Learning and Aggregating Lane Graphs for Urban Automated Driving

Martin Büchner[1*]   Jannik Zürn[1*]   Ion-George Todoran[2]   Abhinav Valada[1]   Wolfram Burgard[3]

[1]University of Freiburg   [2]Woven by Toyota   [3]University of Technology Nuremberg

Abstract

Lane graph estimation is an essential and highly challenging task in automated driving and HD map learning. Existing methods using either onboard or aerial imagery struggle with complex lane topologies, out-of-distribution scenarios, or significant occlusions in the image space. Moreover, merging overlapping lane graphs to obtain consistent large-scale graphs remains difficult. To overcome these challenges, we propose a novel bottom-up approach to lane graph estimation from aerial imagery that aggregates multiple overlapping graphs into a single consistent graph. Due to its modular design, our method allows us to address two complementary tasks: predicting ego-respective successor lane graphs from arbitrary vehicle positions using a graph neural network and aggregating these predictions into a consistent global lane graph. Extensive experiments on a large-scale lane graph dataset demonstrate that our approach yields highly accurate lane graphs, even in regions with severe occlusions. The presented approach to graph aggregation proves to eliminate inconsistent predictions while increasing the overall graph quality. We make our large-scale urban lane graph dataset and code publicly available at http://urbanlanegraph.cs.uni-freiburg.de.

1. Introduction

Most automated driving vehicles rely on the knowledge of their immediate surroundings to safely navigate urban environments. Onboard sensors including LiDARs and cameras provide perception inputs that are utilized in multiple tasks such as localization [7, 21, 27], tracking [4], or scene understanding [20, 24, 26, 37] to aggregate representations of the environment. However, robust planning and control typically require vastly more detailed and less noisy world models in the form of HD map data [12]. In particular, information on lane parametrization and connectivity is essential for both planning future driving maneuvers as well as high-level navigation tasks. Creating and maintaining HD maps in the form of lane graphs is a time-consuming and arduous
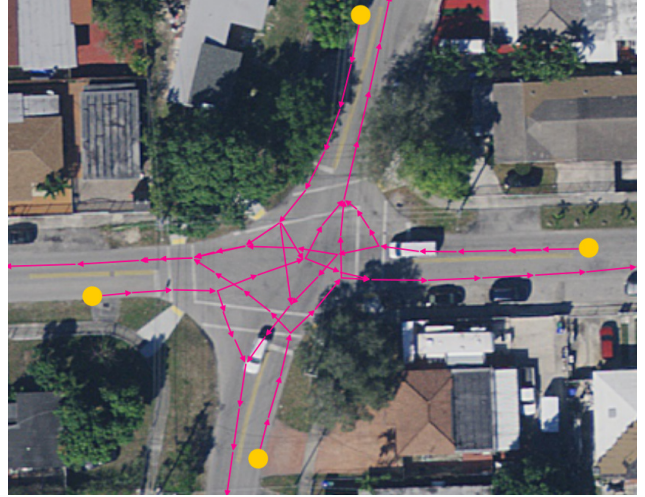

Figure 1. Our approach predicts accurate lane graphs from aerial images of complex urban environments. We visualize the estimated lane graph in magenta and indicate model initialization points with yellow circles.

task due to the large amount of detail required in the annotation and the data curation process including map updates based on local environment changes such as construction sites.

Previous approaches to lane graph estimation have shown shortcomings in predicting lane graphs due to multiple deficiencies: On the one hand, methods using onboard imagery typically degrade at complex real-world intersections and under significant occlusions, e.g., when following another vehicle [5, 6]. On the other hand, methods based on aerial imagery show reduced performance when confronted with occlusions in the bird's-eye-view (BEV) due to, e.g., vegetation or shadows, and suffer from catastrophic drift when unconstrained in out-of-distribution scenarios [30]. Previous works treat intersections and non-intersections inherently differently [15] and thus require elaborated heuristics and post-processing to merge single predictions into a consistent lane graph. Moreover, prior works do not focus on use cases where multiple predicted graphs must be merged into a single consistent solution, which is essential for enabling the automatic generation of highly detailed lane graphs of large contiguous regions.

---

*Equal contribution

Related to the aforementioned challenges, we propose a novel two-stage graph neural network (GNN) approach termed LaneGNN that operates on single aerial color images for lane graph prediction. Inspired by methods in the field of trajectory prediction [8], we formulate a bottom-up approach according to which we place a virtual agent into a local crop of the aerial image and predict reachable successor lane graphs from its positions. To transform multiple disjoint local solutions into a single global solution, we aggregate a global representation by iteratively inferring the lane graph from consecutive poses, ultimately imitating real-world driving behavior. This iterative approach not only increases the predicted area covered but also improves graph accuracy based on data association and rejection. Note that we do not require any human in the loop to perform the graph aggregation. We visualize the output of our graph aggregation procedure in Fig. 1, in which we superimpose the predicted graph on the aerial image input. Using this framework, we envision two applications: ego-centered successor path prediction and full lane graph estimation by aggregation.

To summarize, the main contributions of this work are:

- An innovative bottom-up approach to lane graph estimation in challenging environments that explicitly encodes graph-level lane topology from input aerial images in a scenario-agnostic manner.
- A novel graph aggregation scheme enabling robust and method-agnostic merging of graph-level predictions.
- The large-scale lane graph dataset *UrbanLaneGraph* comprising high-resolution aerial images aligned with dense lane graph annotations aggregated from the Argoverse2 dataset that we make publicly available.
- Extensive experiments and ablation studies demonstrating the significance of our findings.

2. Related Works

In recent years, the prediction of topological road features such as road graphs and lane graphs have been extensively studied. In our discussion, we differentiate between road graph learning and lane graph learning. While road graphs encode the topological connections between road segments, lane graphs describe the locations and connectivity between all lanes, resulting in a spatially much denser graph. Many prior works focus on vehicle trajectory prediction, conditioned on HD map features such as lane centerline and boundary positions [9, 10, 28]. These models do not aim at exclusively predicting the road or lane graph structure from onboard vehicle images or aerial images but instead at predicting future vehicle states such as position and orientation.

Road Graph Learning: Prior works investigate estimating road graphs from both onboard sensors [18] and from aerial images [1, 22, 34] or focus on extracting pixel-level road segmentation from images and extracting graphical road rep-

resentations, i.e., using morphological image operators or graph neural networks to extract the connectivity between different roads within the image [1, 19]. Other approaches investigate iterative methods and interpret road graph prediction as a sequential prediction task [2, 22].

Lane Graph Learning from Vehicle Data: Some earlier works in lane graph learning from onboard vehicle sensors such as cameras and LiDAR formulate lane extraction as an image-based lane centerline regression task [16]. Homayounfar *et al.* [17] aggregate onboard LiDAR data on highways and leverage a recurrent neural network to generate highway lane graphs in an iterative manner. Zhou *et al.* [35] utilize the OpenStreetMap database and a semantic particle filter to accumulate projected semantic predictions from a vehicle ego-view into a map representation. Zhang *et al.* [31] propose an online road map extraction system for a sensor setup onboard a moving vehicle and construct a graph representation of the road network using a fully convolutional neural network. More recently, Can *et al.* proposed two different methods [5, 6] for lane connectivity learning in intersection scenarios from onboard camera images.

Lane Graph Learning from Birds-Eye-View Data: Despite the advantages of leveraging readily available aerial data for training lane graphs, only a few works considered using aerial images as an input modality for the graph learning task. Zürn *et al.* [36] propose a lane centerline regression model jointly with a Graph R-CNN backbone to predict nodes and edges of the lane graph from a local aggregated bird's-eye-view image crop. More recently, He *et al.* [15] propose a two-stage graph estimation pipeline. They first extract lanes at non-intersection areas and subsequently predict the connectivity of each pair of lanes, and extract the valid turning lanes to complete the map at intersections.

In contrast to existing works, we do not estimate the complete lane graph visible in a given crop but only the part of the graph that is reachable from a virtual agent pose located within the crop, simplifying the graph estimation problem based on reduced topological complexity. Furthermore, we leverage a GNN to explicitly model the relationships between graph nodes, allowing us to leverage recent developments in the field of geometric deep learning. This explicit graph encoding and prediction allows us to formulate a model that does not internally differentiate between intersection areas and non-intersection areas, in contrast to some related works. Additionally, we propose a novel large-scale dataset for lane graph estimation from aerial images, allowing the research community to easily evaluate and compare their approaches.

3. Dataset

To evaluate our approach on challenging real-world data, we compiled the *UrbanLaneGraph* dataset. It is a first-of-its-
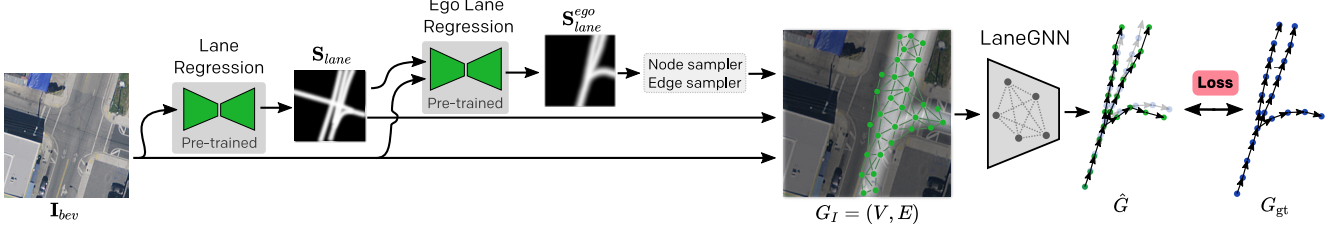
Figure 2. Overview of our LaneGNN model $\mathcal{M}$ predicting successor lane graphs. As a pre-training step, we train lane centerline and ego lane centerline regressor models. The ego lane regression $\mathbf{S}_{lane}^{ego}$ is used as a prior for sampling proposal nodes V in corridors that have a high likelihood of entailing the successor graph. The model learns a binary classification of node and edge scores while also predicting the probability of a node being an endpoint of a given lane segment.

