Appendix

A. Overview

This document is the supplementary material of submission 4378. We provide more details of models, experiments and analysis results in this document. Sec. B introduces more details about our multi-frame 3D object detectors. Sec. C describes the implementation details of our offline tracking. In Sec. D, the network structures and the model training details are described together with figures and tables. Sec. F provides more experiment results and analyses. We also visualize the results after our DetZero in Sec. G.

In addition, we attach several videos to better show the effect of our DetZero. "DetZero_scene1_fv.mp4" is to display the final stable detection and tracking performance. "DetZero_scene2_bev.mp4" is to display the comprehensive detection results by showing true positive, false positive and missing boxes with different colors. "GRM+PRM_Vehicle.mp4" and "GRM+PRM_Pedestrian.mp4" are two examples to show how our refining models work. Please refer to them for better visualization.

B. Multi-frame 3D Object Detection

In this section, we provide more detailed explanations of the multi-frame 3D object detector. Please refer to Table 9 for the detailed ablation study of multi-frame detectors. Firstly, we take CenterPoint [56] as our base detector owing to producing dense detection results, which is beneficial for the downstream refine module.

Multi-frame Input. We accumulate LiDAR sweeps to utilize temporal information and to densify the LiDAR point cloud. The past 4 frames combined with the current frame serve as our input point cloud. To distinguish points from different sweeps, we also follow [9] to add a time offset as an additional attribute to the point cloud. Moreover, 3-frame input (past 2 + current 1) is also used to make up more detection models for boosting the performance in the subsequent model ensembling.

Two-stage Module. To obtain more accurate bounding boxes, we introduce the Point Density-Aware Voxel network (PDV) [13] as the two-stage module to refine the coarse proposals coming from the multi-frame base detector. This model can leverage the voxel point centroid localization and account for point density variations to enhance refining features.

Model Ensembling. Following [19, 14], we use different TTA settings to boost the inference performance: $[0°, \pm 22.5°, \pm 45°, \pm 135°, \pm 157.5°, 180°]$ for global rotation along z-axis, $[0.95, 1.05]$ for global scaling. Besides, different grid sizes of $[0.075, 0.075, 0.15]$m and $[0.1, 0.1, 0.15]$m are used to train both 5-frame and 3-frame input models. Finally, we adopt 3D version WBF [36] to fuse different model results combined with the above TTAs.

Training Details. We use Adam optimizer with one-cycle learning rate policy, with max learning rate 3×10^{-3}, weight decay 0.01 and momentum 0.85 to 0.95. We also adopt the common data augmentations including global rotation, global scaling, translation along z-axis and gt-sampling to train the base detector for 20 epochs. The total batch size is set as 64. The gt-sampling is removed for last 5 epochs training [40]. We train another 6 epochs for two-stage refinement without gt-sampling, while keeping the same batch size and learning rate as the first stage. Besides the general classification and regression loss functions, we also add the IoU loss function [57] to better account for the center-based object detection.

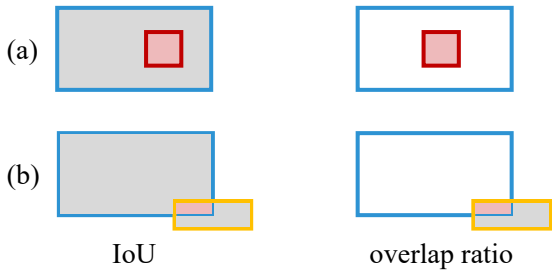

Figure 5. The comparison between traditional IoU based and overlap ratio based calculation. We show two examples here, and the blue box, red box, orange box represent vehicle, pedestrian and cyclist from BEV respectively. And the gray region represents the union of two boxes while the pink region represent the intersection for best view. For (a), the pedestrian FP is totally inside the vehicle, the IoU value between these two boxes is still small, while the overlap ratio equals 1 (the denominator is the same as the numerator). For (b), the gray regions are quite different for these two methods, so the overlap ratio metric could lead to small FPs filtering.

