arXiv:2006.11275v2 [cs.CV] 6 Jan 2021

Center-based 3D Object Detection and Tracking

Tianwei Yin
UT Austin
yintianwei@utexas.edu

Xingyi Zhou
UT Austin
zhouxy@cs.utexas.edu

Philipp Krähenbühl
UT Austin
philkr@cs.utexas.edu

Abstract

Three-dimensional objects are commonly represented as 3D boxes in a point-cloud. This representation mimics the well-studied image-based 2D bounding-box detection but comes with additional challenges. Objects in a 3D world do not follow any particular orientation, and box-based detectors have difficulties enumerating all orientations or fitting an axis-aligned bounding box to rotated objects. In this paper, we instead propose to represent, detect, and track 3D objects as points. Our framework, CenterPoint, first detects centers of objects using a keypoint detector and regresses to other attributes, including 3D size, 3D orientation, and velocity. In a second stage, it refines these estimates using additional point features on the object. In CenterPoint, 3D object tracking simplifies to greedy closest-point matching. The resulting detection and tracking algorithm is simple, efficient, and effective. CenterPoint achieved state-of-the-art performance on the nuScenes benchmark for both 3D detection and tracking, with 65.5 NDS and 63.8 AMOTA for a single model. On the Waymo Open Dataset, CenterPoint outperforms all previous single model method by a large margin and ranks first among all Lidar-only submissions. The code and pretrained models are available at https://github.com/tianweiy/CenterPoint.

1. Introduction

Strong 3D perception is a core ingredient in many state-of-the-art driving systems [1, 50]. Compared to the well-studied 2D detection problem, 3D detection on point-clouds offers a series of interesting challenges: First, point-clouds are sparse, and most regions of 3D space are without measurements [23]. Second, the resulting output is a three-dimensional box that is often not well aligned with any global coordinate frame. Third, 3D objects come in a wide range of sizes, shapes, and aspect ratios, e.g., in the traffic domain, bicycles are near planer, buses and limousines elongated, and pedestrians tall. These marked differences between 2D and 3D detection made a transfer of ideas between the two domain harder [45, 47, 60]. The crux of it is that an axis-aligned 2D box [16, 17] is a poor proxy of a free-form 3D object. One solution might be to classify a different template (anchor) for each object orientation [58, 59], but this unnecessarily increases the computational burden and may introduce a large number of potential false-positive detections. We argue that the main underlying challenge in linking up the 2D and 3D domains lies in this representation of objects.

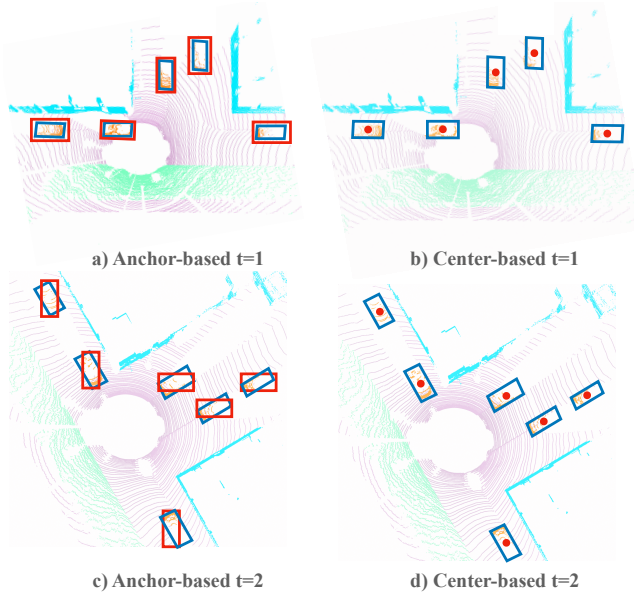


Figure 1: We present a center-based framework to represent, detect and track objects. Previous anchor-based methods use axis-aligned anchors with respect to ego-vehicle coordinate. When the vehicle is driving in straight roads, both anchor-based and our center-based method are able to detect objects accurately (top). However, during a safety-critical left turn (bottom), anchor-based methods have difficulty fitting axis-aligned bounding boxes to rotated objects. Our center-based model accurately detect objects through rotationally invariant points. Best viewed in color.

In this paper, we show how representing objects as points (Figure 1) greatly simplifies 3D recognition. Our

two-stage 3D detector, CenterPoint, finds centers of objects and their properties using a keypoint detector [64], a second-stage refines all estimates. Specifically, CenterPoint uses a standard Lidar-based backbone network, i.e., VoxelNet [56, 66] or PointPillars [28], to build a representation of the input point-cloud. It then flattens this representation into an overhead map-view and uses a standard image-based keypoint detector to find object centers [64]. For each detected center, it regresses to all other object properties such as 3D size, orientation, and velocity from a point-feature at the center location. Furthermore, we use a light-weighted second stage to refine the object locations. This second stage extracts point-features at the 3D centers of each face of the estimated objects 3D bounding box. It recovers the lost local geometric information due to striding and a limited receptive field, and brings a decent performance boost with minor cost.