Dataset	City	Lane Splits/Merges	Total Length [km]
UrbanLaneGraph *(from Argoverse2)*	Palo Alto	4752	796.4
	Austin	8495	531.7
	Miami	7642	850.8
	Pittsburgh	14610	1314.3
	Washington D.C.	2066	739.6
	Detroit	5424	990.5
LaneExtraction [15]	Boston, Seattle, Phoenix, Miami	2262	398.6
NuScenes [11]	Boston	1630	76.9

Table 1. Key statistics of our *UrbanLaneGraph* dataset, aggregated from the Argoverse2 dataset [29], used for our experiments, compared with other recent datasets for lane graph estimation.

kind dataset for large-scale lane graph estimation from aerial images. The dataset contains aerial images from the cities of Austin, Miami, Pittsburgh, Palo Alto, Detroit, and Washington DC. The images have a resolution of $15\,\mathrm{cm}$ per pixel. To obtain the corresponding lane annotations, we leverage the Argoverse2 dataset [29] which entails lane graphs for large sections of the respective cities. The lane graphs in the Argoverse2 dataset are provided on a per-scenario basis, covering only small local areas in one graph. Therefore, we collect all local lane graphs of each city and aggregate them into one globally consistent graph per city, thereby removing inconsistent or redundant nodes or edges. The annotated regions feature a diverse range of environments including urban, suburban, and rural regions with complex lane topologies. Accumulated over all cities, the overall length of all lanes spans over $5.000\,\mathrm{km}$. We split each city into disjoint training and testing regions. We list key dataset statistics in Tab. 1, indicating the scale of our generated dataset compared with other recently proposed datasets containing graph annotations. For more details on the dataset and exemplary visualizations, please refer to Sec. S.1 in the supplementary material.

4. Technical Approach

Our approach is divided into two stages: lane graph learning and lane graph aggregation. In the first stage (Sec. 4.1), we train our GNN model, denoted as LaneGNN, to predict the successor lane graph, entailing the nodes and edges

that can be logically visited from the pose of a virtual vehicle agent. In the second phase (Sec. 4.2), we use our trained LaneGNN model to traverse a large map area. This is achieved by selecting an initial starting pose and predicting the successor lane graph from this pose. Subsequently, we iteratively estimate the traversable lane graph from the current pose and move forward along the predicted graph while aggregating. In the following, we detail both stages of our approach.

4.1. Lane Graph Learning

We formulate the task as a supervised learning problem where a successor lane graph $\hat{G}$ is estimated based on an aerial image $\mathbf{I}_{bev}$. First, a directed graph $G_I = (V, E)$ covering relevant regions is constructed by sampling from likely regions of $\mathbf{I}_{bev}$ (see Fig. 2). For all our models and experiments we choose a spatial resolution of 256×256 pixels. The graph comprises a set of nodes $i \in V$ that are connected via directed edges $E \subseteq \{(i,j)|(i,j) \in V^2 \text{ and } i \neq j\}$ that constitute potentially valid lane graph edges. The graph is attributed using both node features $\mathbf{X} \in \mathbb{R}^{\|V\| \times D}$ and edge features $\mathbf{X}_e \in \mathbb{R}^{\|E\| \times (D_{geo} + D_{bev})}$, where D_{geo} and D_{bev} denote the dimensionality of the involved edge features (see Sec. 4.1). The overall model estimates an output graph $\hat{G} = \mathcal{M}(G_I|\theta)$, where $\mathcal{M}$ is parameterized by network model weights $\theta := (\theta_{reg}, \theta_{GNN})$.

Lane Regression and Graph Construction: We train two regression networks: Firstly, a centerline regression network predicting the likelihood map of lane centerlines $\mathbf{S}_{lane}$, and secondly, a segmentation network predicting all reachable lanes $\mathbf{S}_{lane}^{ego}$ starting from the initial virtual agent pose at the bottom center of $\mathbf{I}_{bev}$. We use identical PSPNet [33] architectures with a ResNet-152 feature extractor for this task. We sample equally-distributed node positions using Halton sequences [14] that are later filtered based on the obtained ego-lane segmentation mask $\mathbf{S}_{lane}^{ego}$, which serves as a region of interest for sampling (see Fig. 2). Directed edges E among nodes are initialized for pairs of nodes with a Euclidean distance $d_{ij} \in [d_{min}, d_{max}]$. Initial node features $\mathbf{X}$ are crafted solely based on their 2D positions $\mathbf{x}_i = (x_i, y_i)$,

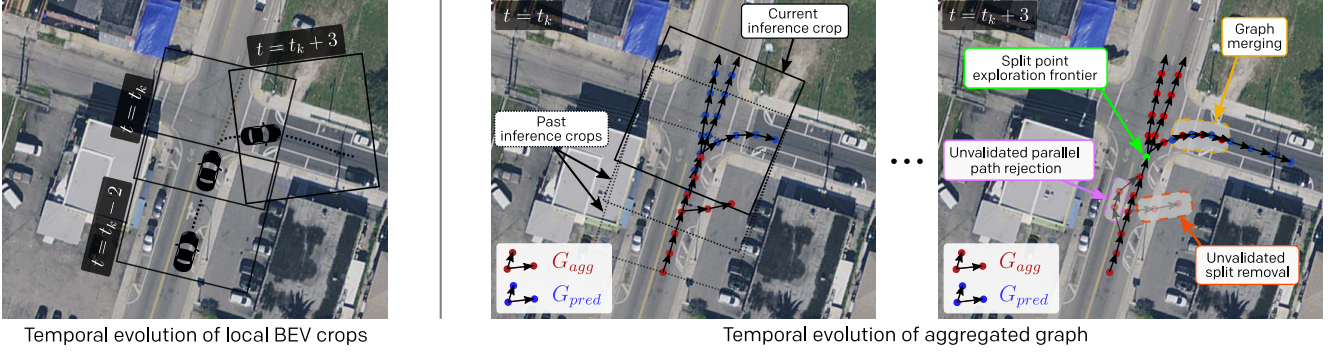Temporal evolution of local BEV crops | Temporal evolution of aggregated graph

Figure 3. In our graph aggregation procedure, we iteratively obtain oriented image crops based on virtual agent poses along the currently predicted successor graph. For each crop, our LaneGNN model predicts a successor graph G_{pred}, which is aggregated into a globally consistent lane graph G_{agg}.

while geometric edge features are defined as

$$\mathbf{x}_{ij,geo} = \left(\tan^{-1} \frac{\Delta y_{ij}}{\Delta x_{ij}}, \sqrt{\Delta x_{ij}{}^2 + \Delta y_{ij}{}^2}, \overline{x}_{ij}, \overline{y}_{ij} \right), \quad (1)$$

where Δx_{ij} and Δy_{ij} represent node position differences and $\overline{x}_{ij}$, $\overline{y}_{ij}$ are edge middle point coordinates. In addition to the geometric edge feature, we generate aerial edge features: Per edge, a small oriented region of $\mathbf{I}_{bev}$ and the lane segmentation $\mathbf{S}_{lane}$ is obtained based on the direction of the edge, which provides $\mathbf{X}_{ij,bev} = [\mathbf{I}^*_{bev}, \mathbf{S}^*_{lane}]$ with dimension $D_{bev} = 4 \times 32 \times 32$.

Feature Encoding and Message Passing: In our approach, we estimate edge probabilities, node probabilities, and whether a node is terminal. While the nodes themselves hold only unidirectional information, we encode the notion of direction using the proposed edge feature as outlined above. We utilize a causal variant of neural message passing as proposed by Brasó *et al.* [3]. By imposing a causality prior, our network encodes predecessor and successor features during message passing. This formulation of message passing renders our approach direction-aware. Initial node and geometric edge features are encoded using multi-layer perceptrons $f^\square_{enc}$ (MLP) while the aerial edge feature is transformed using a ResNet-18 architecture $f^{e,bev}_{enc}$. The geometric and aerial edge features are concatenated and fused subsequently to arrive at various node and edge embeddings $\mathbf{H}^{(0)}_v$ and $\mathbf{H}^{(0)}_e$:

$$f^{e,bev}_{enc}(\mathbf{X}_{e,bev}) = \mathbf{H}^{(0)}_{e,bev}, \ f^{e,geo}_{enc}(\mathbf{X}_{e,geo}) = \mathbf{H}^{(0)}_{e,geo}, \quad (2)$$

$$f^v_{enc}(\mathbf{X}) = \mathbf{H}^{(0)}_v, \ f^e_{fuse}([\mathbf{H}^{(0)}_{e,geo}, \mathbf{H}^{(0)}_{e,bev}]) = \mathbf{H}^{(0)}_e. \quad (3)$$

In the following, multiple message-passing steps are performed using various ReLU-activated MLPs denoted by $f_\square$ as follows. The edge feature is updated based on the current neighboring node features $\mathbf{h}^{(l-1)}_i$, $\mathbf{h}^{(l-1)}_j$ and the edge feature $\mathbf{h}^{(l-1)}_{ij}$:

$$\mathbf{h}^{(l)}_{ij} = f_e \left(\left[\mathbf{h}^{(l-1)}_i, \mathbf{h}^{(l-1)}_j, \mathbf{h}^{(l-1)}_{ij} \right] \right). \quad (4)$$

Messages $\mathbf{m}^{(l)}_{ij}$ are crafted based on either predecessors $\mathcal{N}_{pred}(i)$ or successors $\mathcal{N}_{succ}(i)$ of a node i and propagated based on the constructed adjacency. In the next step, all predecessors and successor messages, respectively, are aggregated using a permutation-invariant sum:

$$\mathbf{h}^{(l)}_{i,pred} = \sum_{j \in \mathcal{N}_{pred}(i)} \underbrace{f^{pred}_v \left(\left[\mathbf{h}^{(l-1)}_j, \mathbf{h}^{(l)}_{ij}, \mathbf{h}^{(0)}_j \right] \right)}_{\mathbf{m}^{(l)}_{ji}}, \quad (5)$$

$$\mathbf{h}^{(l)}_{i,succ} = \sum_{j \in \mathcal{N}_{succ}(i)} \underbrace{f^{succ}_v \left(\left[\mathbf{h}^{(l-1)}_j, \mathbf{h}^{(l)}_{ij}, \mathbf{h}^{(0)}_j \right] \right)}_{\mathbf{m}^{(l)}_{ij}}. \quad (6)$$

Note that the message crafting includes skip connections to initial node embeddings $\mathbf{h}^{(0)}_j$. Nodes are updated by combining the two features using concatenation:

$$\mathbf{h}^{(l)}_i = f_v \left(\left[\mathbf{h}^{(l)}_{i,pred}, \mathbf{h}^{(l)}_{i,succ} \right] \right). \quad (7)$$

Finally, sigmoid-valued edge scores $\hat{e}_{ij}$ and node score $\hat{s}_i$ are predicted from the obtained edge and node embeddings. In a separate network head, we classify each node as being a terminal node or not, denoted as a scalar $\hat{t}_i$. Therefore, we optimize the following combined binary cross entropy:

$$\mathcal{L} = -\sum_{|V|} s_i \log \hat{s}_i - \sum_{|V|} t_i \log \hat{t}_i - \sum_{|E|} e_{ij} \log \hat{e}_{ij}. \quad (8)$$

The ground truth graph G_{GT} as a learning target (s_i, t_i, e_{ij}) is generated based on the map annotations for the given cropped region and the corresponding closest nodes. This is further outlined in the supplementary material in Sec. S.3.