C. Implementation Details of Offline Tracking

Our multi-frame 3D detector is encouraged to generate sufficient bounding boxes. Hence, we utilize pre-processing operations to stable the association of our offline tracker. To be specific, we found that there are many boxes overlapped with each other. And some small boxes are even completely wrapped by other boxes, for example, the vehicle boxes contain pedestrian boxes. In this situation, the traditional IoU based calculation will be invalid, as shown in Fig. 5. Therefore, we adopt a new metric to determine whether a box should be kept or filtered out, which is called *overlap ratio*. For each box (subject), we first calculate pairwise itersection area with other boxes (object), which serve as the numerator. Then, we use the original area of object box as denominator to get the result, and the value range is $[0, 1]$. This overlap ratio can filter out the overlapped boxes

Base detector	Multi-frame	0.075 Voxel	Two-stage	TTA	*Vehicle* (L1 / L2)	*Pedestrian* (L1 / L2)
✓					74.51 / 66.44	70.56 / 63.57
✓	✓				78.61 / 71.07	78.78 / 71.46
✓	✓	✓			79.57 / 72.04	81.09 / 73.16
✓	✓		✓		81.02 / 73.15	80.32 / 72.39
✓	✓	✓	✓		81.17 / 73.29	81.14 / 74.00
✓	✓	✓	✓	✓	**82.57 / 75.09**	**83.23 / 76.47**

Table 9. Effect of each component in our multi-frame Detection module on WOD val set. Metrics are 3D APH of both L1 and L2 difficulties for *Vehicle* and *Pedestrian*.

of small objects as shown in Fig. 5. We also report the quantitative performance in Table 16. In our implementation, we use BEV overlap ratio and set the thresholds as 0.3 for Vehicle, 0.2 for Pedestrian and Cyclist.

In our two-stage data association, the high-score group contains boxes satisfying two options: (1) the confidence score is larger than 0.1, and (2) there are more than 3 (3 for Vehicle, 1 for Pedestrian and Cyclist) points inside the box. Otherwise, the boxes are assigned to low-score group. The threshold used for association is different for the two groups. In high-score group, the new detected boxes are first associated with pre-existing object tracks by BEV IoU (0.3 for Vehicle, 0.15 for Pedestrian and Cyclist). The unmatched boxes are used to generate new object tracks and fed into low-score group for next stage association (0.2 for Vehicle, 0.1 for Pedestrian and Cyclist). After successful association, we would replace the trajectories with matched detected boxes, rather than updating them through Kalman filtering.

In the life cycle management, the birth rate and death rate of an object track are set to 1 and infinite. When any two object tracks overlap with each other and the ratio is larger than the threshold (0.5 for Vehicle, 0.4 for Pedestrian and Cyclist), we will merge them together by keeping the earlier birth object ID. Afterward, any redundant boxes that have not been updated are removed.

D. Implentation Details of Attribute-based Refining

In this section, we provide the details of the network structure, training strategies and loss functions of each refining model.

(a) (b)

Figure 6. The point-to-surface encoding in GRM (a) and point-to-corner encoding in PRM (b). All the distances are three dimensional (x, y, z). Note that there are a few points outside the corresponding proposal box.

D.1. Geometry Refining Model

Encoder Network Structures. In our GRM, the query encoder and value encoder are both PointNet [27] structured. Each layer is built as a multi-layer perceptron (MLP) followed by batch normalization and ReLU activation layer. The query encoder ENC_1 takes as input t randomly selected samples to generate corresponding geometry queries $\mathbf{Q}^{\text{geo}} \in \mathbb{R}^{t \times D}$. Meanwhile, the selected n geometry-aware points (after proposal-to-surface encoding shown in Fig. 6) are fed into the value encoder ENC_2 to generate the global point feature, serving as $\mathbf{V}^{\text{geo}} \in \mathbb{R}^{n \times D}$. The details of point cloud processing are shown in Table 10 and Table 11 respectively.

Index	Input	Operation	Output Shape
(1)	-	geometry points f^{geo}	$t \times 256 \times 11$
(2)	(1)	Linear$(11 \rightarrow 128)$	$t \times 256 \times 128$
(3)	(2)	ReLU, BN	$t \times 256 \times 128$
(4)	(3)	Linear$(128 \rightarrow 128)$	$t \times 256 \times 128$
(5)	(4)	ReLU, BN	$t \times 256 \times 128$
(6)	(5)	Linear$(128 \rightarrow 256)$	$t \times 256 \times 256$
(7)	(5)	Max pooling	$t \times 256$
(8)	(7)	Linear$(256 \rightarrow 256)$	$t \times 256$
(9)	(8)	ReLU, BN	$t \times 256$

Table 10. The architecture of query encoder in GRM. t is the number of randomly selected proposals for each object track. For each proposal, we randomly sample 256 points.

