# TileTracker: Tracking Based Vector HD Mapping using Top-Down Road Images

Mohammad Mahdavian
*Department of Computing Sciences*
*Simon Fraser University*
Burnaby, Canada
mmahdavi@sfu.ca

Mo Chen
*Department of Computing Sciences*
*Simon Fraser University*
Burnaby, Canada
mochen@cs.sfu.ca

Yu Zhang
*NVIDIA Corporation*
Santa Clara, United States of America
yuzh@nvidia.com

***Abstract*—In this paper, we propose a tracking-based HD mapping algorithm for top-down road images, referred to as tile images. While HD maps traditionally rely on perspective camera images, our approach shows that tile images can also be effectively utilized, offering valuable contributions to this research area as it can be start of a new path in HD mapping algorithms. We modified the BEVFormer layers to generate BEV masks from tile images, which are then used by the model to generate divider and boundary lines. Our model was tested with both color and intensity images, and we present quantitative and qualitative results to demonstrate its performance.**

***Index Terms*—HD mapping, top-down images, tracking**

## I. Introduction

Autonomous vehicles rely on accurate environmental data from various sensors, including cameras, lidar, radar, IMUs, and GPS. These sensors work together, with their data fused to generate precise commands for vehicle navigation [1].

Some methods estimate waypoints and navigational commands directly from sensor data in an end-to-end manner [2]–[4]. Large Visual Models (LVMs) derive actions from camera images [5], while Reinforcement Learning (RL) optimizes navigation based on reward signals [6].

High Definition (HD) maps serve as a valuable and detailed resource for navigating autonomous vehicles. To create these maps for various areas and cities, vehicles equipped with multiple sensors, such as cameras, lidar, and IMUs, traverse the roads, capturing necessary data. Human annotators then identify fixed road features like lane dividers, traffic lights, pedestrian crossings, and road boundaries. Following this, deep learning models have been employed to learn and estimate these road features for inclusion in the HD map based on the sensor data [7]–[10]. One well-known method for HD mapping is MapTR [7], [8], which offers an end-to-end, transformer-based [11], unified permutation-equivalent modeling approach. This method models each map element as a point set with a group of equivalent permutations, enabling accurate shape descriptions and stabilizing the learning process. However, MapTR generates road features on a frame-by-frame basis, which can result in noisy and inconsistent HD maps. To address these issues, StreamMapNet [9] utilizes long-sequence temporal modeling of video data. It employs multi-point attention and temporal information to produce large-scale local HD maps with enhanced stability. Additionally, MapTracker [10] approaches the mapping problem as a tracking task, leveraging a memory of latents to ensure consistent reconstructions over time. This method adds a sensor stream into memory buffers of two latent representations: raster latents in bird's-eye-view (BEV) space and vector latents over road elements.

In these types of HD mapping methods, local HD maps are typically generated from perspective view (PV) images. This approach can limit model performance due to the need for PV-to-BEV conversion. In this work, we address this issue by developing HD maps based on MapTracker using top-down road images, known as tile images. We utilize NVIDIA's road dataset to generate sequences of these top-down tile images. Additionally, we modify MapTracker to create BEV segmentation directly from the tile images, enabling the algorithm to generate vectors for local HD maps. These local maps are then integrated to produce comprehensive global HD maps.

This project was completed as part of an internship program at NVIDIA Corporation

## II. Methodology

In this section, we explain our algorithm. As mentioned before, our approach is derived from MapTracker [10] and we have made modifications on this method to be able to generate local HD maps from top-down tile images.

This method employs a robust memory mechanism to accumulate sensor data into two distinct memory representations. The first is a top-down BEV memory of the vehicle's surrounding area, and the second is a vector memory for road elements. At each frame, the algorithm selects a subset of past latent memories for fusion. The vector memory mechanism also uses tracking methods to maintain a sequence of memory latents associated with each road element. In the BEV module, the BEV memory buffer $M_{BEV}(t-1)$ generates the initial BEV features for each frame. Then, a deformable self-attention mechanism enhances the BEV memory features. In the next step, for MapTracker, a spatial deformable cross-attention layer, adapted from BEVFormer [12], converts perspective view image features $I(t)$ into BEV features $M_{BEV}(t)$.