4.2. Iterative Temporal Graph Aggregation

In the second stage of our approach, we aggregate local successor graphs into a globally consistent lane graph. First, we prune the predicted per-crop lane graph, and second, we iteratively aggregate the sparse graphs. In the following, we detail both components.

4

Figure 4. Comparison of raw and pruned lane graph predictions. On the left-hand side, we visualize estimated edge scores $\hat{e}_{ij}$ and node scores $\hat{s}_i$ using the same color scaling. On the right-hand side, we visualize the pruned graph. The circled numbers denote the order of traversal based on estimated terminal node scores.

Pruning: The graph prediction obtained from the LaneGNN model follows the graph connectivity initially generated during sampling. Since the model prediction contains a number of redundant paths with high predicted node and edge scores, we prune the obtained solution to obtain sparse lane representations. We formulate the graph pruning problem as a search problem from a starting node to possibly multiple predicted terminal nodes (see Sec. 4.1). Predicted lane split points should coincide with actual split points. Thus, different branches should share the same set of edges up to a split point. We use Dijkstra's algorithm to iteratively find high-score paths between the initial pose and terminal nodes, ordered from high to low scores until all terminal nodes are reached. Edge scores contained in found paths are set to zero. In Fig. 4, we visualize the output of this step.

Aggregation: We iteratively aggregate predicted successor lane graphs $G_{pred} = (V_{pred}, E_{pred})$ into a globally consistent and complete graph $G_{agg} = (V_{agg}, E_{agg})$ as depicted in Fig. 3. The predicted successor graph G^t_{pred} is added to the current aggregated graph G^{t-1}_{agg} at time step t: $G^t_{agg} \leftarrow \texttt{aggregate}(G^t_{pred}, G^{t-1}_{agg})$. Plausible branches of G_{pred} are merged into G_{agg} and thus extend the aggregated graph with every iteration if new grounds are covered in the respective crops. The next virtual agent pose is extracted from the set of forward-facing edges of G_{agg} given the current pose. Only edges with a significant weight due to previous aggregations are selected for this set. As the node positions differ slightly with each model forward pass we observe roughly similar paths in the lateral sense wrt. the ground-truth graph. However, along the longitudinal dimension of a branch, we observe deviations in node position, which must be circumvented when aggregating the graph. Based on this, we only take the lateral deviation wrt. G^t_{agg} into account when merging G^t_{pred}. Thus, G^t_{agg} is only updated in a lateral sense while the longitudinal misalignment of the two sets of nodes is neglected (see also Sec. S.5 in the supplementary material). The nodes of G^t_{pred} have a

weight of 1 while a node of G^t_{agg} holds a weight equal to the number of merges it has observed so far. If a novel node $i \in V_{pred}$ is not close to any other node in $k \in V_{agg}$ it is added to the aggregated graph including its incident edges. This ultimately allows a weighting-based merging of arbitrary pairs of graphs, which is used for global lane graph estimation (see Sec. 5.4).

We observe that the more graphs we aggregate, the more we are certain about the significance of particular graph branches. Following a weighting-based approach allows us to set certain thresholds that allow modification of G_{agg}. Thus, implausible graph branches, semantically similar parallel paths, redundant edges, and isolated nodes are deleted based on confidence and distance as well as angle criteria (see Fig. 3). As a result, we are able to decrease the number of false positive split and merge points to obtain more consistent global graphs. We observe that this approach greatly improves lane graph prediction accuracy in difficult occluded and out-of-distribution scenarios since the sum of model predictions covering the same region shed light onto what potentially constitutes, e.g., a valid and an invalid branch.

Multiple forward passes naturally lead to a multitude of lane split points (both true positive and false positive splits). We interpret each split point as an element of an exploration frontier. A queue of unexplored, high-probability graph branches is maintained, which is queried in a depth-first manner as soon as the currently traversed branch terminates. Following the weighting-based approach, a branch is only explored if its depth-limited oriented successor tree weight exceeds a certain level of confidence. Due to this flexible approach, we can essentially handle arbitrary lane graph topologies with a single holistic approach. For more details on the graph pruning and aggregation procedures, please refer to the supplementary material. Finally, we apply multiple iterations of Laplacian smoothing to G^t_{pred}, which modifies the original node positions in order to even out position irregularities caused by sampling while keeping the adjacency represented by E^t_{pred} constant.

5. Experimental Results

In the following, we present our experimental findings. We first define and illustrate three tasks on which we benchmark our method. Subsequently, we describe the evaluation metrics and compare against other methods as well as our own baselines. We provide extensive qualitative and quantitative evaluations on our *UrbanLaneGraph* dataset.

5.1. Proposed Tasks

We propose three distinct and complementary tasks. In the first task, successor lane graph prediction *(Successor-LGP, Sec. 5.3)*, we aim at predicting a feasible ego-reachable successor lane graphs from the current pose of the virtual agent. The purpose of the Successor-LGP task is to measure

Table 2. Quantitative results of our model including ablation studies, in comparison with baseline models for the Successor-LGP task on our *UrbanLaneGraph* dataset. P/R denotes Precision/Recall. For our LaneGNN model variants, we denote CMP as causal message passing, aerial node features as AerN, aerial edge features as AerE, and the ego-lane regression with $\mathbf{S}^{ego}_{lane}$. For all metrics, higher values mean better results.

Model	CMP	AerN	AerE	$\mathbf{S}^{ego}_{lane}$	TOPO P/R ↑	GEO P/R ↑	APLS ↑	SDA$_{20}$ ↑	SDA$_{50}$ ↑	Graph IoU ↑
Skeletonized Regression	–	–	–	–	0.597/0.613	0.578/0.601	**0.315**	0.020	0.185	0.180
LaneGraphNet [36]	–	–	–	–	0.0/0.0	0.0/0.0	0.179	0.0	0.0	0.063
	✗	✗	✓	✓	0.549/0.677	0.548/0.671	0.188	0.168	0.323	0.312
	✓	✓	✗	✓	0.562/0.656	0.562/0.655	0.192	0.151	0.298	0.320
LaneGNN (ours)	✓	✗	✗	✓	0.545/0.693	0.545/0.688	0.200	0.188	0.311	0.336
	✓	✗	✓	✗	0.578/0.669	0.577/0.659	0.150	0.132	0.227	0.250
	✓	✗	✓	✓	**0.600/0.699**	**0.599/0.695**	0.202	**0.227**	**0.377**	**0.347**

the prediction quality of potential future driving paths when no HD map coverage is available. In the second task, full lane graph prediction (*Full-LGP*, Sec. 5.4), we evaluate the quality of regionally aggregated lane graphs in the context of HD map estimation. This task aims at measuring the predictive power of our full two-stage model performing lane graph inference and graph aggregation in conjunction. For such purposes, the full ground truth lane graph of a given map area is compared to the aggregated predictions of our model. Finally, we carry out a high-level path planning task (Sec. 5.5) on the predicted lane graphs, intended to analyze the fidelity of routes planned on the predicted graphs.

5.2. Evaluation Metrics

We leverage multiple complementary metrics for performance evaluation as detailed below.

Graph IoU: This metric measures the intersection-over-union (IoU) of two graphs rendered as a binary image [36], where pixels closer than $d = 5$ pixels are assigned the label 1 and the remaining pixels the label 0. Equivalent to the evaluation of semantic segmentation models, we determine the IoU values for the non-zero pixels.

The **APLS metric** sums the differences in optimal path lengths between nodes in the ground truth graph G and the proposal graph G' [25]. The APLS metric scales from 0 (worst) to 1 (best). Formally, it is defined as

$$\text{APLS} = 1 - \frac{1}{N_p} \sum \min \left\{ 1, \frac{|d(v_1, v_2) - d(v'_1, v'_2)|}{d(v_1, v_2)} \right\}, \quad (9)$$

where v_i and v'_i are nodes in G and G', respectively. N_p denotes the number of paths in G and $d(\cdot, \cdot)$ is the path length. For more details, please refer to [25].

TOPO / GEO metrics: Following previous works in road network extraction and lane graph estimation, we use the GEO metric and the TOPO metric. For definitions and details on these metrics, please refer to [15] and to the supplementary material.

Split Detection Accuracy (SDA$_R$): This metric evaluates how accurately a model predicts the lane split within a circle of radius R pixels from a given ground truth lane split.

5.3. Successor Lane Graph Prediction

In the following, we evaluate LaneGNN by ablating and comparing it with two baselines on the Successor-LGP task: morphological skeletonization of the ego-lane regression as well as a modified LaneGraphNet [36] to be used for successor lane graph prediction. We list quantitative results in Tab. 2. Our results demonstrate that none of the baseline methods are capable of estimating accurate lane graphs given the challenging topology of the lane graphs in the dataset. The LaneGraphNet [36] model fails to model the graph for many samples, yielding low scores in all metrics. Despite its simplicity, the skeletonized regression model achieves the highest APLS score and comparably high GEO/TOPO scores. However, it fails to accurately predict lane split points, resulting in low SDA scores. Increasing the number of nodes of the skeleton leads to many more false positive splits and thus deteriorates further.

Regarding different variants of LaneGNN, we find that, e.g., causal message passing (CMP) increases performance over standard message passing, which underlines the significance of the imposed causality prior for lane graph learning. In order to show the efficacy of computationally more demanding aerial image edge features (AerE) compared to uni-directional aerial image node features (AerN), we ablate on this in Tab. 2 as well. We observe stark increases across the TOPO, GEO and SDA metrics when utilizing aerial edge features. Lastly, we replace the ego-lane segmentation mask $\mathbf{S}^{ego}_{lane}$ used for sampling with the standard lane segmentation mask. Our findings show that especially APLS, SDA, and Graph IoU drastically decrease as graph estimation becomes more difficult due to a generally enlarged sampling region (see Fig. 2).

These results are further illustrated in Fig. 5, where we show qualitative comparisons of predictions of our best-performing model with predictions from the two baselines. We find that the quality of predictions by LaneGraphNet [36] is generally low, rendering it unsuitable for the task. The skeletonized regression baseline is capable of following basic lane graph topologies, but lane split points cannot be resolved accurately. In contrast, our LaneGNN model is capable of modeling most graphs with high accuracy; both in intersection areas and in straight road sections. For more

Figure 5. Qualitative results on the Successor-LGP task. We compare predictions of our model with LaneGraphNet [36] and a morphological image skeletonization baseline. Predicted nodes are visualized with points while predicted edges are visualized as directed arrows. We illustrate failure cases in the two rightmost columns. Best viewed zoomed in.

