.3 Position-Sensitive Embedding

Previous embedding-clustering based approach does not take the position of different instances into consideration. When there are two instances with similar appearance in a same image, such model often fails. Inspired by Liu et al. [43], we propose to add coordinate maps in addition to the origianl RGB image as the input of the network. However, achieving a weighting balance between the two aspects (position and appearance) of the final fused embedding can be tricky. Through experiments, we observe that directly training such network from scratch may cause the network to overuse the coordinate information such that the network's learned embedding does not generalize very well and performs poorly on validation set. To address this issue, we apply a two-stage training technique. More specifically, we first train the network with discriminative loss by using the pretrained semantic segmentation weights and re-initializing last few layers. After converging, the network is trained again with the two coordinate input layers added. This approach allows the network to learn how to effectively utilize the coordinates to segment different instances and fuse the position-sensitive information into the final embedding vector.

3.2.4 Clustering

In the discriminative loss function, if we choose the margins such that $\delta_d > \delta_v$, the embedding will move each vector x_i closer to its own cluster center than to all other clusters. To get the instance masks, we need to choose a clustering method to group the vectors x_i into their respective clusters. As the number of instances, i.e., the number of clusters in the embedding space of x_i's, is unknown, the mean-shift algorithm [15] is a good fit in this case. Mean-shift is controlled by a single 'bandwidth' parameter in lieu of the number of clusters. The bandwidth can be viewed as the distance threshold for determining whether a point belongs to the neighbourhood of a cluster center. Because the δ_v term in the loss function plays a similar role as the bandwidth, we choose it as the base value of the bandwidth, and we search within a small range for an optimal value of each semantic category on the training set. This is because we cannot train the discriminative loss to zero in practice, so that the actual optimal bandwidth value for clustering is slightly larger than δ_v. The search process is one-time and straightforward (usually takes few hours to complete). We also accelerate the mean-shift algorithm with discretized bin-seeding (seeds are chosen from a grid of original points). The clustering process adds minimal overhead to the whole pipeline and does not affect the real-time inference speed.

3.2.5 Implementation Details

We set the hyperparameters of the loss function as $\tau_1 = 1, \tau_2 = 0.4, \tau_3 = 0.16$ for the three layer of cascade guidance (1/4, 1/8, 1/16 scale of original image),

method	AP	$AP_{0.5}$	person	rider	car	truck	bus	train	mcycle	bicycle
ICNet[77]	23.1	42.3	8.8	8.2	29.6	29.3	45.3	45.4	9.3	8.5
DeepLabv3+[13]	19.6	40.1	11.9	14.2	28.2	14.4	39.6	22.9	13.1	12.7
ground truth	**59.7**	**77.3**	**41.8**	**73.2**	**31.8**	**78.6**	**76.8**	**71.9**	**58.0**	**45.6**
number of instances in train set			17.9k	1.8k	26.9k	0.5k	0.4k	0.2k	0.7k	3.7k

Table 3.2: Influence of semantic segmentation results on instance segmentation performance. The last row shows the total number of training instances in the Cityscapes dataset.

$\delta_v = 0.25, \delta_d = 1, \delta_r = 6$ for the margins and $\alpha = 1, \beta = 1, \gamma = 0.1$ for the weights of discriminative loss. For training on Cityscapes dataset, we choose Adadelta with the learning rate of 0.003 and polynomial decay policy.

3.3 Results

We train the PanoNet on the Cityscapes train set (2975 images) and test the model on the validation set (500 images). For both training and testing, we use the original high-resolution images without down-sampling. The inference speed measurement is conducted on a single Titan X GPU.

3.3.1 Panoptic Segmentation Metrics

We report the panoptic segementation results on Cityscapes together with other state-of-art methods in Table 3.1. Only our method requires no object detectors or region proposals. We show that the PanoNet runs much faster than all other methods and achieves decent panoptic quality score at the same time. Our inference time is 50 ms, less than 1/4 of the fastest UPSNet. On the other hand, our PQ score is 55.1, which is even better than one method that does not pay attention to the speed.

Some visual examples are shown in Fig. 3.2. Because our method does not need to handle the conflicts of semantic and instance segmentation like other two-stage solutions, there is no large black (unlabeled) area in the images as observed in those methods. The PanoNet is also able to properly group non-connected regions, which cannot be properly segmented by some other instance clustering approaches. The last two rows illustrate some failure modes of our solution. The first image shows semantic segmentation error of some trash cans, and the second one shows a hard instance segmentation case where there are crowded pedestrains with occlusion.

3.3.2 Influence of Semantic Segmentation on Instance Segmentation

As our method does instance clustering on semantic segmentation results, error may accumulate during this process. Compared with the UPSNet, we observe higher performance gap for instance segmentation (things) than semantic segmentation (stuff). To further understand the influence of semantic segmentation on instance segmentation, we replace the semantic segmentation map with a) prediction from a more complex model - DeepLabv3+ [13] and b) ground truth. These results are reported in Table 3.2. We can see that the average precision is significantly higher when the ground truth semantic segmentation is given. This shows that our method is highly efficient on the instance clustering task. The AP is higher on bus and train class because there are usually only one or two instances of these classes in a single image (little clustering work is needed). When using semantic prediction instead of the ground truth, instance segmentation is very prone to error especially when some small regions are incorrectly mislabeled. One of such examples is shown in Fig. 3.3. The highlighted regions are mislabeled as ‘bicycle’ class and clustered into three instances. Though these small regions won’t affect the semantic segmentation metrics much, they significantly degrade the instance performance because of the introduced false positives. The ‘car’ class has the smallest performance gap without the ground truth semantic segmentation, because this class has the most training examples in the dataset. Last, we show that, more accurate semantic segmentation methods, such as DeepLabv3+, does not necessarily lead to performance improvement. The semantic segmentation models that are conservative on ‘things’ categories and predict fewer false positives tend to result in better instance segmentation.

3.3.3 Position-Sensitive Embedding

To study whether adding position-sensitive component to the embedding leads to any improvement on instance segmentation task, we train a variant of PanoNet

method	backbone	AP	AP_{GT}
Discriminative Loss[19]	ResNet38	21.4	37.5
PanoNet w/o pos.	ICNet	20.5	58.9
PanoNet	ICNet	23.1	63.7

Table 3.3: Performance of instance segmentation on Cityscapes validation set. AP_{GT} means the average precision of instance segmentation given the ground truth semantic segmentation.

under the same settings but without providing position information (i.e., removing coordinate map from input). The results are reported in Table 3.3. We observe that position-sensitive embedding increases the average precision. Even with a much more light-weight backbone (ICNet compared to ResNet38), PanoNet outperforms the baseline model.

Position information is especially useful in scenarios where two instances have similar or the same appearance. Fig. 3.4 shows such an example. The image was created by mirroring the left half of the original image to the right half. If we use pure feature embedding to cluster instances, the two mirrored cars cannot be distinguished from each other as expected. But when position information is provided, all the four instances can be correctly predicted as shown in the figure.

