Supplementary Materials for: Once Detected, Never Lost: Surpassing Human Performance in Offline LiDAR based 3D Object Detection

1. Overview

We first present the outline of this supplementary material. §2 contains the detailed network and input designs. The human labeling protocol is presented in §3. The details of bidirectional tracking are in §4. §5 offers detailed detection performance on the WOD test split and detailed tracking performance. §6 elaborates our training and inference scheme. We conduct runtime evaluation in §7, where we compare CTRL with previous art 3DAL. §8 shows the statistics of life cycles of totally missed objects. §9 and §10 showcase some typical examples of incorrect official annotations and the sampled hard cases for relabeling. We present the detailed algorithm of Track Coherence Optimization in §11.

2. Detailed Network Structure

Input. After tracking, for every track, we first use its containing proposals to crop points. Each proposal is expanded by 2 meters in three size dimensions (1 meter on each side). The expanded proposals are used to crop raw points in the corresponding frame. In case of the out-of-memory issue caused by too many points of large objects, we randomly downsample all points in a proposal to 1024 points. Instead, the track length is not limited. The downsampled points are transformed into the pose of the first frame of the track and concatenated together. We just place the points in their transformed positions without any further movement (e.g., centralizing) or editing. We treat the concatenated points in a track as a sample in a mini-batch.

Track Feature Extraction. We use the Sparse UNet in FSD [5] as the backbone to extract the track features (i.e., point-wise features extracted from a full track). The network hyperparameter is the same as the one in FSD, which can be accessed through their official website, we also provide our detailed configuration in the attached file with MMDetection3D format.

Object Feature Extraction. As for the object feature extraction, we first use the proposal to crop points *from all the concatenated track points*, and further extract the features of cropped points. In particular, we adopt PointNet implementation in FSD to extract point features, which is more efficient than the original implementation. We use 6 consecutive PointNets to extract the point features, and eventually, the point features of each object are aggregated by a max pooling operator. We also provide the configuration in our attached code with detailed comments. Since the cropped points are from different time steps, we append a binary flag to the cropped points to distinguish whether a certain point comes from the current frame. "Current frame" here refers to the frame to which the proposal belongs.

3. Human Labeling Protocol

In the main paper, we conduct a human labeling study. Here we present the detailed human labeling protocol.

Data selection. We randomly select 1000 "hard vehicles" to relabel. To this end, we first calculate the maximum IoU between each GT and all predictions in the same frame. Then we randomly sample 1000 GTs with 3D IoU lower than 0.7 and higher than 0.1 as "hard vehicles". For each selected GT, we give all the point clouds throughout its life cycle to annotators.

Labeling. We ask experienced annotators to do the job with professional labeling tools. They carefully label the object poses in three perspective views. Given the multi-frame point clouds, they could play the clip or concatenate them to utilize the temporal information. Although we provide point cloud sequences to annotators, they only need to label the frame containing the aforementioned "hard vehicles". Thus 1000 GTs could sufficiently cover hard cases with high diversity.

4. Details of Bidirectional Tracking

Forward Tracking. We adopt the official codebase of ImmortalTracker to implement the forward tracking. We make some small modifications to its default setting. (1) We do not use the default score threshold to remove low-confidence predictions because we find it harmful to detection performance. (2) We do not use the additional NMS since our base detector has adopted a strict NMS (IoU = 0.2). Other key settings remain unchanged. For example, following ImmortalTracker, we adopt 3D IoU to measure the object matching cost and a bipartite matching algorithm for object association. During the forward tracking, we will fill in the missing predictions by a motion model which is maintained by a Kalman Filter. All tracks longer than 100 frames are extended into the end of the sequence by the motion model. Other tracks are extended 20 frames longer into the future.

Backtracing. After forward tracking, we backtrace each track from its last frame to its first frame to estimate the motion states. Then we backward extend the track into the past using the motion states. Similar to the forward process, all tracks longer than 100 frames are extended into the beginning of the sequence by the motion model. Other shorter tracks are extended 20 frames longer into the past.

5. Detailed Performance

Tracking results. Table 1 and Table 2 show the tracking performance of CTRL in Waymo Open Dataset validation and test split, respectively.

Detection performance on test split. Table 3 showcases our detailed detection performance in the WOD test split.