We have modified the BEVFormer layer to generate BEV features from top-down tile images by utilizing top-down camera intrinsic and extrinsic matrices. In general, the BEVFormer

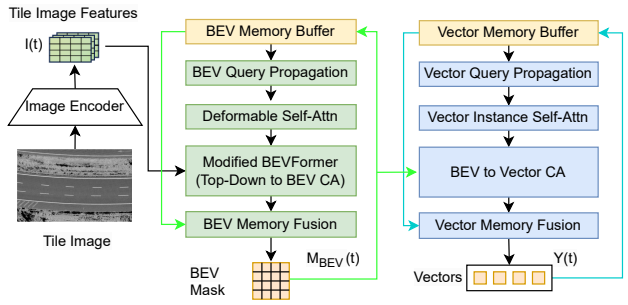


Fig. 1: Main TileTracker model structure. Green and blue boxes show BEV and vector modules, respectively.

processes six camera image features along with their respective camera matrices and related information. Deformable attention [13] is used to integrate the image features into the corresponding areas of the BEV mask. In our approach, the extrinsic camera matrix is constructed by combining the vehicle's translation and rotation matrices, followed by an additional downward rotation to obtain the final extrinsic matrix. In this way, the deformable attention mechanism integrates the top-down tile image features with the entire BEV mask, leading to efficient generation of the BEV mask.

In the next layer, latent features from previous frames that are stored in a buffer get fused with the current frame's features to create the final BEV mask, $M_{BEV}(t)$. The algorithm uses a strided selection of four memories, corresponding to various vehicle positions relative to the current position, for the fusion.

For vector predictions, a transformer-based tracking approach is used to propagate vector queries, combined with new learnable element candidates to generate vector features for new frames. Subsequently, a standard self-attention mechanism, along with BEV-to-Vector cross-attention and vector memory fusion, is employed to estimate vectors, $Y(t)$, containing vector points $p$ for the next frame.

## III. Training

The road elements defined in our dataset include divider and boundary lines. We extract top-down tile images with the size of $(H, W)$ that includes intensity and color images as well as the ground truth (GT) for each line found in the image. We train two separate model with using intensity and color images as our model inputs to investigate their performance. Additionally, we divide all the line types (solid, double solid, dashed, etc.) into two main categories, divider and boundary lines. Each road element in the GT is denoted as $\hat{Y} = (\hat{V}, \hat{c})$ where $\hat{V}$ is 20 points interpolated from the raw vector and $\hat{c}$ specifies the line type. Also, the lines are rasterized into a segmentation image using OpenCV and PIL libraries to generate GT for the BEV mask, $\hat{S}$.

### A. Loss Functions

The loss functions used for training our model, focus on the BEV mask and vector generation. For the BEV masks, we use per-pixel Focal loss [14] and per-class Dice loss [15] between the predictions and the GT defined by

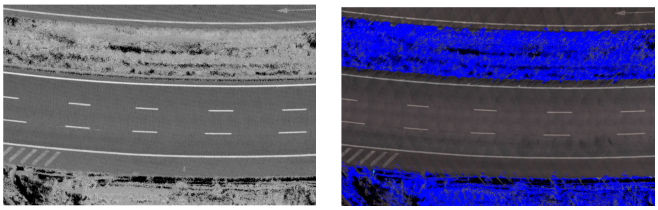

Fig. 2: An example of intensity (left) and color (right) images is shown. These images are generated from 3D colored lidar point clouds and displayed as 2D top-down BEV images, with intensity represented in grayscale and color images in their original colors.

$$\mathcal{L}_{\text{BEV}} = \lambda_1 \mathcal{L}_{\text{focal}}(S(t), \hat{S}(t)) + \lambda_2 \mathcal{L}_{\text{dice}}(S(t), \hat{S}(t)) \quad (1)$$

For the vector generation, we use a tracking style loss which consists of a focal loss, $\mathcal{L}_{\text{focal}}$, and a permutation-invariant line coordinate loss, $\mathcal{L}_{\text{line}}$. Their combination defines $\mathcal{L}_{\text{track}}$ as

$$\mathcal{L}_{\text{track}} = \lambda_3 \mathcal{L}_{\text{focal}}(p, \hat{c}) + \lambda_4 \mathcal{L}_{\text{line}}(V(t), \hat{V}(t)) \quad (2)$$