Table 3. Quantitative evaluation for the Full-LGP task on the test-set of our *UrbanLaneGraph* dataset. We compare a baseline model with graphs aggregated with a naïve aggregation scheme and our iterative temporal aggregation scheme. P/R denotes Precision/Recall. Higher values mean better results.

Model	TOPO P/R ↑	GEO P/R ↑	APLS ↑	Graph IoU ↑
LaneExtraction [15]	0.405/0.507	0.491/0.454	0.072	0.213
Aggregation (naïve)	0.366/0.654	0.523/**0.727**	0.101	0.376
Aggregation (ours)	**0.481/0.670**	**0.649**/0.689	**0.103**	**0.384**

results, please refer to the supplementary material, Sec. S.6.

5.4. Full Lane Graph Prediction

For the Full-LGP task, we compare our approach with LaneExtraction [15]. Since their used graph representation is incompatible with ours, we train it on their provided dataset. To allow for a fair comparison, we evaluate both our method and LaneExtraction only on scenes in the city of Miami, Florida, as it is contained in both of the datasets. We select a testing region that is not part of the training data for either of the models. To initialize our aggregation scheme, we select starting poses obtained from intermediate segmentation predictions including yaw angles of the LaneExtraction model. Tab. 3 lists the evaluation results for the Full-LGP task obtained with LaneExtraction [15] and the results obtained with our LaneGNN model in conjunction with two aggregation schemes: a naïve aggregation scheme baseline

and our full aggregation scheme. The naïve aggregation scheme merges nodes in close proximity while not relying on unvalidated split/merge or parallel path removal as well as the lateral weighting-based merging (Sec. 4.2). Our experiments show that our aggregation scheme outperforms both the LaneExtraction model and the naïve aggregation scheme on nearly all evaluation metrics. We note that our method improves the TOPO/GEO precision metrics while maintaining similar recalls due to better handling of redundant nodes. Fig. 6 illustrates successive aggregations from our model while indicating the used initialization points from the LaneExtraction model. Since our aggregation approach does not differentiate between intersection and non-intersection regions, it does not deteriorate in regions that do not exactly fit this categorization. Furthermore, our model exhibits superior performance in reduced visibility settings introduced by stark illumination changes or road occlusions from vegetation, as illustrated in Fig. 6. One of the decisive assets of our bottom-up method is that it allows to *explore* regions that are missed by LaneExtraction [15] as they are entailed in their predicted segmentation masks. For more qualitative and quantitative results, please refer to the supplementary material, Sec. S.7.

5.5. Path Planning

To illustrate the efficacy of our aggregation scheme, we evaluate the quality of the lane graph on a planning task. We

Figure 6. Qualitative results on the Full-LGP task. We visualize predictions of LaneExtraction [15] (top row) and aggregated LaneGNN predictions (bottom row). Our model is initialized at poses using predicted lane direction masks of LaneExtraction (indicated with yellow circles). Best viewed zoomed in.

Table 4. Quantitative evaluation for the planning task. MMD denotes mean minimum distance, MED denotes mean endpoint distance, and SR denotes the path planning success rate.

Model	MMD [m] ↓	MED [m] ↓	SR ↑
LaneExtraction [15]	157.0	339.4	0.47
Aggregation (ours)	2.2	19.7	0.46

generate 1000 randomly selected starting poses in the Miami graph test area from which a plan to a random goal within the graph must be found, using A* search. We place the points such that a maximal optimal route length of 200 m is not exceeded. To evaluate the planned routes, we compare the mean minimum distance (MMD) and the mean route endpoint distance (MED) between the paths on the predicted graph and the ground truth graph, respectively. We also report the success rate (SR), indicating the number of cases in which a path between start and goal exists. We list the results in Tab. 4. While the SR for our aggregation scheme and the LaneExtraction predictions is similar, the low MMD and MED values of our aggregation scheme indicate that our generated lane graph entails shorter and more direct paths, compared to LaneExtraction. We show additional results in the supplementary material, Sec. S.8.

5.6. Limitations

Due to its bottom-up architecture, the proposed approach performs well for most evaluated scenes in urban and suburban surroundings but struggles with highly complex graph topologies such as multi-lane intersections or roundabouts.

Moreover, due to the iterative formulation of our aggregation scheme, the inference time of our approach increases with the number of nodes and edges. To speed up inference time, future work might include adaptively changing the distance between consecutive virtual agent positions and leveraging efficient neighborhood lookup methods such as k-d trees. Parallel execution of multiple agents would additionally boost run time to match top-down approaches and is feasible in terms of the proposed aggregation scheme.

6. Conclusion

In this work, we presented a novel lane graph estimation framework complemented with a novel dataset comprising aerial images. We showed that formulating the lane graph estimation problem as bottom-up graph neural network approach leveraging agent-centric views yields promising results. In addition, we presented a novel aggregation scheme to merge successive lane graphs to produce large-scale solutions. A first-of-its-kind dataset and benchmark for lane graph estimation from aerial images will enable further research in this field. Future work could address end-to-end training and exploiting further modalities such as onboard vehicle cameras for additional context information.

References

[1] Wele Gedara Chaminda Bandara, Jeya Maria Jose Valanarasu, and Vishal M Patel. Spin road mapper: Extracting roads from aerial images via spatial and interaction space graph reasoning for autonomous driving. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 343–350. IEEE, 2022. 2

[2] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and David DeWitt. Roadtracer: Automatic extraction of road networks from aerial images. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4720–4728, 2018. 2

[3] Guillem Brasó and Laura Leal-Taixé. Learning a neural solver for multiple object tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 4

[4] Martin Büchner and Abhinav Valada. 3d multi-object tracking using graph neural networks with cross-edge modality attention. *IEEE Robotics and Automation Letters*, 7(4):9707–9714, 2022. 1

[5] Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. Structured bird's-eye-view traffic scene understanding from onboard images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15661–15670, 2021. 1, 2

[6] Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. Topology preserving local road network estimation from single onboard camera image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17263–17272, 2022. 1, 2

[7] Daniele Cattaneo, Matteo Vaghi, and Abhinav Valada. Lcdnet: Deep loop closure detection and point cloud registration for lidar slam. *IEEE Transactions on Robotics*, 38(4):2074–2093, 2022. 1

[8] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *Conference on Robot Learning*, pages 86–99. PMLR, 2020. 2

[9] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17222–17231, 2022. 2

[10] Nemanja Djuric, Henggang Cui, Zhaoen Su, Shangxuan Wu, Huahua Wang, Fang-Chieh Chou, Luisa San Martin, Song Feng, Rui Hu, Yang Xu, et al. Multixnet: Multiclass multistage multimodal motion prediction. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 435–442. IEEE, 2021. 2

[11] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *IEEE Robotics and Automation Letters*, 7(2):3795–3802, 2022. 3

[12] Nikhil Gosala and Abhinav Valada. Bird's-eye-view panoptic segmentation using monocular frontal view images. *IEEE Robotics and Automation Letters*, 7(2):1968–1975, 2022. 1

[13] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. 12

[14] J Halton and G Smith. Radical inverse quasi-random point sequence, algorithm 247. *Commun. ACM*, 7(12):701, 1964. 3

[15] Songtao He and Hari Balakrishnan. Lane-level street map extraction from aerial imagery. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2080–2089, 2022. 1, 2, 3, 6, 7, 8, 13, 21

[16] Namdar Homayounfar, Wei-Chiu Ma, Shrinidhi Kowshika Lakshmikanth, and Raquel Urtasun. Hierarchical recurrent attention networks for structured online maps. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3417–3426, 2018. 2

[17] Namdar Homayounfar, Wei-Chiu Ma, Justin Liang, Xinyu Wu, Jack Fan, and Raquel Urtasun. Dagmapper: Learning to map by discovering lane topology. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2911–2920, 2019. 2

[18] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Shenlong Wang, and Raquel Urtasun. Convolutional recurrent network for road boundary extraction. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9512–9521, 2019. 2

[19] Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3438–3446, 2017. 2

[20] Rohit Mohan and Abhinav Valada. Amodal panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21023–21032, 2022. 1

[21] Kürsat Petek, Kshitij Sirohi, Daniel Büscher, and Wolfram Burgard. Robust monocular localization in sparse hd maps leveraging multi-task uncertainty estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4163–4169. IEEE, 2022. 1

[22] Yong-Qiang Tan, Shang-Hua Gao, Xuan-Yi Li, Ming-Ming Cheng, and Bo Ren. Vecroad: Point-based iterative graph exploration for road graphs extraction. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8910–8918, 2020. 2

[23] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991. 11

[24] Abhinav Valada, Ankit Dhall, and Wolfram Burgard. Convoluted mixture of deep experts for robust semantic segmentation. In *IEEE/RSJ International conference on intelligent robots and systems (IROS) workshop, state estimation and terrain perception for all terrain mobile robots*, volume 2, 2016. 1

[25] Adam Van Etten, Dave Lindenbaum, and Todd M Bacastow. Spacenet: A remote sensing dataset and challenge series. *arXiv preprint arXiv:1807.01232*, 2018. 6

[26] Johan Vertens and Wolfram Burgard. Usegscene: Unsupervised learning of depth, optical flow and ego-motion with

semantic guidance and coupled networks. *arXiv preprint arXiv:2207.07469*, 2022. 1

[27] Niclas Vödisch, Daniele Cattaneo, Wolfram Burgard, and Abhinav Valada. Continual slam: Beyond lifelong simultaneous localization and mapping through continual learning. *arXiv preprint arXiv:2203.01578*, 2022. 1

[28] Jingke Wang, Tengju Ye, Ziqing Gu, and Junbo Chen. Ltp: Lane-based trajectory prediction for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17134–17142, 2022. 2

[29] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 3, 11

[30] Zhenhua Xu, Yuxuan Liu, Lu Gan, Yuxiang Sun, Xinyu Wu, Ming Liu, and Lujia Wang. Rngdet: Road network graph detection by transformer in aerial images. *IEEE Transactions on Geoscience and Remote Sensing*, 2022. 1

[31] Li Zhang, Faezeh Tafazzoli, Gunther Krehl, Runsheng Xu, Timo Rehfeld, Manuel Schier, and Arunava Seal. Hierarchical road topology learning for urban map-less driving. *arXiv preprint arXiv:2104.00084*, 2021. 2

[32] Tongjie Y Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984. 15

[33] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 3, 16

[34] Lichen Zhou, Chuang Zhang, and Ming Wu. D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 182–186, 2018. 2

[35] Yiyang Zhou, Yuichi Takeda, Masayoshi Tomizuka, and Wei Zhan. Automatic construction of lane-level hd maps for urban scenes. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6649–6656. IEEE, 2021. 2

[36] Jannik Zürn, Johan Vertens, and Wolfram Burgard. Lane graph estimation for scene understanding in urban driving. *IEEE Robotics and Automation Letters*, 6(4):8615–8622, 2021. 2, 6, 7

[37] Jannik Zürn, Sebastian Weber, and Wolfram Burgard. Trackletmapper: Ground surface segmentation and mapping from traffic participant trajectories. In *6th Annual Conference on Robot Learning*, 2022. 1

Learning and Aggregating Lane Graphs for Urban Automated Driving

- Supplementary Material -

Martin Büchner*, Jannik Zürn*, Ion-George Todoran, Abhinav Valada, and Wolfram Burgard

In our supplementary material, we expand upon multiple aspects of our paper. In Sec. S.1, we visualize exemplary data from our compiled *UrbanLaneGraph* dataset and detail pre- and post-processing methods. We also discuss the proposed benchmark for evaluating lane graph prediction models.

