Lane Graph as Path: Continuity-preserving Path-wise Modeling for Online Lane Graph Construction

Bencheng Liao[1,2,⋆], Shaoyu Chen[2,⋆], Bo Jiang[2], Tianheng Cheng[2], Qian Zhang[3], Wenyu Liu[2], Chang Huang[3], and Xinggang Wang[2,✉]

[1] Institute of Artificial Intelligence, Huazhong University of Science & Technology
[2] School of EIC, Huazhong University of Science & Technology
[3] Horizon Robotics

Abstract. Online lane graph construction is a promising but challenging task in autonomous driving. Previous methods usually model the lane graph at the pixel or piece level, and recover the lane graph by pixel-wise or piece-wise connection, which breaks down the continuity of the lane and results in suboptimal performance. Human drivers focus on and drive along the continuous and complete paths instead of considering lane pieces. Autonomous vehicles also require path-specific guidance from lane graph for trajectory planning. We argue that the path, which indicates the traffic flow, is the primitive of the lane graph. Motivated by this, we propose to model the lane graph in a novel path-wise manner, which well preserves the continuity of the lane and encodes traffic information for planning. We present a path-based online lane graph construction method, termed LaneGAP, which end-to-end learns the path and recovers the lane graph via a Path2Graph algorithm. We qualitatively and quantitatively demonstrate the superior accuracy and efficiency of LaneGAP over conventional pixel-based and piece-based methods on the challenging nuScenes and Argoverse2 datasets under controllable and fair conditions. Compared to the recent state-of-the-art piece-wise method TopoNet on the OpenLane-V2 dataset, LaneGAP still outperforms by 1.6 mIoU, further validating the effectiveness of path-wise modeling. Abundant visualizations in the supplementary material show LaneGAP can cope with diverse traffic conditions. Code is released at `https://github.com/hustvl/LaneGAP`.

Keywords: Autonomous Driving · Online Lane Graph Construction · Path-wise Modeling

1 Introduction

The lane graph contains detailed lane-level traffic information, and serves to provide path-specific guidance for trajectory planning, *i.e.*, an automated vehicle can trace a path from the lane graph as reliable planning prior.

⋆ Equal contribution; ✉ Corresponding author: `xgwang@hust.edu.cn`

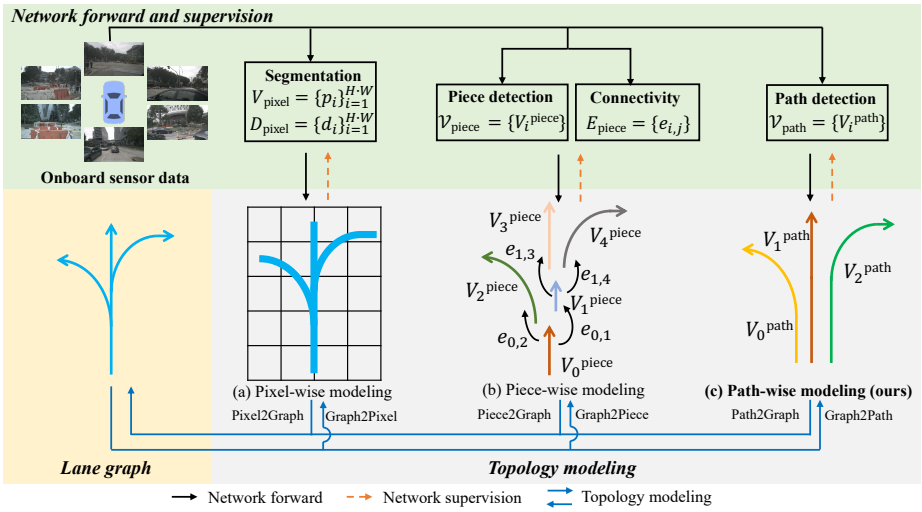

Fig. 1: Modeling comparison. (a) Pixel-wise modeling [22] utilizes a predefined Graph2Pixel algorithm to rasterize the lane graph into a segmentation map and a direction map on dense BEV pixels, and heuristic Pixel2Graph post-processing is needed to recover the lane graph from the predicted segmentation map V_{pixel} and direction map D_{pixel} (direction map is not drawn here for simplicity). **(b)** Piece-wise modeling [6] utilizes a predefined Graph2Piece algorithm to split the lane graph into a set of pieces and the connectivity matrix among pieces, and then it merges the predicted pieces $\mathcal{V}_{\text{piece}}$ to the graph with the Piece2Graph algorithm based on predicted connectivity E_{piece}. **(c)** The proposed path-wise modeling translates the lane graph into complete paths with a predefined Graph2Path algorithm to traverse the graph. We perform path detection and adopt a Path2Graph algorithm to recover the lane graph.

Lane graph is traditionally constructed with an offline map generation pipeline. However, autonomous driving demands a high degree of freshness in lane topology. Hence, online lane graph construction with vehicle-mounted sensors (*e.g.*, cameras and LiDAR) is of great application value.

An intuitive solution is to model the lane graph in a pixel-wise manner and adopt a segmentation-then-vectorization paradigm. For example, HDMapNet [22] predicts a segmentation map and a direction map on dense bird's-eye-view (BEV) features. It then extracts the skeleton from the coarse segmentation map with a morphological thinning algorithm and extracts the graph topology by greedily tracing the single-pixel-width skeleton with the predicted direction map. Pixel-wise modeling incurs heuristic and time-consuming post-processing and often fails in complicated topology (see Fig. 2 row 3, Tab. 10).

Lane graph is also modeled in a piece-wise manner [6], which splits the lane graph into lane pieces at junction points (*i.e.*, merging points and fork points) and predicts an inter-piece connectivity matrix. Based on the connectivity, pieces are linked and merged into the lane graph. However, piece-wise modeling breaks down the continuity of the lane. At complicated road intersections, the frag-

mented short pieces are hard for a neural network to learn to detect (*e.g.*, V_1^{piece} in Fig. 1 (b)), let alone predict the connections.

We argue that the path is the primitive of the lane graph. Human drivers focus on and drive along the continuous and complete paths instead of considering lane pieces. Autonomous vehicles also require path-specific guidance from lane graph for trajectory planning [12]. Continuous paths play an important role in indicating the driving flow, while pixel-wise and piece-wise modeling often fail to merge pixels and pieces into continuous paths (see Fig. 2).

Motivated by this, we propose to model the lane graph in an alternative path-wise manner. We decouple the lane graph into a set of continuous paths with a proposed Graph2Path algorithm, perform path detection through set prediction [8,26], and extract a fine-grained lane graph with a Path2Graph algorithm. Based on this path-wise modeling, we propose an online lane graph construction framework, termed LaneGAP, which feeds onboard sensor data into an end-to-end network for path detection, and further transforms detected paths into a lane graph.