Attention-based Decoder. Our decoder layer follows the classical design, which consists of a multi-head self-attention layer, a multi-head cross-attention layer and an FFN with residual structure. We adopt 1-layer structure in our implementation and the network structure is shown in Fig. 7.

For the multi-head self-attention layer (**SA**), we enrich contextual relationships and feature differences among selected samples. Specifically, we map the object queries $\mathbf{Q}^{\text{geo}}$ by linear projections $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ to form the so-called query, key, and value. For simplicity, we omit the

superscript "geo". Then, the output after **SA** is given by

$$\mathbf{SA}\left(\mathbf{Q}^{\text{geo}}\right) = \left[\sum_{m=0}^{t} \frac{\exp\left(\mathbf{W}_1\mathbf{q}_i\left(\mathbf{W}_2\mathbf{q}_m\right)^T\right)}{\sum_{j=0}^{t}\exp\left(\mathbf{W}_1\mathbf{q}_j\left(\mathbf{W}_2\mathbf{q}_j\right)^T\right)}\mathbf{W}_3\mathbf{q}_m\right] \tag{1}$$

where $[\cdot]$ is a concatenation operation and $\mathbf{Q}^{\text{geo}}$ can be divided into $[\mathbf{q}_1, \cdots, \mathbf{q}_i, \cdots, \mathbf{q}_t], i = 1, ..., t$.

For the multi-head cross-attention layer (**CA**), the refined object queries can aggregate relevant context from global point features for compensating supplementary views. And the calculation is expressed by

$$\mathbf{CA}\left(\mathbf{Q}^{\text{geo}}, \mathbf{K}^{\text{geo}}, \mathbf{V}^{\text{geo}}\right) = \left[\sum_{m=0}^{t} \frac{\exp\left(\mathbf{W}_4\mathbf{q}_i\left(\mathbf{W}_5\mathbf{k}_m\right)^T\right)}{\sum_{j=0}^{t}\exp\left(\mathbf{W}_4\mathbf{q}_j\left(\mathbf{W}_5\mathbf{k}_j\right)^T\right)}\mathbf{W}_6\mathbf{v}_m\right] \tag{2}$$

where $\mathbf{W}_4, \mathbf{W}_5, \mathbf{W}_6$ are linear projections, $\mathbf{K}^{\text{geo}}$ can be divided into $[\mathbf{k}_1, \cdots, \mathbf{k}_t]$, and $\mathbf{V}^{\text{geo}}$ can be divided into $[\mathbf{v}_1, \cdots, \mathbf{v}_t]$.

Training Details. During training, we randomly selected $t = 3$ object proposals as geometry queries. While in inference, the 3 samples are selected with the highest scores. For each query, we predict its size classes (among pre-defined template size classes) and residual sizes for each size class. We use 3 size anchors (length, width, height) for all three classes: $(4.8, 1.8, 1.5), (10.0, 2.6, 3.2), (2.0, 1.0, 1.6)$. The size classes are supervised with a cross-entropy loss $L_{\text{cls}}^{\text{geo}}$ while the residual sizes are supervised with a L1 loss $L_{\text{reg}}^{\text{geo}}$. The total geometry refining loss is $L^{\text{geo}} = 0.1L_{\text{cls}}^{\text{geo}} + 2L_{\text{reg}}^{\text{geo}}$. The final geometry size is the average of these 3 predictions, which is then assigned to all the frame of the object track.

The randomly selected $n = 4096$ geometry-aware points are augmented through randomly flipping along X, Y axes with 50% chance, and randomly rotating around the Z-axis by Uniform$\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$ degrees, and randomly scaling by Uniform$[0.9, 1.1]$. During inference, we also adopt TTA

Index	Input	Operation	Output Shape
(1)	-	geometry points f^{geo}	$n \times 10$
(2)	(1)	Linear$(10 \rightarrow 128)$	$n \times 128$
(3)	(2)	ReLU, BN	$n \times 128$
(4)	(3)	Linear$(128 \rightarrow 128)$	$n \times 128$
(5)	(4)	ReLU, BN	$n \times 128$
(6)	(5)	Linear$(128 \rightarrow 512)$	$n \times 512$
(7)	(5)	Max pooling	128
(8)	(7)	Repeat	$n \times 128$
(9)	(5)(8)	Concatenate	$n \times 640$
(10)	(9)	Linear$(640 \rightarrow 256)$	$n \times 256$
(11)	(10)	ReLU, BN	$n \times 256$

Table 11. The architecture of value encoder in our GRM.