6. Training and Inference Scheme

Training data. For the track-centric learning module, we first generate all tracks in an offline manner. In the whole training set, we obtain 349k vehicle tracks, 293k pedestrian tracks, and 40k cyclist tracks in total. Since the cyclist tracks are much less than the other two categories, **we repeat the cyclist tracks by** $10\times$.

Data augmentations. We regard an input track as a scene of the traditional detectors, so we directly adopt the default data augmentations in these detectors, including global rotation/flip/scaling/translation. In particular, an input track is randomly rotated by $[-0.78, +0.78]$, and flipped in two axes with probability 0.5, and randomly scaled by a factor in $[0.95, 1.05]$, and randomly translated in the vertical direction by at most 0.2 meters. In addition, we add random jittering to each input proposal. Their centers are randomly translated by $[0.2l, 0.2w, 0.1h]$, where l, w, h stand for length, width, and height, respectively. Their widths and lengths are randomly scaled by a factor in $[0.8, 1.2]$. The heights are randomly scaled by a factor in $[0.9, 1.1]$. And their headings are disturbed with a maximum noise of 0.2 rad.

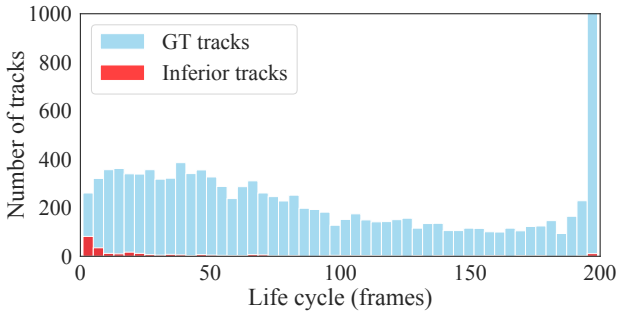


Figure 1: The life cycle histogram of normal GT tracks and inferior tracks. The frame frequency is 10Hz.

Schedule and optimization. We train the model for 24 epochs with a one-cycle schedule. AdamW [7] is adopted as the optimizer and the maximum learning rate is 1e-3. The model is trained on 8 RTX 3090 GPUs, with 16 samples (tracks) on each of them. The whole training takes around 20 hours.

Inference. We adopt so-called batch inference for efficiency. 32 tracks are simultaneously refined in a single forward pass, which offers us high efficiency as we demonstrate in §7.

Track TTA. In the main paper, we adopt conventional double-flip augmentation for the Track TTA. Specifically, a whole input track is first horizontally (x-axis) flipped, resulting in two tracks. The two tracks are then vertically (y-axis) flipped, resulting in four tracks in total. In addition to the double-flip augmentation, we also try different rotations for TTA. In particular, we adopt $[-2\pi/3, 0, +2\pi/3]$ as the rotation angles, leading to a similar performance to the double-flip strategy. Combining double-flip augmentation and rotation further leads to around 0.1 mAP improvement. Note that we do not adopt TTA for the base detector.

7. Runtime Evaluation

Although CTRL is for offline use, resource efficiency is also crucial in production. Thus we perform a simple runtime evaluation on the previous offline method and ours, shown in Table 4. CTRL is around $20\times$ faster than 3DAL since CTRL utilizes efficient FSD as the base detector and does not adopt TTA in the base detector.

	Vehicle			*Pedestrian*			*Cyclist*		
	MOTA↑	MOTP↓	IDS(%)↓	MOTA↑	MOTP↓	IDS(%)↓	MOTA↑	MOTP↓	IDS(%)↓
AB3DMOT* [11]	55.7	16.8	0.40	52.2	31.0	2.74	–	–	–
CenterPoint* [13]	55.1	16.9	0.26	54.9	31.4	1.13	–	–	–
SimpleTrack* [9]	56.1	16.8	0.08	57.8	31.3	0.42	–	–	–
CenterPoint++ [13]	56.1	-	0.25	57.4	–	0.94	–	–	–
Immortal Tracker [10]	56.4	-	0.01	58.2	–	0.26	–	–	–
CTRL (Ours)	71.2	15.1	0.0078	70.3	29.3	0.073	72.5	24.8	0.11

Table 1: Tracking results on WOD validation split (L2). *:from [9].

