Supplementary Document: MapTracker: Tracking with Strided Memory Fusion for Consistent Vector HD Mapping

Jiacheng Chen[1*], Yuefan Wu[1*], Jiaqi Tan[1*], Hang Ma[1], and Yasutaka Furukawa[1,2]

[1]Simon Fraser University [2]Wayve

The supplementary document provides remaining system details (§A,§B,§C) and additional experimental results (§D) as mentioned in the main paper. Please also refer to the video results at our project page `https://map-tracker.github.io`, demonstrating the local and global reconstructions in the video format with the input perspective image streams.

- ◇ §A: Remaining details of MapTracker, including 1) The transformation loss for PropMLP; 2) The query propagation module; and 3) The strided memory selection mechanism.
- ◇ §B: Remaining details/analyses of our consistent vector HD mapping benchmarks, including details on 1) consistent ground truth generation; 2) the track extraction algorithm for detection-based baseline approaches; and 3) the consistency-aware mAP (C-mAP).
- ◇ §C: Details of our online merging algorithm that generates the global vector HD maps from per-frame reconstructions.
- ◇ §D: Additional experimental results and analyses, including 1) The C-mAP results of all methods using the track extraction algorithm with different look-back parameters; 2) Check the temporal consistency of MapTR's ground truth data using our consistent-aware benchmarks; and 3) Additional qualitative results.

A Remaining Details of MapTracker

This section explains the remaining details of MapTracker (refer §3 of the main paper).

A.1 Transformation loss details

The transformation loss $\mathcal{L}_{\text{trans}}$ trains the PropMLP to ensure the latent transformation maintains the geometry and class type. The inputs and outputs of the PropMLP are described in §3.3, Figure 2, and Figure 3 of the main paper. For the propagated vector latents $\mathbf{M}^*_{\text{VEC}}(t-1)$ from $t-1$ to t, we apply the vector output heads to get the predictions $\mathbf{Y}^*(t-1)$. We then derive the ground truth $\hat{\mathbf{Y}}^*(t-1)$ by directly applying the transformation matrix to the ground

truth $\hat{\mathbf{Y}}(t-1)$. Since we have the optimal bipartite match $\boldsymbol{\omega}(t-1)$ of $t-1$, the transformation loss is defined by

$$\begin{aligned}\mathcal{L}_{\text{trans}} = \mathcal{L}_{\text{focal}}(\{\hat{c}_i^*(t-1)\}|_{\boldsymbol{\omega}(t-1)}, \{p_i^*(t-1)\}) \\ + \mathcal{L}_{\text{line}}(\{\hat{V}_i^*(t-1)\}|_{\boldsymbol{\omega}(t)}, \{V_i^*(t-1)\}),\end{aligned}$$

where $\hat{\mathbf{Y}}^*(t-1) = \{\hat{V}_i^*(t-1), \hat{c}_i^*(t-1)\}$ and $\mathbf{Y}^*(t-1) = \{V_i^*(t-1), p_i^*(t-1)\}$.

A.2 Memory fusion details

§3.2 and §3.3 of the main paper have explained our BEV and vector memory fusion. We provide more implementation details here.

Strided memory selection. We present the concrete implementation steps of our strided memory selection in the following. Firstly, we sort all memory entries based on the distance to the current location. Then, we select one closest memory entry for each stride value, starting from the farthest stride (*i.e.*, 15m). If the memory buffer contains less than four history latents, we simply take all.

Vector fusion layer. For the per-instance cross-attention of the vector fusion layer, we compute the absolute value of the relative frame difference from the history to the current frame, encode it with the sin/cos positional encoding, and use the encoding as the position encoding for the key and values of the cross-attention.

B Remaining Details of Benchmark Contributions

This section presents thorough details of our consistent vector HD mapping benchmarks, complementing §4 of the main paper.

B.1 Consistent ground truth

We review the typical problems of the two existing ground-truth data and demonstrate the improved quality of our consistent ground truth.

Pedestrain-crossing (nuScenes and Argoverse2). Figure 1 shows typical failure cases in MapTR [2] and StreamMapNet's [4] ground truth (GT) for the pedestrian crossing class. The examples are from the nuScenes dataset, where the raw annotations are many small polygon pieces. A merging algorithm must merge the pieces to get a correct global polygon for each pedestrian crossing instance. MapTR's merging algorithm merges all small polygons with overlaps in a brute-force way, making some polygons merge and split when crossing the perception boundary and introducing temporal inconsistency. StreamMapNet's merging algorithm considers the orientation of each polygon piece to avoid merging orthogonal pedestrian crossings, thus achieving better temporal consistency. However, the algorithm sometimes fails to handle noisy small pieces near the perception boundary. We borrow the merging algorithm from StreamMapNet

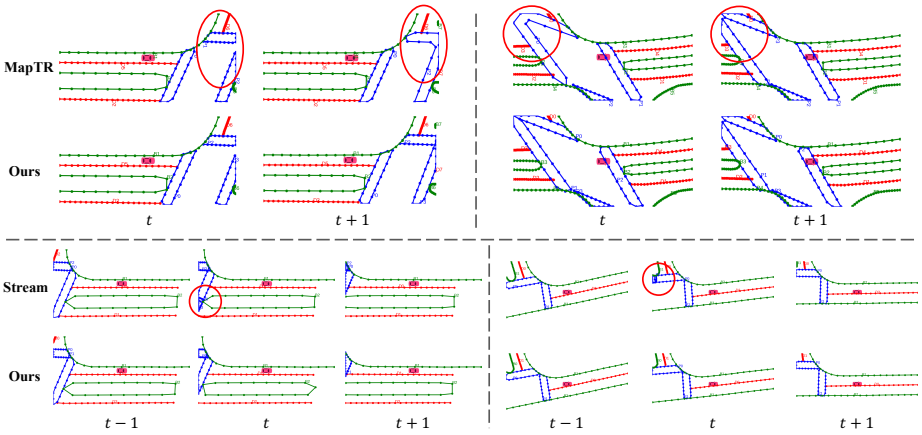

Fig. 1: Typical examples of problematic pedestrian crossing annotations in existing ground truth for nuScenes. (Top) MapTR's ground truth merges or splits nearby pedestrian crossings at the perception boundary, leading to temporal inconsistencies. (Bottom) StreamMapNet's ground truth does not have the above merge/split issue but sometimes fails to fuse small polygons (from raw annotations) into a global one.

and impose more geometric constraints as conditions when merging the polygon pieces, resulting in almost perfect temporal consistency.