In Sec. S.2 we give additional detail on evaluation metrics. In Sec. S.3, we discuss the sampling of the annotated graph into a representation suitable to train our LaneGNN model. In Sec. S.4, we provide additional explanations on the model architectures, the training procedures, and hyperparameter selection. In Sec. S.5, we explain our graph aggregation in more detail compared to the main manuscript. Finally, in Sec.. S.6, S.7, and S.8, we provide additional ablation studies and evaluations for our Successor-LGP, Full-LGP, and Planning tasks, respectively.

S.1. UrbanLaneGraph Dataset Details

As detailed in the main manuscript, we introduced a large-scale lane graph dataset including aligned high-resolution aerial images. In the following, we give a thorough description of the dataset curation process and how we designed the lane graph prediction benchmark.

S.1.1. Dataset Curation

S.1.1.1 Annotation Pre-Processing

Our graph annotation source is the large-scale Argoverse2 [29] autonomous driving dataset, which includes HD map annotations, including lane graphs, for all scenarios entailed in the dataset. In the context of the dataset, a scenario is a small-scale region ($50\,\mathrm{m}$ diameter). Our goal is to estimate arbitrarily large lane graphs, rendering the per-scenario graph annotation scheme insufficient. We aggregated all scenario graph annotations into a per-city global graph. However, we found that many scenarios overlap while the nodes in the respective overlaps do not have a perfect positional match. Therefore, we implemented an annotation merging procedure, producing the desired globally consistent ground-truth graph.

S.1.1.2 Image-Graph Alignment

The coordinate system of the annotations is not consistent with our aerial image coordinate frame. We therefore, transform the graph annotations into the image coordinate system

employing the Kabsch-Umeyama [23] algorithm, in which a set of selected point pairs in the source and in the target frame are aligned, minimizing a least-squares objective function. The solution to the minimization problem comprises the optimal translation, rotation, and scaling that maps points from one frame into the other.

S.1.1.3 Regional Train-Test Splits

We split the dataset into disjoint train and test splits on a geospatial basis. Concretely, for the experiments carried out in this work, we select a challenging subset of the annotated regions within each city as a separate test split. The remaining regions are leveraged for model training. For all tasks and models, we consistently use the same training and testing regions.

S.1.1.4 Graph Sampling into Crops

In order to generate samples for training our LaneGNN model, we sample the dataset into crops. As illustrated in our manuscript, for our successor lane graph prediction task, the input to our model is a $256\,\mathrm{px} \times 256\,\mathrm{px}$ crop of the original birds-eye view image. A sample consists of the aerial image crop and the lane successor graph that starts at the bottom center position of the crop. Crop positions are selected according to the position of nodes in the lane graph annotations. We also crop the annotated graph in order to use only graph nodes and edges as learning targets that are visible in the respective crop.

As described in the main manuscript, for aggregating the locally predicted successor graphs into a global graph, we facilitate an iterative aggregation scheme where the position and orientation of the next crop are determined according to the graph prediction in the current crop. This procedure can be interpreted as imitation learning from expert data, which are the crops present in the training data. This can produce unstable trajectories that do not follow lanes when deploying the model if the training data distribution does not cover the full distribution of crops from arbitrary positions and orientations. We, therefore, add Gaussian noise w.r.t the crop position (x_{crop}, y_{crop}) and orientation γ_{crop} when sampling training data crops. Concretely, we sample $x_{crop} \sim \mathcal{N}(x_{gt}, 5)$, $y_{crop} \sim \mathcal{N}(y_{gt}, 5)$, and $\gamma_{crop} \sim \mathcal{N}(\gamma_{gt}, 0.3)$. The units are px and rad, respectively. This crop sampling scheme allows the model

Figure S.1. Exemplary visualization of aerial imagery with a pixel-aligned lane graph for multiple US cities. Our *UrbanLaneGraph* dataset features a wide range of environments, including rural, suburban, and urban areas. The graph annotations feature a wide range of topological complexity scales from straight road sections to large-scale intersection scenarios with multiple incoming and outgoing lanes. The aerial images provided with the dataset have challenging visual properties such as prominent shadows, and occlusions of streets due to trees and other vegetation.

to recover from crop positions and orientations that are not well-aligned with the ground-truth graph.

S.1.1.5 Centerline Regression Data

The centerline regression targets are obtained by rendering an inverse signed distance function from the graph as an image with the same domain as the aerial image crop. Outputs of models trained on this are visualized in Fig. S.4 (lane centerline regression) and in Fig. S.3 (ego lane centerline regression).

S.1.2. Benchmark

We envision the previously described dataset as a reference for evaluating future approaches. To facilitate easier and quantitatively fair comparison between different approaches, we provide an easy-to-use graph prediction benchmarking API. For calculating all our metrics and generating visualizations, we leverage the networkx [13] Python library.
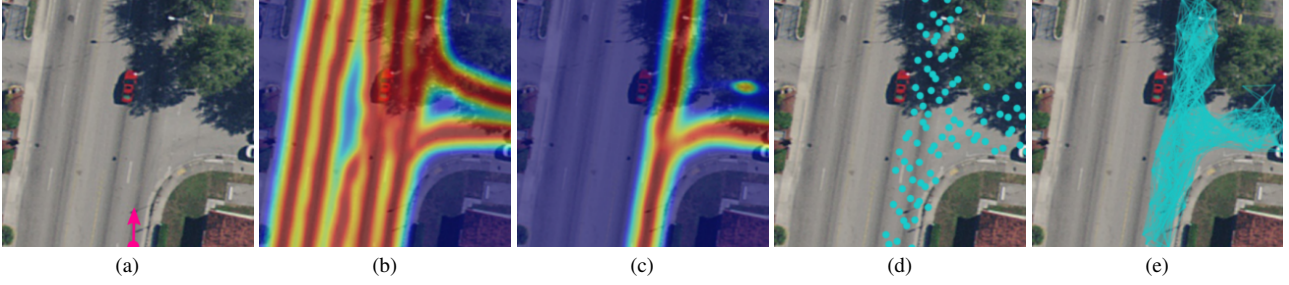
(a) (b) (c) (d) (e)

Figure S.2. Visualization of the data modalities that serve as the input to our LaneGNN model. From left to right: a) An aerial BEV image of a local scene, including the virtual agent pose for this crop. b) The regression output $\mathbb{S}_{lane}$ of our lane regression model. c) The regression output $\mathbb{S}_{lane}^{ego}$ of the agent-centric ego lane regressor. d) The lane graph node proposal that potentially holds the successor graph. e) The edge proposal list densely connects neighboring proposal nodes.
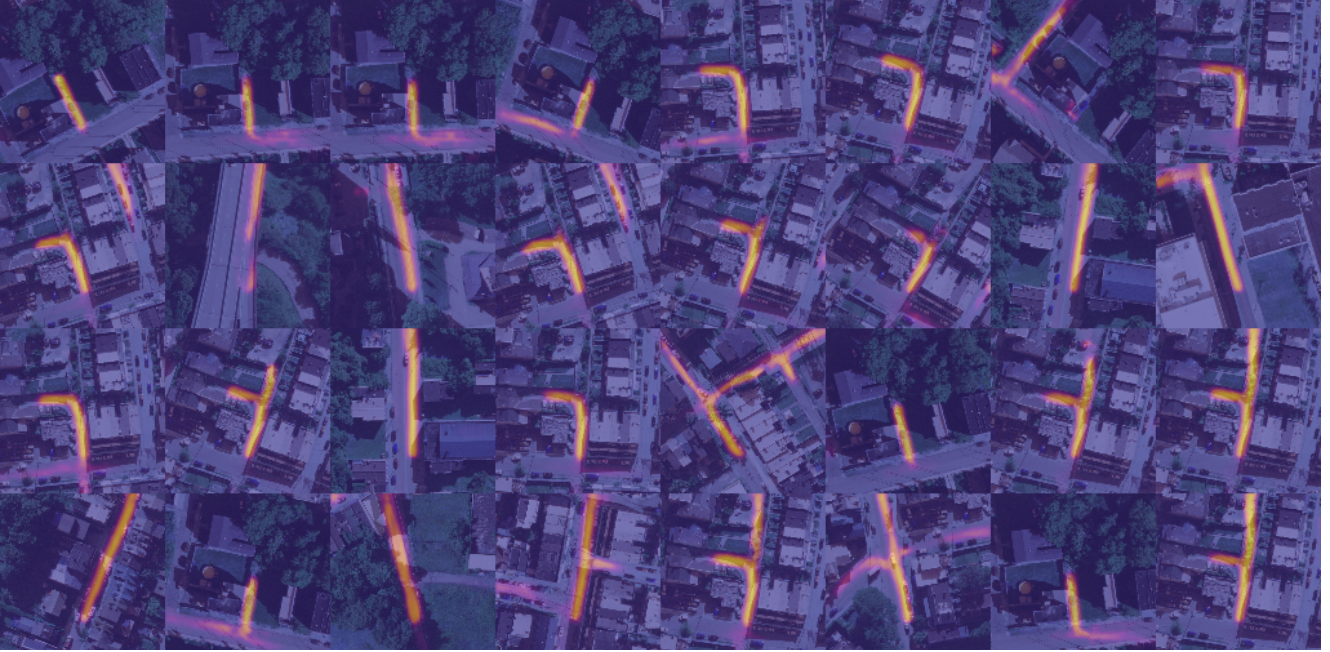

Figure S.3. Exemplary visualizations of our ego lane regression model on random crops from the test split of our dataset.

The overall information flow of our benchmarking system is visualized in Fig. S.5. Given an aerial image, the to-be-evaluated model predicts a graph g_pred. The LaneGraphEvaluator expects g_pred to be a networkx-type object; the conversion of the model output into networkx format may be implemented by the authors in a to_networkx() function. Given a predicted graph in networkx format, our evaluator queries the ground-truth graph annotations and crops the annotations to cover the same region as the predicted graph. Subsequently, all described metrics are evaluated and exported. Optionally, visualizations to rasterized or vector image formats may be produced. Finally, path-planning experiments on the predicted and ground-truth graphs may be conducted, evaluated, and visualized.