To fairly evaluate across different modelings and emphasize the unique properties of the lane graph (*i.e.*, connection, direction, and split/merge around the junction points), we carefully design a graph-centric TOPO metric to evaluate both holistic topology and local junction topology, which is compatible with different modelings.

We compare LaneGAP with the pixel-wise modeling method HDMapNet [22] and piece-wise modeling methods STSU [6], MapTR [26] under strictly fair conditions (encoder, model size, training schedule, *etc.*) on the challenging nuScenes [5] and Argoverse2 [45] datasets, which covers diverse graph topology and traffic conditions. With only camera input, LaneGAP achieves the best graph construction quality both quantitatively (see Tab. 1) and qualitatively (see Fig. 2), while running at the fastest inference speed (2x faster than HDMapNet). We further validate the superiority of path-wise modeling on TopoNet [23] (1.6 higher mIoU), a state-of-the-art piece-wise modeling method on the OpenLane-V2 dataset [43]. The extensive experiments show that modeling the lane graph at the path level is reasonable and promising. We hope LaneGAP can serve as a fundamental module of the self-driving system and boost the development of downstream motion planning.

Our contributions can be summarized as follows:

- We propose to model the lane graph in a novel path-wise manner, which well preserves the continuity of the lane and encodes traffic information for planning.
- Based on our path-wise modeling, we present an online lane graph construction method, termed LaneGAP. LaneGAP end-to-end learns the path and constructs the lane graph via the designed Path2Graph algorithm.
- We design a novel graph-centric metric to evaluate the unique properties of lane graph construction (*i.e.*, connection, direction, and junction), which is compatible with different modelings.

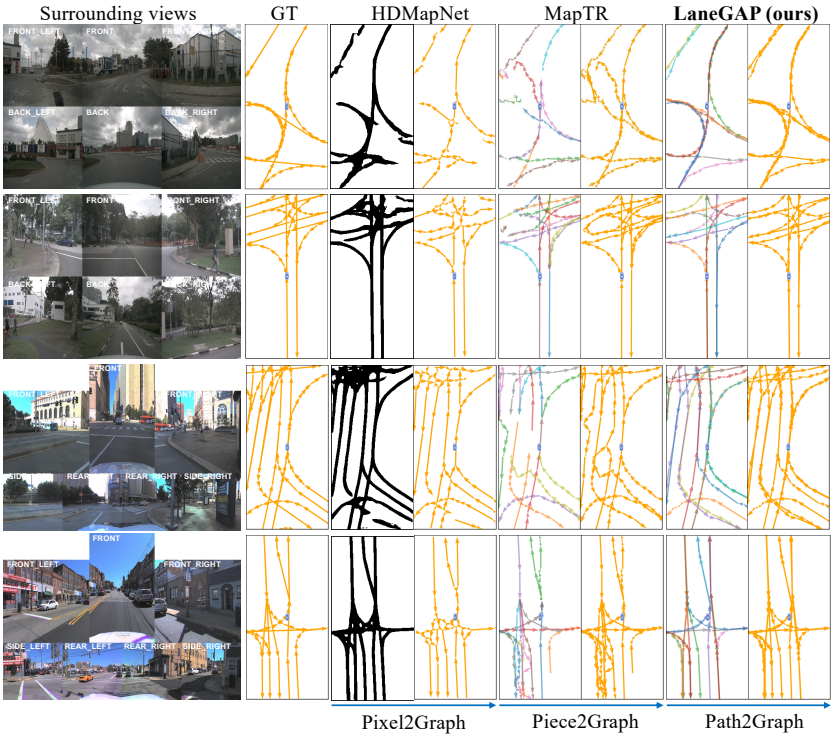

Fig. 2: Qualitative comparison of pixel-wise HDMapNet, piece-wise MapTR, and path-wise LaneGAP on the nuScenes and Argoverse2 val splits. The top 2 rows are from the nuScenes val split, and the bottom 2 rows are from the Argoverse2 val split. Different colors indicate different instances. More qualitative comparisons are available in the supplementary.

– We qualitatively and quantitatively demonstrate the superior accuracy and efficiency of LaneGAP over pixel-based and piece-based methods. LaneGAP can cope with diverse traffic conditions, especially for road intersections with complicated lane topology.

2 Related Work

Lane detection. Lane detection only considers predicting and evaluating the lane divider lines without spatial relation (merging and fork). Since most lane detection datasets only provide front-view images, previous lane detection methods [16, 17, 28, 29, 41, 44] were stuck in predicting lines with a small curvature in a limited horizontal FOV. BezierLaneNet [15] uses a fully convolutional network to predict Bezier lanes defined with 4 Bezier control points. PersFormer [9] proposes a Transformer-based architecture for spatial transformation and unifies 2D and 3D lane detection.

Online HD map construction. Online HD map construction can be seen as an advanced setting of lane detection, consisting of lines and polygons with various semantics in the local 360° FOV perception range of ego-vehicle. With advanced 2D-to-BEV modules [35], previous online HD map construction methods cast it into semantic segmentation task on the transformed BEV features [24, 31–34, 51]. Building vectorized semantic HD map online achieves increasing interests nowadays [13, 22, 26, 30, 38, 40], HDMapNet [22] follows a segmentation-then-vectorization paradigm. To achieve end-to-end learning [8, 14, 53], VectorMapNet [30] adopts a two-stage pipeline for vectorized HD map learning. MapTR [26] proposes a unified permutation-equivalent modeling to exploit the undirected nature of semantic HD map and designs a parallel end-to-end framework. BeMapNet [38] and PivotNet [13] propose Bezier-based representation and pivot-based representation for modeling map geometry. While the above works focus on map elements without physical directions and lane graph topology, we aim to fill the gap in this work.

Road graph construction. There is a long history of extracting the road graph from remote sensor data (*e.g.*, aerial imagery and satellite imagery). Many works [2, 4, 20, 36, 52] frame the road graph as a pixel-wise segmentation problem and utilizes morphological post-processing methods to extract the road graph. RoadTracer [1] uses an iterative search process to extract graph topology step by step. Some works [11, 25, 37, 42, 48] follow this sequential generation paradigm. Different from the above works, we focus on the online, ego-centric setting with vehicle-mounted sensors to produce more fine-grained lane-level graph.

