CenterPoint++ submission to the Waymo Real-time 3D Detection Challenge

Tianwei Yin
UT Austin
yintianwei@utexas.edu

Xingyi Zhou
UT Austin
zhouxy@cs.utexas.edu

Philipp Krähenbühl
UT Austin
philkr@cs.utexas.edu

Abstract

In this report, we present our "CenterPoint++" solution for the CVPR 2021 Waymo real-time 3D detection challenge. Our submission builds upon our previous CenterPoint 3D recognition framework. CenterPoint detects centers of objects using a keypoint detector on top of a VoxelNet backbone. It regresses to other attributes, including offsets, 3D size, and 3D orientation, from extracted features at the center of the object. In a second stage, CenterPoint refines these estimates using additional point features on the object. We improve the original CenterPoint baseline with an IoU-aware confidence rectification module and multiple design choices change to optimize for real-time inference. Our final model achieves 72.8 *mAPH on the Waymo 3D detection test set while running at* 17.5*FPS. The code is available at* https://github.com/tianweiy/CenterPoint.

1. Introduction

The Waymo real-time 3D detection challenge at CVPR'21 is among the first few popular competitions that encourages a good speed/accuracy tradeoff. The challenge requires the model to accurately detect 3D objects within a $70ms$ latency constraint. We develop our submission based on our previous CenterPoint [16] detector and introduce multiple modifications (tricks) for real-time inference. Specifically, inspired by [14], we add an IoU-aware confidence rectification module to alleviate the misalignment between localization accuracy and classification confidence. We also apply multiple tricks including random translation augmentation, multi-sweep temporal point cloud input [1], and dynamic voxelization [18]. Ablation studies verify the effectiveness of the proposed method. We will first review our CenterPoint 3D detection framework [16] in Section 2 and Section 3.

2. Preliminaries

2D CenterNet [17] phrases object detection as keypoint estimation. It takes an input image and predicts a $w \times h$ heatmap $\hat{Y} \in [0,1]^{w \times h \times K}$ for each of K classes. Each local maximum (i.e., pixels whose value is greater than its eight neighbors) in the output heatmap corresponds to the center of a detected object. It additionally regresses to a size map $\hat{S} \in \mathbb{R}^{w \times h \times 2}$ that stores the width and height at the object centers. During testing, the object sizes are read out at each peak location.

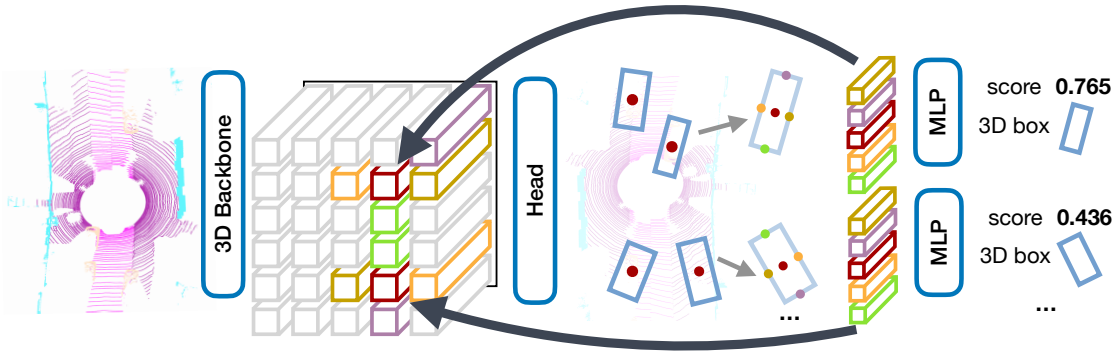
3D Detection Let $\mathcal{P} = \{(x, y, z, r)_i\}$ be an orderless point-cloud of 3D location (x, y, z) and reflectance r measurements. 3D object detection aims to predict a set of 3D object bounding boxes $\mathcal{B} = \{b_k\}$ in the bird eye view from this point-cloud. Each bounding box $b = (u, v, d, w, l, h, \alpha)$ consists of a center location (u, v, d), relative to the objects ground plane, and 3D size (w, l, h), and rotation expressed by yaw α.

