Beyond 3D Siamese Tracking: A Motion-Centric Paradigm for 3D Single Object Tracking in Point Clouds
Supplementary Material

1. Implementation Details

Network Architecture

The target segmentation network is a PointNet segmentation network [3], where each input point is firstly processed by a 5-layer MLP with output channel sizes of 64, 64, 64, 128, 1024. A max-pooling is then applied along the point dimension to obtain a 1024-dim global embedding, which is then copied and concatenated with the output of the second layer (64-dim). The concatenated features are then processed by another 5-layer per-point MLP with output channel sizes 512, 256, 128, 128, 2. Every layer of the MLP except the last one has batch normalization and ReLU. The output logits are used to extract the target points from the input.

At the 1^{st}-stage, we use a vanilla PointNet [3] to encode the spatial-temporal target points (with the temporal channel) into a 256-dim embedding. The PointNet includes a 4-layer per-point MLP with output sizes 64, 128, 256, 512, a point-wise max-pooling layer, and another MLP with output sizes 512, 256. We have batch normalization and ReLU for every layer of the MLP. On top of the encoded embedding, we independently apply two MLPs with 3 hidden layers (128,128,128) to obtain the motion state (6-dim) and the **R**elative **T**arget **M**otion (RTM) for previous box refinement (4-dim). The motion state includes a 4-dim RTM and a 2-dim motion classification logit.

At the 2^{nd}-stage, we use a similar PointNet [3] as in the 1^{st}-stage to regress a 4-dim RTM on the denser target point cloud (without the temporal channel). The PointNet includes a 4-layer per-point MLP with output sizes 64, 128, 256, 512, a point-wise max-pooling layer, and another MLP with output sizes 512, 256, 4. We have batch normalization and ReLU for every layer of the MLP except the last one.

Training & Inference

We train our models using the Adam optimizer with batch size 256 and an initial learning rate 0.001, which is decayed by 10 times every 20 epochs. The training takes ~ 4 hours to converge on a V100 GPU for the KITTI Cars. During the inference, the model tracks a target frame-by-frame in a point cloud sequence given the target BBox at the first

Algorithm 1 Workflow of the 1^{st}-stage

Input: Segmented target points $\widetilde{\mathcal{P}}_{t-1,t}$ and possible target BBox $\mathcal{B}_{t-1}$ at the previous frame.

Output: A relative target motion state (including a RTM $\mathcal{M}_{t-1,t}$ and 2D binary motion state logits), a refined target BBox $\widetilde{\mathcal{B}}_{t-1}$ at the previous frame, and a coarse target BBox $\mathcal{B}_t$ at the current frame.

1: Use a PointNet to encode $\widetilde{\mathcal{P}}_{t-1,t}$ to an embedding $\mathcal{E}$.
2: Obtain an RTM with respect to $\mathcal{B}_{t-1}$ by applying an MLP to the embedding $\mathcal{E}$.
3: Obtain the refined BBox $\widetilde{\mathcal{B}}_{t-1}$ at the previous frame by transforming $\mathcal{B}_{t-1}$ using the RTM predicted in *step 2*.
4: Obtain the motion state by applying another MLP to the embedding $\mathcal{E}$. The the motion state includes an RTM $\mathcal{M}_{t-1,t} \in \mathbb{R}^4$ and a 2D logit indicating whether the target is dynamic or not.
5: If the target is dynamic, obtain the coarse $\mathcal{B}_t$ by transforming $\widetilde{\mathcal{B}}_{t-1}$ using the RTM $\mathcal{M}_{t-1,t}$. Otherwise, set $\mathcal{B}_t = \widetilde{\mathcal{B}}_{t-1}$.

Algorithm 2 Workflow of the 2^{nd}-stage

Input: Segmented target points $\widetilde{\mathcal{P}}_{t-1,t}$, the coarse target BBox $\mathcal{B}_t$ at the current frame and the motion state predicted in the 1^{st}-stage.

Output: A refined target BBox $\widetilde{\mathcal{B}}_t$ at the current frame.