S.2. Evaluation Metrics Details

In the following, we detail the GEO and TOPO metrics, proposed by He *et al.* [15] which were used, among others metrics, to evaluate or experiments.

S.2.1. GEO Metric

The GEO metric aims to quantify the quality of the spatial position of vertices in the predicted graph, ignoring any topological properties (the existence of edges between the vertices). To this end, following He *et al.* [15], we densely interpolate the predicted graph $G_{pred} = \{V_{pred}, E_{pred}\}$ and the ground-truth graph $G_{gt} = \{V_{gt}, E_{gt}\}$ such that any two adjacent vertices have the same distance from each other. Subsequently, a 1:1 matching between V_{pred} and V_{gt} is computed, giving rise to the matching precision and matching

Figure S.4. Exemplary visualizations of our lane regression model on random crops from the test split of our dataset.
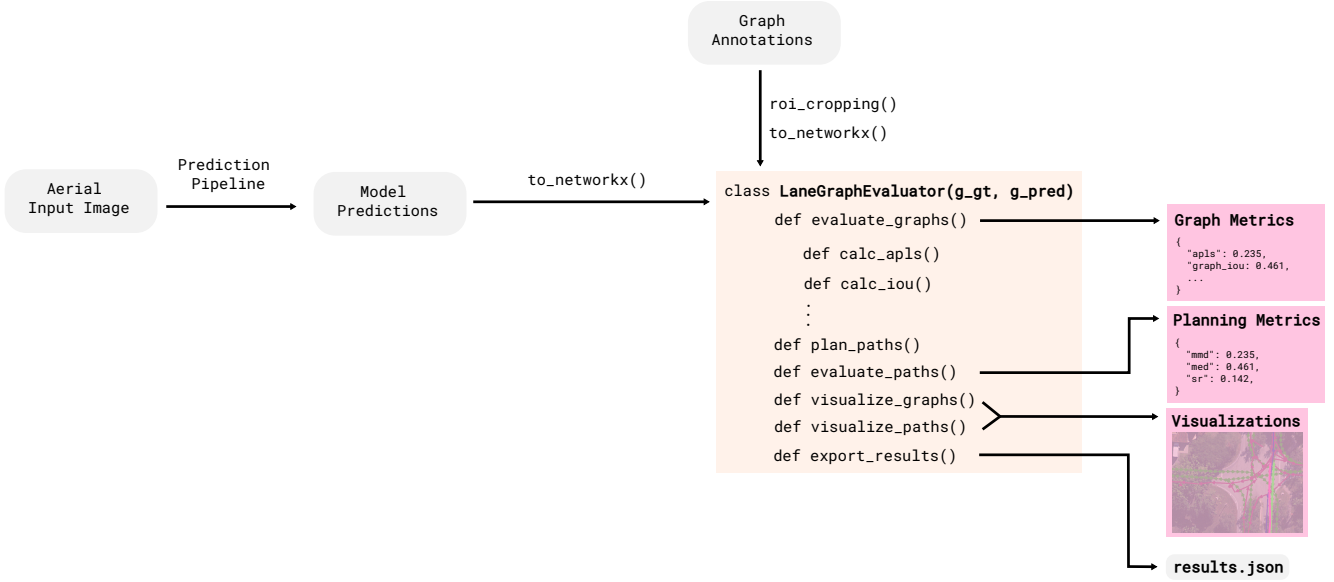

Figure S.5. Basic information flow for our graph evaluation benchmark. We provide basic routes for evaluating and visualizing lane graph predictions and annotations. Our evaluator expects the predicted graph(s) g_pred and the ground-truth graph annotations g_gt as networkx objects.

recall, denoted as GEO precision and GEO recall.

S.2.2. TOPO Metric

Building on top of the GEO metric, the TOPO metric takes vertex connectivity through edges into account as well. Given a vertex pair (V_{gt}, V_{pred}) obtained from the GEO metric, subgraphs $S_{gt}^{V_{gt}}$ and $S_{pred}^{V_{pred}}$ are created by walking a maximum distance D on the graphs. We select $D = 50\,\mathrm{m}$.

The so-created sub-graphs may be compared according to the GEO metric and averaged over all sub-graphs to obtain the overall TOPO metric. The graph-walking procedure allows penalizing missing links or false-positive graph branches as they will produce poorly aligned sub-graphs S_{gt} and S_{pred}.

S.3. Target Graph Sampling for Lane Graph Learning

As described in the main manuscript, our LaneGNN model learns to predict node and edge scores for a proposal graph. The ground-truth graph according to the dataset annotations, however, cannot without modification be used as a learning target since its node positions do not correspond to the node points obtained from the Halton-sequence-based node sampling mechanism (see S.3.1). In the following, we describe how a target graph used for learning the LaneGNN model can be obtained from lane graph annotations.
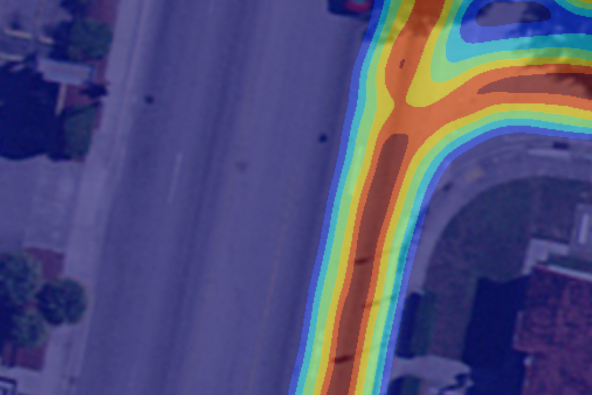

Figure S.6. The ego lane regressor model output, binned into 6 discrete thresholds from 0 to 1 for visualization purposes. The region of sampled node proposals for our LaneGNN model is sensitive to the choice of threshold cutoff value.

S.3.1. Node Sampling

In order to generate uniformly distributed 2D node positions in the rectangular image domain, we use Halton sequences. Their advantage over a per-dimension uniform random sampling of points in a 2D rectangle is their comparatively low discrepancy, leading to an approximately equal spacing of neighboring points across the 2D plane, rendering them the preferred choice for sampling random uniformly but non-regular distributed node proposals. For the experiments in our manuscript, we generate $N_{node} = 400$ Halton points for each sample. Finally, we filter the generated node positions based on the obtained ego centerline regression mask. Thus, the effective number of nodes is vastly decreased on average. We visualize the sampled Halton points for an exemplary scene in Fig. S.2d.

S.3.2. Edge Sampling

Due to the large number of possible edges in the number of nodes $|E| = \mathcal{O}(N_{node}^2)$, we sample edge proposals between pairs of nodes that have a Euclidean distance $d \in [d_{min}, d_{max}]$ such that only neighboring node are assigned a proposal edge. We visualize the sampled edges

for an exemplary scene in Fig. S.2e. Furthermore, we only include edges in the edge proposal list that do not span over regions with low ego centerline regression output.

S.3.3. Node and Edge Scoring

Target node scores are a function of the proposal node distance to the ground-truth graph. More precisely, we score the node according to $s(n_i) = \left(1 - d_{L2}(n_i, G_{gt})\right)^8$ where $d_{L2}(\cdot, \cdot)$ denotes the euclidean distance. We also assign a binary *is-endpoint* label to each node. As described in the main manuscript, in addition to predicting node scores, our LaneGNN model also entails a node classification head, discriminating between lane endpoint nodes and non-endpoint nodes. This distinction allows for efficient proposal graph pruning during model inference. The positive *is-endpoint* label is assigned to the closest node in the proposal graph the ground-truth graph endpoint for each endpoint in the ground-truth graph.

For edge scoring, we empirically found that a binary scoring function (in contrast to a continuous scoring function for the node scores) leads to favorable graph learning performance. To produce this scoring, we leverage a two-step process: First, we evaluate a similar function to the node scoring function but add an angle penalty to the scoring function which penalizes a difference in the relative angle between the proposal edge and the closest edge e_{gt}^{prox} in the ground-truth graph. Concretely: $s(e_{ij}) = \left(1 - d_\angle(e_{ij}, e_{gt}^{prox}) d_{L2}(e_{ij}, G_{gt})\right)^8$. Second, we find minimum-cost paths from the lane starting node to the endpoint node(s) through the proposal graph, where the edge traversal cost is the reciprocal of the edge score. All edges along the minimum-cost paths from the start node to the end nodes are assigned a score of one while all remaining edges are assigned a score of zero.

S.4. Model Details

S.4.1. Image Skeletonization Baseline

Our image skeletonization baseline is obtained by first thresholding the predicted ego centerline regression model output to generate a binary image of high-likelihood successor lane graph regions. We subsequently skeletonize [32] the binary image and obtain a 1-pixel wide representation of this region (On-pixels in the binary image). Finally, we convert this representation into a graph by considering all On-pixels that have $n = 1$ neighbor pixels (lane starting point or endpoint) or $n >= 3$ neighbor pixels (lane split point), forming graph nodes. To obtain the graph edges, we add a graph edge for all pairs of nodes that are connected by regions of On-pixels that do not contain any other nodes between the two nodes in consideration.

Table S.1. Details on the used architecture of the LaneGNN model $\mathcal{M}$. All ReLU-activated MLP layers except for the ones under *Classification* which are ReLU-activated up to the last layer followed by Sigmoid.