The center-based representation has several key advantages: First, unlike bounding boxes, points have no intrinsic orientation. This dramatically reduces the object detector's search space while allowing the backbone to learn the rotational invariance of objects and rotational equivariance of their relative rotation. Second, a center-based representation simplifies downstream tasks such as tracking. If objects are points, tracklets are paths in space and time. CenterPoint predicts the relative offset (velocity) of objects between consecutive frames, which are then linked up greedily. Thirdly, point-based feature extraction enables us to design an effective two-stage refinement module that is much faster than previous approaches [44–46].

We test our models on two popular large datasets: Waymo Open Dataset [48], and nuScenes Dataset [6]. We show that a simple switch from the box representation to center-based representation yields a 3-4 mAP increase in 3D detection under different backbones [28, 56, 66, 67]. Two-stage refinement further brings an additional 2 mAP boost with small ($< 10\%$) computation overhead. Our best single model achieves 71.8 and 66.4 level 2 mAPH for vehicle and pedestrian detection on Waymo, 58.0 mAP and 65.5 NDS on nuScenes, outperforming all published methods on both datasets. Notably, in NeurIPS 2020 nuScenes 3D Detection challenge, CenterPoint is adopted in 3 of the top 4 winning entries. For 3D tracking, our model performs at 63.8 AMOTA outperforming the prior state-of-the-art by 8.8 AMOTA on nuScenes. On Waymo 3D tracking benchmark, our model achieves 59.4 and 56.6 level 2 MOTA for vehicle and pedestrian tracking, respectively, surpassing previous methods by up to 50%. Our end-to-end 3D detection and tracking system runs near real-time, with 11 FPS on Waymo and 16 FPS on nuScenes.

2. Related work

2D object detection predicts axis-algined bounding box from image inputs. The RCNN family [16, 17, 21, 43] finds a category-agnostic bounding box candidates, then classify and refine it. YOLO [42], SSD [33], and RetinaNet [32] directly find a category-specific box candidate, sidestepping later classification and refinement. Center-based detectors, e.g. CenterNet [64] or CenterTrack [63], directly detect the implicit object center point without the need for candidate boxes. Many 3D object detectors [20, 45, 47, 60] evolved from these 2D object detector. We argue center-based representation [63, 64] is a better fit in 3D application comparing to axis-aligned boxes.

3D object detection aims to predict three dimensional rotated bounding boxes [15, 28, 31, 39, 56, 60, 61]. They differ from 2D detectors on the input encoder. Vote3Deep [12] leverages feature-centric voting [51] to efficiently process the sparse 3D point-cloud on equally spaced 3D voxels. VoxelNet [66] uses a PointNet [40] inside each voxel to generate a unified feature representation from which a head with 3D sparse convolutions [18] and 2D convolutions produces detections. SECOND [56] simplifies the VoxelNet and speeds up sparse 3D convolutions. PIXOR [57] project all points onto a 2D feature map with 3D occupancy and point intensity information to remove the expensive 3D convolutions. PointPillars [28] replaces all voxel computation with a pillar representation, a single tall elongated voxel per map location, improving backbone efficiency. MVF [65] and Pillar-od [52] combine multiple view features to learn a more effective pillar representation. Our contribution focuses on the output representation, and is compatible with any 3D encoder and can improve them all.

VoteNet [38] detects objects through vote clustering using point feature sampling and grouping. In contrast, we directly regress to 3D bounding boxes through features at the center point without voting. Wong et al. [55] and Chen et al. [8] used similar multiple points representation in the object center region (i.e., point-anchors) and regress to other attributes. We use a single positive cell for each object and use a keypoint estimation loss.

Two-stage 3D object detection. Recent works considered directly applying RCNN style 2D detectors to the 3D domains [9, 44–46, 61]. Most of them apply RoIPool [43] or RoIAlign [21] to aggregate RoI-specific features in 3D space, using PointNet-based point [45] or voxel [44] feature extractor. These approaches extract region features from 3D Lidar measurements (points and voxels), resulting in a prohibitive run-time due to massive points. Instead, we extract sparse features of 5 surface center points from the intermediate feature map. This makes our second stage very efficient and keeps effective.

3D object tracking. Many 2D tracking algorithms [2, 4, 27, 54] readily track 3D objects out of the box. However, dedicated 3D trackers based on 3D Kalman filters [10, 53] still have an edge as they better exploit the three-dimensional motion in a scene. Here, we adopt a much simpler approach

following CenterTrack [63]. We use a velocity estimate together with the point-based detection to track centers of objects through multiple frames. This tracker is much faster and more accurate than dedicated 3D trackers [10, 53].

3. Preliminaries

2D CenterNet [64] rephrases 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 8 neighbors) in the output heatmap corresponds to the center of a detected object. To retrieve a 2D box, CenterNet regresses to a size map $\hat{S} \in \mathbb{R}^{w \times h \times 2}$ shared between all categories. For each detection object, the size-map stores its width and height at the center location. The CenterNet architecture uses a standard fully convolutional image backbone and adds a dense prediction head on top. During training, CenterNet learns to predict heatmaps with rendered Gaussian kernels at each annotated object center $\mathbf{q}_i$ for each class $c_i \in \{1 \dots K\}$, and regress to object size S at the center of the annotated bounding box. To make up for quantization errors introduced by the striding of the backbone architecture, CenterNet also regresses to a local offset $\hat{O}$.