	Vehicle			*Pedestrian*			*Cyclist*		
	MOTA↑	MOTP↓	IDS(%)↓	MOTA↑	MOTP↓	IDS(%)↓	MOTA↑	MOTP↓	IDS(%)↓
InceptioLidar	65.58	15.70	0.14	64.52	29.54	0.20	65.12	25.42	0.52
HorizonMOT3D	64.07	15.77	0.19	64.15	30.67	0.50	62.13	25.45	0.18
MFMS_Track	63.14	15.65	0.07	63.85	30.19	0.28	62.83	25.44	0.69
CasTrack	63.66	15.79	0.05	64.79	30.24	0.24	59.34	25.30	0.09
ImmortalTracker [10]	60.55	16.22	0.01	60.60	31.20	0.18	61.61	27.41	0.10
CTRL (Ours)	74.29	14.26	0.02	74.21	28.95	0.05	71.37	25.18	0.07

Table 2: Comparison with state-of-the-art online tracker on WOD test leaderboard (L2).

	Detection	Tracking	Refine	Total
3DAL	900s	3s	25s	928s
CTRL	36s	4s	6s	46s

Table 4: Time cost of processing a 20s sequence.

8. Short-life Failure Cases

In the main paper, we have discussed the totally missed objects by CTRL. One of their characteristics is that they are likely to belong to short-life tracks. We briefly present the statistics in the main paper. Here we depict the life cycle histogram of inferior tracks. The **inferior tracks** refer to the GT tracks containing more than 10% missed objects. Figure 1 shows the life cycle distribution of all GT tracks and the inferior tracks. As can be seen, the major part of inferior tracks has very short life cycles, which are usually shorter than 2.5 seconds.

9. Examples of Incorrect Annotations

In the §5.4 of the main paper, we calculate BEV IoU instead of the 3D IoU because we find there are some incorrect annotations in the official WOD ground-truth boxes. Figure 2 shows some typical cases.

10. Examples of Hard Cases

In the §5.4 of the main paper, we randomly sample some hard cases to relabel. Figure 4 qualitatively shows these cases.

11. Details of Track Coherence Optimization (TCO)

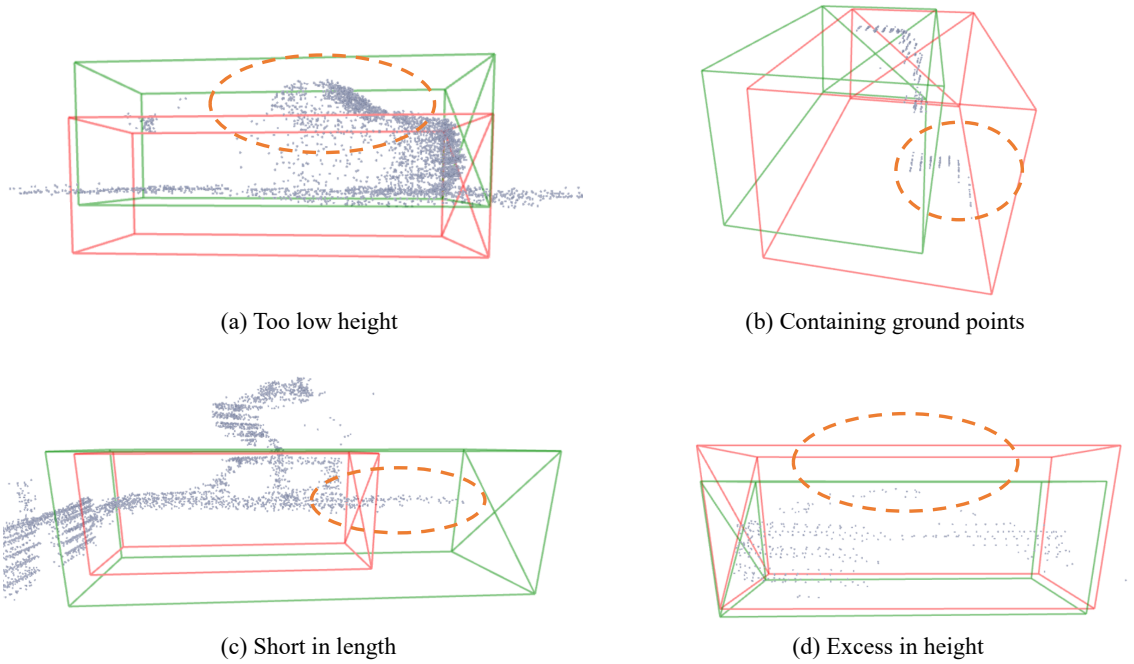
Overall pipeline. The pipeline of TCO comprises four steps: box size alignment, object shape extraction, multi-way registration, and pose quality evaluation.