Modern 3D object detectors [3, 5, 15, 19] uses a 3D encoder that quantizes the point-cloud into regular bins. A point-based network [9] then extracts features for all points inside a bin. The 3D encoder then pools these features into its primary feature representation. Most of the computation happens in the backbone network, which operates solely on these quantized and pooled feature representations. The output of a backbone network is a map-view feature-map $\mathbf{M} \in \mathbb{R}^{W \times L \times F}$ of width W and length L with F channels in a map-view reference frame. Both width and height directly relate to the resolution of individual voxel bins and the backbone network's stride. Common backbones include VoxelNet [15, 19] and PointPillars [5]. After the backbone, the inputs are converted into a bird-eye view feature map $\mathbf{M}$. In the next section, we introduce the center-based detection head on top of existing 3D backbones (VoxelNet or PointPillars).

3. CenterPoint

Figure 1 shows the overall framework of the CenterPoint model. Let $\mathbf{M} \in \mathbb{R}^{W \times H \times F}$ be the output of the 3D backbone. The first stage of CenterPoint predicts a class-specific heatmap, object size, a sub-voxel location refinement, and rotation. A second stage then refines the object score and locations.

Center heatmap head. The training objective of the



(a) Point. cloud (b) Map-view features (c) First stage: Centers and 3D boxes (d) Second stage: Score and 3D boxes

Figure 1: Overview of our CenterPoint framework. We rely on a standard 3D backbone that extracts map-view feature representation from Lidar point-clouds. Then, a 2D CNN architecture detection head finds object centers and regress to full 3D bounding boxes using center features. From this box prediction, we extract point features at the 3D centers of each face of the estimated 3D bounding box, and pass them into a MLP to predict an IoU-guided confidence score and box regression refinement. Best viewed in color.

heatmap output follows 2D CenterNet [17]. However, hyper-parameters need to change. Objects in a top-down map view are sparser than in an image. In map-view, distances are absolute, while an image-view distorts them by perspective. Consider a road scene, in map-view the area occupied by vehicles small, but in image-view, a few large objects may occupy most of the screen. Furthermore, the compression of the depth-dimension in perspective projection naturally places object centers much closer to each other in image-view. To counteract this, we increase the positive supervision for the target heatmap Y by enlarging the Gaussian peak rendered at each ground truth object center. Specifically, we set the Gaussian radius to $\sigma = \max(f(wl), \tau)$, where $\tau = 2$ is the smallest allowable Gaussian radius, and f is a radius function defined in CornerNet [6]. In this way, CenterPoint maintains the center-based target assignment's simplicity; the model gets denser supervision from nearby pixels.

Regression heads. We store several object properties at center-features of objects: a sub-voxel location refinement $o \in \mathbb{R}^2$, height-above-ground $h_g \in \mathbb{R}$, the 3D size $s \in \mathbb{R}^3$, and a yaw rotation angle $(\sin(\alpha), \cos(\alpha)) \in [-1, 1]^2$. The sub-voxel location refinement o reduces the quantization error from voxelization and striding of the backbone network. The height-above-ground h_g helps localize the object in 3D and adds the missing elevation information removed by the map-view projection. The orientation prediction uses the sine and cosine of the yaw angle as a continuous regression target. Combined with box size, these regression heads provide the full state information of the 3D bounding box. Each output uses its own head. We train all outputs using an L1 loss at the ground truth center location. We regress to logarithmic size to better handle boxes of various shapes. At inference time, we extract all properties by indexing into dense regression head outputs at each object's peak location.

CenterPoint combines all heatmap and regression losses in one common objective and jointly optimizes them. It simplifies and improves previous anchor-based 3D detectors. However, all object properties are currently inferred from the object's center-feature, which may not contain sufficient information for accurate object localization. For example, in autonomous driving, the sensor often only sees the side of the object, but not its center. Next, we improve CenterPoint by using a second refinement stage with a light-weight point-feature extractor.

3.1. Two-Stage CenterPoint

We use CenterPoint unchanged as a first stage. The second stage extracts additional point-features from the output of the backbone. We extract one point-feature from the 3D center of each face of the predicted bounding box. Note that the bounding box center, top and bottom face centers all project to the same point in map-view. We thus only consider the four outward-facing box-faces together with the predicted object center. For each point, we extract a feature using bilinear interpolation from the backbone map-view output $\mathbf{M}$. Next, we concatenate the extracted point-features and pass them through an MLP. The second stage predicts a class-agnostic confidence score and box refinement on top of one-stage CenterPoint's prediction results.