Lane graph construction. The lane graph is traditionally constructed with an offline pipeline [3, 19, 47]. [19] proposes a multi-step training pipeline to construct the lane graph of aerial images based on pixel-wise modeling. [3] proposes a bottom-up approach to aggregate multiple local aerial lane graphs into a globally consistent graph. CenterlineDet [47] proposes a DETR-like decision-making transformer network to iteratively update the global lane graph with vehicle-mounted sensors. Recently, STSU [6] shifts the offline lane graph construction to the online, ego-centric setting with vehicle-mounted sensors. It models the lane graph as a set of disjoint pieces split by junction points and a set of connections among those pieces. Based on STSU, recent works [7, 23, 46] advance this piece-wise setting and push forward the performance. Different from the above graph modelings, we regard the path as the primitive of the lane graph, and model the lane graph in a novel path-wise manner.

3 Method

In this section, we first describe how to translate the directed lane graph into a set of directed paths in Sec. 3. Then we introduce the online path detection framework in Sec. 3.2. And we describe how to translate paths back to the lane graph in Sec. 3.3. An overview of our method is exhibited in Fig. 3.

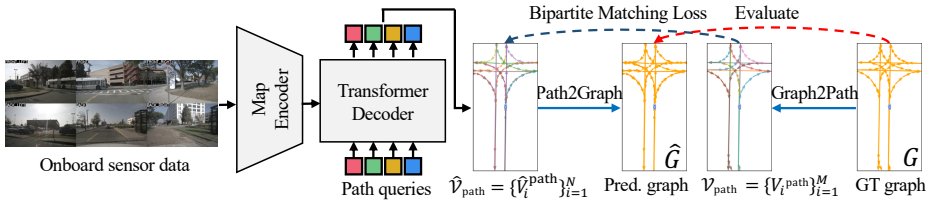

Fig. 3: Overview of LaneGAP.

3.1 Graph2Path

We propose a simple Graph2Path algorithm to translate the directed lane graph into a set of paths according to the direction and connection information encoded in the lane graph. The pseudo-code of Graph2Path is shown in Alg. 1.

Given the ground truth Lane graph G, which is typically a directed graph in the local map around the ego-vehicle, we first extract the root vertices Ω_{root} and leaf vertices Ω_{leaf}. Then we pair the root vertices and leaf vertices and then utilize depth-first-search (DFS) algorithm to find the valid path V^{path}. Finally, we can translate the ground truth directed lane graph G to a set of directed paths $\mathcal{V}_{\text{path}} = \{V_i^{\text{path}}\}_{i=1}^{M}$, where M is the number of ground truth paths.

3.2 Path Representation and Learning

Inspired by advanced set detection methods [8, 53], we propose an end-to-end network, LaneGAP, to predict all paths simultaneously in a single stage, as illustrated in Fig. 3. Our network consists of an encoder that encodes the features from the onboard sensor data, and a query-based Transformer decoder that performs set detection by decoding a set of paths $\hat{\mathcal{V}}_{\text{path}} = \{\hat{V}_i^{\text{path}}\}_{i=1}^{N}$ from the encoded features. To parameterize the path, we utilize two widely-used types of representations, Polyline [26, 30] and Bezier [6, 38], where Polyline offers high flexibility in describing the path, while Bezier provides a smoother representation.

Polyline representation. Polyline representation models the arbitrary directed path as an ordered set of N_{p} points $V_{\text{poly}}^{\text{path}} = \{p_j \in \mathbb{R}^2 | j = 0, 1, 2, ..., N_{\text{p}} - 1\}$. We directly regress polyline points and utilize deformable attention [53] to exploit the local information along each Polyline path, where the keys and values are the local features along the Polyline path.

Bezier representation. Bezier representation models the directed path as an ordered set of N_{b} control points $V_{\text{Bezier}}^{\text{path}} = \{b_j \in \mathbb{R}^2 | j = 0, 1, 2, ..., N_{\text{b}} - 1\}$. Bezier is a parametric curve, where the point B on the line can be calculated by the weighted sum of control points $V_{\text{Bezier}}^{\text{path}}$:

$$B = \sum_{j=0}^{N_{\text{b}}-1} C_{N_{\text{b}}-1}^{j} t^j (1-t)^{N_{\text{b}}-1-j} b_j, 0 \leq t \leq 1. \tag{1}$$

Given the Bezier control points set $V^{\text{path}}_{\text{Bezier}}$ and the sampled interval set $T = \{t_k \in \mathbb{R} | 0 \leq t_k \leq 1, k = 0, 1, 2, ..., K-1\}$, we can calculate the curve $\mathcal{B} = \{B_k \in \mathbb{R}^2 | 0, 1, 2, ..., K-1\}$ with matrix multiplication:

$$\mathcal{B} = \mathit{\Gamma} \times V^{\text{path}}_{\text{Bezier}}, \tag{2}$$

where weight matrix $\mathit{\Gamma}$ is a $K \times N_{\text{b}}$ matrix and $\mathit{\Gamma}(k, j) = C^{j}_{N_{\text{b}}-1} t_k^j (1-t_k)^{N_{\text{b}}-1-j}$. For Beizer representation, we use the same network as Polyline representation to regress Bezier control points directly. To enable exploiting the local information along the Bezier path with offline control points, we sample the Bezier path to get online points $\mathcal{B}$ based on Eq. 2 and perform deformable attention, where the keys and values are the local features around the sampled points $\mathcal{B}$ along the Bezier path. We denote this design as Bezier deformable attention, which enables the Transformer decoder to aggregate features along the Bezier path.

Learning. We use the encoder to transform the input onboard sensor data into unified BEV features F_{BEV} [10,24,49]. Then we utilize the Transformer decoder to predict a set of paths based on a set of random initialized learnable path queries [26]. With the above path representation, we can cast the predicted path into an ordered set of points with a fixed number N_v of points on the path, where $\hat{V}^{\text{path}}_i = \{\hat{v}_j \in \mathbb{R}^2 | j = 0, 1, 2, ..., N_v - 1\}$. We modify the bipartite matching loss used in [8] to fit in the path detection setting:

$$\begin{aligned}
\mathcal{L}_{\text{bipartite}}(\hat{\mathcal{V}}_{\text{path}}, \mathcal{V}_{\text{path}}) &= \sum_{i=1}^{N} [\mathcal{L}_{\text{Focal}}(\hat{p}_{\hat{\sigma}(i)}, c_i) + \quad \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{path}}(\hat{V}^{\text{path}}_{\hat{\sigma}(i)}, V^{\text{path}}_i)], \\
\mathcal{L}_{\text{path}}(\hat{V}^{\text{path}}_{\hat{\sigma}(i)}, V^{\text{path}}_i) &= \sum_{j=0}^{N_v - 1} L_1(\hat{v}_j, v_j),
\end{aligned} \tag{3}$$