1: Extract $\widetilde{\mathcal{P}}_{t-1} \in \mathbb{R}^{M_{t-1} \times 3}$ and $\widetilde{\mathcal{P}}_t \in \mathbb{R}^{M_t \times 3}$ from $\widetilde{\mathcal{P}}_{t-1,t} \in \mathbb{R}^{(M_{t-1}+M_t) \times 4}$ according to the timestamp.
2: If the target is dynamic, transform $\widetilde{\mathcal{P}}_{t-1}$ to $\hat{\mathcal{P}}_{t-1}$ using the RTM $\mathcal{M}_{t-1,t}$. Otherwise, simply set $\hat{\mathcal{P}}_{t-1} = \widetilde{\mathcal{P}}_{t-1}$.
3: Form a denser target point cloud $\hat{\mathcal{P}}_t \in \mathbb{R}^{(M_{t-1}+M_t) \times 3}$ by merging $\hat{\mathcal{P}}_{t-1}$ and $\widetilde{\mathcal{P}}_t$.
4: Transform $\hat{\mathcal{P}}_t$ to the canonical coordinate system defined by $\mathcal{B}_t$.
5: Apply a PointNet on the canonical $\hat{\mathcal{P}}_t$ to regress a RTM.
6: Obtain the refined BBox $\widetilde{\mathcal{B}}_t$ by transforming $\mathcal{B}_t$ using the RTM predicted in *step 5*.

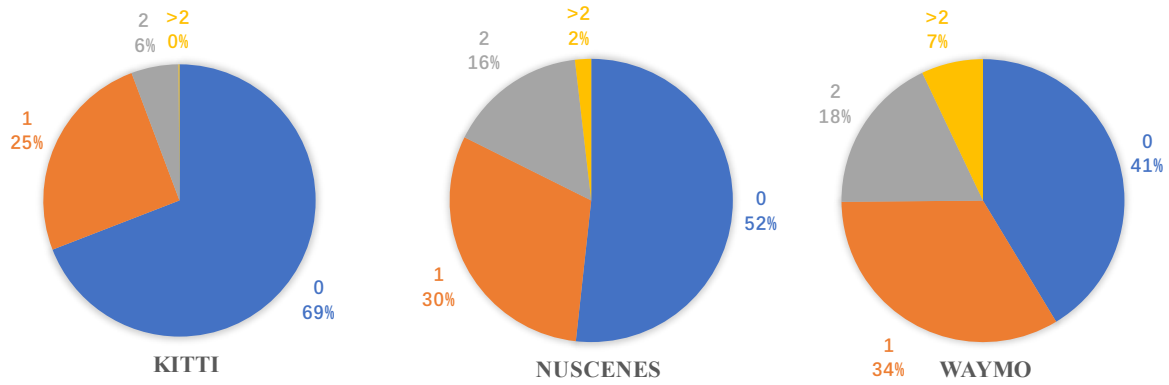

Figure 1. Distributions of distractors for **Cars (Vehicles)** in KITTI, NuScenes, and Waymo Open Dataset. We enlarge each target BBox by **2 meters** and count the distractors inside. Objects with the same category as the target are regarded as distractors.

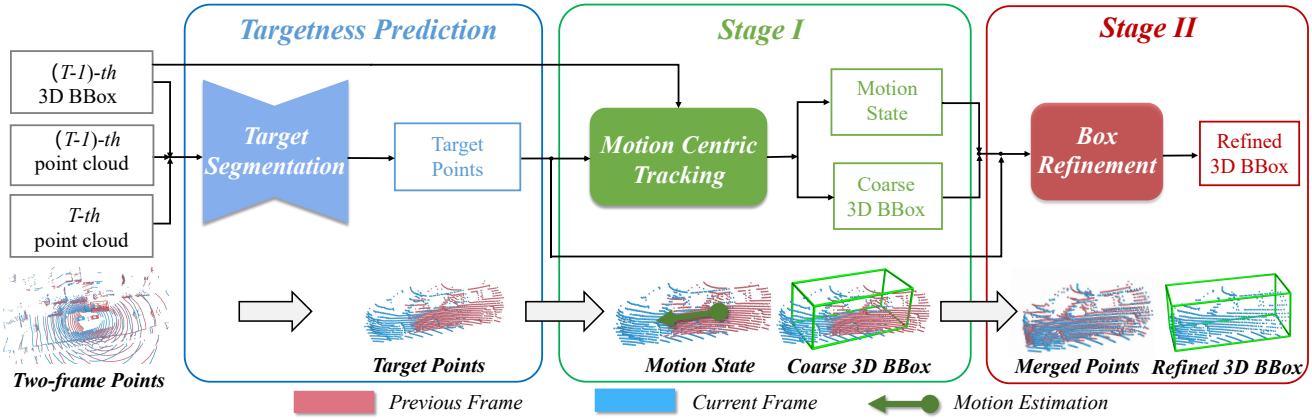

Figure 2. The pipeline of M^2-Track. The data flow is illustrated with **black** lines.

frame.