For class-agnostic confidence score prediction, we follow [4, 7, 10, 12] and use a score target I guided by the box's 3D IoU with the corresponding ground truth bounding box:

$$I = \min(1, \max(0, 2 \times IoU_t - 0.5)) \quad (1)$$

where IoU_t is the IoU between the t-th proposal box and the

ground-truth. We train using a binary cross entropy loss:

$$L_{score} = -I_t \log(\hat{I}_t) - (1 - I_t) \log(1 - \hat{I}_t) \quad (2)$$

where $\hat{I}_t$ is the predicted confidence score. During the inference, we directly use the class prediction from one-stage CenterPoint and computes the final confidence score as the geometric average of the two scores $\hat{Q}_t = \sqrt{\hat{Y}_t * \hat{I}_t}$ where $\hat{Q}_t$ is the final prediction confidence of object t and $\hat{Y}_t = \max_{0 \leq k \leq K} \hat{Y}_{p,k}$ and $\hat{I}_t$ are the first stage (after rectification) and second stage confidence of object t, respectively.

For box regression, the model predicts a refinement on top of first stage proposals, and we train the model with L1 loss. Our two-stage CenterPoint simplifies and accelerates previous two-stage 3D detectors that use expensive PointNet-based feature extractor and RoIAlign operations [10, 11].

4. CenterPoint Variants

To further improve the real-time 3D detection performance of CenterPoint, we explore multiple modifications below.

IoU-Aware Confidence Rectification Module A common issue for 2D and 3D object detection is the misalignment between localization accuracy and classification confidence [7, 14]. To alleviate this problem, we follow CIA-SSD [14] to process the first stage predicted confidence score with an IoU-Aware confidence rectification module. Specifically, we add one additional regression head to predict the IoU between object detections and corresponding ground truth boxes. We supervise the IoU prediction with an L1 loss and rectify the confidence score $\hat{Y}_t = \max_{0 \leq k \leq K} \hat{Y}_{p,k}$ at inference time with the predicted IoU IoU_t

$$\hat{C}_t = \hat{Y}_t \times IoU_t^{\beta} \quad (3)$$

where β is a hyperparmeter and empirically set to 2 in all experiments.

Temporal Multi-sweep Point Cloud Input The Waymo dataset contains temporal point cloud sequences which can be utilized to produce a denser point-cloud and enable a better orientation estimation. Specifically, we follow [1] to append a timestamp feature to each lidar point in the current frame and last two frames. We then transform and merge Lidar points of the last two frames into the current frame using the vehicle's pose information. This simple multi-sweep point cloud aggregation gives a large 2.5 mAPH boost (see experiments).

Dynamic Voxelization Previous methods [10, 15, 16] use a CPU-based voxelization implementation which is slow especially for multi-sweep point cloud inputs. The voxelization step alone can take more than 50ms making it unsuitable for real-time inference. MVF [18] proposed the dynamic voxelization to speed up voxelization and avoid information loss during the quantization process. In our submission, we reimplement the dynamic voxelization using parallel computation on GPU. The overall implementation is less than ten lines of code using PyTorch and reduces the latency by an order of magnitude. Please refer to our opensourced code for details.

5. Experiments

Waymo Open Dataset. Waymo Open Dataset [13] contains 798 training sequences and 202 validation sequences for vehicles, pedestrians, and cyclists. The point-clouds contain 64 lanes of Lidar corresponding to 180k points every 0.1s. The official 3D detection evaluation metrics include 3D bounding box mean average precision (mAP) and mAP weighted by heading accuracy (mAPH). The mAP and mAPH are based on an IoU threshold of 0.7 for vehicles and 0.5 for pedestrians and cyclists.

Our Waymo model uses a detection range of $[-75.2\text{m}, 75.2\text{m}]$ for the X and Y axis, and $[-2\text{m}, 4\text{m}]$ for the Z axis. The voxel size is $(0.1\text{m}, 0.1\text{m}, 0.15\text{m})$ following PV-RCNN [10].

Implementation Details. We use the same VoxelNet architecture as prior works [10, 15]. We transform and merge Lidar points of the last two frames into the current frame to produce a denser point-cloud and enable a better orientation estimation. For data augmentation, we use random flipping along both X and Y axis, and global scaling with a random factor from $[0.95, 1.05]$. We also apply a random global rotation between$[-\pi/4, \pi/4]$ and a random translation between $[-0.5m, 0.5m]$. The random translation is not used in previous methods [10, 16] but we found it to be useful on the Waymo dataset. We adopt ground-truth sampling [15] on Waymo to deal with the long tail class distribution, which copies and pastes points inside an annotated box from one frame to another frame.