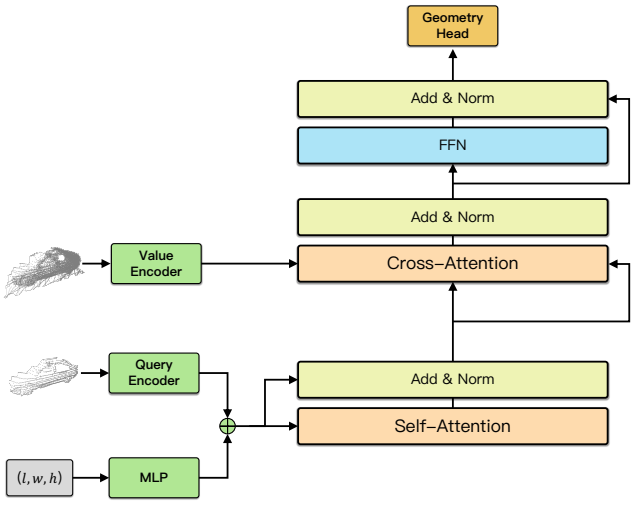

Figure 7. The network structure of decoder in GRM.

settings, in which the scaling operation can boost the performance at most while the flipping and rotation along z-axis operations lead to slight improvements.

We use Adam optimizer with a one-cycle decay policy to separately train the model for each class. The initial learning rate is 0.001 and the batch size is set to 128. The total epochs are 30 for Vehicle, 100 for Pedestrian and 500 for Cyclist. In total, we have extracted around 44K vehicle tracks, 18K pedestrian tracks and 0.5K cyclist tracks for training. Ground-truth boxes are assigned to every frame of the object track (frames with no matched ground-truth are skipped, such as the non-point objects).

D.2. Position Refining Model

Encoder Network Structures. The encoders in PRM are similar to those in GRM. Each object track is padded with zeros to the length of the whole sequence, such as 200 for WOD. The full processing procedures are shown in Tabel 12.

Attention-based Decoder. The full attention-based process is the same as mentioned by Eq. 1 and Eq. 2. And the network structure is shown in Fig. 8.

Training Details. The object tracks short than 7 are deprecated during training, and we also adopt a random frame deprecation as the additional data augmentation. Random flipping operation could boost the performance at most by stabilizing the trajectories during inference. The residual distances between each tracked box to the randomly-selected proposal's center are supervised with an L1 loss $L_{\text{reg}}^{\text{ce}}$. For heading prediction, we also utilize a bin-based classification and residual degrees. We use 12 heading anchors, each bin accounts for 30 degrees from 0 to 360 degrees. The total loss is $L^{\text{pos}} = L_{\text{reg}}^{\text{ce}} + 0.1L_{\text{cls}}^{\text{yaw}} + 2L_{\text{reg}}^{\text{yaw}}$. We train total 50 epochs for *Vehicle* with a batch size of 96, 100 epochs for *Pedestrian* with a batch size of 128, and 200

Index	Input	Operation	Output Shape
(1)	-	position points f^{geo}	$200 \times 256 \times 32$
(2)	(1)	Linear$(32 \rightarrow 128)$	$200 \times 256 \times 128$
(3)	(2)	ReLU, BN	$200 \times 256 \times 128$
(4)	(3)	Linear$(128 \rightarrow 128)$	$200 \times 256 \times 128$
(5)	(4)	ReLU, BN	$200 \times 256 \times 128$
(6)	(5)	Linear$(128 \rightarrow 256)$	$200 \times 256 \times 256$
(7)	(6)	Max pooling	200×256
(8)	(7)	Linear$(256 \rightarrow 256)$	200×256
(9)	(8)	ReLU, BN	200×256

Table 12. The architecture of query encoder in PRM. The object track is padded to the length of 200. For each proposal of the object track, we randomly sample 256 points.

epochs for *Cyclist* with a batch size of 64. The optimizer setting is the same as our GRM.