In Argoverse2, the raw annotations of each pedestrian crossing are two line segments, making the ground truth processing easier. However, MapTR and StreamMapNet's processing codes sometimes produce open-loop curves at the perception boundary. We fix their codes to always produce closed polygon loops.

Divider (Argoverse2). Figure 2 shows the failure cases of the lane divider class in MapTR and StreamMapNet's ground truth. The examples are from Argoverse2, where the raw annotations are many short divider segments, and a merging algorithm should merge the segments belonging to the same divider instance. MapTR employs a graph-tracing algorithm to connect the line segments, where each segment is a node, and segments of the same instance are connected by tracing from a root to the leaf. However, some annotations are corrupted with incomplete graph information, making the graph tracing algorithm fail completely and miss entire dividers. For StreamMapNet, owing to its unstable threshold-based rules to connect divider segments, it sometimes fails to produce correct long dividers, leading to temporal inconsistency.

To obtain better ground truth, we first fix the graph-tracing algorithm to avoid missing entire dividers. To handle the noisy/corrupted graph information from the annotations, we then borrow the threshold-based rules from StreamMapNet to further connect the dividers produced by the graph tracing algorithm.

B.2 Track extraction algorithm

We show details of the track extraction algorithm in Algorithm 1. This algorithm forms tracks for 1) our consistent ground truth to generate the temporal

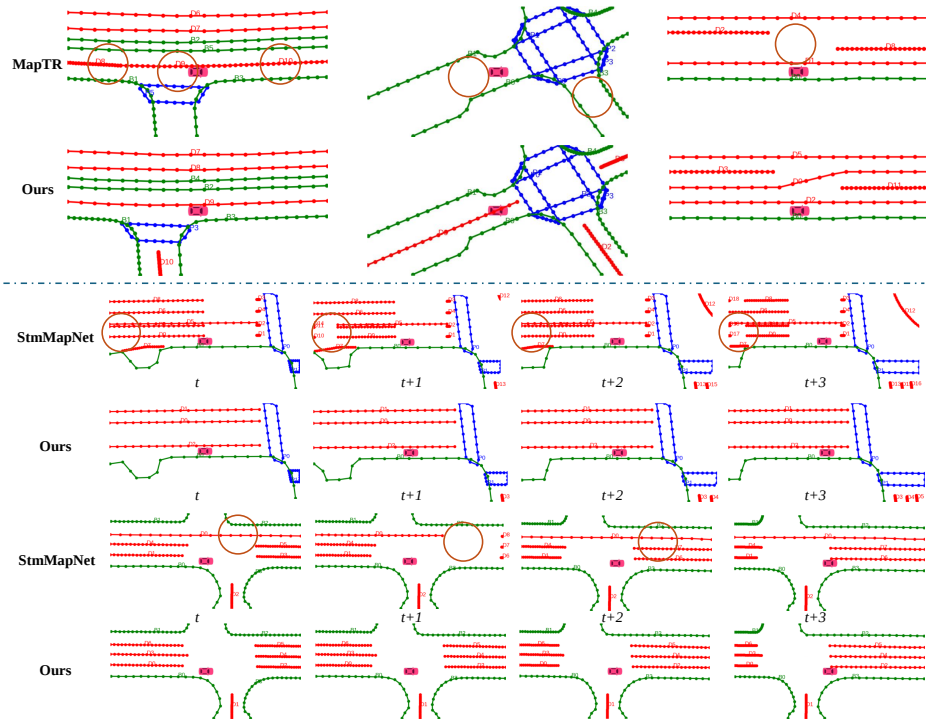

Fig. 2: Typical failure cases in MapTR and StreamMapNet's ground truth dividers on Argoverse2. (Top) MapTR's ground truth fails to properly merge short divider segments into a global one, and the entire dividers can even be missing due to the failure of its graph tracing algorithm. (Bottom) StreamMapNet's ground-truth dividers suffer from temporal inconsistency (split and merge constantly as the car goes).

alignments and 2) the baseline methods to form tracks from per-frame reconstructions. The algorithm has a "look-back" hyper-parameter N, specifying how many previous frames to check when determining the temporal correspondence (*i.e.*, assigning a global ID to an element in the current frame). Larger look-back parameters better tolerate missing reconstructions by re-identification but greatly slow down the entire vector HD mapping pipeline.

The ground-truth track formation uses $N = 1$ (See §4.1 of the main paper). In the main paper, the baselines use $N = 1$ to have similar real-time inference speeds for fair evaluations. A large N improves the C-mAP metric for all the methods but is computationally expensive due to the per-instance rasterization and the bipartite matching. For example, the track extraction algorithm alone is almost 4 times more expensive than the MapTRv2 baseline (with ResNet50) when $N = 5$. As reference, this appendix (§D.1) provides additional results when $N = 3$ or $N = 5$.