3.3.4 Runtime and Memory Analysis

We report the computational complexity of our model in Table 3.4. PanoNet shows excellent performance-accuracy trade-off. To our best knowledge, PanoNet is the first panoptic segmentation model that runs in real time on a single GPU with 2-mega-pixel high-resolution input images.

Our model has advantages over other methods for three main reasons. First, for applications such as autonomous driving and augmented reality, real-time performance is highly desired. The inference speed of PanoNet is at 20 fps and significantly faster than all other approaches, making it favorable especially when hardware resources are limited. Second, the highest-end GPUs on the market usually have 12 GB memory, while many other consumer GPU products have only 4 GB to 8 GB memory. For high-resolution images, the listed models usually have to crop the image into several smaller sub-images to process separately simply because the model cannot be fit into the GPU's memory. This makes PanoNet stand out as it only requires 3GB memory for images as large as 2048 × 1024. Thus, our whole model can be easily fit into most GPUs so that the output of a complete image is generated in a single shot. Last, PanoNet is easy to train beacause of it contains many fewer parameters. Most other panoptic segmentaion models use 16 GPUs or more [64] to train, while our training process only takes 4 GPUs thanks to the compactness of the model.

method	PQ	# p.	mem.	fps
Li et al. [36]	53.8	-	48 GB	< 0.5
MR-CNN-PSP [32]	58.0	92M	$>$ 12 GB	0.9
Panoptic FPN [33]	58.1	-	$>$ 12 GB	2
UPSNet [64]	59.3	46M	-	4.2
DeeperLab (Xception-71) [67]	56.5	-	-	3.2
DeeperLab (Wider MNV2) [67]	52.3	-	-	6.7
DeeperLab (Light Wider MNV2) [67]	48.1	-	-	10.2
DeeperLab (Light Wider MNV2) [67]	39.3	-	-	24
PanoNet	55.1	12M	3 GB	20

Table 3.4: Panoptic quality (PQ) and computational complexity comparison of PanoNet and other methods on full-scale Cityscapes images. '# p.' means the number of parameters of the model. Metrics that are not reported are left as '-'.

3.3.5 Sharable Weights

Currently our PanoNet model consists of two independent ICNet branches. However, it is possible for these two branches to share the weights of shallow layers to further compress the model. We train a variant of PanoNet with shared weights except for the final three layers leading to the prediction. This yields 52.3 PQ, which is 5% lower than the baseline. As PanoNet is already light-weight and fast, further study on weight sharing and model compression is beyond the scope of this paper.

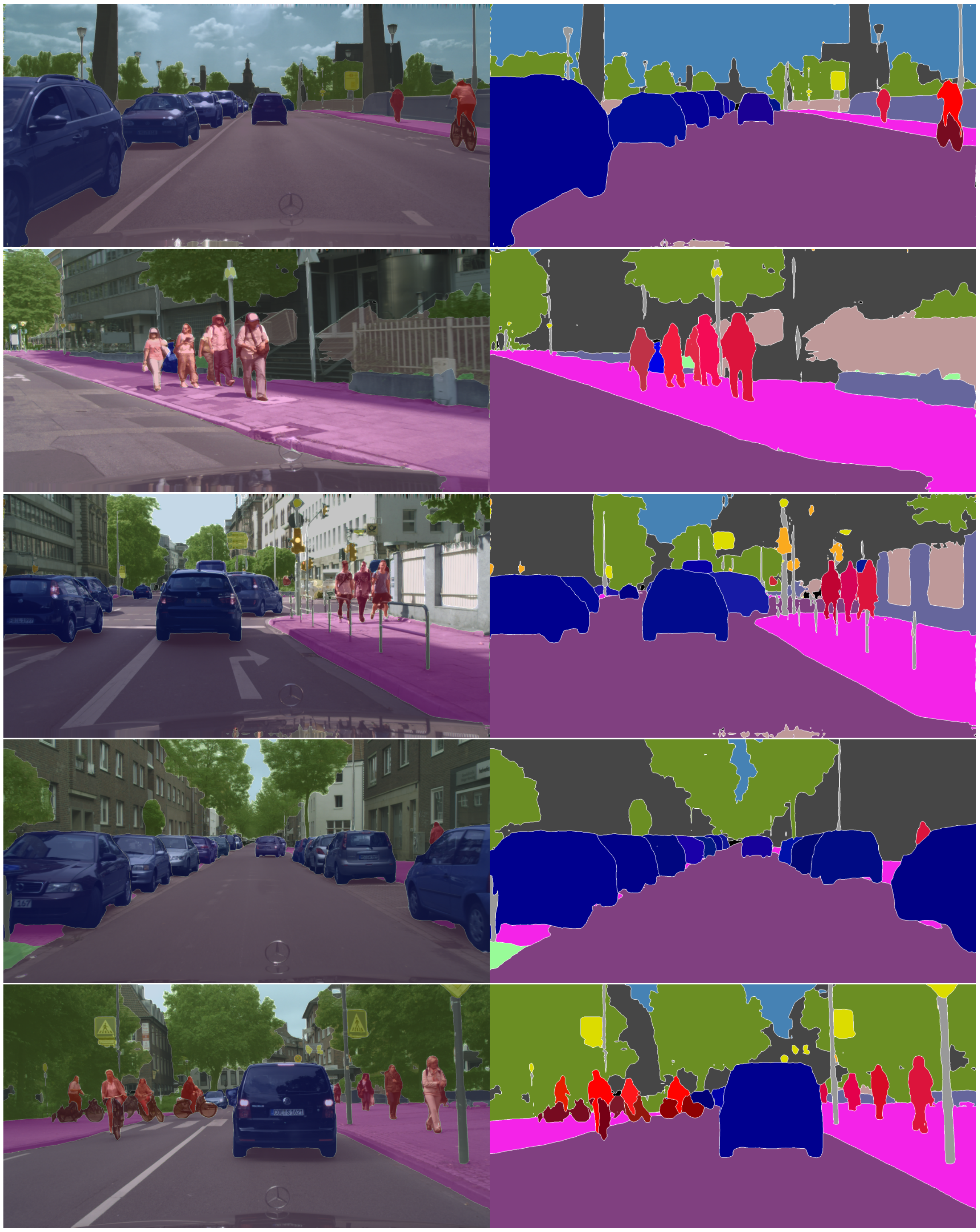

Figure 3.2: Visual prediction of PanoNet on Cityscapes dataset (1 of 2). First column: overlayed results; second column: corresponding panoptic segmentation results.