At test time, the detector produces K heatmaps and dense class-agnostic regression maps. Each local maxima (peak) in the heatmaps corresponds to an object, with confidence proportional to the heatmap value at the peak. For each detected object, the detector retrieves all regression values from the regression maps at the corresponding peak location. Depending on the application domain, Non-Maxima Suppression (NMS) may be warranted.

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. Without loss of generality, we use an egocentric coordinate system with sensor at $(0, 0, 0)$ and yaw$= 0$.

Modern 3D object detectors [20, 28, 56, 66] uses a 3D encoder that quantizes the point-cloud into regular bins. A point-based network [40] 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 [56, 66] and PointPillars [28].

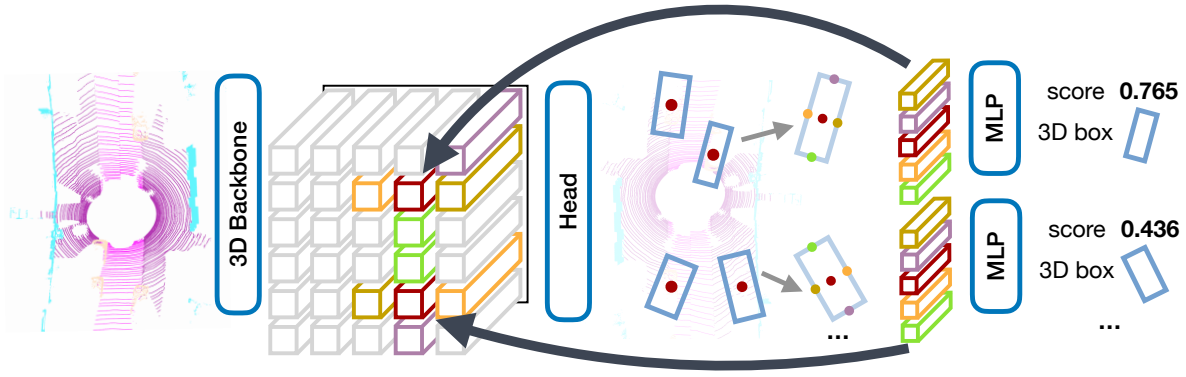
With a map-view feature map $\mathbf{M}$, a detection head, most commonly a one- [32] or two-stage [43] bounding-box detector, then produces object detections from some predefined bounding boxes anchored on this overhead feature-map. As 3D bounding boxes come with various sizes and orientation, anchor-based 3D detectors have difficulty fitting an axis-aligned 2D box to a 3D object. Moreover, during the training, previous anchor-based 3D detectors rely on 2D Box IoU for target assignment [44, 56], which creates unnecessary burdens for choosing positive/negative thresholds for different classes or different dataset. In the next section, we show how to build a principled 3D object detection and tracking model based on point representation. We introduce a novel center-based detection head but rely on existing 3D backbones (VoxelNet or PointPillars).

4. CenterPoint

Figure 2 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, rotation, and velocity. All outputs are dense predictions.

Center heatmap head. The center-head's goal is to produce a heatmap peak at the center location of any detected object. This head produces a K-channel heatmap $\hat{Y}$, one channel for each of K classes. During training, it targets a 2D Gaussian produced by the projection of 3D centers of annotated bounding boxes into the map-view. We use a focal loss [29, 64]. 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. Following the standard supervision of CenterNet [64] results in a very sparse supervisory signal, where most locations are considered background. 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 [29]. In this way, CenterPoint maintains the simplicity of the center-based target assignment; 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 $(\sin(\alpha), \cos(\alpha)) \in \mathbb{R}^2$. The sub-voxel location refinement o reduces the quantization error



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

Figure 2: 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. This box prediction is used to extract point features at the 3D centers of each face of the estimated 3D bounding box, which are passed into MLP to predict an IoU-guided confidence score and box regression refinement. Best viewed in color.

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. At training time, only ground truth centers are supervised using an L1 regression loss. 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.

Velocity head and tracking. To track objects through time, we learn to predict a two-dimensional velocity estimation $\mathbf{v} \in \mathbb{R}^2$ for each detected object as an additional regression output. The velocity estimate is special, as it requires two input map-views the current and previous time-step. It predicts the difference in object position between the current and the past frame. Like other regression targets, the velocity estimation is also supervised using L1 loss at the ground truth object's location at the current time-step.

At inference time, we use this offset to associate current detections to past ones in a greedy fashion. Specifically, we project the object centers in the current frame back to the previous frame by applying the negative velocity estimate and then matching them to the tracked objects by closest distance matching. Following SORT [4], we keep unmatched tracks up to $T = 3$ frames before deleting them. We update each unmatched track with its last known velocity estimation. See supplement for the detailed tracking algorithm diagram.

CenterPoint combines all heatmap and regression losses in one common objective and jointly optimizes them. It simplifies and improves previous anchor-based 3D detectors (see experiments). However, all properties of the object 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.

4.1. 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 [26, 30, 44, 46] 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. The training is supervised with 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 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 [44, 45].

4.2. Architecture

All first-stage outputs share a first 3×3 convolutional layer, Batch Normalization [25], and ReLU. Each output then uses its own branch of two 3×3 convolutions separated by a batch norm and ReLU. Our second-stage uses a shared two-layer MLP, with a batch norm, ReLU, and Dropout [22] with a drop rate of 0.3, followed by two branches of three fully-connected layers, one for confidence score and one for box regression prediction.