We also borrow the transformation loss, $\mathcal{L}_{\text{trans}}$, from StreamMapNet to enforce the query transformation to maintain the vector shape and line type. Therefore the complete loss function would become

$$\mathcal{L} = \mathcal{L}_{\text{BEV}} + \mathcal{L}_{\text{track}} + \lambda_5 \mathcal{L}_{\text{trans}} \quad (3)$$

### B. Model Training Details

The training has three main stages. 1) Pre-training the image backbone and BEV encoder to generate BEV masks, 2) Warming up the vector decoders while keeping the remaining model components frozen, 3) Training the complete model structure.

The loss weights are set as follow: $\lambda_1 = 10.0$, $\lambda_2 = 1.0$, $\lambda_2 = 5.0$, $\lambda_2 = 50.0$ and $\lambda_5 = 0.1$. We use AdamW as our optimizer and initial learning rate of 5e-4 which is decreased to 1.5e-6 using a cosine learning rate scheduler.

## IV. Experiments

We train our model using 8 NVIDIA V100 GPUs and the three stages need 12, 4 and 24 epochs, respectively. Also the batch size of 4 is considered for all training stages. In this section, we explain our dataset generation and preprocessing.

### A. Dataset

We use data collected from various roads and tracks to train our model. This dataset includes top-down tile intensity and color images covering a 30×60 meters area, GT for different divider line types and boundary lines within the images, and the car's GPS location. In the data generation process, a 3D colored LiDAR system was employed to capture track data. The resulting color and intensity images provide 2D top-down views, displaying colored and grayscale representations of the captured points. A sample of them is provided in Fig. 2.

It is crucial to preprocess the dataset to ensure it is suitable for model training. The input data must be continuous with

TABLE I: Our model's (TileTracker) performance compared with our baseline (StreamTileNet) over different inputs (intensity and color images) and output categories (divider and boundary lines). The AP is measured for different thresholds and their average.

| Input | Category | Model | AP | | | Ave |
|---|---|---|---|---|---|---|
| | | | @0.5m | @1.0m | @1.5m | AP |
| Intensity | Divider | TileTracker | 11.64 | 19.60 | 26.12 | 19.12 |
| | | StreamTileNet | 11.95 | 18.37 | 23.73 | 18.01 |
| | Boundary | TileTracker | 2.84 | 8.74 | 15.68 | 9.09 |
| | | StreamTileNet | 1.65 | 6.62 | 13.09 | 7.12 |
| Color | Divider | TileTracker | 8.61 | 15.71 | 21.76 | 15.36 |
| | | StreamTileNet | 7.57 | 14.03 | 20.29 | 13.96 |
| | Boundary | TileTracker | 2.84 | 8.74 | 15.68 | 9.09 |
| | | StreamTileNet | 1.79 | 6.00 | 11.90 | 6.56 |

reasonable frame differences, as our tracking algorithm relies on finding similarities between consecutive frames. Consequently, the tracks are generated so that two consecutive frames have a minimum distance of 1 meter, though this is not always guaranteed. In some instances, the distance between frames may exceed 5 meters, in which case we split them into separate tracks. We only consider tracks with at least 20 frames and limit each track to a maximum of 500 frames. This preprocessing results in a total of 48K samples, with 40K samples allocated for training and 8K samples for testing.

### *B. Baseline*

As our baseline, we train the popular temporal method, StreamMapNet, after applying the same modifications used in BEVFormer to enable training with tile images. This modified version of the model is referred to as StreamTileNet.

### *C. Quantitative Results*

We present quantitative results for our method and the baseline. Table I shows the Average Precision (AP) across thresholds (0.5, 1, and 1.5 meters) for different input types (intensity and color images) and output categories (divider and boundary lines). TileTracker consistently outperforms StreamTileNet, demonstrating superior tracking. Intensity images yield better accuracy for divider lines, while differences for boundary lines are minimal, suggesting intensity images are more effective due to higher contrast. Despite using a larger dataset than nuScenes [16] (40K compared to 28K), our results indicate that additional data and training could further improve performance due to complicated nature of our data.