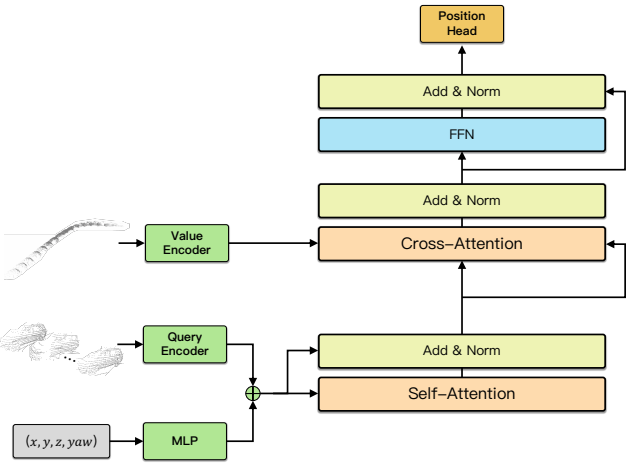

Figure 8. The network structure of decoder in PRM.

D.3. Confidence Refining Model

For the first classification branch of our CRM, we use different IoU thresholds to determine the positive and negative samples. Specifically, τ_h is set to 0.7 for *Vehicle*, 0.5 for *Pedestrian* and *Cyclist*, τ_l is set to 0.35, 0.25 and 0.25 respectively. We use binary labels 0 and 1 for supervision. For both two branches, we use BCE loss to supervise the predictions. And the total loss $L^{\text{conf}} = L^{\text{conf}}_{\text{cls}} + L^{\text{conf}}_{\text{iou}}$. We train total 30 epochs for *Vehicle* with a batch size of 256, 50 epochs for *Pedestrian* with a batch size of 256, and 100 epochs for *Cyclist* with a batch size of 64. The optimizer setting is the same as our GRM.

E. Details of the Human Label Study

We keep the same setting as 3DAL [29], and the randomly selected 5 sequences from the WOD val set are listed in Table 13. We directly utilize their human labeling results rather than repeat the whole labeling task. In summary, there are 12 experienced labels to annotate the 15 labeling tasks (3 sets of re-labels for each sequence) and obtain 2.3k labels. Then, the human APs are computed by comparing them with the WOD's released ground-truth labels and using the number of points in boxes as human label scores.

Sequence
segment-17703234244970638241_220_000_240_000
segment-15611747084548773814_3740_000_3760_000
segment-11660186733224028707_420_000_440_000
segment-10243601436120575203580_000_3600_000
segment-6491418762940479413_6520_000_6540_000

Table 13. The list of selected sequences from WOD val set for human label study.

Besides, we also report the statistical results of auto labels (on 90% train set) in Table 14 to better show that the auto labels contain fewer boxes than ground-truth, especially for the hard cases (object points are smaller than 5). Therefore, as shown in Fig. 9, the student model trained with auto labels would generate fewer false positives than training with ground-truth especially when the score is larger than 0.2. Besides, when we remove the boxes by cutting different scores, the student model trained with auto labels can preserve more true positive boxes, which proves that the model is more confident in the easy samples. We infer that the model can focus more on the easy samples with better convergence.

F. More Experiments

Comparison on different distances. To better evaluate the effect of our DetZero, we report the performance on different distances. As shown in Tabel 15, for both *Vehicle* and *Pedestrian*, the improvements are increasing while the distances are from near to far. It proves that the current performance bottleneck of object detection exists at the farther

	Vehicle		*Pedestrian*	
	≥ 5 pts	< 5 pts	≥ 5 pts	< 5 pts
Ground-truth	3,435,724	495,419	1,535,584	303,308
Auto labels	2,867,096	165,924	1,290,236	158,074

Table 14. The comparison between ground-truth and auto labels. Boxes in auto labels have an IoU larger than thresholds (0.7 for *Vehicle* and 0.5 for *Pedestrian*) would be kept for statistics.