5. Experiments

We evaluate CenterPoint on Waymo Open Dataset and nuScenes dataset. We implement CenterPoint using two 3D encoders: VoxelNet [56, 66, 67] and PointPillars [28], termed CenterPoint-Voxel and CenterPoint-Pillar respectively.

Waymo Open Dataset. Waymo Open Dataset [48] contains 798 training sequences and 202 validation sequences for vehicle and pedestrian. The point-clouds are captured with a 64 lanes Lidar, which produces about 180k Lidar points every 0.1s. The official 3D detection evaluation metrics include the standard 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. For 3D tracking, the official metrics are Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTP) [3]. The official evaluation toolkit also provides a performance breakdown for two difficulty levels: LEVEL_1 for boxes with more than five Lidar points, and LEVEL_2 for boxes with at least one Lidar point.

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. CenterPoint-Voxel uses a $(0.1\text{m}, 0.1\text{m}, 0.15\text{m})$ voxel size following PV-RCNN [44] while CenterPoint-Pillar uses a grid size of $(0.32\text{m}, 0.32\text{m})$.

nuScenes Dataset. nuScenes [6] contains 1000 driving sequences, with 700, 150, 150 sequences for training, validation, and testing, respectively. Each sequence is approximately 20-second long, with a Lidar frequency of 20 FPS. The dataset provides calibrated vehicle pose information for each Lidar frame but only provides box annotations every ten frames (0.5s). nuScenes uses a 32 lanes Lidar, which produces approximately 30k points per frame. In total, there are 28k, 6k, 6k, annotated frames for training, validation, and testing, respectively. The annotations include 10 classes with a long-tail distribution. The official evaluation metrics are an average among the classes. For 3D detection, the main metrics are mean Average Precision (mAP) [13] and nuScenes detection score (NDS). The mAP uses a bird-eye-view center distance $< 0.5\text{m}, 1\text{m}, 2\text{m}, 4\text{m}$ instead of standard box-overlap. NDS is a weighted average of mAP and other attributes metrics, including translation, scale, orientation, velocity, and other box attributes [6]. After our test set submission, the nuScenes team adds a new neural planning metric (PKL) [37]. The PKL metric measures the influence of 3D object detection for down-streamed autonomous driving tasks based on the KL divergence of a planner's route (using 3D detection) and the ground truth trajectory. Thus, we also report the PKL metric for all methods that evaluate on the test set.

For 3D tracking, nuScenes uses AMOTA [53], which penalizes ID switches, false positive, and false negatives and is averaged among various recall thresholds.

For experiments on nuScenes, we set the detection range to $[-51.2\text{m}, 51.2\text{m}]$ for the X and Y axis, and $[-5\text{m}, 3\text{m}]$ for Z axis. CenterPoint-Voxel use a $(0.1\text{m}, 0.1\text{m}, 0.2\text{m})$ voxel size and CenterPoint-Pillars uses a $(0.2\text{m}, 0.2\text{m})$ grid.

Training and Inference. We use the same network designs and training schedules as prior works [44, 67]. See supplement for detailed hyper-parameters. During the training of two-stage CenterPoint, we randomly sample 128 boxes with 1:1 positive negative ratio [43] from the first stage predictions. A proposal is positive if it overlaps with a ground truth annotation with at least 0.55 IoU [44]. During inference, we run the second stage on the top 500 predictions after Non-Maxima Suppression (NMS). The inference times are measured on an Intel Core i7 CPU and a Titan RTX GPU.

5.1. Main Results

3D Detection We first present our 3D detection results on the test sets of Waymo and nuScenes. Both results use a single CenterPoint-Voxel model. Table 1 and Table 2 summarize our results. On Waymo test set, our model achieves 71.8 level 2 mAPH for vehicle detection and 66.4 level 2 mAPH for pedestrian detection, surpassing previous methods by 7.1% mAPH for vehicles and 10.6% mAPH for pedestrians. On nuScenes (Table 2), our model outperforms the last-year challenge winner CBGS [67] with multi-scale inputs and multi-model ensemble by 5.2% mAP and 2.2% NDS. Our

Difficulty	Method	Vehicle		Pedestrian	
		mAP	mAPH	mAP	mAPH
Level 1	StarNet [36]	61.5	61.0	67.8	59.9
	PointPillars [28]	63.3	62.8	62.1	50.2
	PPBA [36]	67.5	67.0	69.7	61.7
	RCD [5]	72.0	71.6	-	-
	Ours	**80.2**	**79.7**	**78.3**	**72.1**
Level 2	StarNet [36]	54.9	54.5	61.1	54.0
	PointPillars [28]	55.6	55.1	55.9	45.1
	PPBA [36]	59.6	59.1	63.0	55.8
	RCD [5]	65.1	64.7	-	-
	Ours	**72.2**	**71.8**	**72.2**	**66.4**

Table 1: State-of-the-art comparisons for 3D detection on Waymo test set. We show the mAP and mAPH for both level 1 and level 2 benchmarks.