where $\hat{\sigma}$ is the optimal assignment between a set of predicted paths and a set of ground truth paths computed by the Hungarian algorithm, c_i is the target class label, and $\mathcal{L}_{\text{Focal}}(\hat{p}_{\hat{\sigma}(i)}, c_i)$ is the classification loss defined in [27]. L_1 loss is utilized between the matched predicted path $\hat{V}^{\text{path}}_{\hat{\sigma}(i)}$ and sampled ground truth path V^{path}_i. To enhance the BEV features, we introduce an auxiliary BEV segmentation branch ϕ_{seg} to predict the BEV segmentation mask, and the auxiliary BEV segmentation loss is defined as:

$$\mathcal{L}_{\text{auxseg}} = \mathcal{L}_{\text{CE}}(\phi_{\text{seg}}(F_{\text{BEV}}), M_{\text{BEV}}), \tag{4}$$

where M_{BEV} is the ground truth. The total loss is defined as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{path}} + \mathcal{L}_{\text{auxseg}}, \tag{5}$$

where $\mathcal{L}_{\text{cls}}$ is the path-wise classification loss using Focal loss [27].

Algorithm 1: Pseudo-code of Graph2Path.

Input: directed lane graph G
Output: directed paths $\mathcal{V}$

Initialize: $\mathcal{V} \leftarrow \emptyset$,
  root vertices set $\Omega_{\text{root}} \leftarrow \emptyset$,
  leaf vertices set $\Omega_{\text{leaf}} \leftarrow \emptyset$
for *v in G.vertices* **do**
  if *v.in_degree* $== 0$ **then**
    $\Omega_{\text{root}} \leftarrow \Omega_{\text{root}} \cup \{v\}$
  end
  if *v.out_degree* $== 0$ **then**
    $\Omega_{\text{leaf}} \leftarrow \Omega_{\text{leaf}} \cup \{v\}$
  end
end
for *root v in* Ω_{root} **do**
  for *leaf v in* Ω_{leaf} **do**
    $V^{\text{path}} =$ **findpath**$(root\ v, leaf\ v, G)$
    if V^{path} *is not None* **then**
      $\mathcal{V} \leftarrow \mathcal{V} \cup \{V^{\text{path}}\}$
    end
  end
end
Return: $\mathcal{V}$

Algorithm 2: Pseudo-code of Path2Graph.

Input: a set of paths $\mathcal{V} = \{V_i^{\text{path}}\}$
Output: directed lane graph G

1 Initialization: $G = \text{DirGraph}()$
2 **for** V^{path} *in* $\mathcal{V}$ **do**
3 $V = \{v_j\} \leftarrow$ **discret_to_vert_seq**(V^{path})
4 **for** v_j *in* V **do**
5 **if** v_j *is the last vertice of* V **then**
6 G.**add_vertice**(v_j)
7 **break**
8 **end**
9 G.**add_vertice**(v_j)
10 G.**add_edge**(v_j, v_{j+1})
11 **end**
12 **end**
13 $G =$ **merge_vertices_by_overlap**(G)
14 Return: G

The for loops here can be implemented in a parallel paradigm. **merge_vertices_by_overlap** merges the overlapped vertices of different paths to reduce the redundancy of G without affecting the accuracy of graph topology.

3.3 Path2Graph

The predicted continuous paths encode sufficient traffic information and can be directly applied to downstream motion planning. To further recover the graph structure of lane topology and extract merging and fork information, we convert the predicted paths $\hat{\mathcal{V}} = \{\hat{V}_i^{\text{path}}\}$ into a directed lane graph $\hat{G}$ with a designed Path2Graph algorithm in Alg. 2.

We discretize the path into point sequences $V = \{v_j\}$. The discretized points are regarded as vertices and the adjacent relation between successive points is regarded as edges of vertices. We add these vertices and edges to the directed graph (line 4 to 11 in Alg. 2). The vertices registered in the directed graph G on one path may have spatial overlapping with vertices on other paths. We merge the overlapped vertices into one vertex if their distance is below a threshold (line 13 in Alg. 2). The merged vertex inherits the predecessors and successors of the pair of overlapped vertices.

4 Metrics

Previous works mainly design a network based on one sort of graph modelings (pixel-wise or piece-wise) and choose a metric (pixel-level IoU or instance-level Chamfer distance mAP) friendly to the modeling. To fairly evaluate and compare across different modelings, we adapt the TOPO metric [19] to measure the correctness of the overall directed graph construction. The uniqueness of lane graph construction lies in the junction points, without which the lane graph degenerates into a set of disjoint lanes. To emphasize the quality of the subgraph around the junction points, we propose a new metric, Junction TOPO, which specifically evaluates the accuracy of the local directed graph formed by traversing around the junction points on the directed lane graph.

TOPO metric. Given the predicted directed lane graph $\hat{G}$ and ground truth directed lane graph G, we interpolate them so that the distances between any two connected vertices are $0.15m$, and get predicted directed lane graph $\hat{G} = (\hat{V}, \hat{E})$ and ground truth directed lane graph $G = (V, E)$ where $\hat{V}, V$ are the sets of interpolated vertices and $\hat{E}, E$ are the sets of edges among vertices encoding direction and connection. For $\hat{V}$ and V, a pair of vertices is considered a candidate match if the distance between the two vertices is less than $0.45m$. And we utilize maximal one-to-one matching among those candidate pairs to find final matched vertices $P_{\text{pair}} = \{(\hat{v}, v)_i\}_{i=1}^{N_{\text{pair}}}$, then we traverse the directed graph around the paired vertices $\hat{v}$ and v for less than $7.5m$ to get subgraphs $\hat{S}_{\hat{v}}$ and S_v on $\hat{G}$ and G. We compute the precision $\text{Pre}(\hat{S}_{\hat{v}}, S_v) = \frac{N_{\text{subpair}}}{|\hat{S}_{\hat{v}}|}$ and recall $\text{Rec}(\hat{S}_{\hat{v}}, S_v) = \frac{N_{\text{subpair}}}{|S_v|}$ between the vertices of predicted subgraph $\hat{S}_{\hat{v}}$ and the vertices of ground truth subgraph S_v, where the matching part follows the previous procedure with $0.45m$ threshold. Finally, we report the TOPO precision and recall defined as:

$$\begin{aligned} \text{Precision}_{\text{TOPO}} &= \frac{\sum_{i=1}^{N_{\text{pair}}} \text{Pre}(\hat{S}_{\hat{v}}, S_v)}{|\hat{V}|}, \\ \text{Recall}_{\text{TOPO}} &= \frac{\sum_{i=1}^{N_{\text{pair}}} \text{Rec}(\hat{S}_{\hat{v}}, S_v)}{|V|}. \end{aligned} \tag{6}$$

Junction TOPO metric. The TOPO metric focuses on the topology correctness of the overall directed lane graph, it does not highlight the correctness of the subgraph formed by traversing from the junction points, which plays a key role in determining the driving choices across different lanes. To bridge this gap, we propose the Junction TOPO metric, which only reports the precision and recall of the junction subgraph. Given the N_{junction} junction points of the ground truth lane graph, we get pairs of subgraphs $(\hat{S}_{\text{junction}}, S_{\text{junction}})$ by traversing the directed graphs $\hat{G}$ and G less than $7.5m$ from junction point. For each subgraph pair, we calculate the precision $\text{Pre}(\hat{S}_{\text{junction}}, S_{\text{junction}})$ and recall $\text{Rec}(\hat{S}_{\text{junction}}, S_{\text{junction}})$.