Detailed Workflow

The overall pipeline with the data flow of M^2-Track is presented in Fig. 2. The detailed description of the 1^{st}-stage and 2^{nd}-stage are provided in Alg. 1 and Alg. 2, respectively.

4DOF RTM Transformation

We focus on the 4DOF RTM within two successive frames. Given a 4D RTM $(\Delta x, \Delta y, \Delta z, \Delta\theta)$, we can construct a transformation matrix $\mathcal{T} \in \mathbb{R}^{4\times4}$ as follows:

$$\begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) & 0 & \Delta x \\ \sin(\Delta\theta) & \cos(\Delta\theta) & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

We define the above process as a function $\mathcal{T} : \mathbb{R}^4 \mapsto \mathbb{R}^{4\times4}$.

Point Transformation. Given any object point $p = (x, y, z)$ in a BBox $\mathcal{B} = (x_b, y_b, z_b, \theta_b, w_b, l_b, h_b)$, we can transform it using an RTM $= (\Delta x, \Delta y, \Delta z, \Delta\theta)$ under the homogeneous coordinates:

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \\ 1 \end{bmatrix} = \mathcal{T}(\mathcal{B}[:4]) \times \mathcal{T}(\text{RTM}) \times \mathcal{T}(\mathcal{B}[:4])^{-1} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2}$$

Here $\hat{p} = (\hat{x}, \hat{y}, \hat{z})$ denotes the transformed point. And $\mathcal{B}[:4] = (x_b, y_b, z_b, \theta_b)$ is the 4DOF pose of the BBox $\mathcal{B}$. Note that we transform all scenes from different datasets to the same *right-handed* coordinate system with the z-axis pointing upward.

Box Transformation. For a BBox $\mathcal{B} = (x, y, z, \theta, w, l, h)$, we transform its center using Eqn. 2. The transformed BBox is $\hat{\mathcal{B}} = (\hat{x}, \hat{y}, \hat{z}, \theta + \Delta\theta, w, l, h)$, where $(\hat{x}, \hat{y}, \hat{z})$ is the transformed center, $(\theta + \Delta\theta)$ is the transformed heading angle, and w, l, h are the width, length and height of the BBox which remain unchanged.

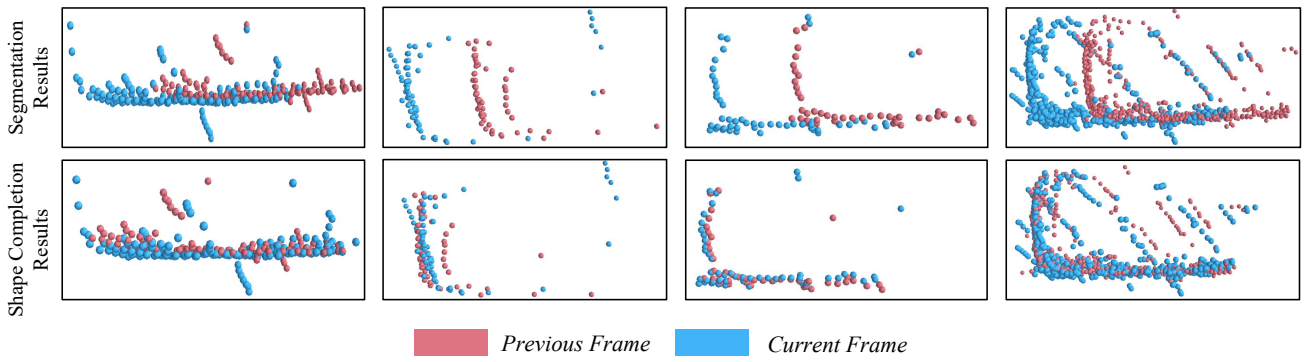

Figure 3. Visualization of the target segmentation and motion-assisted shape completion. Pictures in the same column are from the same case. The completion results demonstrate that our model learns good enough relative target motions.