Method	mAP↑	NDS↑	PKL↓
WYSIWYG [23]	35.0	41.9	1.14
PointPillars [28]	40.1	55.0	1.00
CVCNet [7]	55.3	64.4	0.92
PointPainting [49]	46.4	58.1	0.89
PMPNet [62]	45.4	53.1	0.81
SSN [68]	46.3	56.9	0.77
CBGS [67]	52.8	63.3	0.77
Ours	**58.0**	**65.5**	**0.69**

Table 2: State-of-the-art comparisons for 3D detection on nuScenes test set. We show the nuScenes detection score (NDS), and mean Average Precision (mAP).

Difficulty	Method	MOTA↑		MOTP↓	
		Vehicle	Ped.	Vechile	Ped.
Level 1	AB3D [48, 53]	42.5	38.9	18.6	34.0
	Ours	**62.6**	**58.3**	**16.3**	**31.1**
Level 2	AB3D [48, 53]	40.1	37.7	18.6	34.0
	Ours	**59.4**	**56.6**	**16.4**	**31.2**

Table 3: State-of-the-art comparisons for 3D tracking on Waymo test set. We show MOTA, and MOTP. ↑ is for higher better and ↓ is for lower better.

model is also much faster, as shown later. A breakdown along classes is contained in the supplementary material. Our model displays a consistent performance improvement over all categories and shows more significant improvements in small categories (+5.6 mAP for traffic cone) and extreme-aspect ratio categories (+6.4 mAP for bicycle and +7.0 mAP for construction vehicle). More importantly, our model significantly outperforms all other submissions under the neural planar metric (PKL), a hidden metric evaluated by the organizers after our leaderboard submission. This highlights the generalization ability of our framework.

Method	AMOTA↑	FP↓	FN↓	IDS↓
AB3D [53]	15.1	**15088**	75730	9027
Chiu et al. [10]	55.0	17533	33216	950
Ours	**63.8**	18612	**22928**	**760**

Table 4: State-of-the-art comparisons for 3D tracking on nuScenes test set. We show AMOTA, the number of false positives (FP), false negatives (FN), id switches (IDS), and per-category AMOTA. ↑ is for higher better and ↓ is for lower better.

Encoder	Method	Vehicle	Pedestrain	mAPH
VoxelNet	Anchor-based	66.1	54.4	60.3
	Center-based	**66.5**	**62.7**	**64.6**
PointPillars	Anchor-based	64.1	50.8	57.5
	Center-based	**66.5**	**57.4**	**62.0**

Table 5: Comparison between anchor-based and center-based methods for 3D detection on Waymo validation. We show the per-calss and average LEVEL_2 mAPH.

Encoder	Method	mAP	NDS
VoxelNet	Anchor-based	52.6	63.0
	Center-based	**56.4**	**64.8**
PointPillars	Anchor-based	46.2	59.1
	Center-based	**50.3**	**60.2**

Table 6: Comparison between anchor-based and center-based methods for 3D detection on nuScenes validation. We show mean average precision (mAP) and nuScenes detection score (NDS).

3D Tracking Table 3 shows CenterPoint's tracking performance on the Waymo test set. Our velocity-based closest distance matching described in Section 4 significantly outperforms the official tracking baseline in the Waymo paper [48], which uses a Kalman-filter based tracker [53]. We observe a 19.4 and 18.9 MOTA improvement for vehicle and pedestrian tracking, respectively. On nuScenes (Table 4), our framework outperforms the last challenge winner Chiu et al. [10] by 8.8 AMOTA. Notably, our tracking does not require a separate motion model and runs in a negligible time, $1ms$ on top of detection.

5.2. Ablation studies

Center-based vs Anchor-based We first compare our center-based one-stage detector with its anchor-based counterparts [28, 56, 67]. On Waymo, we follow the state-of-the-art PV-RCNN [44] to set the anchor hyper-parameters: we use two anchors per-locations with $0°$and $90°$; The positive/ negative IoU thresholds are set as $0.55/0.4$ for vehicles and $0.5/0.35$ for pedestrians. On nuScenes, we follow the

	Vehicle			Pedestrian		
Rel. yaw	0°-15°	15°-30°	30°-45°	0°-15°	15°-30°	30°-45°
# annot.	81.4%	10.5%	8.1%	71.4%	15.8%	12.8%
Anchor-based	67.1	**47.7**	45.4	55.9	32.0	26.5
Center-based	**67.8**	46.4	**51.6**	**64.0**	**42.1**	**35.7**

Table 7: Comparison between anchor-based and center based methods for detecting objects of different heading angles. The ranges of the rotation angle and their corresponding portion of objects are listed in line 2 and line 3. We show the LEVEL_2 mAPH for both methods on the Waymo validation.

Method	Vehicle			Pedestrian		
	small	medium	large	small	medium	large
Anchor-based	58.5	**72.8**	64.4	29.6	60.2	60.1
Center-based	**59.0**	72.4	**65.4**	**38.5**	**69.5**	**69.0**

Table 8: Effects of object size for the performance of anchor-based and center-based methods. We show the per-class LEVEL_2 mAPH for objects in different size range: small 33%, middle 33%, and large 33%

anchor assignment strategy from the last challenge winner CBGS [67]. All other parameters are the same as our CenterPoint model.