Undirected versions. The above metrics calculate the precision and recall by traversing the directed graph $\hat{G}$ and G, ignoring the predecessor vertices. To

Dataset	Method	Modeling	Directed Graph						Undirected Graph						Param.	FPS_{net}	FPS
			Junction			TOPO			Junction			TOPO					
			Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1			
nusc	HDMapNet	Pixel-wise	0.594	0.442	0.507	0.608	0.443	0.513	0.526	0.445	0.482	0.563	0.447	0.498	35.4M	17.8	6.8
	STSU	Piece-wise	0.449	0.393	0.419	0.416	0.376	0.395	0.413	0.404	0.408	0.405	0.375	0.389	35.3M	15.8	15.1
	MapTR	Piece-wise	0.540	0.459	0.496	0.504	0.432	0.465	0.493	0.465	0.478	0.487	0.429	0.456	36.0M	15.6	15.0
	LaneGAP	Path-wise	0.591	0.539	**0.564**	0.547	0.511	**0.529**	0.543	0.518	**0.530**	0.534	0.496	**0.514**	35.9M	16.5	**15.6**
av2	HDMapNet	Pixel-wise	0.656	0.449	0.533	0.635	0.463	0.535	0.596	0.473	0.528	0.607	0.470	0.530	35.4M	15.2	4.6
	STSU	Piece-wise	0.562	0.414	0.477	0.462	0.436	0.449	0.512	0.416	0.459	0.449	0.431	0.440	35.3M	13.0	12.4
	MapTR	Piece-wise	0.589	0.469	0.522	0.520	0.481	0.500	0.543	0.468	0.503	0.507	0.477	0.491	36.0M	13.1	12.5
	LaneGAP	Path-wise	0.663	0.575	**0.616**	0.578	0.552	**0.565**	0.627	0.541	**0.581**	0.569	0.546	**0.557**	35.9M	13.8	**13.0**

Table 1: Quantitative comparison of lane graph modeling under fair conditions (the same encoder, comparable model size, the same training schedule, and vision-only modality). FPS_{net} and FPS of all the experiments are measured on the same machine with one NVIDIA Geforce RTX 3090 GPU and one 24-core AMD EPYC 7402 2.8 GHz CPU, where FPS_{net} is benchmarked with only network forward.

evaluate the complete connections, we turn the directed graphs into undirected graphs and repeat the calculation defined above for two metrics.

5 Experiments

5.1 Dataset

We conduct experiments on two popular and large-scale datasets, *i.e.*, nuScenes [5] and Argoverse2 [45]. nuScenes [5] dataset consists of 1000 scene sequences. Each sequence is sampled in 2Hz frame rate and provides LiDAR point cloud and RGB images from 6 surrounding cameras, which covers 360° horizontal FOV of the ego-vehicle. The dataset provides the 2D lane graph without height information in the form of lane centerline and covers diverse online driving conditions. For the online setting of lane graph construction, we set the perception ranges as $[-15.0m, 15.0m]$ for the X-axis and $[-30.0m, 30.0m]$ for the Y-axis, and preprocess the dataset following [22, 26, 30]. We train on the nuScenes train set and evaluate on the val set. The experiments on nuScenes dataset are conducted using 6 surrounding-view images by default. Argoverse2 [45] dataset contains 1000 scene logs and 7 surrounding cameras with 360° horizontal FOV. Each log is sampled in 10Hz frame rate and provides 3D lane graph. Due to the limitations of pixel-wise modeling in performing 3D segmentation (piece-wise and path-wise modelings can simply predict an extra z-coordinate to support 3D lane graph construction), we omit the height information for fair comparison across different modelings. Other experimental settings remain the same as those for nuScenes.

5.2 Baselines

Implementation details. All the baselines use the same encoder consisting of ResNet50 [18] and GKT [10] to transform the 360° horizontal FOV multi-camera images to BEV features at the resolution of 200×100 (*i.e.*, the BEV grid size is

Method	$\text{HDMapNet}^*_{\text{pixel}}$	$\text{STSU}^*_{\text{piece}}$	$\text{VectorMapNet}^*_{\text{piece}}$	$\text{MapTR}^*_{\text{piece}}$	$\text{TopoNet}^*_{\text{piece}}$	$\text{LaneGAP}^\dagger$
mIoU	18.3	31.1	25.0	35.7	39.0	**40.6**

Table 2: Quantitative comparison on Openlane-V2 dataset. * means the results are directly taken from TopoNet paper [23]. $\text{LaneGAP}^\dagger$ is our proposed variant based on $\text{TopoNet}_{\text{piece}}$ to validate the effectiveness of path-wise modeling, which only adapts the lane piece detection part of TopoNet for learning path-wise representation.

$0.3m$). All the models are controlled in comparable parameter sizes and trained for long enough epochs to ensure convergence (110 epochs on nuScenes and 24 epochs on Argoverse2). We train all the experiments on 8 NVIDIA GeForce RTX 3090 GPUs with a total batch size of 32 (6 view images on nuScenes and 7 view images on Argoverse2). Further details about training settings are provided in the supplementary material. The only differences lie in the specialized decoders designed for respective modelings, which are detailed as follows.

Pixel-wise modeling. We select HDMapNet [22] as the pixel-wise modeling baseline. It adopts a U-Net structure with a ResNet18 to output a binarized segmentation map and a direction map with two branches. Meanwhile, a cross-entropy loss is applied to the segmentation map, and L_2 loss is applied to the (-1,1) normalized direction map [19,52]. To exclude the quantization error brought by classification on dense pixels, the output resolution of the U-Net-shaped [39] segmentation model is set to 400×200, which means the grid size is $0.15m$ (the perception range is $60m \times 30m$), the same as the graph interpolation size. The Pixel2Graph follows [19, 22].