2. More Analysis

Distractor Statistics

We count the number of distractors in each object's neighborhood in the training set of KITTI [2], NuScenes [1], and Waymo Open Dataset (WOD) [5] respectively. Specifically, we enlarge each target BBox by 2 meters and count the number of annotated BBoxs which not only intersect with the enlarged area but also have the same category as the target. Fig. 1 illustrates the distributions of distractors for cars/vehicles. It shows that more than two-thirds (**69%**) of the regions in KITTI are free of distractors. While in NuScenes and WOD, distractors are very common, especially for WOD. Besides, though we only consider a pretty small neighborhood around the target, some regions in NuScenes and WOD have even more than two distractors. We do the same analysis on the pedestrians in KITTI and find that 68.3% of the regions has at least 1 distractor(s). All of these observations, together with our main experiment results, prove that M^2-Track is much more robust to distractors than previous matching-based approaches.

Larger search area for NuScenes Cars

By default, we enlarge the (predicted) target BBox at previous frame by **2 meters** and collect points inside to generate the inputs. This strategy is also adopted in P2B [4] and BAT [6] to generate their search areas. The 2 meters larger area is sufficient for KITTI and WOD, where keyframes are sampled at 10Hz. However, NuScenes only provides keyframes at 2Hz. Thus, the target may move more than 2 meters even within two consecutive keyframes. For a fair comparison, we only report our results with 2 meters in the main manuscript. In Tab. 1, we re-evaluate our performance on NuScenes Cars with **5 meters** larger search area. Note that using a larger area does not incur more computational cost because we keep the number of sampled points unchanged. As shown in Tab. 1, we can further improve the performance of M^2-Track by using larger search areas. However, due to the increase of distractors and sparsity, larger search areas instead harm the performance of P2B and BAT.

Table 1. Larger search area for NuScenes Cars.

Method	Success	Precision
BAT [6] (2m)	40.73	43.29
BAT [6] (5m)	37.14 ↓ 3.59	39.92 ↓ 3.37
P2B [4] (2m)	38.81	43.18
P2B [4] (5m)	38.48 ↓ 0.33	42.15 ↓ 1.03
M^2-Track (2m)	55.85	65.09
M^2-Track (5m)	58.35 ↑ 2.50	67.04 ↑ 1.95

Limitations

Unlike appearance matching, our motion-centric model requires a good variety of motion in the training data to ensure its generalization on data sampled with different frequencies. For instance, our model suffers from considerable performance degradation if trained with 2Hz data but tested with 10Hz data because the motion distribution of the 2Hz and 10Hz data differs significantly. But fortunately, we can aid this using a well-design motion augmentation strategy.

3. Visualization

Target Segmentation

Our model depends on the target segmentation to learn the relative target motion (RTM). The first row in Fig. 3 shows our target segmentation results. We can see that most segmented points are from the target objects, demonstrating the effectiveness of spatial-temporal learning.

Motion-assisted Shape Completion

In the 2^{nd}-stage, we leverage the RTM $\mathcal{M}_{t-1,t}$ to complete the target point cloud at the current frame. As shown in the second row in Fig. 3, our method correctly merges the point clouds from two consecutive frames, using the predicted RTM. These results demonstrate that the RTMs are correctly modeled by our method.