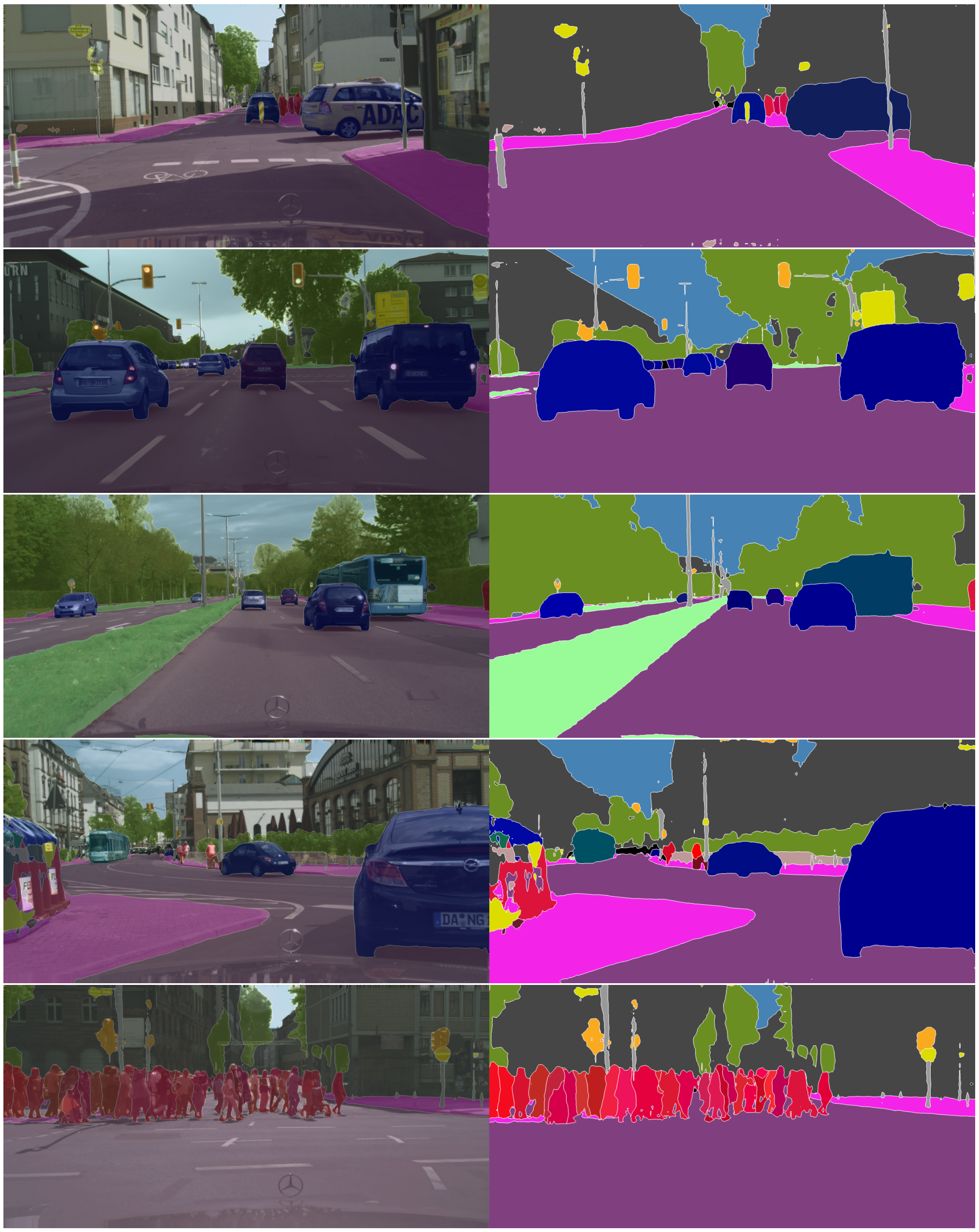

Figure 3.2: Visual prediction of PanoNet on Cityscapes dataset (2 of 2). First column: overlayed results; second column: corresponding panoptic segmentation results. The last two rows shows some failure mode.

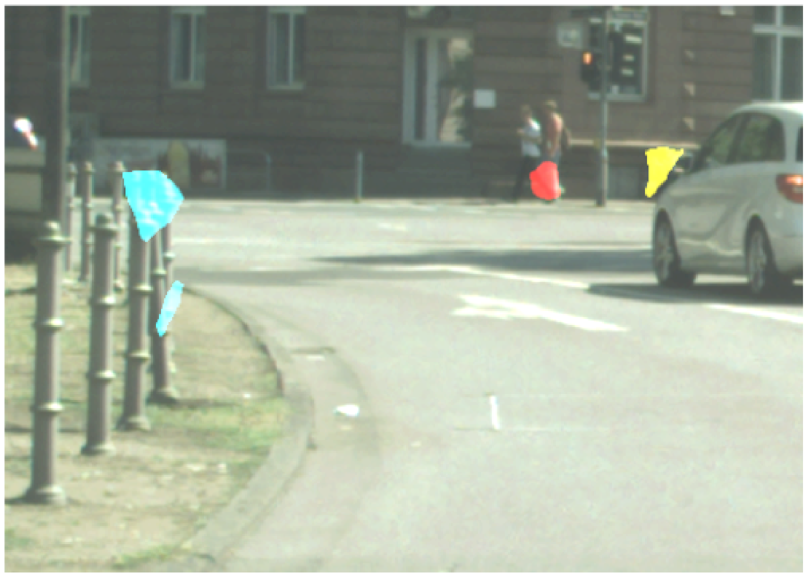

Figure 3.3: Zoomed-in view of mislabeled semantic regions. The highlighted regions are all mislabeled as 'bicycle' class by the semantic segmentation branch. They are clustered into three 'bicycle' instances, resulting in three false positives.

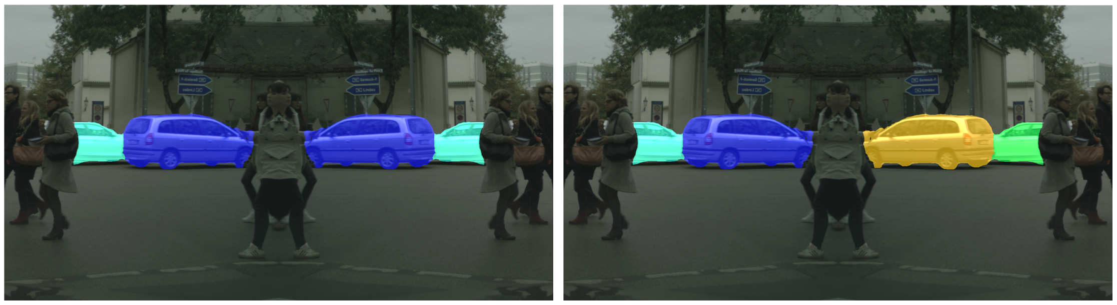

Figure 3.4: Instance segmentation on an artificially created image (mirrored about the axis in the middle). Left: prediction result of non-position-sensitive embedding, the two mirrored cars are grouped into one single instance; right: prediction result of position-sensitive embedding.

Chapter 4

PanoNet3D

4.1 Related Work

4.1.1 Point Cloud Representation

Deep learning architectures take different formats of point clouds as input. The first class consumes raw point clouds directly, including PointNet [54], PointNet++ [55], and PointRCNN [58]. 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 usually contains over 50,000 points. So these networks usually need to down-sample input, losing resolution of raw data.