Algorithm 1 Track Generation Algorithm

Input: Sequence of predicted vectors $\{V(t), t = 1, ..., T\}$, Sequence of predicted scores $\{p(t), t = 1, ..., T\}$, filter threshold $\tau = 0.4$, look-back frame number N
Output: $\{M_{\text{ID}}(t), t = 1, ...T\}$, $M_{\text{ID}}(t)$ records the global id of positive predictions in $V(t)$
for $t = 1 : T$ **do**
  Init $M_{\text{ID}}(t)$ as an empty mapping
  Obtain a subset of positive vectors $V'(t) = \{V_j(t) \in V(t), p_j(t) > \tau\}$
  if $t = 0$ **then**
    Assign a new global id to each $v'_j(t)$, update $M_{\text{ID}}(t)$
    continue
  end if
  for $k = 1 : N$ **do**
    Transform $V'(t-k)$ to the current frame, $V'_t(t-k)$= Affine$(V'(t-k), P^t_{t-k})$
    Do bipartite matching between $V'(t)$ with $V'_t(t-k)$ using the IoU between rasterized masks, store the optimal bipartite matching as $B_{rec}(k)$
  end for
  for $k = 1 : N$ **do**
    for $v'_j(t)$ in $V'(t)$
      if $v'_j(t)$ doesn't have a global id in $M_{\text{ID}}(t)$, and $v'_j(t)$ in $B_{rec}(k)$ **then**
        Get the global id of $v'_j(t)$'s matched instance in $M_{\text{ID}}(t-k)$,
        Assign this id to $v'_j(t)$, update $M_{\text{ID}}(t)$
      end if
    end for
  end for
  for $v'_j(t)$ in $V'(t)$
    if $v'_j(t)$ doesn't have a global id in $M_{\text{ID}}(t)$ **then**
      Assign a new global id to $v'_j(t)$, update $M_{\text{ID}}(t)$
    end if
  end for
end for

B.3 Consistent-aware mAP

Algorithm 2 presents the algorithmic details for computing our Consistent-aware mAP (C-mAP). Note that the algorithm computes the C-mAP of one distance threshold, while the actual evaluation metrics compute the results of three distances and take the average. The consistency check is at the line 11 of the algorithm. The definition of the C-mAP metric does not include the track extraction algorithm and does not introduce extra hyperparameters. An ideal vector HD mapping method should explicitly predict tracks of reconstruction like MapTracker, instead of relying on an external algorithm for track formation.

The conventional distance-based mAP (reported in the main paper) considers all predictions, including those with low confidence scores, when computing the area under curve to get per-class average precision (AP). However, since the tracks are only defined for positive predictions (negative predictions do not have a "global ID"), the computation of C-mAP excludes negative predictions. Therefore, the value of C-mAP can never reach the conventional mAP, even

Algorithm 2 Consist-aware mAP

Input: Predicted vectors V, GT vectors $\hat{V}$, Predicted Global ID I, GT Global ID $\hat{I}$, Predicted scores P, a Chamfer distance threshold σ (*e.g.*, 0.5m)
Output: The C-mAP on the test set
for each sequence in the test set **do**
  B_{rec} records the matching between predictions and GT across the sequence
  for each timestep t **do**
    Obtain the optimal bipartite matching between $V(t)$ and $\hat{V}(t)$, denoted as $I(t) \rightarrow \hat{I}(t)$, and sort $V(t)$ in descending order based on $P(t)$
    for $v_j(t)$ in $V(t)$
      if $v_j(t)$ has a matched GT vector with Chamfer distance $\leq \sigma$ **then**
        Define $\hat{I}_j$ as the global ID of the GT vector that matches $v_j(t)$
        if $\hat{I}_j$ has existed in B_{rec} **then**
          if its matched prediction ID is not I_j // Consistency check
            Consider $v_j(t)$ as FP (False Positive)
          else
            Consider $v_j(t)$ as TP (True Positive)
          end if
        else
          Consider $v_j(t)$ as TP (True Positive), and update B_{rec}
        end if
      else
        Consider $v_j(t)$ as FP
      end if
      Record the TP/FP for $v_j(t)$, along with its score $p_j(t)$
    end for
    Get TP, FP, and scores for $V(t)$
  end for
  Get TP, FP, and scores for the entire sequence
end for
Sort TP and FP of all sequences with the scores in descending order to calculate the AP, get the consistency-aware AP (C-AP).
The C-mAP is the average C-AP across all classes

with perfectly predicted temporal correspondences. To set up an upper bound for C-mAP when all consistency checks are passed, §D.1 reports the results of "$\overline{\text{C-mAP}}$", which computes C-mAP by ignoring the consistency checks.

C Details of Online Merging Algorithm

Algorithm 3 presents the high-level pseudo-code for our online merging algorithm, which merges per-frame reconstructions into a global vector HD map. Note that our merging algorithm is simple and not perfect, and may occasionally fail to accurately merge the per-frame data. However, the main goal of implementing this merging algorithm is to investigate and analyze the consistency of per-frame reconstructions. More advanced algorithms can be designed and implemented if we need high-quality merged global maps.

Algorithm 3 Online Merging Algorithm

Input: Predicted set $Y(t)$ for each timestep t;
Dictionary $D[I]$ records the merged vectors, I represents the global ID of the predicted vectors
for each timestep t **do**
  for each pair (V_i, c_i) in $Y(t)$ **do**
    $I_i \leftarrow$ Global ID of V_i
    if I_i is not in D **then**
      $D[I_i] = V_i$
      continue
    else
      if c_i == Pedestrian Crossing **then**
        $D[I_i] =$ `MergeCrossing`$(D[I_i], V_i)$ // Merge crossing by finding the convex hull that contains all points in $D[I_i]$ and V_i
      else if c_i == Lane Divider **then**
        $D[I_i] =$ `MergeDivider`$(D[I_i], V_i)$ // Merge divider by interpolating $D[I_i]$ and V_i
      else if c_i == Road Boundary **then**
        $D[I_i] =$ `MergeBoundary`$(D[I_i], V_i)$ // Merge boundary by interpolating $D[I_i]$ and V_i
      end if
    end if
  end for
end for

D Additional Experimental Results and Analyses

This section presents additional experimental results and analyses, complementing §5 of the main paper.

D.1 Full C-mAP results

As discussed in §B.2, the track extraction algorithm is more robust to temporal inconsistency in the reconstructions when using higher look-back parameters. Table 1 and Table 2 in this appendix extend the Table 1 and Table 2 of the main paper by providing C-mAP results with different look-back parameters. The first row of each method is the same as the results reported in the main paper, and MapTracker directly uses the predicted track. §D.3 contains qualitative results with different look-back parameters. We analyze the results of the three methods below.