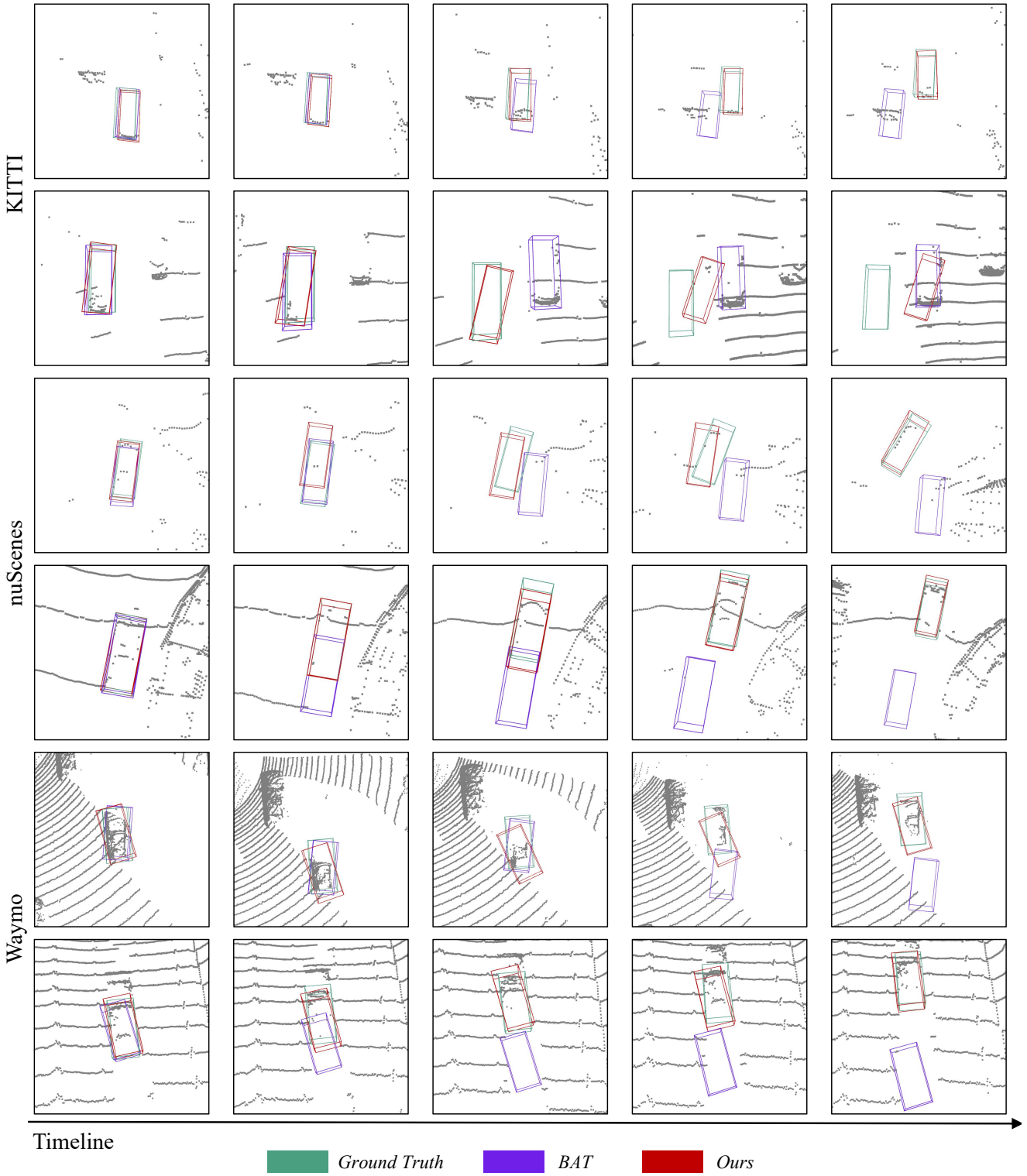

Figure 4. Visualization results for Cars.

Advantageous Cases

More qualitative comparison results are in Fig. 4 and Fig. 5. We also provide animated results in the attached video. We can observe that our M^2-Track consistently shows its ad-