We train the model using AdamW [8] optimizer with one-cycle learning rate policy [2], with max learning rate 3e-3, weight decay 0.01, and momentum 0.85 to 0.95. We use a batch size of 16 evenly distributed across 4 V100 GPUs. We train the model for 20 epochs which takes about 3 days. During inference, we apply dynamic voxelization [18] and run the VoxelNet backbone in FP16 mode to reduce latency.

Here we present results related to the Waymo Real-time 3D detection challenge. More ablations and analysis are available in our CenterPoint paper [16].

5.1. Main Result

Table 3 shows the CVPR 2021 Waymo real-time 3D detection challenge leaderboard. Our submission ranked second among all entries. Our detection performance is comparable with the top-performing submission (72.8 mAPH vs. 73.1

CenterPoint [16]	Random Translation	Temporal	Dynamic Voxelization	IoU Rectification	Two-stage	mAPH↑
✓						65.1
✓	✓					65.8
✓	✓	✓				68.3
✓	✓	✓	✓			68.3
✓	✓	✓	✓	✓		70.8
✓	✓	✓	✓	✓	✓	**71.6**

Table 1: Ablation studies for 3D detection on Waymo validation set. We ablate each component of our submission compared to a single-frame single-stage CenterPoint baseline.

Point Cloud IO	Voxelization	VoxelNet Backbone	BEV Neck	Head	Post-processing	Two-stage
6ms	2ms	38ms	4ms	3ms	2ms	3ms

Table 2: Runtime analysis of different steps of our model during inference. Latecny is measured on a Titan RTX GPU.

mAPH) with a faster runtime (57.1 ms vs. 60.1 ms). A latency breakdown of our model is provided in Table 2.

5.2. Ablation Studies

Table 1 ablates the improvement of our entry based on our original CenterPoint [16]. We redesign the baseline model to find a good tradeoff between speed and accuracy. We first enhancing the single-frame baseline with 3-sweep temporal point cloud following [1]. This brings 2.5mAPH improvements with $\sim$ 12ms overhead. Adding dynamic voxelization decreases the latency of voxelization by an order of magnitude (from $> 50ms$ to $< 3ms$) due to the fast parallel computation on GPU. IoU rectification and Two-stage refinement give 3.3mAPH further improvement and our final model achieves 71.6mAPH on the Waymo validation set. For our final submission, we train our model on the joint dataset of Waymo training and validation splits. This gives a slight improvement for test set accuracy (72.8 vs. 72.6).

6. Conclusion

In this report, we demonstrate how to apply CenterPoint 3D detection framework to the Waymo real-time 3D detection challenge. Our CenterPoint detector is widely used in state-of-the-art methods on the popular Waymo Open [13], and nuScenes [1] datasets. Notably, in NeurIPS 2020 nuScenes 3D Detection challenge, CenterPoint forms the basis of 3 of the top 4 winning entries. On the Waymo 3D detection benchmark, our CenterPoint++ submission achieves 72.8 mAPH while running at 17.5 FPS. We hope our method and open-sourced code [1] can contribute to the switch from anchor-based to point-based 3D object representation in the future.

Method	mAPH↑	latency↓
HorizonLidar3DV2	73.1	60.1
Ours	72.8	57.1
AFDetV2	72.6	55.9
X_Autonomous3D	70.5	68.4
HorizonLidar3DV2-Lite	70.0	46.9
HIKIVISION_LIDAR	67.3	54.1
Light-FMFNet	62.2	62.3
QuickDet	59.9	67.8

Table 3: State-of-the-art comparisons for 3D detection on Waymo leaderboard. We show the mean average precision weighted by heading accuracy (mAPH), and inference latency.

References

[1] 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. *CVPR*, 2020.

[2] Sylvain Gugger. The 1cycle policy. https://sgugger.github.io/the-1cycle-policy.html, 2018.

[3] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. *CVPR*, 2020.

[4] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. *ECCV*, 2018.

[5] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *CVPR*, 2019.

[6] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. *ECCV*, 2018.

[7] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. *CVPR*, 2019.

[8] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, 2017.

[1] https://github.com/tianweiy/CenterPoint

[10] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. *CVPR*, 2020.

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

[12] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *TPAMI*, 2020.

[13] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: An open dataset benchmark. *CVPR*, 2020.

[14] Sijin Chen Li Jiang Chi-Wing Fu Wu Zheng, Weiliang Tang. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. In *AAAI*, 2021.

[15] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018.

[16] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021.

[17] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019.

[18] 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. In *Conference on Robot Learning*, pages 923–932. PMLR, 2020.

[19] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CVPR*, 2018.