MapTRv2 gets huge boosts on C-mAP with increased look-back parameters, especially on the nuScenes dataset. This suggests that MapTRv2 suffers from poor temporal consistency, and the predicted road elements frequently disappear and reappear within 2 or 3 consecutive frames.

StreamMapNet also benefits from higher look-back parameters. Note that we tried to derive the tracks from StreamMapNet's hidden query propagation but found almost no temporal correspondence – very few propagated elements stay

Table 1: Full C-mAP on nuScenes [1].†: Epochs for our multi-frame training.

Method	Epoch	Lookback	C-AP$_p$	C-AP$_d$	C-AP$_b$	C-mAP	$\overline{\text{C-mAP}}$
MapTRv2	110	1 frame	55.8	43.8	57.9	50.5	64.9
		3 frames	61.5	54.0	61.3	58.9	
		5 frames	62.5	55.4	62.2	60.0	
StreamMapNet	110	1 frame	58.6	53.5	57.1	56.4	65.9
		3 frames	62.8	59.7	58.9	60.5	
		5 frames	63.1	60.8	59.3	61.0	
MapTracker	72†	∅ (predicted)	75.4	65.0	66.9	69.1	72.5
		1 frame	75.5	65.9	67.6	69.7	
		3 frames	76.3	66.8	68.2	70.4	
		5 frames	76.9	67.0	68.4	70.7	

Table 2: Full C-mAP on Argoverse2 [3]. †: Epochs for our multi-frame training.

Method	Epoch	Lookback	C-AP$_p$	C-AP$_d$	C-AP$_b$	C-mAP	$\overline{\text{C-mAP}}$
MapTRv2	24*4	1 frame	58.4	52.5	57.4	56.1	67.7
		3 frames	63.2	62.1	62.8	62.7	
		5 frames	63.9	64.1	63.0	63.7	
StreamMapNet	72	1 frame	63.0	53.3	56.2	57.5	65.8
		3 frames	65.5	58.8	58.9	61.0	
		5 frames	65.6	59.5	59.0	61.4	
MapTracker	35†	∅ (predicted)	70.8	68.3	66.0	68.3	72.8
		1 frame	72.6	69.5	67.4	69.8	
		3 frames	73.2	71.0	68.3	70.8	
		5 frames	73.4	71.3	68.3	71.0	

positive in the next frame. This is mainly because the detection-based formulation cannot exploit tracking labels, and the model treats the propagated information as extra conditions without capturing explicit temporal relationships. StreamMapNet's C-mAP results are worse than MapTRv2 on Argoverse2, further indicating the limitations of its temporal modeling designs.

MapTracker predicts tracks and obtains good results without the track extraction algorithm. Note that the track extraction algorithm and our VEC module use different thresholds for determining positive road elements. Our VEC module uses higher thresholds and output fewer positive elements, leading to slightly lower C-mAP compared to the result of using the track extraction algorithm with $N = 1$. When increasing the look-back parameters, the C-mAP of MapTracker can also keep improving – Although MapTracker obtains much more consistent reconstructions than the baselines, the predicted temporal correspondences are sometimes incorrect, and road elements are still occasionally unstable (*i.e.*, disappear and reappear within several frames).

As explained in §B.3, $\overline{\text{C-mAP}}$ is the upper bound of C-mAP when the reconstructions pass all consistency checks. MapTracker gets very close to the $\overline{\text{C-mAP}}$ when we use the track extraction algorithm with $N = 5$ to trade running time for track quality – The gap (1.8) is much smaller than the gap of the baselines, again demonstrating our superior consistency.

D.2 Check MapTR ground truth

Table 3 investigates the quality of MapTR's ground truth by evaluating its data on our consistent benchmarks. The tracks of MapTR's ground truth are extracted using Algorithm 1 with look-back parameter 1. The results in the table are consistent with what we have analyzed in §B.1: 1) the pedestrian crossings of MapTR data suffer from temporal inconsistencies on both datasets; 2) the dividers have severe issues (*e.g.*, entirely dropped) on Argoverse2.

Table 3: Evaluating MapTR's ground-truth data with our consistent benchmarks. The goal is to understand the temporal consistency of MapTR's ground truth.

	C-AP$_p$	C-AP$_d$	C-AP$_b$	C-mAP
nuScenes	80.6	100	100	93.5
Argoverse2	95.6	65.7	100	87.1

According to the table, we observe failure in Pedestrian Crossing for both datasets and severe misalignment in Argoverse2, which matches the conclusion of our observation mentioned in the main paper.

D.3 More qualitative results

Figure 3 to Figure 8 present additional qualitative comparisons. We show two additional rows for each example compared to Figure 4 of the main paper: **(the second row, "Merged LB5"** is the merged results for all the methods when using the track extraction algorithm with look-back parameter $N = 5$; **(the third row, "Unmerged")** is the raw unmerged results that simply place the reconstructions of all frames together. For the first row, MapTracker uses the predicted tracks while the baselines use the track extraction algorithm with look-back parameter $N = 1$.

Note that since our VEC module and the track extraction algorithm use different thresholds for determining the positive predictions, MapTracker's results in the first and second rows of each example can be slightly different. The second row of MapTracker sometimes has more positive road elements than the first row, and we use the predicted tracks to plot the unmerged results. Comparing each row of each example, MapTracker's results are cleaner and more consistent, further validating our contributions toward consistent vector HD mapping.