Some networks simply treat point clouds as a bird-eye-view (BEV) image, e.g., AVOD [34] and Complex-YOLO [59]. This particularly works well on detectors of LiDAR point clouds as we usually only care about x-y (2D) localization of objects. This formatting allows mature 2D image detection frameworks to be re-applied on point clouds at the cost of partly losing vertical geometric structure information.

Another type of point cloud formatting is voxelization. Examples include VoxelNet [80], SECOND [65], and PIXOR [66]. Voxelized point cloud usually has finite spatial size with pooling as the technique to convert per-point features to per-voxel features.

Recently, LaserNet [48] shows that when the size of training dataset is large enough, detection networks performing on perspective view of LiDAR point clouds (range images) can achieve performance close to popular BEV detectors. Similarly, MVF [81] extracts semantics from both range images and BEV images with two 2D convolutional towers. For LiDAR point clouds, range image format is dense and has no range limits compared to other BEV based representations.

4.1.2 Object Detection

Object detection has traditionally been studied on 2D images. Various Convolutional Neural Network (CNN) based detectors are proposed since R-CNN [23]. 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) [56] 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) [45] 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 [42] to alleviate the problem of foreground-background class imbalance, single-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 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 [65], PointPillars [35], PIXOR [66], and LaserNet [48]. Single-stage point cloud detector is more favorable for autonomous driving application due to its simplicity and fast inference speed.

4.1.3 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.

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 [65] in order to significantly reduce time and memory consumption. PointPillars [35] 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.

Temporal Aggregation

Many detectors aggregate multiple consecutive LiDAR sweeps and show that temporal information can improve detection results. FaF [47] treats temporal information as an additional dimension of input tensor, i.e., multiple frames are appended along a new dimension to create a 4D tensor. SECOND [65] proposes a simpler solution that adds relative temporal stamp to each point as an extra input channel. We need to pay special attention t

o ego motion during temporal aggregation as the reference coordinate system shifts with ego vehicle's movement.

Data Augmentation

Data augmentation is extremely important for applying LiDAR detection on autonomous driving scenarios, as such real world datasets usually have severe problem of class imbalance. For example, about half of labelled instances in NuScenes [8] dataset are cars. A copy-and-paste augmentation schematic are used in many popular detectors including SECOND [65], PointPillars [35], and CBGS [82]. This method crops ground truth bounding boxes from other frames and pastes them onto current frame's ground plane. Hu et al. [29] argues that maintaining correct visibility during augmentation makes significant improvements on detection results. The visibility information can either be explicitly expressed or naturally encoded in range images.

4.2 Method

The structure of PanoNet3D is illustrated in Fig. 4.1. This framework can be divided into two stages. 1) **Feature extraction stage**: A 2D FCN generates deep semantic features from projected pseudo range images. Meanwhile, a geometric decorator generates each point's raw geometric features including its global coordinates and local position relative to the center of its residing voxel. The semantic features and geometric features are aggregated and passed to the next stage. 2) **Detection stage**: Per-point features are converted to per-voxel features by 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 details of each component of the network in following sections.

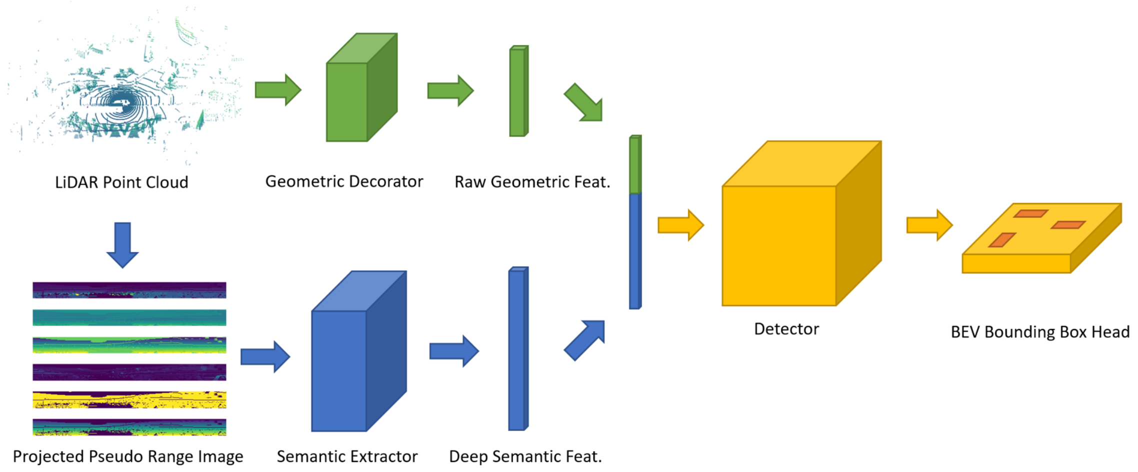

Figure 4.1: PanoNet3D's framework for point cloud detection. The top branch takes LiDAR point cloud as input and decorates raw point features with several simple local geometric features. The lower branch converts point cloud to pseudo range image and feeds it into a 2D FCN to get per-pixel deep semantic 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.

4.2.1 Pseudo Range Image Feature 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 are not even available), we manually project point clouds back to 2D range images with evenly spaced projection angles. For NuScenes [8] dataset, we choose 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 points are mapped to a same pixel on range image. In this case, we simply keep the closest point and neglect 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 [48], 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. 4.2) pseudo range image of LiDAR.

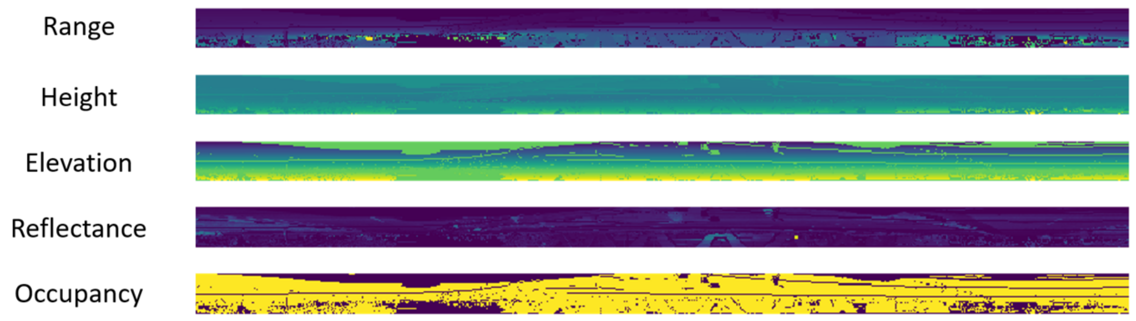

Figure 4.2: An example of projected pseudo range image with five channels. From top to bottom: range r, height h, elevation angle ϕ, reflectance i, and occupancy mask m.