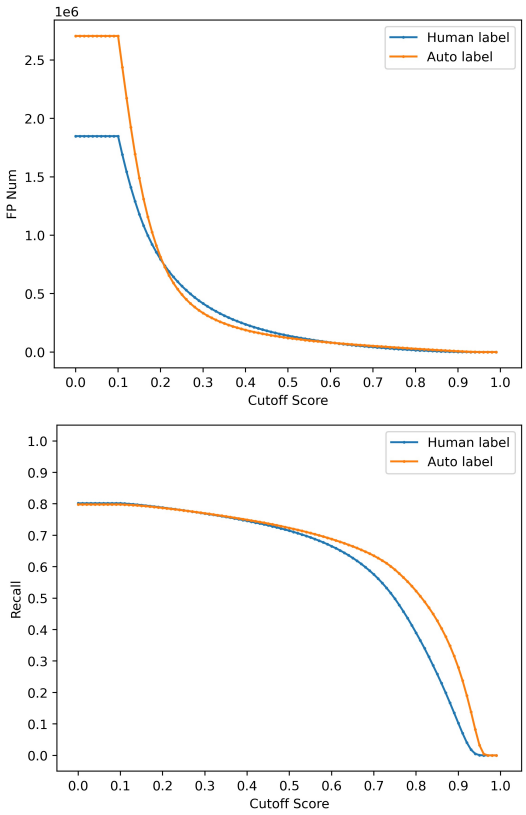

Figure 9. The number of false positive boxes curve and the recall curve on WOD val set. The curves decrease by cutting scores from 0 to 1.0.

	Total		*0-30m*		*30-50m*		*50+m*	
	L1	L2	L1	L2	L1	L2	L1	L2
Upstream	82.57	75.24	94.25	93.35	81.54	75.25	63.28	51.32
Full	89.06	82.92	96.27	95.52	88.41	83.97	77.80	65.70
improve	+6.49	+7.68	+2.02	+2.17	+6.87	+8.72	**+14.52**	**+14.38**
Upstream	83.07	76.34	86.09	82.26	82.09	75.39	77.35	65.22
Full	87.06	81.01	89.25	85.71	86.14	80.84	83.08	71.94
improve	+3.99	+4.67	+3.16	+3.45	+4.05	+5.47	**+5.73**	**+6.72**

Table 15. Performance evaluation of different distances on WOD val set. Metrics are standard 3D APH of both L1 and L2 difficulties for *Vehicle* (first group) and *Pedestrian* (second group).

range. And our DetZero could utilize the long-term temporal context to optimize these boxes located at the beginning and the end of an object track. In addition, the improvements of objects with L2 difficulty are larger than those of L1 difficulty, which draws the same conclusion as Table 8.

Offline tracking generates complete tracks. We list the top SOTA tracking methods on Waymo 3D tracking leaderboard [2] in Table 20. Our DetZero ranks 1st place by outperforming previous SOTA performance with 9.97-point MOTA (L2) for all classes. Compared to our own upstream results, we still keep a huge performance improvement with 5.84-point MOTA (L2) for all classes.

[2]We report the performance of 3D detection and tracking till 2023-03-08 23:59 GMT.

	Vehicle (0.7 / 0.5)		*Pedestrian* (0.5 / 0.3)	
	Recall	Precision	Recall	Precision
Detection	83.6 / 95.6	13.4 / 15.3	88.9 / 97.1	6.7 / 7.3
IoU filer	75.3 / 92.7	52.6 / 65.1	83.8 / 95.0	17.8 / 20.2
OR filer	75.2 / 92.4	55.8 / 69.0	82.7 / 93.5	20.7 / 23.4
Offline Trk.	75.4 / 91.8	66.2 / 81.9	81.2 / 91.3	35.7 / 40.3

Table 16. Performance comparison of our offline tracking. Metrics are 3D Recall and Precision under different IoU thresholds for *Vehicle* (0.7 / 0.5) and *Pedestrian* (0.5 / 0.3). OR filter is the filtering operation based on the overlap ratio.

We also show the effect of generating sufficient complete object tracks in Table 16. The first row shows that the detection results contain huge false-positive boxes, resulting in very low precision performance. Traditional IoU-based filtering operations will loose the effect when facing overlapped boxes. As a comparison, our overlap ratio based filtering would further remove these boxes, especially under a loose threshold. Finally, the whole offline tracking procedure would further remove FPs while keeping a slightly-low Recalls.