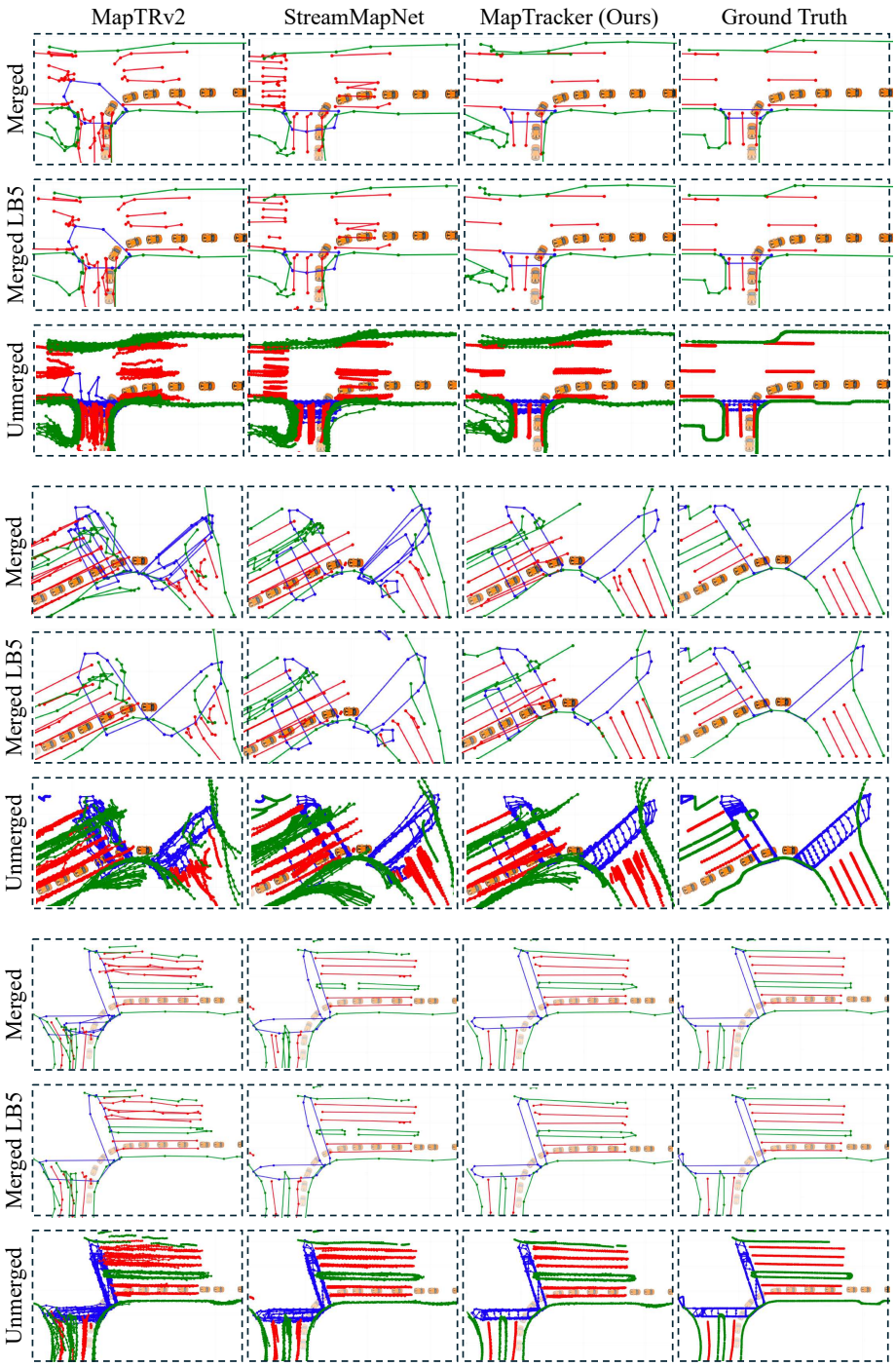

Fig. 3: Additional qualitative results on the nuScenes dataset.