As is shown in Table 5, on Waymo dataset, simply switching from anchors to our centers gives 4.3 mAPH and 4.5 mAPH improvements for VoxelNet and PointPillars encoder, respectively. On nuScenes (Table 6) CenterPoint improves anchor-based counterparts by 3.8-4.1 mAP and 1.1-1.8 NDS across different backbones. To understand where the improvements are from, we further show the performance breakdown on different subsets based on object sizes and orientation angles on the Waymo validation set.

We first divide the ground truth instances into three bins based on their heading angles: 0°to 15°, 15°to 30°, and 30°to 45°. This division tests the detector's performance for detecting heavily rotated boxes, which is critical for the safe deployment of autonomous driving. We also divide the dataset into three splits: small, medium, and large, and each split contains $\frac{1}{3}$ of the overall ground truth boxes.

Table 7 and Table 8 summarize the results. Our center-based detectors perform much better than the anchor-based baseline when the box is rotated or deviates from the average box size, demonstrating the model's ability to capture the rotation and size invariance when detecting objects. These results convincingly highlight the advantage of using a point-based representation of 3D objects.

One-stage vs. Two-stage In Table 9, we show the comparison between single and two-stage CenterPoint models using 2D CNN features on Waymo validation. Two-stage refinement with multiple center features gives a large accuracy boost to both 3D encoders with small overheads (6ms-7ms). We also compare with RoIAlign, which densely samples 6×6 points in the RoI [44, 46], our center-based feature aggregation achieved comparable performance but is faster and simpler.

Encoder	Method	Vehicle	Ped.	$T_{proposal}$	T_{refine}
VoxelNet	First Stage	66.5	62.7	71ms	-
	+ Box Center	68.0	64.9	71ms	5ms
	+ Surface Center	**68.3**	65.3	71ms	6ms
	Dense Sampling	68.2	**65.4**	71ms	8ms
PointPillars	First Stage	66.5	57.4	56ms	-
	+ Box Center	67.3	57.4	56ms	6ms
	+ Surface Center	**67.5**	57.9	56ms	7ms
	Dense Sampling	67.3	**57.9**	56ms	8ms

Table 9: Compare 3D LEVEL_2 mAPH for VoxelNet and PointPillars encoders using single stage, two stage with 3D center features, and two stage with 3D center and surface center features on Waymo validation.

Methods	Vehicle	Pedestrian	Runtime
BEV Feature	68.3	65.3	77ms
w/ VSA [44]	68.3	65.2	98ms
w/ RBF Interpolation [20, 41]	**68.4**	**65.7**	89ms

Table 10: Ablation studies of different feature components for two stage refinement module. VSA stands for Voxel Set Abstraction, the feature aggregation methods used in PV-RCNN [44]. RBF uses a radial basis function to interpolate 3 nearest neighbors. We compare bird-eye view and 3D voxel features using LEVEL_2 mAPH on Waymo validation.

The voxel quantization limits two-stage CenterPoint's improvements for pedestrian detection with PointPillars as pedestrians typically only reside in 1 pixel in the model input.

Two-stage refinement does not bring an improvement over the single-stage CenterPoint model on nuScenes in our experiments. We think the reason is that the nuScenes dataset uses 32 lanes Lidar, which produces about 30k Lidar points per frame, about $\frac{1}{6}$ of the number of points in the Waymo dataset, which limits the potential improvements of two-stage refinement. Similar results have been observed in previous two-stage methods like PointRCNN [45] and PV-RCNN [44].

Effects of different feature components In our two-stage CenterPoint model, we only use features from the 2D CNN feature map. However, previous methods propose to also utilize voxel features for second stage refinement [44, 46]. Here, we compare with two voxel feature extraction baselines:

Voxel-Set Abstraction. PV-RCNN [44] proposes the Voxel-Set Abstraction (VSA) module, which extends Point-

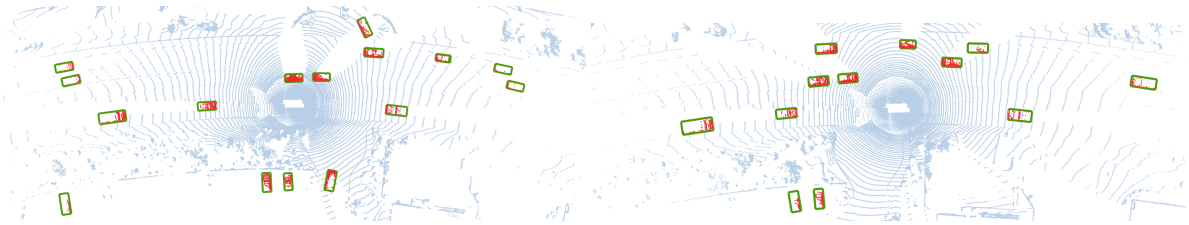


Figure 3: Example qualitative results of CenterPoint on the Waymo validation. We show the raw point-cloud in blue, our detected objects in green bounding boxes, and Lidar points inside bounding boxes in red. Best viewed on screen.