Effect of point cloud information encoding. We show the ablation of point cloud information encoding methods used in GRM and PRM. For every experiment, we randomly selected 20% sequences (160) of the original train set for training, and evaluate the performance on whole val set (202 sequences). We also report the Accuracy performance by object's motion state, which is calculated by its ground-truth trajectory. In Table 17, our point-to-surface encoding method yields the largest gains. In Table 18, the point-to-corner encoding method yields the largest gains compared to point-to-center encoding. Because the tracked boxes have already provided efficient geometry information, which could be efficiently utilized by our encoding method. We also find that the improvements after position refining are much higher than those after geometry refining, which further demonstrates the effect of our PRM on removing jitters and smoothing trajectories through attending global motion information.

xyzi	p2s	score	ALL box	ALL track	Static box	Dynamic box
✓			78.08	66.87	76.18	84.12
✓	✓		78.50	67.36	76.60	84.43
✓	✓	✓	**78.56**	**67.42**	**76.66**	**84.51**

Table 17. Effect of the different point encoding method for GRM. Metrics are Accuracy under standard IoU (0.7 for *Vehicle*) for both box-level and track-level statistics. We split the objects based on its ground-truth motion state for better comparison.

xyzi	p2ce	p2co	score	ALL box	ALL track	Static box	Dynamic box
✓				78.95	68.78	78.36	80.81
✓	✓		✓	80.98	71.95	80.71	81.84
✓	✓	✓		81.84	72.83	81.30	83.55
✓	✓	✓	✓	**81.99**	**73.22**	**81.47**	**83.60**

Table 18. Effect of the different point encoding method for PRM. Metrics are Accuracy under standard IoU (0.7 for *Vehicle*) for both box-level and track-level statistics. We split the objects based on its ground-truth motion state for better comparison.

# of query	ALL box	ALL track	Static box	Dynamic box
1	78.29	66.81	76.38	84.27
2	78.48	67.23	76.46	84.34
3	78.56	**67.42**	**76.66**	**84.51**
5	**78.57**	67.32	**76.66**	84.50

Table 19. Effect of the different number of geometry queries used in GRM. Metrics are Accuracy under standard IoU (0.7 for *Vehicle*) for both box-level and track-level statistics. We split the objects based on its ground-truth motion state for better comparison.

Effect of the number of geometry queries. We show the empirical performance by selecting different object samples as geometry queries. As shown in Table 19, the performance increases while the number of queries increases, which could be viewed as another data augmentation method. Note that the performance gaps among them may not be very stable and we finally select 3 queries in our whole processing.

G. Qualitative Results

In this section, we show the qualitative comparisons after our attribute-refining module in Fig. 10 and Fig. 11.

Method	Rank	Frames	MOTA L2	*Vehicle* (MOTA ↑ /MOTP↓) L1	*Vehicle* (MOTA ↑ /MOTP↓) L2	*Pedestrian* (MOTA ↑ /MOTP↓) L1	*Pedestrian* (MOTA ↑ /MOTP↓) L2	*Cyclist* (MOTA ↑ /MOTP↓) L1	*Cyclist* (MOTA ↑ /MOTP↓) L2
DetZero (Full)	1	200	**75.05**	**79.04 / 14.09**	**75.97 / 14.18**	**77.60 / 28.76**	**76.03 / 28.76**	**73.24 / 23.77**	**73.16 / 23.77**
DetZero (Upstream)	–	200	69.21	71.02 / 15.47	67.96 / 15.47	71.56 / 29.90	70.00 / 29.90	69.75 / 24.24	69.67 / 24.24
InceptioLidar*	2	10	65.08	68.78 / 15.68	65.58 / 15.70	66.38 / 29.54	64.52 / 29.54	65.19 / 25.42	65.12 / 25.42
HorizonMOT3D [42]	3	5	63.45	67.30 / 15.75	64.07 / 15.77	65.88 / 30.67	64.15 / 30.67	62.20 / 25.45	62.13 / 25.45
MFMS_Track*	4	4	63.27	66.45 / 15.65	63.14 / 15.65	65.47 / 30.19	63.85 / 30.19	62.90 / 25.44	62.83 / 25.44
CasTrack*	5	5	62.60	66.95 / 15.79	63.66 / 15.79	66.39 / 30.22	64.79 / 30.24	59.41 / 25.30	59.34 / 25.30
ImmortalTracker [41]	6	2	60.92	63.77 / 16.22	60.55 / 16.22	62.20 / 31.17	60.60 / 31.20	61.68 / 27.41	61.61 / 27.41
OptMOT*	7	2	60.85	65.47 / 16.16	62.18 / 16.16	60.02 / 30.58	58.31 / 30.58	62.14 / 26.97	62.06 / 26.97
SimpleTrack [24]	8	2	60.18	63.53 / 16.19	60.30 / 16.23	61.75 / 31.09	60.13 / 31.14	60.18 / 27.35	60.12 / 27.35
CenterPoint [56]	11	2	58.67	62.58 / 16.30	59.38 / 16.37	58.28 / 31.13	56.64 / 31.16	60.06 / 27.62	60.00 / 27.62