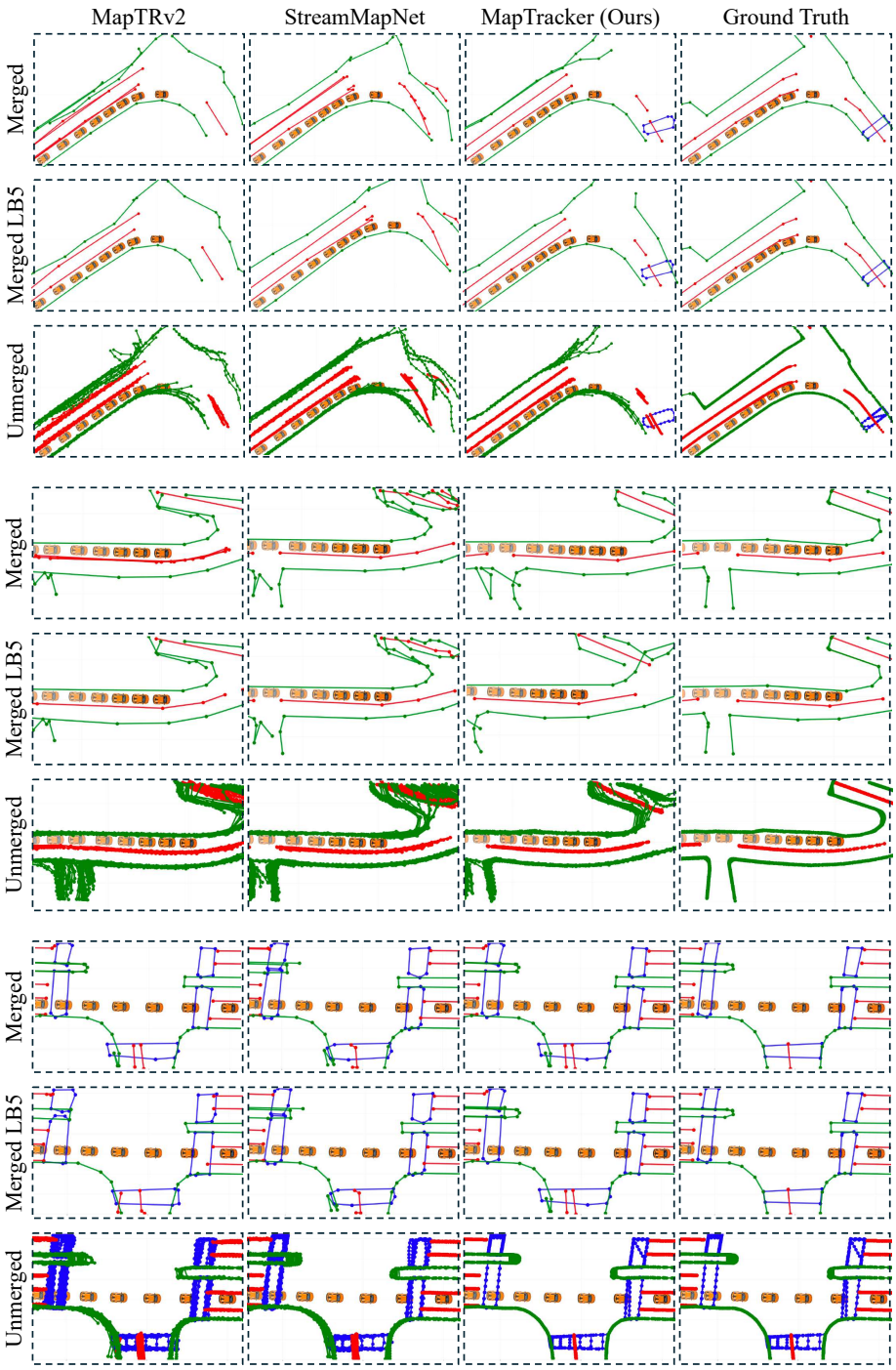

Fig. 4: Additional qualitative results on the nuScenes dataset.

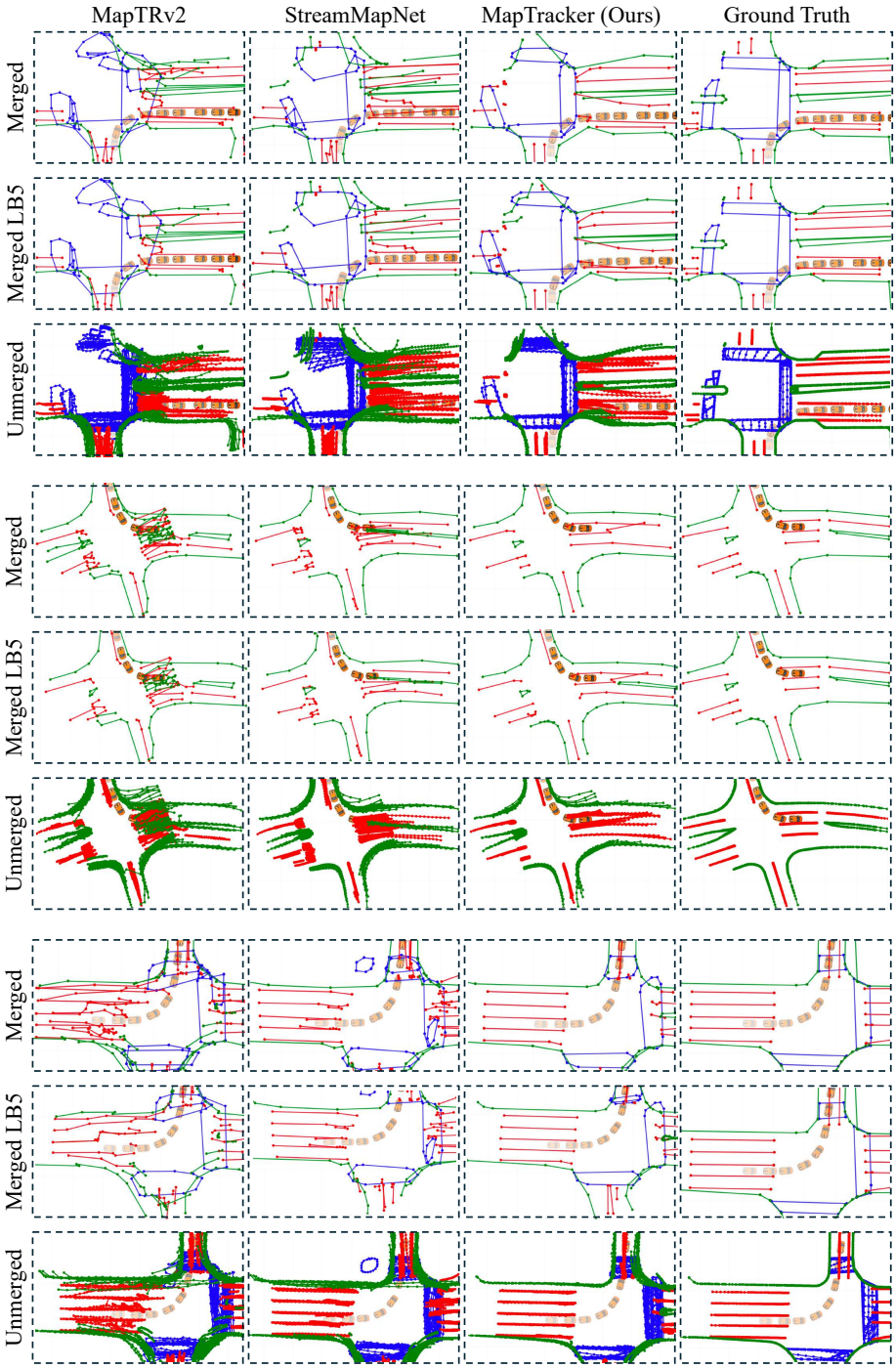

Fig. 5: Additional qualitative results on the nuScenes dataset.

