Center-based 3D Object Detection and Tracking — Supplementary Materials

A. Tracking algorithm

Algorithm 1: Center-based Tracking

Input : $T^{(t-1)} = \{(\mathbf{p}, \mathbf{v}, c, \mathbf{q}, id, a)_j^{(t-1)}\}_{j=1}^{M}$: Tracked objects in the previous frame, with center $\mathbf{p}$, ground plane velocity $\mathbf{v}$, category label c, other bounding box attributes $\mathbf{q}$, tracking id id, and inactive age a (active tracks will have $a = 0$).
$\hat{D}^{(t)} = \{(\hat{\mathbf{p}}, \hat{\mathbf{v}}, \hat{c}, \hat{\mathbf{q}})_i^{(t)}\}_{i=1}^{N}$: Detections in the current frame in descending confidence.

Output : $T^{(t)} = \{(\mathbf{p}, \mathbf{v}, c, \mathbf{q}, id, a)_{j=1}^{K}\}$: Tracked Objects.

Hyper parameters: Matching distance threshold τ; Max inactive age A.
Initialization: Tracks $T^{(t)}$, and matches $\mathcal{S}$ are initialized as empty sets.
$T^{(t)} \leftarrow \emptyset, \mathcal{S} \leftarrow \emptyset$
$F \leftarrow Cost(\hat{D}^{(t)}, T^{(t-1)})$ // $F_{ij} = ||\hat{\mathbf{p}}_i^{(t)} - \hat{\mathbf{v}}, \mathbf{p}_j^{(t-1)}||_2$
for $i \leftarrow 1$ *to* N **do**
  $j \leftarrow \arg\min_{j \notin \mathcal{S}} F_{ij}$
  // Class-wise distance threshold τ_c
  if $\mathbf{F}_{ij} \leq \tau_c$ **then**
    // Associate with tracked object
    $a_i^{(t)} \leftarrow 0$
    $T^{(t)} \leftarrow T^{(t)} \cup \{(\hat{D}_i^{(t)}, id_j^{(t-1)}, a_i^{(t)})\}$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$ // Mark track j as matched
  end
  else
    // Initialize a new track
    $a_i^{(t)} \leftarrow 0$
    $T^{(t)} \leftarrow T^{(t)} \cup \{(\hat{D}_i^{(t)}, newID, a_i^{(t)})\}$
  end
end
for $j \leftarrow 1$ *to* M **do**
  if $j \notin \mathcal{S}$ **then**
    // Unmatched tracks
    if $T.a_j^{(t-1)} < A$ **then**
      $T.a_j^{(t)} \leftarrow T.a_j^{(t-1)} + 1$
      $T.p_j^{(t)} \leftarrow T.p_j^{(t-1)} + T.v_j^{(t-1)}$ // Update the center location
      $T^{(t)} \leftarrow T^{(t)} \cup \{T_j^{(t-1)}\}$
    end
  end
end
Return $T^{(t)}$

B. Implementation Details

Our implementation is based on the open-sourced code of CBGS [14][1]. CBGS provides implementations of PointPillars [6] and VoxelNet [13] on nuScenes. For Waymo experiments, we use the same architecture for VoxelNet and increases the output stride to 1 for PointPillars [6] following the dataset's reference implementation[2].

A common practice [1, 9, 11, 14] in nuScenes is to transform and merge the Lidar points of non-annotated frames into its following annotated frame. This produces a denser point-cloud and enables a more reasonable velocity estimation. We follow this practice in all nuScenes experiments.

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 use a random global rotation between $[-\pi/8, \pi/8]$ for nuScenes [14] and $[-\pi/4, \pi/4]$ for Waymo [8]. We also use the ground-truth sampling [10] on nuScenes to deal with the long tail class distribution, which copies and pastes points inside an annotated box from one frame to another frame.

For nuScenes dataset, we follow CBGS [14] to optimize the model using AdamW [7] optimizer with one-cycle learning rate policy [4], with max learning rate 1e-3, weight decay 0.01, and momentum 0.85 to 0.95. We train the models with batch size 16 for 20 epochs on 4 V100 GPUs.

We use the same training schedule for Waymo models except a learning rate 3e-3, and we train the model for 30 epochs following PV-RCNN [8]. To save computation on large scale Waymo dataset, we finetune the model for 6 epochs with second stage refinement modules for various ablation studies. All ablation experiments are conducted in this same setting.