Stage	Layer	Transformation				Parametrization
Encoding	Aerial edge features	$f_{enc}^{e,bev}(\mathbf{X}_{e,bev})$	$=$	$\mathbf{H}_{e,bev}^{(0)}$	$\in \; \mathbb{R}^{B \times E \times 64}$	ResNet-18 (non-pretrained)
	Geometric edge features	$f_{enc}^{e,geo}(\mathbf{X}_{e,geo})$	$=$	$\mathbf{H}_{e,geo}^{(0)}$	$\in \; \mathbb{R}^{B \times E \times 16}$	MLP(4, 8, 16), ReLU
	Node features	$f_{enc}^{v}(\mathbf{X})$	$=$	$\mathbf{H}_{v}^{(0)}$	$\in \; \mathbb{R}^{B \times N \times 16}$	MLP(2, 8, 16), ReLU
Fusion	Edge Feature Fusion	$f_{fuse}^{e}([\mathbf{H}_{e,bev}^{(0)}, \mathbf{H}_{e,geo}^{(0)}])$	$=$	$\mathbf{H}_{e}^{(0)}$	$\in \; \mathbb{R}^{B \times E \times 32}$	MLP(16 + 64, 64, 32), ReLU
Message Passing $l=1...L$	Edge feature update	$f_e(\mathbf{H}_i^{(l-1)}, \mathbf{H}_{ij}^{(l-1)})$	$=$	$\mathbf{H}_e^{(l)}$	$\in \; \mathbb{R}^{B \times E \times 32}$	MLP(16 + 16 + 32, 64, 32), ReLU
	Node feature update	$f_v^{pred}(\mathbf{H}_i^{(l-1)}, \mathbf{H}_{ij}^{(l-1)}, \mathbf{H}_i^{(0)})$	$=$	$\mathbf{H}_{v,pred}^{(l)}$	$\in \; \mathbb{R}^{B \times E \times 32}$	MLP(16 + 16 + 32, 64, 32), ReLU
		$f_v^{succ}(\mathbf{H}_i^{(l-1)}, \mathbf{H}_{ij}^{(l-1)}, \mathbf{H}_i^{(0)})$	$=$	$\mathbf{H}_{v,succ}^{(l)}$	$\in \; \mathbb{R}^{B \times E \times 32}$	MLP(16 + 16 + 32, 64, 32), ReLU
		$f_v(\mathbf{H}_{i,pred}^{(l)}, \mathbf{H}_{i,succ}^{(l)})$	$=$	$\mathbf{H}_v^{(l)}$	$\in \; \mathbb{R}^{B \times E \times 32}$	MLP(16 + 16 + 32, 64, 32), ReLU
Classification	Edge Activation Scores	$f_{cls}^{e}(\mathbf{H}_e^{(L)})$	$=$	$\mathbb{E}_e$	$\in \; \mathbb{R}^{B \times E \times 1}$	MLP(32, 16, 8, 1), ReLU + Sigmoid
	Node Activation Scores	$f_{cls}^{v}(\mathbf{H}_v^{(L)})$	$=$	$\mathbb{S}_v$	$\in \; \mathbb{R}^{B \times E \times 1}$	MLP(16, 8, 4, 1), ReLU + Sigmoid
	Terminal Node Scores	$f_{cls}^{v,t}(\mathbf{H}_v^{(L)})$	$=$	$\mathbb{T}_v$	$\in \; \mathbb{R}^{B \times N \times 1}$	MLP(16, 8, 4, 1), ReLU + Sigmoid

Table S.2. Training details for the three models contained in the full LaneGNN model $\mathcal{M}$.

	Batch Size	Epoch	Learning Rate	Weight Decay
Lane Regressor	8	50	10^{-3}	10^{-3}
Ego Lane Regressor	8	50	10^{-3}	10^{-3}
Graph Neural Network	2	100	10^{-3}	10^{-4}

S.4.2. Centerline Regression Architectures

For the *centerline regression* and the *ego centerline regression* models, we use the same model architectures. In both cases, we use a PSPNet architecture [33] with a ResNet-152 backbone. The number of input channels is 3 (RGB color channels) for the centerline regression model while it is 4 (RGB + 1 channel output of centerline regression model) for the ego centerline regression model. Additional training parameters can be found in Tab. S.2.

S.4.3. LaneGNN Architecture

The graph neural network architecture detailed in Table S.1 is trained for 100 epochs using a learning rate of 10^{-3} and a batch size of 2. The used optimizer is Adam with weight decay $\lambda = 10^{-4}$ and $\beta = (0.9, 0.999)$. Empirically, we found that the network is relatively insensitive to variations in the learning rate, batch size, and the number of message passing steps $L = 6$. This is further demonstrated in Tab. S.3.

S.5. Graph Aggregation

In order to aggregate our successive predictions into a globally consistent solution we take the parallelizable approach of running multiple drive()-instances (see Algo. 1) each starting at a different initial pose $\mathbf{p}_i = (x_i, y_i, \gamma_i)$ up to a certain maximum number of steps or a maximum number of branches is reached. Each initial pose $\mathbf{p}_i$ could either be the actual pose obtained from localization when driving or a pose inferred from a segmentation mask of the aerial image that also includes yaw regression. Next, we follow the procedure described in Algo. 1, which starts *driving* on the birds-eye view image. Sequentially, we predict a successor lane graph G_{pred} that is pruned via multiple runs of Dijkstra's algorithm using the predicted terminal node scores $\mathbb{T}_v$ as target nodes while neglecting already traversed corridors between runs.

Before merging the predicted graph G_{pred} into G_{agg}, we transform each predicted graph into global coordinates and employ multiple iterations of Laplacian smoothing:

$$\mathbf{X} \leftarrow (\mathbf{I} - \gamma \mathbf{L}) \mathbf{X}, \tag{1}$$

which smoothens the original node positions $\mathbf{X}$ based on the graph Laplacian $\mathbf{L}$ of the undirected representation of G_{pred}. The graph Laplacian is given by

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2}$$

where $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ is the adjacency matrix. Hereby, the position of each node is influenced by its first-degree neighbors. The scalar γ denotes a smoothing intensity parameter while $\mathbf{I}$ is the identity matrix. The smoothing evens out position irregularities obtained from the initial Halton sampling. In the following, we aggregate the smoothed graph G_{pred} with a consistent representation of all prior predictions G_{agg} (see Sec. S.5.1) and traverse it by steadily exploring the edge that shows the highest successor-tree weight up to a depth of 10. The depth is limited because
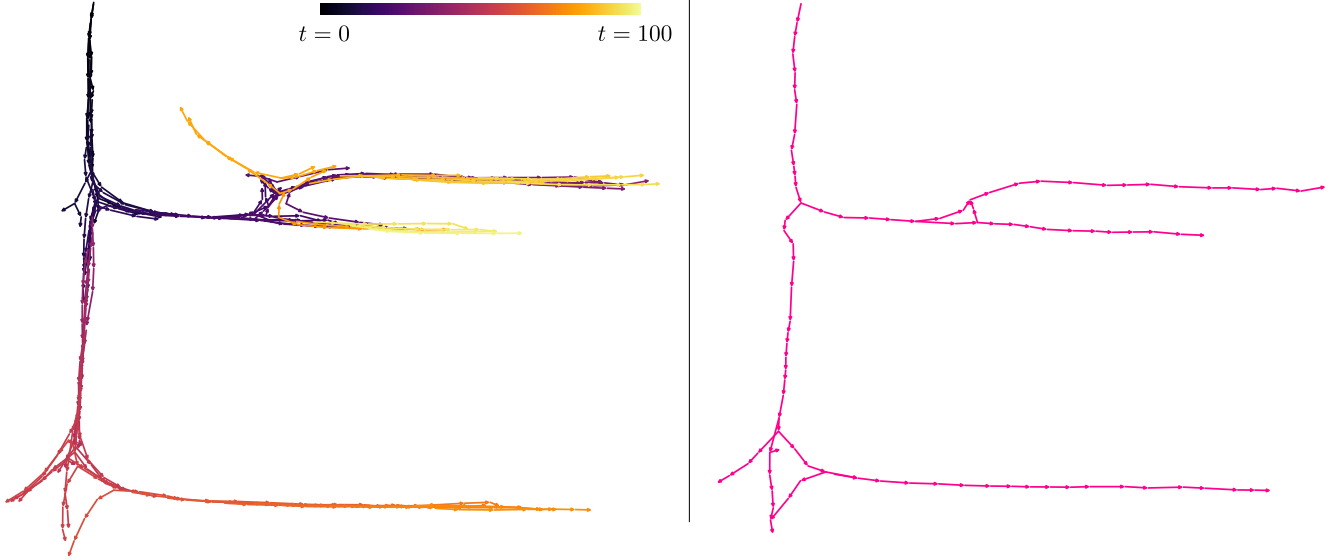
Figure S.7. Left: Visualization of consecutive successor graph predictions for $t \in [0, 1, ..., 100]$ time-steps. Right: The aggregated graph.

potentially existing loop closures of the lane graph would induce infinite tree weights. As predicted intersections provide multiple future branches of which only one is explored at a time, we queue the remaining ones for later exploration. As soon as the next edge is selected, we *step forward* by one edge length, which however is subject to hyperparameter optimization. After updating the pose, our approach is able to make predictions on new grounds while further updating and improving the aggregated lane graph representation G_{agg}. This is illustrated over 100 time steps in Fig. S.7.

Depending on the mode of operation, we are able to terminate a current branch as soon as a pose shows significant similarity to an already visited pose stored in a list. In addition, we also terminate a branch if a certain maximum number of steps is traveled across a number of branches combined or the number of explored branches exceeds a certain value. Furthermore, every single branch can also be terminated after a designated number of steps. If a branch suddenly terminates, we filter the currently aggregated graph G_{agg} by node out-degrees of two and larger to obtain split points with unvisited edges/poses that provide grounds for further exploration. The edge with the largest successor-tree weight is investigated next if exceeds a certain threshold. We have listed all used parameters in Sec. S.7.

Each and every aggregated output G_{agg} produced by one of the drive()-instances are finally aggregated one by one using the function aggregate() provided with Algo. 2, which is able to merge arbitrary graphs. In general, it would also be possible to aggregate multiple returns of drive() in an agglomerative fashion to increase inference speed for HD map creation. If necessary as a postprocessing step, we delete all but one parallel branch with the same source and

end node if they do not show any intermediate branching and contain less than six consecutive edges.

Overall, we observe an average runtime of $1\,\mathrm{s}$ per model forward-pass and aggregation step on our development machine (NVIDIA RTX 3090, AMD EPYC CPU @ 3.65GHz). With parallel execution, processing a map tile ($10^6\,\mathrm{m}^2$) requires $\sim 7\,\mathrm{min}$.