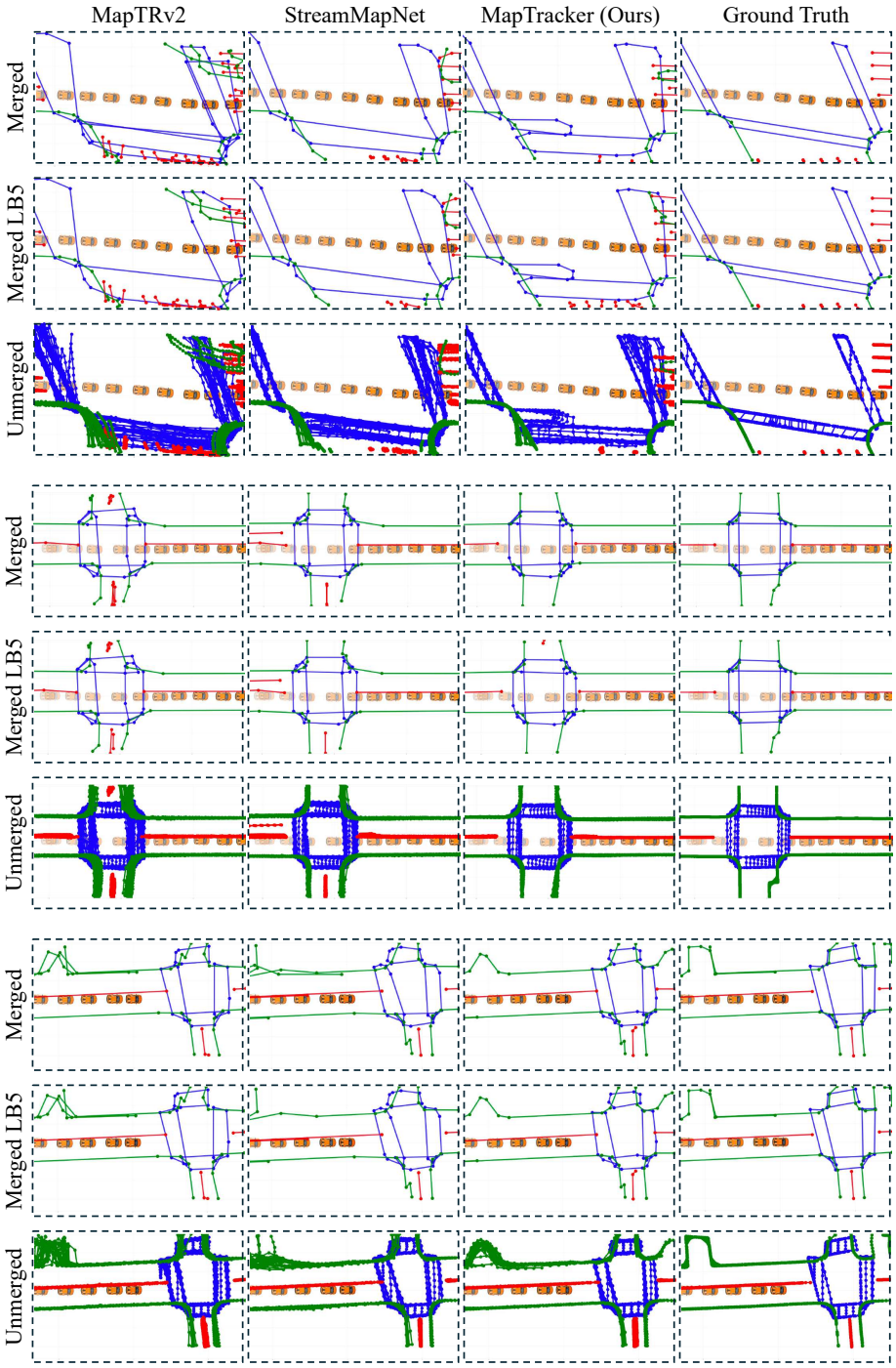

Fig. 6: Additional qualitative results on the Argoverse2 dataset.