Box size alignment. The first step of TCO is to specify a frame in a track as *base frame*. For quasi-rigid objects, we assume their 3D sizes keep unchanged throughout the whole track. So we first align all sizes in the track to the box size in the base frame.

Object shape extraction. Afterwards, we need to extract object shapes from the complete scenes for further registration. To this end, we first expand all proposals by 1 meter only in the height dimension, which potentially keeps more foreground points without increasing too much background clutter. Then we use the expanded boxes to crop points in their corresponding time steps. We use the term "object shape" to denote the cropped point clouds. In a track, only frames containing more than 60 cropped points will be used for further processing. These cropped points (shapes) are then transformed into their canonical box coordinate.

Methods	mAPH L2	*Vehicle* 3D AP/APH L1	*Vehicle* 3D AP/APH L2	*Pedestrian* 3D AP/APH L1	*Pedestrian* 3D AP/APH L2	*Cyclist* 3D AP/APH L1	*Cyclist* 3D AP/APH L2
CTRL (Ours)	82.52	90.08/89.17	84.41/83.51	90.17/87.13	85.64/82.61	84.06/83.23	82.27/81.44
HRI_ADLAB_HZ	81.32	86.77/86.40	80.19/79.83	88.59/86.01	83.84/81.27	85.67/84.84	83.69/82.87
LoGoNet_Ens	81.02	88.33/87.87	82.17/81.72	88.98/85.96	84.27/81.28	83.10/82.16	80.98/80.06
MT-Net v2 [2]	80.00	87.54/87.12	81.20/80.79	87.62/84.89	82.33/79.66	82.80/81.74	80.58/79.54
BEVFusion-TTA [6]	79.97	87.96/87.58	81.29/80.92	87.64/85.04	82.19/79.65	82.53/81.67	80.17/79.33
LidarMultiNet-TTA [12]	79.94	87.64/87.26	80.73/80.36	87.75/85.07	82.48/79.86	82.77/81.84	80.50/79.59
MPPNetEns [3]	79.60	87.77/87.37	81.33/80.93	87.92/85.15	82.86/80.14	80.74/79.90	78.54/77.73

Table 3: Top-performing detectors in the leaderboard of Waymo Open Dataset. The results are up to March 8th.



(a) Too low height

(b) Containing ground points

(c) Short in length

(d) Excess in height

Figure 2: **Typical examples of incorrect official Waymo annotations.** The red boxes are ground-truth boxes and the green boxes are predictions. We use dashed circles to indicate the incorrect parts. For those incorrectly annotated objects, our predictions are actually more reasonable. For reference, the four examples are from scenes with timestamps 1507130389559851, 1507165348439487, 1507217534286345, 1507310198254864, respectively.

Multi-way registration. For the next shape registration part, we adopt a multi-way registration [4] with pose graph optimization. To reduce overhead, we design a sparse pose graph instead of the conventional dense pose graph. The sparse pose graph has two key elements: nodes and sparse edges. A node is the point cloud (i.e., shape) P_i associated with a transformation matrix M_i which transforms P_i into the base object shape P_{base}. For each frame, we only connect it with the previous k frames and the succeeding k frames to construct edges, resulting in $2k$ edges. So the constructed pose graph is sparse. In practice, k larger than 5 could achieve good performance, and we let $k = 10$ in our experiments. For each edge, we have a transformation matrix $T_{i,j}$ aligning shape P_i to shape P_j. We use point-to-point ICP [1] to estimate the all mentioned transformations. We optimize $\mathcal{M} = \{M_i\}$ via:

$$\min_{\mathcal{M}} \sum_{i,j} \sum_{(p,q)\in K_{ij}} \|M_i p - M_j q\|_2^2. \quad (1)$$

In Equation 1, K_{ij} is the set of point pairs between P_i and P_j, and their pair relations are obtained by matching points in $T_{i,j}P_i$ and points in P_j with a nearest-neighbor manner.