For the feature extractor, we adopt the simple Semantic FPN (SFPN) design in [33]. 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 [27] without the first layer (conv1). For each projected LiDAR points, the SFPN generates a 64-dimensional feature vector extracted from the pseudo range image.

4.2.2 Voxelization and Geometric Decorator

The raw input 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 [35]. Pillarization can be seen as a special type of voxelization with only one layer of voxels vertically. We decorate each point's global position $[x, y, z]$ with its distance to the LiDAR origin r and its position relative 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 can be applied on each voxel to extract more local features like VoxelNet [80].

4.2.3 Feature Aggregation

We use the simplest way of aggregating semantic feature and geometric feature by concatenation. For those points that are not assigned with semantic feature, their embeddings are padded with zeros. For each voxel, locally aggregated feature representations are generated from point-wise embeddings via symmetric pooling operations: we apply element-wise max pooling on high-dimensional semantic feature and average pooling on low-dimensional geometric feature.

4.2.4 Temporal Aggregation

When multi-frame data is available, we add timestamp t as an additional feature to each point. Such temporal aggregation mainly affects the semantic feature extractor of the network. For example, when we aggregate 10 consecutive sweeps, the point cloud is 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.

Temporal Multi-Frame Fusion

We retrieve the pseudo range image at each frame respectively, and then concatenate them along a new dimension to form a batch of images as input. This is equivalent to running the same feature extractor on every individual frame.

Spatial Multi-Frame Fusion

We transform all points to the key-frame's coordinate system and increase the resolution of the pseudo range image to allow more points to be projected. If the new linear resolution is n times as large as before (single-frame resolution), we will need to find the optimal value of n. Fig. 4.3 shows the occupancy map of different n. The range image becomes sparse and inefficient for feature extraction when n is too large. We want the occupancy rate τ to be close to the original one. For NuScenes dataset (20 Hz frame rate), we choose $n = 2$ for 10-frame aggregation. Notice that the range image has only $4\times$ pixels while the point cloud have $10\times$ points. When multiple points are projected to the same pixel, we prioritize those with timestamps closer to the key-frame. This allows us to enhance the resolution of input efficiently without too much redundancy caused by close or repeating points.

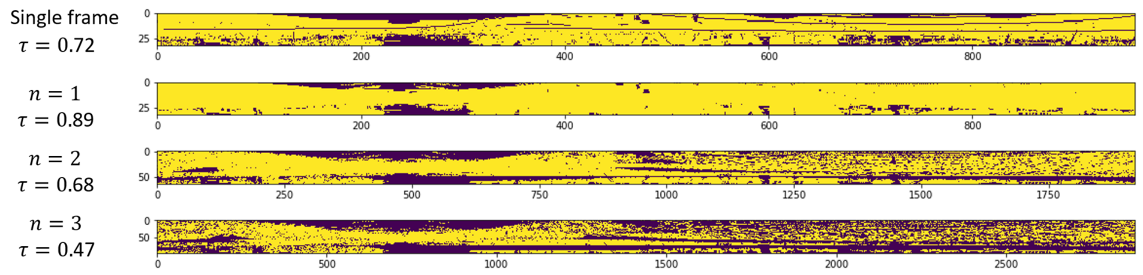

Figure 4.3: Occupancy maps showing the results of spatial multi-frame fusion. Yellow color indicates the pixel is occupied by a projected LiDAR point. τ is the occupancy rate (number of occupied pixels over number of total pixels). From top to bottom: original single frame, 10-frame aggregation with $n = 1$, $n = 2$, $n = 3$.

4.2.5 Detector

As discussed in Section 4.2.2, 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 a semantic FPN as backbone (similar to the range image feature extractor, but functions as a Region Proposal Network (RPN) instead) to get the final feature map. For 3D voxel input with 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.

For bounding box regression head, we follow the same multi-group head design as in CBGS [82], where the 10 classes in NuScenes are categorized into 6 groups:

(car), (truck, construction vehicle), (bus, trailer), (barrier), (motorcycle, bicycle), (pedestrian, traffic cone). We also adopts the same bounding box parameterization and anchor design. Detailed detector structure is illustrated in Fig. 4.4.

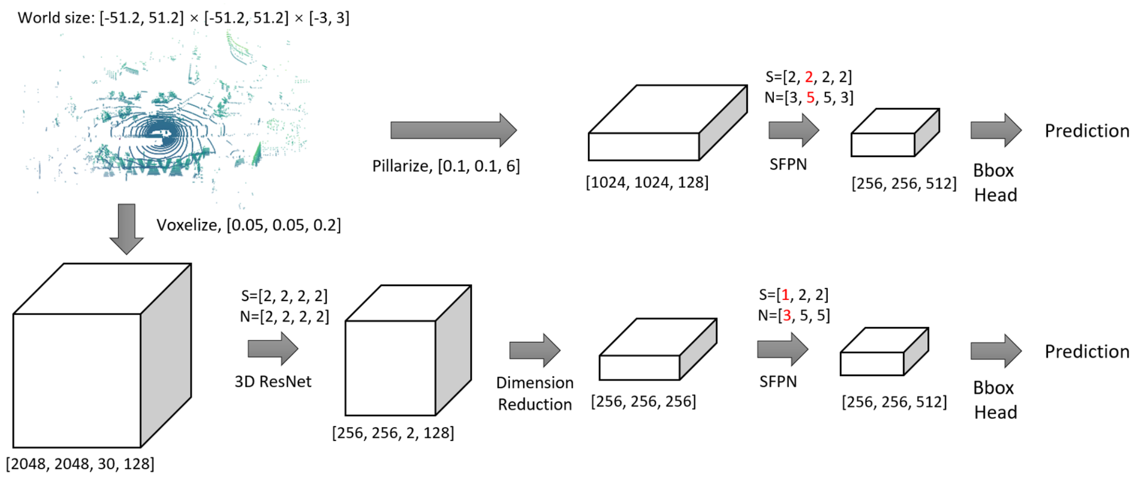

Figure 4.4: Structure of detector with 2D pillars or 3D voxels as input. The initial feature is 128 dimensional. We limit the size of the whole scene to $[-51.2, 51.2] \times [-51.2, 51.2] \times [-3, 3]$ meters in x, y, z direction. The networks consist of a few layers of ResNet basic blocks. S denotes 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.

4.2.6 Data Augmentation

We use similar data augmentation schematics used in SECOND [65] and CBGS [82]. Ground truth boxes are cropped and saved offline, and then pasted onto the current frame's ground plane. Additionally, we allow the augmented object randomly rotate around the LiDAR within 45 degrees (its distance to the center of frame remains unchanged). As newly pasted objects may occlude with other objects, we remove all annotations that have less than 3 points projected on the pseudo range image. By doing so, the objects that should not be visible to the LiDAR are easily filtered out. 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]).

4.2.7 Implementation Details

Our implementation is based on CBGS's [82] official code base[1]. All object classes share the same 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 [30], one-cycle policy [1] (max learning rate: 0.0001, division factor: 5), and 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.