Difficulty	Method	Vehicle		Pedestrian	
		mAP	mAPH	mAP	mAPH
Level 1	DOPS [35]	56.4	_	_	
	PointPillars [28]	56.6	_	59.3	_
	PPBA [36]	62.4	_	66.0	_
	MVF [65]	62.9	_	65.3	_
	Huang et al. [24]	63.6	_	_	
	AFDet [14]	63.7	_	_	
	CVCNet [7]	65.2	_	_	
	Pillar-OD [52]	69.8	_	72.5	_
	PV-RCNN [44]	74.4	73.8	61.4	53.4
	CenterPoint-Pillar(ours)	76.1	75.5	76.1	65.1
	CenterPoint-Voxel(ours)	**76.7**	**76.2**	**79.0**	**72.9**
Level 2	PV-RCNN [44]	65.4	64.8	53.9	46.7
	CenterPoint-Pillar(ours)	68.0	67.5	68.1	57.9
	CenterPoint-Voxel(ours)	**68.8**	**68.3**	**71.0**	**65.3**

Table 11: State-of-the-art comparisons for 3D detection on Waymo validation.

Detector	Tracker	AMOTA↑	AMOTP↓	T_{track}	T_{tot}
CenterPoint-Voxel	Point	**63.7**	**0.606**	1ms	62ms
CBGS [67]	Point	59.8	0.682	1ms	$> 182ms$
CenterPoint-Voxel	M-KF	60.0	0.765	73ms	135ms
CBGS [67]	M-KF	56.1	0.800	73ms	>254ms

Table 12: Ablation studies for 3D tracking on nuScenes validation. We show combinations of different detectors and trackers. CenterPoint-* are our detectors. Point is our proposed tracker. M-KF is short for Mahalanobis distance-based Kalman filter, as is used in the last challenge winner Chiu et al. [10]. T_{track} denotes tracking time and T_{tot} denotes total time for both detection and tracking.

Net++ [41]'s set abstraction layer to aggregate voxel features in a fixed radius ball.

Radial basis function (RBF) Interpolation. PointNet++ [41] and SA-SSD [20] use a radial basis function to aggregate grid point features from three nearest non-empty 3D feature volumes.

For both baselines, we combine bird-eye view features with voxel features using their official implementations. Table 10 summarizes the results. It shows bird-eye view features are sufficient for good performance while being more efficient comparing to voxel features used in the literatures [20, 41, 44].

To compare with prior work that did not evaluate on Waymo test, we also report results on the Waymo validation split in Table 11. Our model outperforms all published methods by a large margin, especially for the challenging pedestrian class(+18.6 mAPH) of the level 2 dataset, where boxes contain as little as one Lidar point.

3D Tracking. Table 12 shows the ablation experiments of 3D tracking on nuScenes validation. We compare with last year's challenge winner Chiu et al. [10], which uses mahalanobis distance-based Kalman filter to associate detection results of CBGS [67]. We decompose the evaluation into the detector and tracker to make the comparison strict. Given the same detected objects, using our simple velocity-based closest point distance matching outperforms the Kalman filter-based Mahalanobis distance matching [10] by 3.7 AMOTA (line 1 vs. line 3 and line 2 vs. line4). There are two sources of improvements: 1) we model the object motion with a learned point velocity, rather than modeling 3D bounding box dynamic with a Kalman filter; 2) we match objects by center point-distance instead of a Mahalanobis distance of box states or 3D bounding box IoU. More importantly, our tracking is a simple nearest neighbor matching without any hidden-state computation. This saves the computational overhead of a 3D Kalman filter [10] (73ms vs. 1ms).

6. Conclusion

We proposed a center-based framework for simultaneous 3D object detection and tracking from the Lidar point-clouds. Our method uses a standard 3D point-cloud encoder with a few convolutional layers in the head to produce a bird-eye-view heatmap and other dense regression outputs. Detection is a simple local peak extraction with refinement, and tracking is a closest-distance matching. CenterPoint is simple, near real-time, and achieves state-of-the-art performance on the Waymo and nuScenes benchmarks.

References

[1] Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *RSS*, 2019. 1

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. *ICCV*, 2019. 2

[3] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. Citeseer. 5

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *ICIP*, 2016. 2, 4

[5] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. *arXiv preprint arXiv:2005.09927*, 2020. 6

[6] 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, 5, 11

[7] 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. 6, 8, 12

[8] Qi Chen, Lin Sun, Zhixin Wang, Kui Jia, and Alan Yuille. Object as hotspots: An anchor-free 3d object detection approach via firing of hotspots. *ECCV*, 2020. 2

[9] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. *ICCV*, 2019. 2

[10] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *arXiv:2001.05673*, 2020. 2, 3, 6, 8

[11] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. *ICCV*, 2017. 11

[12] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. *ICRA*, 2017. 2

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 5

[14] Runzhou Ge, Zhuangzhuang Ding, Yihan Hu, Yu Wang, Sijia Chen, Li Huang, and Yuan Li. Afdet: Anchor free one stage 3d object detection. *arXiv preprint arXiv:2006.12671*, 2020. 8

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *CVPR*, 2012. 2

[16] Ross Girshick. Fast r-cnn. *ICCV*, 2015. 1, 2

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014. 1, 2

[18] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. *CVPR*, 2018. 2

[19] Sylvain Gugger. The 1cycle policy. https://sgugger.github.io/the-1cycle-policy.html, 2018. 11

[20] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. Structure aware single-stage 3d object detection from point cloud. *CVPR*, 2020. 2, 3, 7, 8