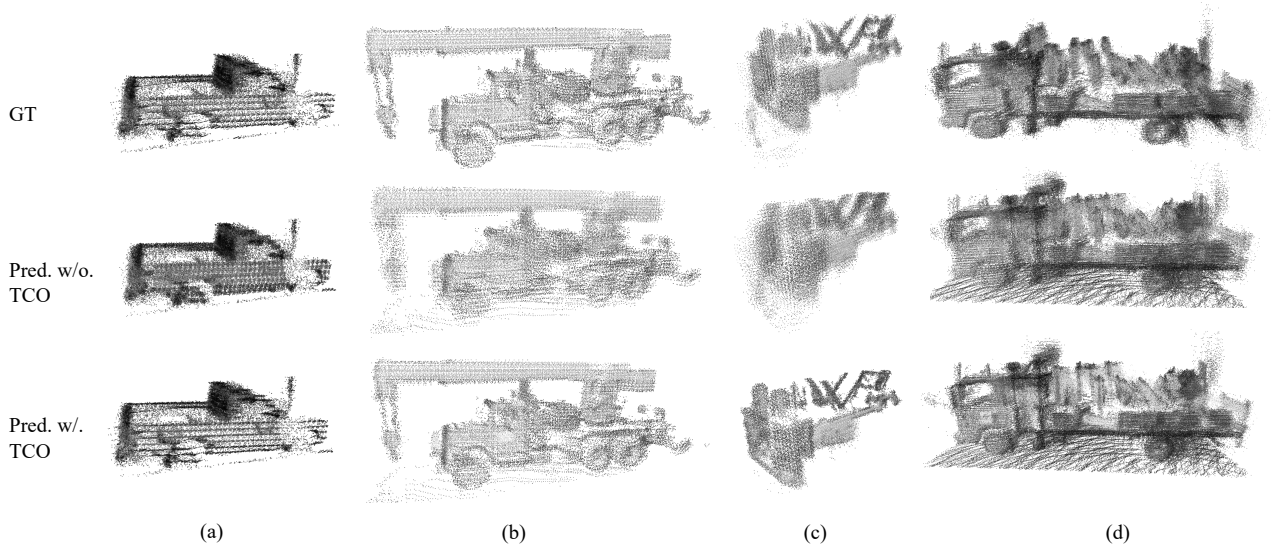


Figure 3: **Qualitative results of track coherence optimization (TCO).** We reconstruct object shapes by aligning the poses of boxes and concatenating object points from multiple frames. After TCO, the reconstructed shapes are much more clear, even better than the ones reconstructed by GT object poses.

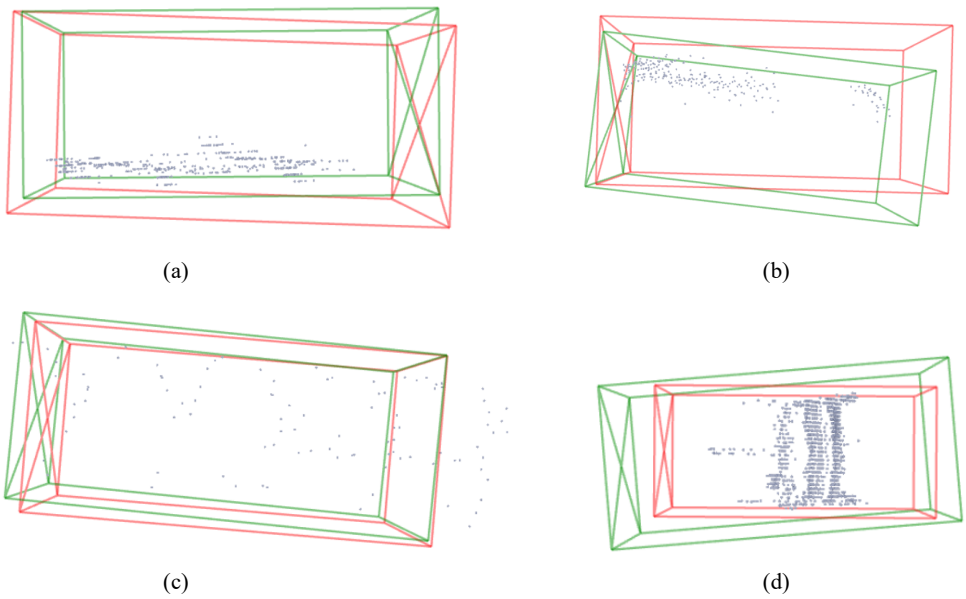


Figure 4: **Typical examples of our relabeled hard cases.** The red boxes are ground-truth boxes and green boxes are predictions. They usually contain a few points or have partial shapes.