4.3 Results

We first compare quantitative performance of our model against other SOTA models on NuScenes dataset. Qualitative results (visualization of predictions) are shown in Fig. 4.5. Next, we conduct ablation studies to explain how we make the decisions during network design and show where the performance improvements come from.

4.3.1 Main Results

We submitted the results of our method to the NuScenes test server. In Tab. 4.1, we compare PanoNet3D against other models on the NuScenes detection leaderboard. Our overall mAP on test-set surpasses the current first-place method CBGS [82] by 1.7%. For fairness, we also compare our method against CBGS's reproducible performance on NuScenes validation set in Tab. 4.2. 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 and the average mAP is 7.7% higher than CBGS.

4.3.2 Ablation Study

Tab. 4.3 shows a series ablation studies. Based on these results, we can make the following key observations.

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

	car	truck	bus	trailer	cons.	pedes.	mcycle	bicycle	cone	barrier	mAP
Point Pillars [35]	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9	30.5
SARPNET [70]	59.9	18.7	19.4	18.0	11.6	69.4	29.8	14.2	44.6	38.3	32.4
CBGS [82]	**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 4.1: Detection mAP by categories compared on NuScenes test set.

	car	truck	bus	trailer	cons.	pedes.	mcycle	bicycle	cone	barrier	mAP
CBGS* [82]	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 4.2: Detection mAP by categories compared on NuScenes validation set. *: reproduced with official released code and our experimental setup. Second line shows the result of our model without aggregation of range-image-based semantic features (row i. in Tab. 4.3).

Baseline Comparison (a.-c., g.-h.)

The major difference between PanoNet3D and traditional detectors is its range-image-based semantic feature extractor. Without the aggregation of extracted semantic features, our pillar-based detector should have similar framework to PointPillar's [35] except for backbone design. Our pillar-based baseline model achieves better performance than PointPillars. We hypothesize this might be because our SFPN backbone is able to utilize multi-level features more efficiently. On the other hand, for voxel-based detectors, our model has close performance to CBGS without semantic feature extractor, showing that the final improvements against CBGS do not come from the different backbone.

Range Image Semantic Feature Extractor (c.-f., g.-l., i.-m.)

For single-frame pillar-based detectors, semantic features extracted from range images significantly improve the average mAP by 13.7%. With this help of semantic feature extractor, our single-frame model is able to achieve comparable results against multi-frame models. For multi-frame voxel-based detectors, semantic features also improve the average mAP by over 6%. From Tab. 4.2, 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 Euclidean space, which helps the detector to achieve a better overall understanding of the scene.

Pillar or Voxel (e.-f., k.-l.)

We show that for single-frame input pillar-based detector performs slightly better, while for multi-frame input voxel-based detector is more favorable. One possible explanation is that pillar-based detector is sufficient for single-frame input and can prevent over-fitting caused by complex 3D convolutions. Multi-frame input has much denser point clouds whose features can not be well extracted by the simple pillar feature extractor.

BEV Resolution (j.-k., l.-m.)

Finer grid of voxelization usually leads 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%, and increasing BEV resolution from 0.1m to 0.05m improves mAP of 10-frame voxel-based detector by 1.4%.

	Method	Range image feat.	# Input frames	Voxelization	BEV resolution(m)	mAP
a.	Point Pillars [35]	-	1	Pillar	0.25	24.0
b.	Point Pillars [35]	-	10	Pillar	0.25	29.5
c.	Ours	✗	1	Pillar	0.25	31.5
d.	CGBS [82]	-	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 [82]	-	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 4.3: 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.

4.3.3 Voxel Feature Pooling Methods

We test all pooling methods during aggregating point-wise features to voxel-wise features. With all combinations shown in Tab. 4.4, we conclude that max pooling on semantic features with average pooling on geometric features yields the best results.

Deep semantic feat. aggr.	Raw geometric feat. aggr.	mAP
Max	Max	44.6
Max	Average	45.2
Average	Max	44.3
Average	Average	44.9

Table 4.4: Study on pooling methods during voxel-wise feature aggregation. The experiment is done with a single-frame pillar detector as baseline.

4.3.4 Temporal Aggregation Methods

We also experiment with different temporal aggregation approaches, the results of which are shown in Tab. 4.5. 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 frame or the input dataset changes, we might need to change n accordingly to fit 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 4.5: Results of different multi-frame temporal aggregation approaches.