Table 20. Performance comparison on the Waymo 3D tracking leaderboard. Metrics are standard 3D MOTA and MOTP by both L1 and L2 difficulties. Anonymous submissions are marked with $*$.

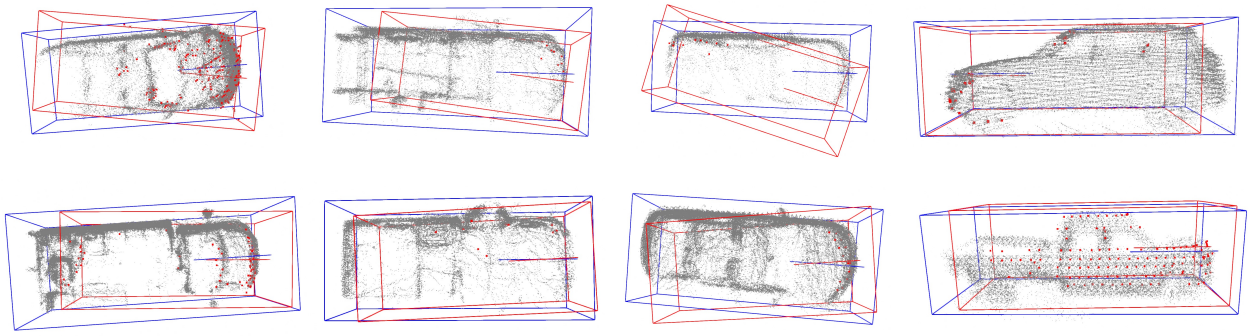

Figure 10. The visualization of GRM results on WOD val set. The red boxes are selected from one frame of the object track, and corresponding points are also colored with red. The refining boxes with precise sizes are colored with blue.

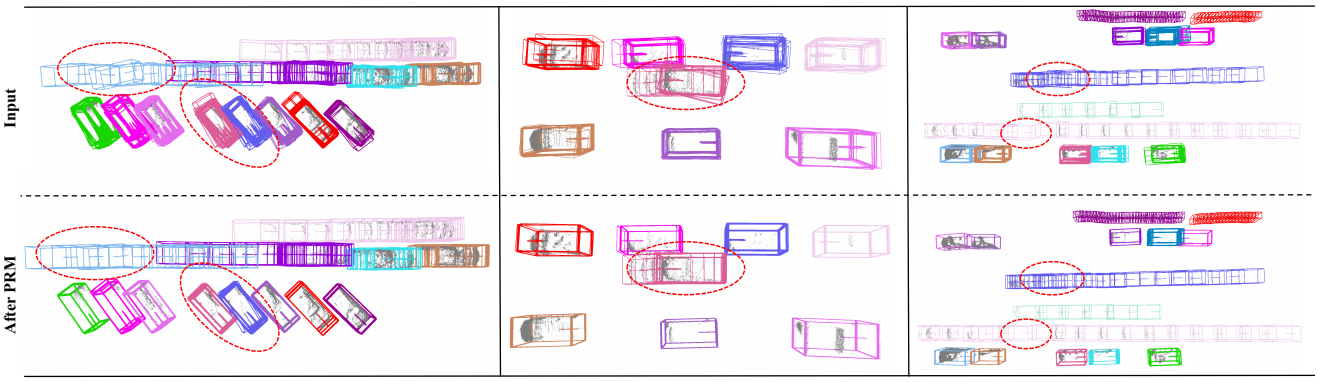

Figure 11. The visualization of PRM results on WOD val set. The first row is the input object tracks, and the second row is the corresponding results after PRM. We use red dotted circles to mark the important cases.