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. *ICCV*, 2017. 2

[22] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *JMLR*, 2012. 5

[23] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. *CVPR*, 2020. 1, 6, 12

[24] Rui Huang, Wanyue Zhang, Abhijit Kundu, Caroline Pantofaru, David A Ross, Thomas Funkhouser, and Alireza Fathi. An lstm approach to temporal 3d object detection in lidar point clouds. *ECCV*, 2020. 8

[25] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015. 5

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

[27] H. Karunasekera, H. Wang, and H. Zhang. Multiple object tracking with attention to appearance, structure, motion and size. *IEEE Access*, 2019. 2

[28] 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. 2, 3, 5, 6, 8, 11, 12

[29] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. *ECCV*, 2018. 3

[30] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. *CVPR*, 2019. 4

[31] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. *CVPR*, 2019. 2

[32] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *ICCV*, 2017. 2, 3

[33] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. *ECCV*, 2016. 2

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019. 11

[35] Mahyar Najibi, Guangda Lai, Abhijit Kundu, Zhichao Lu, Vivek Rathod, Thomas Funkhouser, Caroline Pantofaru, David Ross, Larry S Davis, and Alireza Fathi. Dops: Learning to detect 3d objects and predict their 3d shapes. *CVPR*, 2020. 8

[36] Jiquan Ngiam, Benjamin Caine, Wei Han, Brandon Yang, Yuning Chai, Pei Sun, Yin Zhou, Xi Yi, Ouais Alsharif, Patrick Nguyen, et al. Starnet: Targeted computation for object detection in point clouds. *arXiv preprint arXiv:1908.11069*, 2019. 6, 8

[37] Jonah Philion, Amlan Kar, and Sanja Fidler. Learning to evaluate perception models using planner-centric metrics. *CVPR*, 2020. 5

[38] Charles R. Qi, Or Litany, Kaiming He, and Leonidas Guibas. Deep hough voting for 3d object detection in point clouds. *ICCV*, 2019. 2

[39] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. *CVPR*, 2018. 2

[40] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CVPR*, 2017. 2, 3

[41] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 7, 8

[42] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *CVPR*, 2017. 2

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015. 2, 3, 5

[44] 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. 2, 3, 4, 5, 6, 7, 8, 11

[45] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. *CVPR*, 2019. 1, 2, 5, 7

[46] 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. 2, 4, 7

[47] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: An euler-region-proposal for real-time 3d object detection on point clouds. *ECCV*, 2018. 1, 2

[48] 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. 2, 5, 6

[49] Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. *CVPR*, 2020. 6, 11, 12

[50] Dequan Wang, Coline Devin, Qi-Zhi Cai, Philipp Krähenbühl, and Trevor Darrell. Monocular plan view networks for autonomous driving. *IROS*, 2019. 1

[51] Dominic Zeng Wang and Ingmar Posner. Voting for voting in online point cloud object detection. *RSS*, 2015. 2

[52] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. *ECCV*, 2020. 2, 8

[53] Xinshuo Weng and Kris Kitani. A Baseline for 3D Multi-Object Tracking. *IROS*, 2020. 2, 3, 5, 6

[54] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *ICIP*, 2017. 2

[55] Kelvin Wong, Shenlong Wang, Mengye Ren, Ming Liang, and Raquel Urtasun. Identifying unknown instances for autonomous driving. *CORL*, 2019. 2

[56] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 2018. 2, 3, 5, 6, 11

[57] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. *CVPR*, 2018. 2

[58] Xue Yang, Qingqing Liu, Junchi Yan, Ang Li, Zhiqiang Zhang, and Gang Yu. R3det: Refined single-stage detector with feature refinement for rotating object. *arXiv:1908.05612*, 2019. 1

[59] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Xian Sun, and Kun Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects. *ICCV*, 2019. 1

[60] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. *CVPR*, 2020. 1, 2, 11

[61] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. *ICCV*, 2019. 2

[62] 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. 6, 12

[63] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *ECCV*, 2020. 2, 3

[64] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv:1904.07850*, 2019. 2, 3

[65] 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. *CORL*, 2019. 2, 8

[66] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. *CVPR*, 2018. 2, 3, 5, 11

[67] 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. 2, 5, 6, 7, 8, 11, 12

[68] 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. 6, 12

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 [67][1]. CBGS provides implementations of PointPillars [28] and VoxelNet [66] on nuScenes. For Waymo experiments, we use the same architecture for VoxelNet and increases the output stride to 1 for PointPillars [28] following the dataset's reference implementation[2].

A common practice [6, 49, 60, 67] 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 [67] and $[-\pi/4, \pi/4]$ for Waymo [44]. We also use the ground-truth sampling [56] 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 [67] to optimize the model using AdamW [34] optimizer with one-cycle learning rate policy [19], 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 [44]. 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 [11] 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 13.

D. nuScenes Detection Challenge

As a general framework, CenterPoint is complementary to contemporary methods and was used by three of the top 4

[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 [23]	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 [28]	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 [49]	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 [7]	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 [62]	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 [68]	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 [67]	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 13: 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 [49]	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 14: Ablation studies for 3D detection on nuScenes validation.

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 [67] by 14.3 mAP and 8.1 NDS. We report some improved results in Table 14. We use PointPainting [49] 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.

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