### *D. Qualitative Results*

Qualitative results are presented in two formats: unmerged and merged. In the unmerged format, the model outputs are depicted as 20 individual vector points, plotted sequentially according to the vehicle's pose. In the merged format, the points are connected based on overlapping vectors, resulting in continuous lines. As shown in Fig. 3 to Fig. 6, the model generally performs well. However, its performance seems better on straights and curved paths compared to sharp turns. This discrepancy is likely due to the predominance of straight and slight curved paths in our dataset. By incorporating more complex turns and curves, we can further improve the model's performance across a wider range of path types.

## V. Conclusion

In this paper, we developed a tracking-based HD mapping algorithm for top-down tile images. Building on MapTracker, we modified the BEVFormer layer to generate BEV masks from the tile images. The model was tested using both intensity and color images, with intensity images yielding better results due to the higher contrast between the lines and the background. While the model demonstrated reasonable performance, incorporating more complex paths into the dataset could further improve its effectiveness, which we leave as future work.

## References


[1] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, "Sensor and sensor fusion technology in autonomous vehicles: A review," *Sensors*, vol. 21, no. 6, p. 2140, 2021.

[2] P. Agand, M. Mahdavian, M. Savva, and M. Chen, "Letfuser: Lightweight end-to-end transformer-based sensor fusion for autonomous driving with multi-task learning," *arXiv preprint arXiv:2310.13135*, 2023.

[3] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *Pattern Analysis and Machine Intelligence (PAMI)*, 2022.

[4] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-enhanced autonomous driving using interpretable sensor fusion transformer," in *Conference on Robot Learning*. PMLR, 2023, pp. 726–737.

[5] L. Chen, O. Sinavski, J. Hünermann, A. Karnsund, A. J. Willmott, D. Birch, D. Maund, and J. Shotton, "Driving with llms: Fusing object-level vector modality for explainable autonomous driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 14 093–14 100.

[6] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, J. Araluce, and L. M. Bergasa, "Reinforcement learning-based autonomous driving at intersections in carla simulator," *Sensors*, vol. 22, no. 21, p. 8373, 2022.

[7] B. Liao, S. Chen, X. Wang, T. Cheng, Q. Zhang, W. Liu, and C. Huang, "Maptr: Structured modeling and learning for online vectorized hd map construction," *arXiv preprint arXiv:2208.14437*, 2022.

[8] B. Liao, S. Chen, Y. Zhang, B. Jiang, Q. Zhang, W. Liu, C. Huang, and X. Wang, "Maptrv2: An end-to-end framework for online vectorized hd map construction," *arXiv preprint arXiv:2308.05736*, 2023.

[9] T. Yuan, Y. Liu, Y. Wang, Y. Wang, and H. Zhao, "Streammapnet: Streaming mapping network for vectorized online hd map construction," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 7356–7365.

[10] J. Chen, Y. Wu, J. Tan, H. Ma, and Y. Furukawa, "Maptracker: Tracking with strided memory fusion for consistent vector hd mapping," *arXiv preprint arXiv:2403.15951*, 2024.

[11] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, 2017.

[12] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *European conference on computer vision*. Springer, 2022, pp. 1–18.

[13] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[14] T. Lin, "Focal loss for dense object detection," *arXiv preprint arXiv:1708.02002*, 2017.

[15] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*. Ieee, 2016, pp. 565–571.