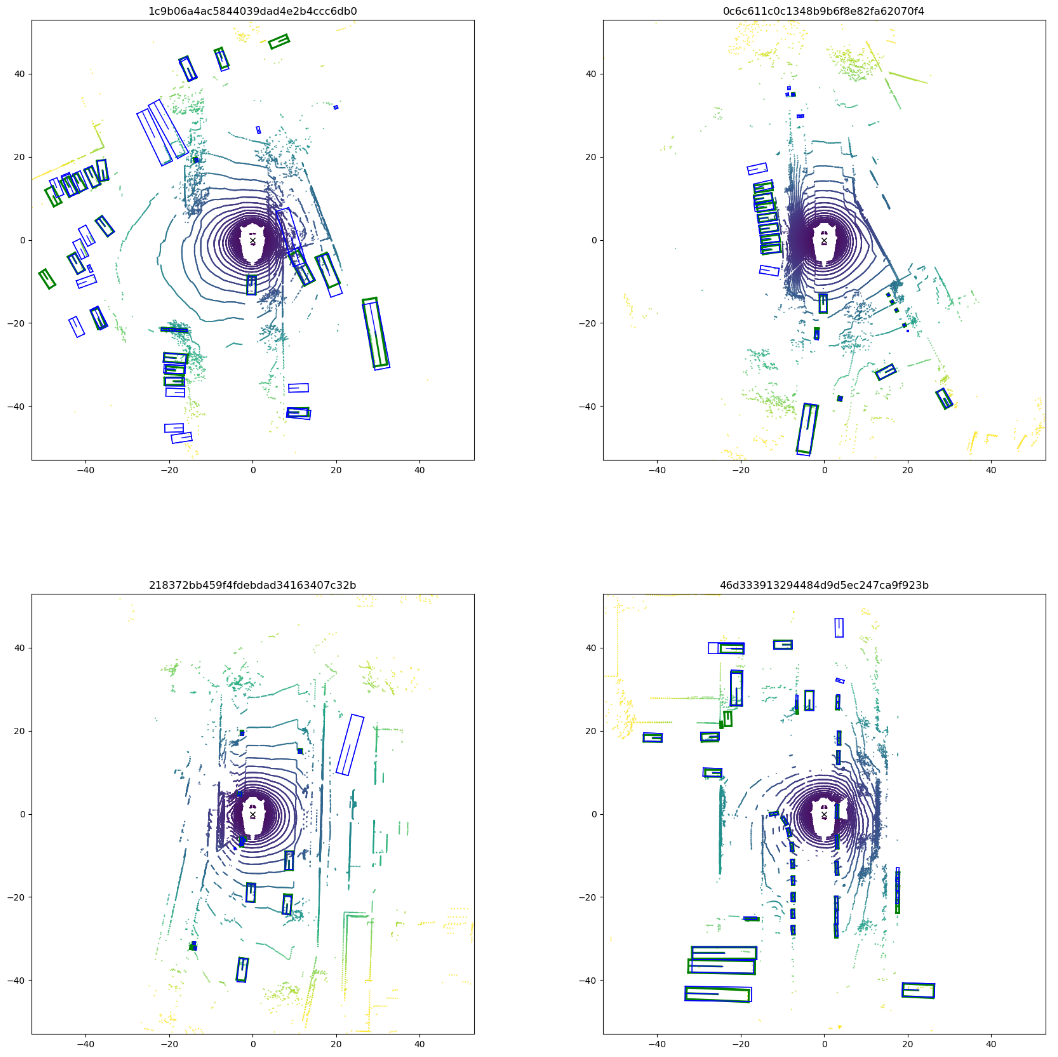

Figure 4.5: Detection examples of PanoNet3D on NuScenes dataset (1 of 2). Ground truths are annotated in green boxes and detection results are annotated in blue boxes.

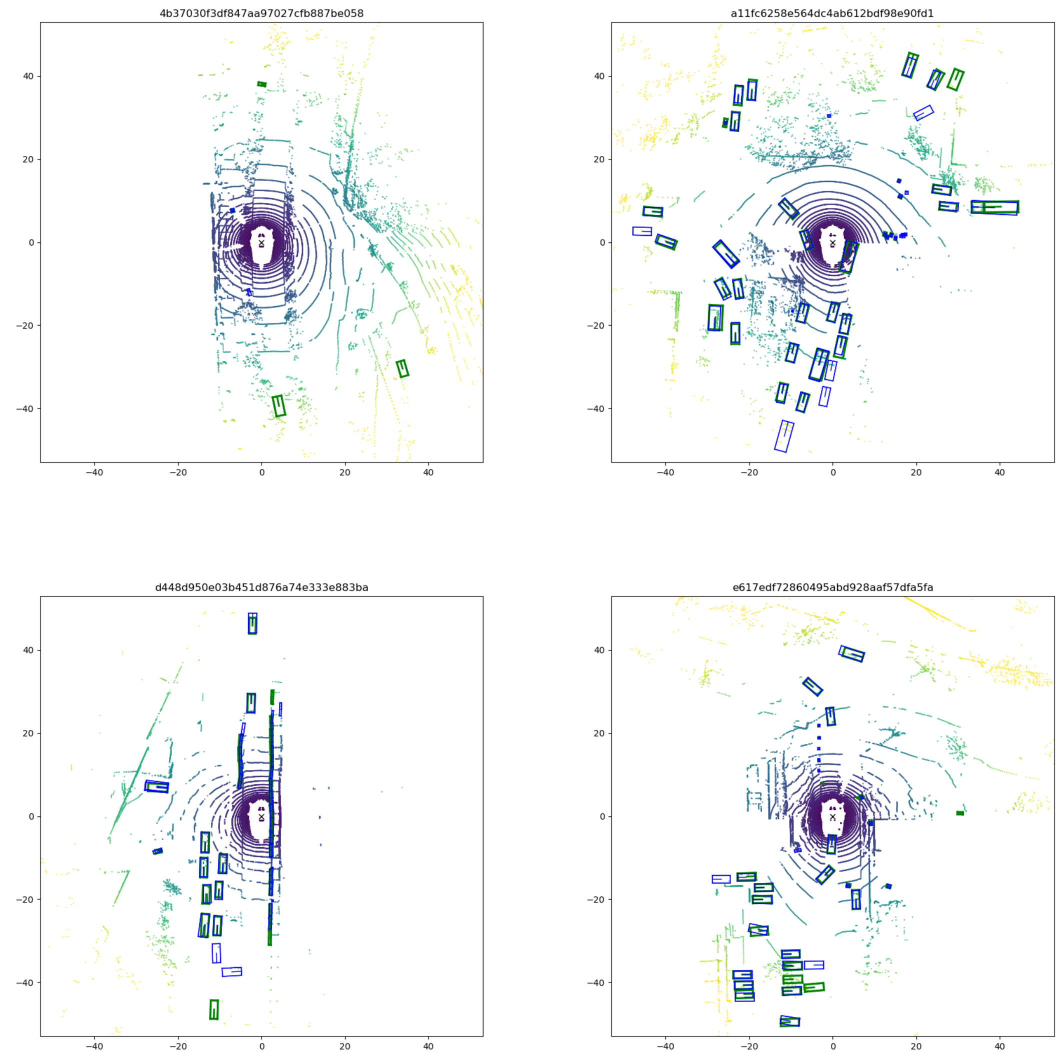

Figure 4.5: Detection examples of PanoNet3D on NuScenes dataset (2 of 2). Ground truths are annotated in green boxes and detection results are annotated in blue boxes.

Chapter 5

Conclusions

We explore the possibility of combining both semantic and geometric understanding for modern visual recognition tasks. This idea is tested with two types of data commonly used in autonomous driving scenarios: 2D images (on perspective plane) and 3D point clouds (in real-world Euclidean space). For image data, we add spatial information of each pixel to original color channels to generate instance-specific embeddings. For LiDAR point cloud data, we enhance each point's raw geometric coordinates with deep semantic features to improve 3D detection performance. Multiple experiments show that both geometric and semantic information are important for deep learning frameworks to achieve better overall understanding of the scene.

Bibliography

[1] Another data science student's blog – The 1cycle policy. URL https://sgugger.github.io/the-1cycle-policy.html.

[2] Anurag Arnab and Philip H S Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1209–1218, 2017.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (12):2481–2495, 2017.

[4] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2858–2866, 2017.

[5] Piotr Bilinski and Victor Prisacariu. Dense decoder shortcut connections for single-pass semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6596–6605, 2018.

[6] Keshav Bimbraw. Autonomous cars: Past, present and future: A review of the developments in the last century, the present scenario and the expected future of autonomous vehicle technology. In *ICINCO 2015 - 12th International Conference on Informatics in Control, Automation and Robotics, Proceedings*, volume 1, pages 191–198. SciTePress, 2015. ISBN 9789897581229. doi: 10.5220/0005540501910198.

[7] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. COCO-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE Conference onComputer Vision and Pattern Recognition*, pages 1209–1218, 2018.

[8] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. 3 2019. URL http://arxiv.org/abs/1903.11027.

[9] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Semantic image segmentation with deep convolutional nets and

fully connected crfs. *CoRR*, abs/1412.7062, 2015. URL http://arxiv.org/abs/1412.7062.

[10] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3640–3649, 2016.

[11] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 441–450, 2017.

[12] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[13] 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, 2018.

[14] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv preprint arXiv:1802.02611*, 2018.

[15] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):603–619, 2002.

[16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[17] Jifeng Dai, Kaiming He, and Jian Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3992–4000, 2015.

[18] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016.