The maximum correspondence distance of each pair is 2 meters.

Pose quality evaluation. To avoid the failure of ICP, we employ Chamfer Distance (CD) [8] to evaluate the quality of the optimized pose for each frame. For object shape P_i, we use CD to measure its distance to P_j:

$$CD_{ij} = \frac{1}{|P_i|}\sum_{x\in P_i}\min_{y\in P_j}\|x-y\|_2 + \frac{1}{|P_j|}\sum_{y\in P_j}\min_{x\in P_i}\|y-x\|_2. \tag{2}$$

For object shape P_i in a track, we define its pose quality

as[1]:

$$Q_i = \frac{CD_{i,i-1} + CD_{i,i+1}}{2}. \tag{3}$$

Then we define

$$\Delta Q_i = Q_i - Q'_i. \tag{4}$$

where Q'_i is pose quality after TCO. If ΔQ_i is positive, it means that the optimized pose becomes better. Thus, only optimized poses with a positive ΔQ_i are retained, and other poses are not utilized.

Then we assume the optimal solution of Equation 1 is $\mathcal{M}^* = \{M_i^*\}$. To get better bounding box parameters, we simply use the inverse transformation of M_i^* to adjust the initial pose of box B_i. We show the qualitative results of TCO in Figure 3.

[1]The Chamfer Distance here is calculated between two shapes aligned by their poses. We omit the transformation for simplicity

References

[1] Paul J Besl and Neil D McKay. Method for Registration of 3-D Shapes. In *Sensor fusion IV: control paradigms and data structures*. Spie, 1992. 4

[2] Shaoxiang Chen, Zequn Jie, Xiaolin Wei, and Lin Ma. MT-Net Submission to the Waymo 3D Detection Leaderboard. *arXiv preprint arXiv:2207.04781*, 2022. 4

[3] Xuesong Chen, Shaoshuai Shi, Benjin Zhu, Ka Chun Cheung, Hang Xu, and Hongsheng Li. MPPNet: Multi-Frame Feature Intertwining with Proxy Points for 3D Temporal Object Detection. In *ECCV*. Springer, 2022. 4

[4] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust Reconstruction of Indoor Scenes. In *CVPR*, 2015. 4

[5] Lue Fan, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Fully Sparse 3D Object Detection. In *NeurIPS*, 2022. 1

[6] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. *arXiv preprint arXiv:2205.13542*, 2022. 4

[7] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[8] Ziqi Pang, Zhichao Li, and Naiyan Wang. Model-free Vehicle Tracking and State Estimation in Point Cloud Sequences. In *IROS*. IEEE, 2021. 5

[9] Ziqi Pang, Zhichao Li, and Naiyan Wang. Simpletrack: Understanding and Rethinking 3d Multi-object Tracking. In *ECCVW*. Springer, 2023. 3

[10] Qitai Wang, Yuntao Chen, Ziqi Pang, Naiyan Wang, and Zhaoxiang Zhang. Immortal Tracker: Tracklet Never Dies. *arXiv preprint arXiv:2111.13672*, 2021. 3

[11] Xinshuo Weng and Kris Kitani. A Baseline for 3D Multi-object Tracking. *arXiv preprint arXiv:1907.03961*, 2019. 3

[12] Dongqiangzi Ye, Weijia Chen, Zixiang Zhou, Yufei Xie, Yu Wang, Panqu Wang, and Hassan Foroosh. LidarMutliNet: Unifying LiDAR Semantic Segmentation, 3D Object Detection, and Panoptic Segmentation in a Single Multi-task Network. *arXiv preprint arXiv:2206.11428*, 2022. 4

[13] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D Object Detection and Tracking. *arXiv preprint arXiv:2006.11275*, 2020. 3