[16] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

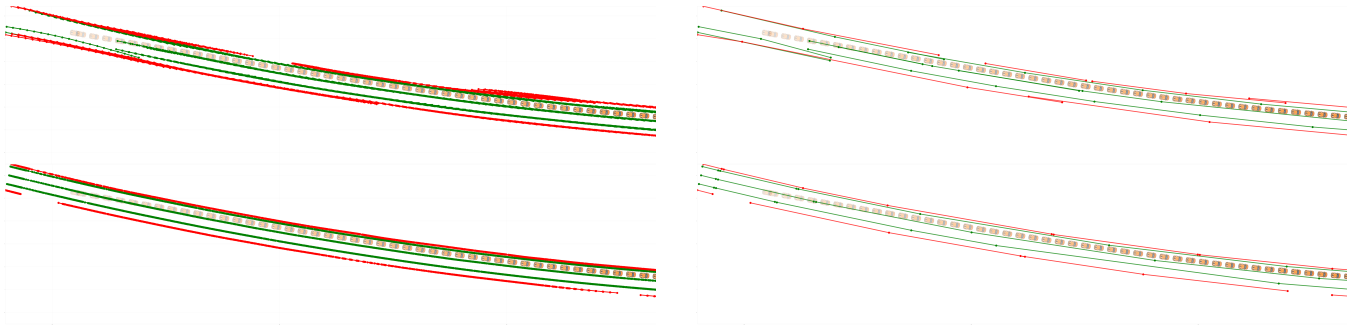

Fig. 3: Unmerged and merged predictions (top left and right) as well as Unmerged and merged GTs (bottom left and right) for a curved path

Fig. 4: Unmerged and merged predictions (top left and right) as well as Unmerged and merged GTs (bottom left and right) for a path containing straight paths and turns

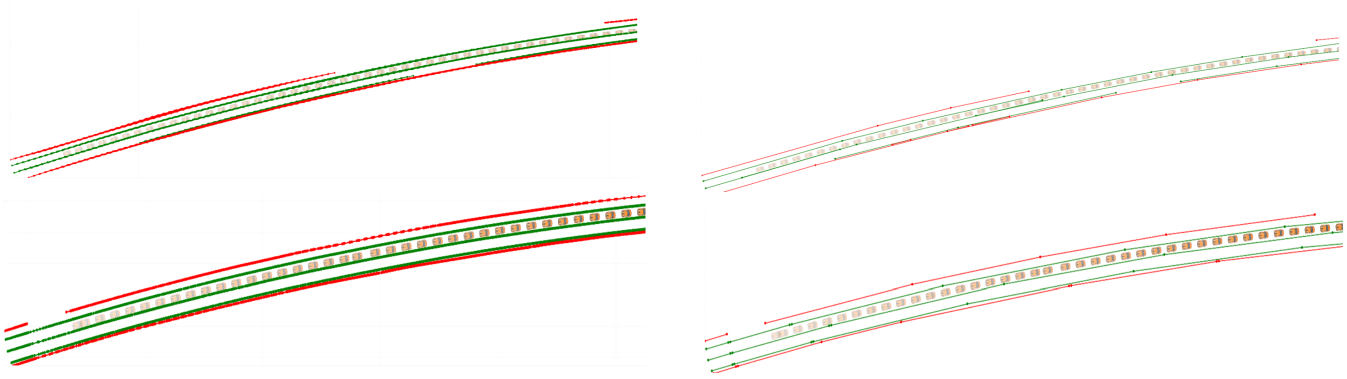

Fig. 5: Unmerged and merged predictions (top left and right) as well as Unmerged and merged GTs (bottom left and right) for a curved path

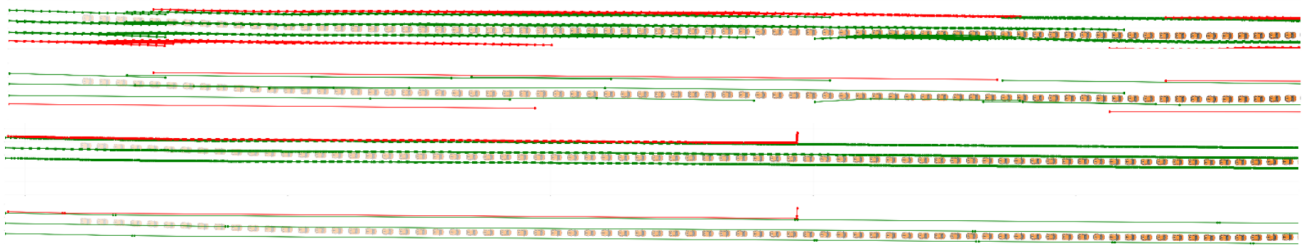

Fig. 6: Unmerged and merged predictions (two top figures) as well as Unmerged and merged GTs (two bottom figures) for a straight path