Piece-wise modeling. STSU [6] is selected as the piece-wise modeling baseline, which consists of a detection branch to detect Bezier lane pieces with 4 control points and an association branch to predict the connectivity between lane pieces. It stacks 6 Transformer decoder layers to iteratively refine the predictions. L_1 loss and Focal loss are applied to the matched (0,1) normalized pieces and cross-entropy loss is applied to the connectivity. More recently, MapTR [26] proposes a much stronger Transformer decoder for the detection branch. We further implement an additional piece-wise baseline by simply replacing the original Beizer piece detection branch of STSU with MapTR Polyline piece detection branch. We set the number of Polyline points to 20. The Piece2Graph follows [6].

Path-wise modeling. The path-wise LaneGAP adopts the same decoder as the piece-wise MapTR. We use the 30-point Polyline to represent the path. We stack 6 deformable Transformer decoder layers and regress $(0, 1)$ normalized coordinates. During training, L_1 loss and Focal loss are applied to the matched paths, and cross-entropy loss is applied to the auxiliary BEV segmentation branch. As for the Path2Graph, We discretize the path at the interval of $0.15m$, and the merging threshold is set to $0.15m$, the same as the graph interpolation size.

5.3 Quantitative Comparision

Tab. 1 compares path-wise modeling with pixel-wise modeling and piece-wise modeling w.r.t. their accuracy, model size, FPS_{net}, FPS.

Highlights. Path-wise modeling achieves the best F_1 score on the both subgraph around the junction points and the overall graph across two popular datasets (nuScenes and Argoverse2), while running at the fastest inference speed. The designed Path2Graph algorithm has negligible cost in translating predicted paths into a directed lane graph (from 16.5 FPS to 15.6 FPS).

Path-wise *vs*. pixel-wise. Thanks to the high-resolution output of the segmentation model, pixel-wise modeling exhibits comparable precision for subgraph construction and overall graph construction. Nevertheless, the notorious fragmentation and oversmooth issues of segmentation methods make it hard to maintain continuity across pixels and distinguish the fine-grained subgraph around the junction points, leading to much lower recall compared to path-wise modeling. In comparison, path-wise modeling demonstrates higher lane graph construction quality and 2x faster inference speed.

Path-wise *vs*. piece-wise. Path-wise modeling outperforms piece-wise modeling on all metrics. Even equipped with the same advanced Transformer decoder proposed in [26], our path-wise LaneGAP still significantly outperforms piece-wise MapTR, which validates the effectiveness.

5.4 Quantitative Comparision on Openlane-V2 Dataset

Tab. 2 compare path-wise modeling against state-of-the-art piece-wise method TopoNet on OpenLane-V2 [43] dataset, which is a large-scale perception and reasoning dataset developed from nuScenes and Argoverse2. To further validate the superiority of path-wise modeling over piece-wise modeling under fair setting, we adapt the lane piece detection part of TopoNet for learning path-wise representation, while removing all the related piece-wise connection modules of TopoNet. This variant is denoted as $\text{LaneGAP}^{\dagger}_{\text{path}}$, distinguishing it from the original implementation, $\text{TopoNet}_{\text{piece}}$. All other hyper-parameters and the training recipe remain consistent with $\text{TopoNet}_{\text{piece}}$.

Results. Since the evaluation metric proposed in TopoNet is not applicable to other modelings. We employ the mIoU metric for comparison, which is also utilized in TopoNet paper for benchmarking against the pixel-wise HDMapNet. As shown in Tab. 2, $\text{LaneGAP}^{\dagger}$ achieves 1.6 higher mIoU than $\text{TopoNet}_{\text{piece}}$ in $50m \times 100m$ perception range, thereby validating the effectiveness of path-wise modeling.

5.5 Qualitative Comparison

Fig. 2 compares baselines on complicated lane graphs with more than 6 junction points across different datasets. Failure case studies and more qualitative comparisons are provided in the supplementary material.

$\mathcal{L}_{\text{auxseg}}$	Junction TOPO			TOPO			Param.	FPS
	Prec.	Rec.	F_1	Prec.	Rec.	F_1		
✗	0.544	0.516	0.529	0.421	0.484	0.450	35.9M	15.6
✓	0.554	0.524	0.539	0.432	0.493	0.461	35.9M	15.6

Table 3: Auxiliary BEV segmentation branch.

Polyline Points	Junction TOPO			TOPO			Param.	FPS
	Prec.	Rec.	F_1	Prec.	Rec.	F_1		
10	0.550	0.515	0.532	0.423	0.481	0.450	35.9M	16.3
30	0.554	0.524	0.539	0.432	0.493	0.461	35.9M	15.6
40	0.566	0.548	0.557	0.440	0.509	0.472	35.9M	15.1

Table 4: Polyline path representation.

Bezier Points	Junction TOPO			TOPO			Param.	FPS
	Prec.	Rec.	F_1	Prec.	Rec.	F_1		
3	0.480	0.483	0.481	0.309	0.447	0.365	35.9M	16.1
5	0.514	0.507	0.510	0.375	0.477	0.477	35.9M	15.8
10	0.498	0.503	0.501	0.334	0.463	0.388	35.9M	15.6

Table 5: Bezier path representation.

Method	L2 (m) ↓				Collision (%) ↓			
	1s	2s	3s	Avg.	1s	2s	3s	Avg.
VAD-Tiny	0.46	0.76	1.12	0.78	0.21	0.35	0.58	0.38
VAD-Tiny+MapTR	0.43	0.74	1.06	0.74	0.20	0.31	0.51	0.34
VAD-Tiny+LaneGAP	**0.42**	**0.70**	**0.99**	**0.70**	**0.18**	**0.27**	**0.43**	**0.29**

Table 6: Comparison on downstream end-to-end planner VAD.

Method	Modeling	Junction TOPO			TOPO		
		Prec.	Rec.	F_1	Prec.	Rec.	F_1
HDMapNet	Pixel	0.401	0.328	0.361	0.415	0.349	0.379
STSU	Piece	0.309	0.297	0.303	0.298	0.273	0.285
MapTR	Piece	0.369	0.357	0.363	0.363	0.317	0.338
LaneGAP	Path	0.384	0.400	**0.392**	0.384	0.392	**0.388**

Table 7: Comparing on large perception range $50m \times 100m$.

Method	Modeling	Junction TOPO			TOPO		
		Prec.	Rec.	F_1	Prec.	Rec.	F_1
HDMapNet	Pixel	0.464	0.312	0.373	0.468	0.318	0.379
STSU	Piece	0.318	0.241	0.274	0.296	0.245	0.268
MapTR	Piece	0.391	0.298	0.338	0.369	0.304	0.334
LaneGAP	Path	0.458	0.356	**0.400**	0.447	0.358	**0.398**

Table 8: Comparing on new split.