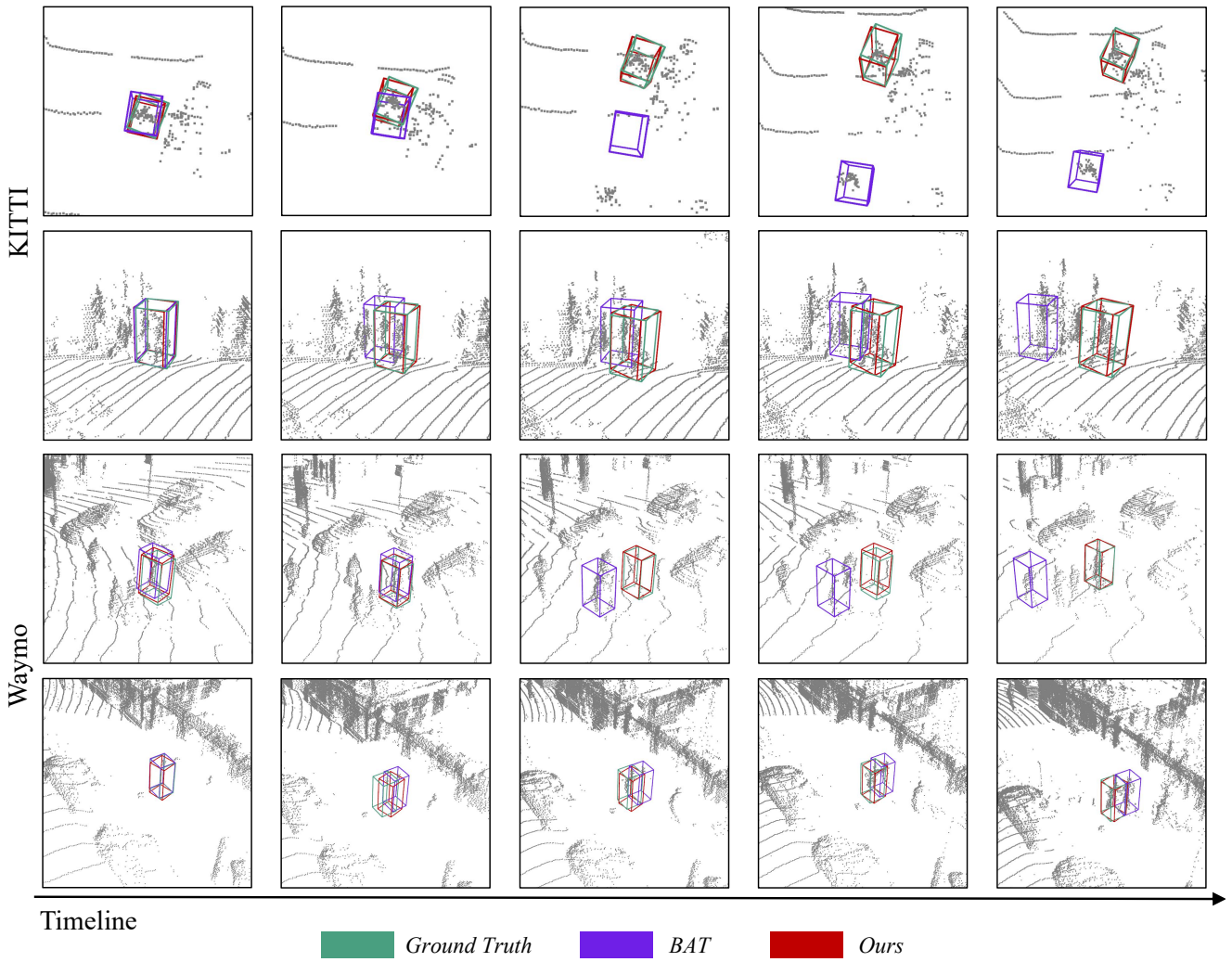

Figure 5. Visualization results for Pedestrians.

vantage when the scene is sparse, the relative target motion is large, or distractors exist in the target's neighborhood. However, since our M^2-Track only takes LiDAR point clouds as input, it fails on extremely sparse scenarios where the number of target points is almost zero (*e.g.* the second row in Fig. 4). Actually, this is a common issue for LiDAR-based SOT and could be probably solved by using multi-modal data (*e.g.* RGB images)

References

[1] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 11621–11631, 2020. 3

[2] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3354–3361, 2012. 3

[3] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017. 1

[4] Haozhe Qi, Chen Feng, Zhiguo Cao, Feng Zhao, and Yang Xiao. P2b: Point-to-box network for 3d object tracking in point clouds. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6329–6338, 2020. 3

[5] 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: Waymo open dataset. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2446–2454, 2020. 3

[6] Chaoda Zheng, Xu Yan, Jiantao Gao, Weibing Zhao, Wei Zhang, Zhen Li, and Shuguang Cui. Box-aware feature enhancement for single object tracking on point clouds. In *Int. Conf. Comput. Vis.*, pages 13199–13208, 2021. 3