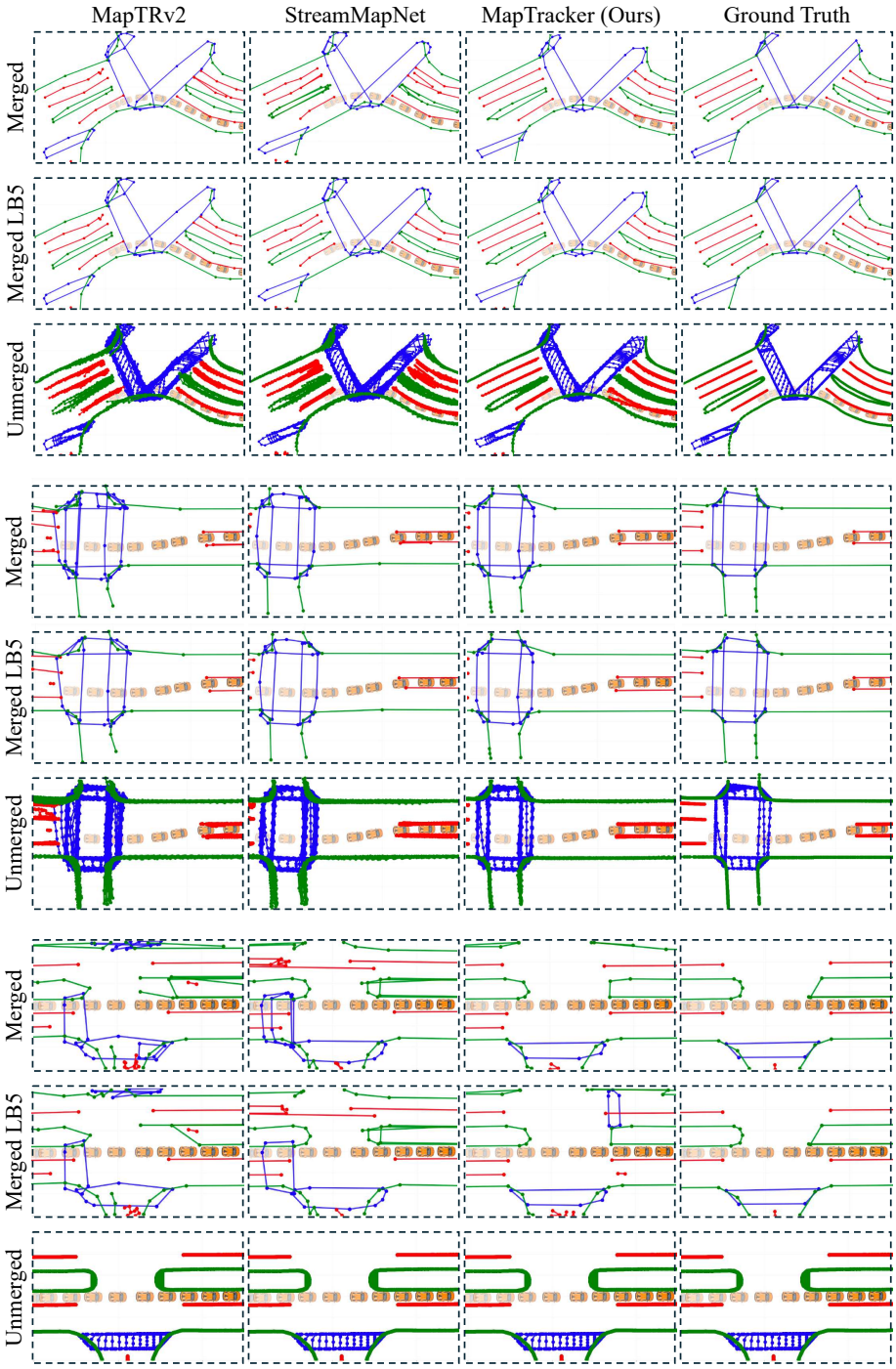

Fig. 7: Additional qualitative results on the Argoverse2 dataset.

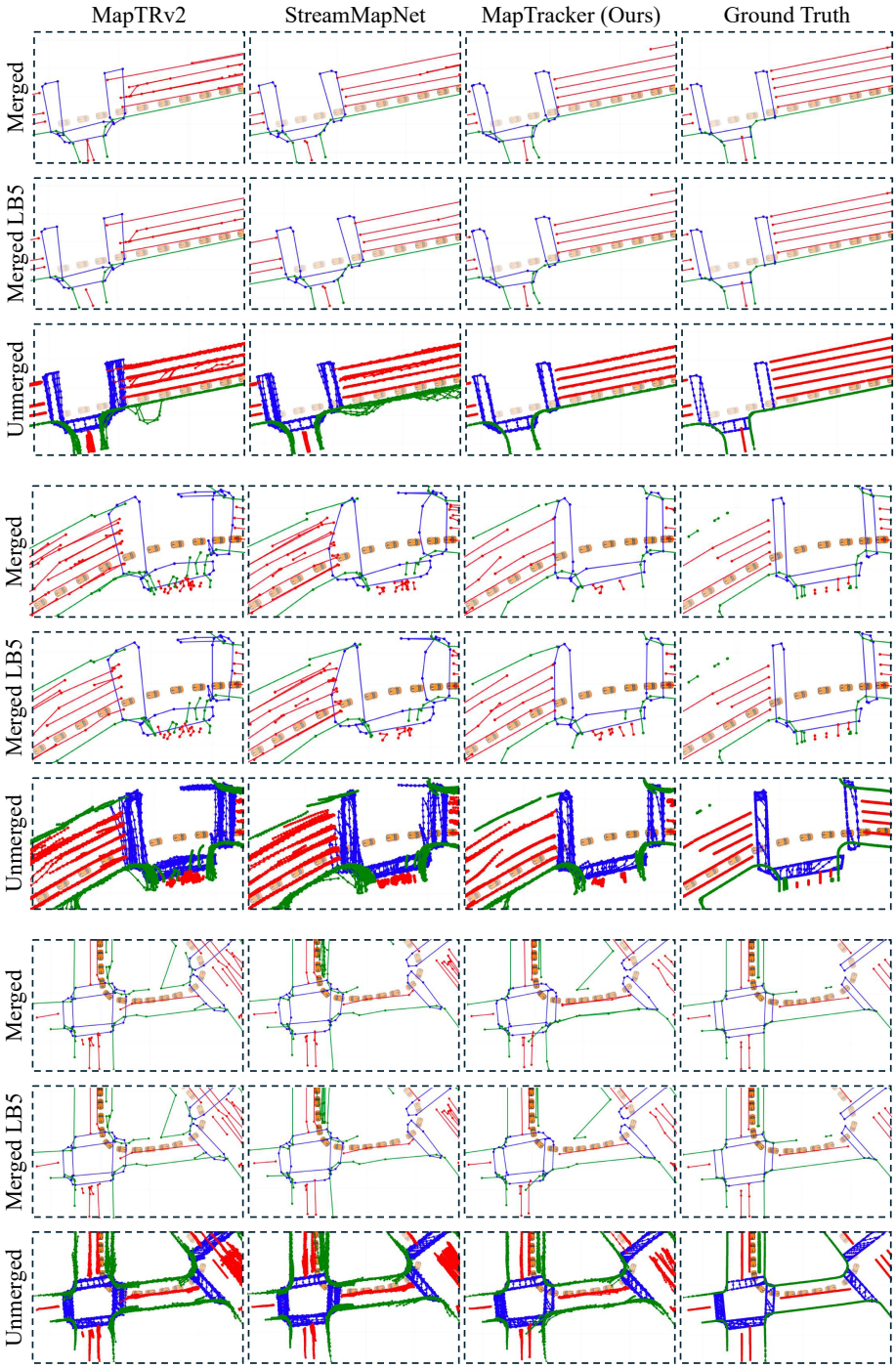

Fig. 8: Additional qualitative results on the Argoverse2 dataset.

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020)
2. Liao, B., Chen, S., Zhang, Y., Jiang, B., Zhang, Q., Liu, W., Huang, C., Wang, X.: Maptrv2: An end-to-end framework for online vectorized hd map construction. arXiv preprint arXiv:2308.05736 (2023)
3. Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J.K., et al.: Argoverse 2: Next generation datasets for self-driving perception and forecasting. arXiv preprint arXiv:2301.00493 (2023)
4. Yuan, T., Liu, Y., Wang, Y., Wang, Y., Zhao, H.: Streammapnet: Streaming mapping network for vectorized online hd map construction. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 7356–7365 (2024)