	HDMapNet	STSU	MapTR	LaneGAP
mIoU on nusc	48.1	38.2	43.6	**49.2**
mIoU on av2	55.0	44.7	49.3	**55.9**

Table 9: Comparing on mIoU metric.

Post-Proc.	Junction TOPO			TOPO		
	Prec.	Rec.	F_1	Prec.	Rec.	F_1
Pixel	0.898	0.678	0.773	0.913	0.693	0.788
Piece	1.000	1.000	1.000	1.000	1.000	1.000
Path	1.000	1.000	1.000	1.000	1.000	1.000

Table 10: Post-processing accuracy.

Highlights. Path-wise modeling demonstrates better lane graph construction quality than pixel-wise and piece-wise modeling on extremely challenging lane graphs, well preserving the continuity of lane.

Path-wise *vs*. pixel-wise. As shown in Fig. 2, the pixel-wise HDMapNet performs well on non-junction area. But for the subgraph around the junction points, the segmentation model struggles to distinguish the fine-grained topology, and the post-processing is prone to fail to generate a decent vectorized lane graph, aligning with the much lower Junction TOPO metrics in Tab. 1. On the contrary, LaneGAP can capture the fine-grained topology.

Path-wise *vs*. piece-wise. The quality of the lane graph constructed by piece-wise modeling depends on the accuracy of both piece detection and connectivity prediction. As shown in Fig. 2, the piece-wise MapTR either detects jagged lane pieces or produces wrong connections, leading to an incomplete and inaccurate lane graph after Piece2Graph post-processing. In contrast, our proposed path-wise modeling encodes the connectivity between pieces into the continuous path representation, which is easy to learn and robust.

5.6 Ablation Study

We ablate the design choices with a 24-epoch training schedule of path-wise modeling on nuScenes by default without specification. And we report Junction TOPO and TOPO only on the directed lane graph.

Effectiveness of auxiliary BEV segmentation branch. Tab. 3 shows the auxiliary BEV segmentation branch can improve the performance by 1% F_1 of Junction TOPO and 1.1% F_1 of TOPO without adding cost at inference.

Polyline path representation. As shown in Tab. 4, the accuracy increases with adding more points for Polyline modeling under a 24-epoch schedule, while the inference speed decreases. We choose 30-point Polyline as default setting.

Bezier path representation. Tab. 5 shows that adding the number of control points from 3 to 5 improves 2.9% F_1 of Junction TOPO and 11.2% F_1 of TOPO. While adding the number of control points from 5 to 10 incurs an accuracy drop.

Comparison on end-to-end planner VAD. Tab. 6 provides comparison on end-to-end planner VAD [21] by including an extra centerline lane graph branch. Since VAD utilizes instance-wise queries, HDMapNet is not applicable, we only compare with piece-wise MapTR. Path-wise modeling enhances planning performance and outperforms piece-wise modeling. Notably, LaneGAP's path queries offer a complete depiction of the lane graph, unlike MapTR's centerline piece queries, whose connectivity predictions are not directly usable in VAD.

Large perception range. In Tab. 7, we compare the performance on a large perception range ($50m \times 100m$) using the same training recipe of Tab. 1. LaneGAP still outperforms other methods.

Generalization on new split. StreamMapNet [50] proposes a new split of the nuScenes dataset, which splits the train/val set by the geolocation and reduces the map overlap. We further compare the performance on this new split in Tab. 8 using the same training recipe. LaneGAP maintains consistent superiority.

Comparison on mIoU metric. In Tab. 9, we further compare the performance of different modelings with mIoU metric following HDMapNet [22]. We render the prediction on the BEV map with $0.75m$ line width. The results further validate the superiority of LaneGAP.

Post-processing accuracy. In Tab. 10, we show the accuracy of the lane graphs transformed by respective post-processings given the ground-truth segmentation maps, sets of pieces and connections, and set of paths. The results indicate that Pixel2Graph post-processing is prone to fail in handling the fine-grained topology of the subgraph around junction points, as illustrated in Fig. 2.

6 Conclusion

In this work, we present an online lane graph construction method LaneGAP based on novel path-wise modeling. We qualitatively and quantitatively demonstrate the superiority of LaneGAP over pixel-based and piece-based methods. LaneGAP can serve as a fundamental module of the self-driving system and facilitate downstream motion prediction and planning, which we leave as future work.

Acknowledgement

This work was partly supported by the National Natural Science Foundation of China (NSFC 62276108).

References

1. Bastani, F., He, S., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S., DeWitt, D.: Roadtracer: Automatic extraction of road networks from aerial images. In: CVPR (2018)
2. Batra, A., Singh, S., Pang, G., Basu, S., Jawahar, C., Paluri, M.: Improved road connectivity by joint learning of orientation and segmentation. In: CVPR (2019)
3. Büchner, M., Zürn, J., Todoran, I.G., Valada, A., Burgard, W.: Learning and aggregating lane graphs for urban automated driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13415–13424 (2023)
4. Buslaev, A., Seferbekov, S., Iglovikov, V., Shvets, A.: Fully convolutional network for automatic road extraction from satellite imagery. In: CVPR (2018)
5. 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: CVPR (2020)
6. Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Structured bird's-eye-view traffic scene understanding from onboard images. In: ICCV (2021)
7. Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Topology preserving local road network estimation from single onboard camera image. In: CVPR (2022)
8. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV (2020)
9. Chen, L., Sima, C., Li, Y., Zheng, Z., Xu, J., Geng, X., Li, H., He, C., Shi, J., Qiao, Y., Yan, J.: Persformer: 3d lane detection via perspective transformer and the openlane benchmark. In: ECCV (2022)
10. Chen, S., Cheng, T., Wang, X., Meng, W., Zhang, Q., Liu, W.: Efficient and robust 2d-to-bev representation learning via geometry-guided kernel transformer. arXiv preprint arXiv:2206.04584 (2022)
11. Chu, H., Li, D., Acuna, D., Kar, A., Shugrina, M., Wei, X., Liu, M.Y., Torralba, A., Fidler, S.: Neural turtle graphics for modeling city road layouts. In: ICCV (2019)
12. Dauner, D., Hallgarten, M., Geiger, A., Chitta, K.: Parting with misconceptions about learning-based vehicle motion planning. In: Conference on Robot Learning (CoRL) (2023)
13. Ding, W., Qiao, L., Qiu, X., Zhang, C.: Pivotnet: Vectorized pivot learning for end-to-end hd map construction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3672–3682 (2023)
14. Fang, Y., Liao, B., Wang, X., Fang, J., Qi, J., Wu, R., Niu, J., Liu, W.: You only look at one sequence: Rethinking transformer in vision through object detection. NeurIPS (2021)
15. Feng, Z., Guo, S., Tan, X., Xu, K., Wang, M., Ma, L.: Rethinking efficient lane detection via curve modeling. In: CVPR (2022)
16. Garnett, N., Cohen, R., Pe'er, T., Lahav, R., Levi, D.: 3d-lanenet: end-to-end 3d multiple lane detection. In: ICCV (2019)
17. Guo, Y., Chen, G., Zhao, P., Zhang, W., Miao, J., Wang, J., Choe, T.E.: Gen-lanenet: A generalized and scalable approach for 3d lane detection. In: ECCV (2020)