[19] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function. *arXiv preprint arXiv:1708.02551*, 2017.

[20] Dan Deng, Haifeng Liu, Xuelong Li, and Deng Cai. PixelLink: Detecting scene text via instance segmentation. *arXiv preprint arXiv:1801.01315*, 2018.

[21] Mark Everingham, S M Ali Eslami, Luc Van Gool, Christopher K I Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[22] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[23] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra 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.

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

[25] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 447–456, 2015.

[26] Mohammad Havaei, Axel Davy, David Warde-Farley, Antoine Biard, Aaron Courville, Yoshua Bengio, Chris Pal, Pierre-Marc Jodoin, and Hugo Larochelle. Brain tumor segmentation with deep neural networks. *Medical image analysis*, 35:18–31, 2017.

[27] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. ISBN 9781467388504. doi: 10.1109/CVPR.2016.90.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[29] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: exploiting visibility for 3D object detection. 12 2019. URL http://arxiv.org/abs/1912.04986.

[30] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 12 2015.

[31] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *CVPR*,

volume 3, page 9, 2017.

[32] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. *arXiv preprint arXiv:1801.00868*, 2018.

[33] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic Feature Pyramid Networks. *arXiv preprint arXiv:1901.02446*, 2019.

[34] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven 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. ISBN 9781538680940. doi: 10.1109/IROS.2018.8594049.

[35] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar 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. URL http://arxiv.org/abs/1812.05784.

[36] Ruiyu Li, Kaican Li, Yi-Chun Kuo, Michelle Shu, Xiaojuan Qi, Xiaoyong Shen, and Jiaya Jia. Referring Image Segmentation via Recurrent Refinement Networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2018.

[37] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2359–2367, 2017.

[38] Xiaodan Liang, Yunchao Wei, Xiaohui Shen, Jianchao Yang, Liang Lin, and Shuicheng Yan. Proposal-free network for instance-level object segmentation. *arXiv preprint arXiv:1509.02636*, 2015.

[39] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3194–3203, 2016.

[40] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Cvpr*, pages 1925–1934, 2017.

[41] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[42] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal

loss for dense object detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[43] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Advances in Neural Information Processing Systems*, pages 9628–9639, 2018.

[44] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[45] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37, 2016.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[47] Wenjie Luo, Bin Yang, and Raquel 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. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00376.

[48] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl 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. URL http://arxiv.org/abs/1903.08701.

[49] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4990–4999, 2017.

[50] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative embedding: End-to-end learning for joint detection and grouping. In *Advances in Neural Information Processing Systems*, pages 2277–2287, 2017.

[51] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.

[52] Viet-Quoc Pham, Satoshi Ito, and Tatsuo Kozakaya. BiSeg: Simultaneous instance segmentation and semantic segmentation with fully convolutional networks. *arXiv preprint arXiv:1706.02135*, 2017.

[53] Pedro H O Pinheiro and Ronan Collobert. Recurrent convolutional neural networks for scene labeling. In *31st International Conference on Machine Learning (ICML)*, number EPFL-CONF-199822, 2014.

[54] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January, pages 77–85. Institute of Electrical and Electronics Engineers Inc., 11 2017. ISBN 9781538604571. doi: 10.1109/CVPR.2017.16.

[55] Charles R. Qi, Li Yi, Hao Su, and Leonidas 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. URL http://arxiv.org/abs/1706.02413.

[56] 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.

[57] Baoguang Shi, Xiang Bai, and Serge Belongie. Detecting Oriented Text in Natural Images by Linking Segments. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3482–3490, 2017.

[58] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019-June:770–779, 12 2018. URL http://arxiv.org/abs/1812.04244.

[59] Martin Simon, Stefan Milz, Karl Amende, and Horst-Michael Gross. Complex-YOLO: Real-time 3D object detection on point clouds. 3 2018. URL http://arxiv.org/abs/1803.06199.

[60] Jonas Uhrig, Marius Cordts, Uwe Franke, and Thomas Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition*, pages 14–25, 2016.

[61] Jonas Uhrig, Eike Rehder, Björn Fröhlich, Uwe Franke, and Thomas Brox. Box2Pix: Single-shot instance segmentation by assigning pixels to object boxes. In *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[62] Jessica Van Brummelen, Marie O'Brien, Dominique Gruyer, and Homayoun Najjaran. Autonomous vehicle perception: The technology of today and tomorrow, 4 2018. ISSN 0968090X.

[63] Fangting Xia, Peng Wang, Liang-Chieh Chen, and Alan L Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. In *European Conference on Computer Vision*, pages 648–663, 2016.

[64] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. UPSNet: A unified panoptic segmentation network. *arXiv preprint*, page arXiv:1901.03784, 1 2019.

[65] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors (Switzerland)*, 18(10), 10 2018. ISSN 14248220. doi: 10.3390/s18103337.

[66] Bin Yang, Wenjie Luo, and Raquel 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. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00798.

[67] Tien-Ju Yang, Maxwell D Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. DeeperLab: Single-Shot Image Parser. *arXiv preprint arXiv:1902.05093*, 2019.

[68] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *Proceedings of the IEEE Computer SocietyConference on Computer Vision and Pattern Recognition*, pages 702–709, 2012.

[69] Shunyu Yao, Tzu Ming Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, Bill Freeman, and Josh Tenenbaum. 3D-aware scene manipulation via inverse graphics. In *Advances in Neural Information Processing Systems*, pages 1891–1902, 2018.

[70] Yangyang Ye, Houjin Chen, Chi Zhang, Xiaoli Hao, and Zhaoxiang Zhang. SARPNET: Shape attention regional proposal network for liDAR-based 3D object detection. *Neurocomputing*, 379:53–63, 2 2020. ISSN 18728286. doi: 10.1016/j.neucom.2019.09.086.

[71] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *Proceedings of International Conference on Learning Representations*, 2016.

[72] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: common practices and emerging technologies. 6 2019. URL http://arxiv.org/abs/1906.05113.

[73] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[74] Ziyu Zhang, Alexander G Schwing, Sanja Fidler, and Raquel Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2614–2622, 2015.

[75] Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 669–677, 2016.

[76] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2881–2890, 2017.

[77] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. ICNet for Real-Time Semantic Segmentation on High-Resolution Images. In *The European Conference on Computer Vision (ECCV)*, 9 2018.

[78] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H S Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1529–1537, 2015.

[79] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, page 4, 2017.

[80] Yin Zhou and Oncel 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. ISBN 9781538664209. doi: 10.1109/CVPR.2018.00472.

[81] Yin Zhou, Pei Sun, Yu Zhang, Dragomir Anguelov, Jiyang Gao, Tom Ouyang, James Guo, Jiquan Ngiam, and Vijay Vasudevan. End-to-end multi-view fusion for 3D object detection in lidar point clouds. 10 2019. URL http://arxiv.org/abs/1910.06528.

[82] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3D object detection. 8 2019. URL http://arxiv.org/abs/1908.09492.