For the nuScenes test set submission, we use a input grid size of $0.075m \times 0.075m$ and add two separate deformable convolution layers [3] in the detection head to learn different features for classification and regression. This improves CenterPoint-Voxel's performance from 64.8 NDS to 65.4 NDS on nuScenes validation. For the nuScenes tracking benchmark, we submit our best CenterPoint-Voxel model with flip testing, which yields a result of 66.5 AMOTA on nuScenes validation.

C. nuScenes Performance across classes

We show per-class comparisons with state-of-the-art methods in Table 1.

[1] https://github.com/poodarchu/Det3D
[2] https://github.com/tensorflow/lingvo/tree/master/lingvo/tasks/car

Method	mAP	NDS	Car	Truck	Bus	Trailer	CV	Ped	Motor	Bicycle	TC	Barrier
WYSIWYG [5]	35.0	41.9	79.1	30.4	46.6	40.1	7.1	65.0	18.2	0.1	28.8	34.7
PointPillars [6]	30.5	45.3	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
PointPainting [9]	46.4	58.1	77.9	35.8	36.2	37.3	15.8	73.3	41.5	24.1	62.4	60.2
CVCNet [2]	55.3	64.4	82.7	46.1	46.6	49.4	**22.6**	79.8	**59.1**	**31.4**	65.6	69.6
PMPNet [12]	45.4	53.1	79.7	33.6	47.1	43.1	18.1	76.5	40.7	7.9	58.8	48.8
SSN [15]	46.4	58.1	80.7	37.5	39.9	43.9	14.6	72.3	43.7	20.1	54.2	56.3
CBGS [14]	52.8	63.3	81.1	48.5	54.9	42.9	10.5	80.1	51.5	22.3	70.9	65.7
Ours	**58.0**	**65.5**	**84.6**	**51.0**	**60.2**	**53.2**	17.5	**83.4**	53.7	28.7	**76.7**	**70.9**

Table 1: State-of-the-art comparisons for 3D detection on nuScenes test set. We show the NDS, mAP, and mAP for each class. Abbreviations: construction vehicle (CV), pedestrian (Ped), motorcycle (Motor), and traffic cone (TC).

Method	mAP	NDS
Baseline	57.1	65.4
+ PointPainting [9]	62.7	68.0
+ Flip Test	64.9	69.4
+ Rotation	66.2	70.3
+ Ensemble	67.7	71.4
+ Filter Empty	68.2	71.7

Table 2: Ablation studies for 3D detection on nuScenes validation.

D. nuScenes Detection Challenge

As a general framework, CenterPoint is complementary to contemporary methods and was used by three of the top 4 entries in the NeurIPS 2020 nuScenes detection challenge. In this section, we describe the details of our winning submission which significantly improved 2019 challenge winner CBGS [14] by 14.3 mAP and 8.1 NDS. We report some improved results in Table 2. We use PointPainting [9] to annotate each lidar point with image-based instance segmentation results generated by a Cascade RCNN model trained on nuImages[3]. This improves the NDS from 65.4 to 68.0. We then perform two test-time augmentations including double flip testing and point-cloud rotation around the yaw axis. Specifically, we use [0°, ± 6.25°, ± 12.5°, ± 25°] for yaw rotations. Theses test time augmentations improve the NDS from 68.0 to 70.3. In the end, we ensemble five models with input grid size between $[0.05m, 0.05m]$ to $[0.15m, 0.15m]$ and filter out predictions with zero number of points, which yields our best results on nuScenes validation, with 68.2 mAP and 71.7 NDS.

E. Experiments of Training Hyperparmeters

We have investigated the influence of different hyperparameters on nuScenes validation. We tried different hyperparameters on CenterPoint-Pillars and we list some results here. The flip-augmentations along both axis improves single X-asis flipping by 2.4mAP. Log normalization of the regression targets give the same performance as no normalization. And the cosine-sine rotation representation performs similarly with directional classifier approaches used in [8, 10].

[3] acquired from https://github.com/open-mmlab/mmdetection3d/tree/master/configs/nuimages

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] Qi Chen, Lin Sun, Ernest Cheung, Kui Jia, and Alan Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. *NeurIPS*, 2020.

[3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *ICCV*, 2017.

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

[5] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. *CVPR*, 2020.

[6] 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.

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

[8] 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.

[9] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. *CVPR*, 2020.

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

[11] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. *CVPR*, 2020.

[12] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. *CVPR*, 2020.

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

[14] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv:1908.09492*, 2019.

[15] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. *ECCV*, 2020.