18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
19. He, S., Balakrishnan, H.: Lane-level street map extraction from aerial imagery. In: WACV (2022)
20. He, S., Bastani, F., Jagwani, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Elshrif, M.M., Madden, S., Sadeghi, M.A.: Sat2graph: Road graph extraction through graph-tensor encoding. In: ECCV (2020)
21. Jiang, B., Chen, S., Xu, Q., Liao, B., Chen, J., Zhou, H., Zhang, Q., Liu, W., Huang, C., Wang, X.: Vad: Vectorized scene representation for efficient autonomous driving. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8340–8350 (2023)
22. Li, Q., Wang, Y., Wang, Y., Zhao, H.: Hdmapnet: An online hd map construction and evaluation framework. In: ICRA (2022)
23. Li, T., Chen, L., Wang, H., Li, Y., Yang, J., Geng, X., Jiang, S., Wang, Y., Xu, H., Xu, C., Yan, J., Luo, P., Li, H.: Graph-based topology reasoning for driving scenes. arXiv preprint arXiv:2304.05277 (2023)
24. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J.: Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: ECCV (2022)
25. Li, Z., Wegner, J.D., Lucchi, A.: Topological map extraction from overhead images. In: ICCV (2019)
26. Liao, B., Chen, S., Wang, X., Cheng, T., Zhang, Q., Liu, W., Huang, C.: MapTR: Structured modeling and learning for online vectorized HD map construction. In: ICLR (2023), https://openreview.net/forum?id=k7p_YAO7yE
27. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
28. Liu, R., Chen, D., Liu, T., Xiong, Z., Yuan, Z.: Learning to predict 3d lane shape and camera pose from a single image via geometry constraints. In: AAAI (2022)
29. Liu, R., Yuan, Z., Liu, T., Xiong, Z.: End-to-end lane shape prediction with transformers. In: WACV (2021)
30. Liu, Y., Wang, Y., Wang, Y., Zhao, H.: Vectormapnet: End-to-end vectorized hd map learning. arXiv preprint arXiv:2206.08920 (2022)
31. Liu, Y., Yan, J., Jia, F., Li, S., Gao, Q., Wang, T., Zhang, X., Sun, J.: Petrv2: A unified framework for 3d perception from multi-camera images. arXiv preprint arXiv:2206.01256 (2022)
32. Liu, Z., Chen, S., Guo, X., Wang, X., Cheng, T., Zhu, H., Zhang, Q., Liu, W., Zhang, Y.: Vision-based uneven bev representation learning with polar rasterization and surface estimation. arXiv preprint arXiv:2207.01878 (2022)
33. Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D., Han, S.: Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. arXiv preprint arXiv:2205.13542 (2022)
34. Lu, J., Zhou, Z., Zhu, X., Xu, H., Zhang, L.: Learning ego 3d representation as ray tracing. In: ECCV (2022)
35. Ma, Y., Wang, T., Bai, X., Yang, H., Hou, Y., Wang, Y., Qiao, Y., Yang, R., Manocha, D., Zhu, X.: Vision-centric bev perception: A survey. arXiv preprint arXiv:2208.02797 (2022)
36. Mattyus, G., Luo, W., Urtasun, R.: Deeproadmapper: Extracting road topology from aerial images. In: ICCV (2017)
37. Mi, L., Zhao, H., Nash, C., Jin, X., Gao, J., Sun, C., Schmid, C., Shavit, N., Chai, Y., Anguelov, D.: Hdmapgen: A hierarchical graph generative model of high definition maps. In: CVPR (2021)

38. Qiao, L., Ding, W., Qiu, X., Zhang, C.: End-to-end vectorized hd-map construction with piecewise bezier curve. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13218–13228 (2023)
39. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI (2015)
40. Shin, J., Rameau, F., Jeong, H., Kum, D.: Instagram: Instance-level graph modeling for vectorized hd map learning. arXiv preprint arXiv:2301.04470 (2023)
41. Tabelini, L., Berriel, R., Paixao, T.M., Badue, C., De Souza, A.F., Oliveira-Santos, T.: Keep your eyes on the lane: Real-time attention-guided lane detection. In: CVPR (2021)
42. Tan, Y.Q., Gao, S.H., Li, X.Y., Cheng, M.M., Ren, B.: Vecroad: Point-based iterative graph exploration for road graphs extraction. In: CVPR (2020)
43. Wang, H., Li, T., Li, Y., Chen, L., Sima, C., Liu, Z., Wang, B., Jia, P., Wang, Y., Jiang, S., Wen, F., Xu, H., Luo, P., Yan, J., Zhang, W., Li, H.: Openlane-v2: A topology reasoning benchmark for unified 3d hd mapping. In: NeurIPS (2023)
44. Wang, J., Ma, Y., Huang, S., Hui, T., Wang, F., Qian, C., Zhang, T.: A keypoint-based global association network for lane detection. In: CVPR (2022)
45. 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)
46. Wu, D., Chang, J., Jia, F., Liu, Y., Wang, T., Shen, J.: TopoMLP: An simple yet strong pipeline for driving topology reasoning. In: The Twelfth International Conference on Learning Representations (2024), `https://openreview.net/forum?id=OgTW5JUFTW`
47. Xu, Z., Liu, Y., Sun, Y., Liu, M., Wang, L.: Centerlinedet: Road lane centerline graph detection with vehicle-mounted sensors by transformer for high-definition map creation. arXiv preprint arXiv:2209.07734 (2022)
48. Xu, Z., Sun, Y., Liu, M.: icurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving. IEEE Robotics and Automation Letters **6**(2), 1097–1104 (2021)
49. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)
50. 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)
51. Zhou, B., Krähenbühl, P.: Cross-view transformers for real-time map-view semantic segmentation. In: CVPR (2022)
52. Zhou, L., Zhang, C., Wu, M.: D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In: CVPRW (2018)
53. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: deformable transformers for end-to-end object detection. In: ICLR (2021)