S.5.1. Lateral Aggregation Scheme

In the context of lane graphs, particular longitudinal node coordinates are merely a consequence of the chosen sampling distance. The proposed lateral graph aggregation scheme disregards deviations in longitudinal node position. Thus, we merge newly predicted graphs into an existing graph while updating it solely based on lateral deviations. Before doing so, we remove unvalidated splits and merges from the already existing aggregated graph by disregarding all splits or merges that do not exhibit a successor-edge tree of at least length three or do not show sufficient successor-tree (splits) or predecessor-tree (merges) weights. To do so, we construct local temporary aggregation graphs $LocalAggGraph(\mathbf{A})$ for each newly predicted node $\mathbf{A} \in G_{pred}$ as depicted in Fig. S.8. This is done to decrease the number of distance calculations necessary, which is crucial for large graphs G_{agg} containing thousands of nodes. Thus, each $LocalAggGraph(\mathbf{A})$ represents the region of G_{agg} in immediate vicinity of node $\mathbf{A} \in V_{pred}$ whereas $G_{pred} = (V_{pred}, E_{pred})$. In the next step, we obtain the nearest edge contained in G_{agg} as well as the lateral distance a to that edge. Similarly, we obtain the closest and second closest nodes $\mathbf{I}$ and $\mathbf{II}$ incident to that edge. If the lateral distance $a < a_{thresh}$, we see the grounds for updat-

17

Algorithm 1: drive($\mathbf{p}_{\text{init}}, \mathbf{I}_{\text{sat}}$)

```
 1  G_agg ← InitializeEmptyAggGraph()
 2  p ← InitializePoseOnMap(p_init)
 3  p ← PadSatImageSymmetrically(I_bev)
 4  stepCounter = 0
 5  branchAlive = True
 6  branchAge = 0
 7  branchCounter = 0
 8  numFutBranches = 1
 9  do
10      if stepCounter > maxSteps ∨ branchCounter >
          maxNumBranches ∨ branchAge >
          maxBranchAge then
11          break
12      if branchAlive then
13          branchAge = branchAge + 1
14      else
15          succEdges, numFutBranches ←
              GetUntraversedEdgesAtSplits(G_agg)
16          p ← GetNewPoseOfMaxScoreEdge(succEdges)
17      G_I, S_lane^ego, S_lane ← ConstrAttribGraphManifold(I_bev, p)
18      E_e, S_v, T_e ← PredictSuccessorGraph(G_I)
19      Ĝ ← PruneTraverseLaneGraph(G_I, E_e, S_v, T_e)
20      G_pred ← TransformLaneGraphToGlobalCoords(Ĝ)
21      G_pred ← ApplyLaplacianSmoothing(G_pred)
          G_agg ← Aggregate(G_agg, G_pred)
22      branchAlive, p ← StepForwAlongCurrBranch(p, G_agg)
23  while numFutBranches ∨ branchAlive;
24  return G_agg
```

ing the node positions of **I** and **II** using a weighting-based scheme involving the weights of the involved aggregated nodes (obtained from previous aggregations) as well as the initial weight of each newly predicted node **A**. To do so, we calculate the angles α, β and, γ as given by Eq. 3 in order to further calculate the lengths b_1 and b_2 as shown in Fig. S.8:

$$\alpha = \arccos\left(\frac{a}{c_1}\right), \quad \beta = \arctan\left(\frac{d_y^{\mathbf{AI}}}{d_x^{\mathbf{AI}}}\right), \quad \gamma = \frac{\pi}{2} - \alpha - \beta \quad (3)$$

$$b_1 = c_1 \sin(\alpha), \qquad b_2 = c_2 \sin\left(\arccos\left(\frac{a}{c_2}\right)\right). \quad (4)$$

These lengths are used to measure the relative influence of **A** onto **I** as well as **II**, respectively. We create temporary nodes $\mathbf{A}'$ and $\mathbf{A}''$ to ease merging (Eq. 5) in the next step:

$$\mathbf{A}' = \mathbf{A} + b_1 \begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \end{bmatrix}, \quad \mathbf{A}'' = \mathbf{A} + b_2 \begin{bmatrix} \cos(\gamma) \\ \sin(\gamma) \end{bmatrix}. \quad (5)$$

Finally, we update the position of **I** and **II** using a weighting-based approach as described with Eq. 6 and Eq. 7 in the following:

$$\mathbf{I}^* = \frac{1}{\omega_{agg,I} + \omega_{A,I}} \left(\omega_{agg,I} \begin{bmatrix} I_x \\ I_y \end{bmatrix} + \omega_{A,I} \begin{bmatrix} A'_x \\ A'_y \end{bmatrix} \right), \quad (6)$$

$$\mathbf{II}^* = \frac{1}{\omega_{agg,II} + \omega_{A,II}} \left(\omega_{agg,II} \begin{bmatrix} II_x \\ II_y \end{bmatrix} + \omega_{A,II} \begin{bmatrix} A''_x \\ A''_y \end{bmatrix} \right), \quad (7)$$
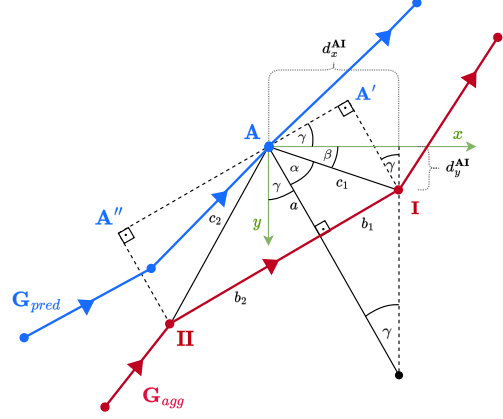

Figure S.8. Geometric visualization of our lateral graph aggregation scheme. We visualize the predicted graph G_{pred} in blue and the graph to be merged into G_{agg} in red.

Figure S.9. Additional qualitative results of our LaneGNN model. In the top row, we illustrate success cases. In the bottom row, we show interesting failure cases. The depicted graphs are not Laplace-smoothed.

where $\omega_{A,I}$ and $\omega_{A,II}$ are defined using c_1, c_2 as follows:

$$\omega_{A,I} = 1 - \frac{c_1}{c_1 + c_2}, \quad \omega_{A,II} = 1 - \frac{c_2}{c_1 + c_2}. \quad (8)$$

We collect all aggregation information using a merging map. All other nodes $m \in V_{pred}$ that were not mapped to any other node $k \in V_{agg}$ are added as new nodes to G_{agg}. Similarly, we add edges $(m, l) \in E_{pred}$ to E_{agg} in a slightly different fashion depending on whether one of the involved nodes has been already mapped to a node $k \in V_{agg}$. Finally, our `aggregate()` function returns G_{agg}.

S.6. Extended Results: Successor-LGP

In the following section, we perform additional ablation studies for our LaneGNN model on the Successor-LGP task.

18

Algorithm 2: aggregate(G_{pred}, G_{agg})

1 Add weight to each node $m \in V_{pred}$ based on weight equal to path length to starting pose
2 Compute edge angles and mean node angles based on predecessor and successor edge angles for G_{pred} and G_{agg}
3 $G_{agg} \leftarrow removeUnvalidatedSplitsMerges(G_{agg})$
4 $\mathbf{D}^{L2}_{agg,pred}$ Compute pair-wise Euclidean distance between V_{agg} and V_{pred}.
5 $\mathbf{D}^{\angle}_{agg,pred}$ Compute pair-wise mean angle distance between V_{agg} and V_{pred}.
6 $\mathbf{B}_{agg,pred} \leftarrow \left(\mathbf{D}^{L2}_{agg,pred} < \lambda\right) \wedge \left(\mathbf{D}^{L2}_{agg,pred} < \psi\right)$
7 **for** $A \in V_{pred}$ **do**
8 $\quad$ $LocalAggGraph(\mathbf{A}) = EmptyGraph()$
9 $\quad$ **for** $k \in V_{agg}$ **do**
10 $\quad\quad$ **if** $\mathbf{B}_{agg,pred}(\boldsymbol{A}, k) == 1$ **then**
11 $\quad\quad\quad$ $LocalAggGraph(\mathbf{A}) \leftarrow AddAggEdges(k)$
12 $\quad\quad$ **if** $LocalAggGraph(\mathbf{A})$ *not empty* **then**
13 $\quad\quad\quad$ $(\mathbf{II}, \mathbf{I}), a \leftarrow getNearestEdgeAndLatDist(\mathbf{A}, LocalAggGraph(\mathbf{A}))$
14 $\quad\quad\quad$ $\mathbf{I}, \mathbf{II} \leftarrow getNearestNodes(m, LocalAggGraph(\mathbf{A}), (\mathbf{II}, \mathbf{I}))$
15 $\quad\quad\quad$ **if** $a < a_{thresh}$ **then**
16 $\quad\quad\quad\quad$ Compute $\mathbf{I}^*$ and $\mathbf{II}^*$ as detailed in Eq. 6 and Eq. 7
17 $\quad\quad\quad\quad$ $G_{agg} \leftarrow UpdateAggGraph(\mathbf{I}^*, \mathbf{II}^*)$
18 **for** $m \in V_{pred}$ **if** m *unmapped to* G_{agg} **do**
19 $\quad$ $G_{agg} \leftarrow AddUnmappedNode(G_{agg}, m)$
20 **for** $(m, l) \in E_{pred}$ **do**
21 $\quad$ **if** m *not mapped to* $G_{agg} \wedge l$ *not mapped to* G_{agg} **then**
22 $\quad\quad$ $G_{agg} \leftarrow AddUnconstrainedEdge(G_{agg}, (m, l))$
23 $\quad$ **if** m *mapped to* $G_{agg} \wedge l$ *not mapped to* G_{agg} **then**
24 $\quad\quad$ $G_{agg} \leftarrow AddLeadingEdge(G_{agg}, (m, l))$
25 $\quad$ **if** m *not mapped to* $G_{agg} \wedge l$ *mapped to* G_{agg} **then**
26 $\quad\quad$ $G_{agg} \leftarrow AddTrailingEdge(G_{agg}, (m, l))$
27 **return** G_{agg}

S.6.1. Quantitative Results

S.6.1.1 Additional LaneGNN Ablation

In addition to our observation regarding low sensitivity to minor parameter variations (see Sec. S.4.3), we present a parameter study in Tab. S.3 for the LaneGNN architecture as well as the training parameters. The model is trained using both 200 as well as 400 node samples, which is a hyperparameter that is chosen in the preprocessing stage. Note that the effective number of nodes covering the drivable corridor can be much lower in reality due to ego lane segmentation masking. We observe significantly higher recalls while the precision is roughly the same when increasing from 200 to 400 nodes. The remaining metrics, however, indicate a more drastic performance difference with plummeting values across $APLS, SDA_{20}, SDA_{50}$ and $GraphIoU$ for fewer nodes.

In addition to the number of nodes, we show that the inclusion of the node loss term improves recall significantly while also showing higher values across all other metrics. The node regression outputs $\mathbb{S}_v$ are not used during graph traversal but still induce significant performance improvements as they guide the network towards reasonable corridors. The batch size shows the best performance for a value of 2 or 4

depending on the evaluated metric while batch size 1 produces drastically worse outputs. In addition to the batch size, we have observed that a number of 6 to 8 message passing steps produces high precision as well as high recall. A GNN depth of 0 essentially renders the network a classic edge classifier without leveraging the graph structure, which results in drastically reduced performance. Our experiments show that the produced output graph is often not even traversable (using our pruning approach) under that limitation.

Lastly, we train three different GNN architectures: a small-, medium- and, large-size LaneGNN. Our findings support our chosen network architecture presented in the main paper and underpin that a larger network does not necessarily perform better given the test sets across all 4 cities. Overall, we observe that the parameter set used in the main paper (underlined values) yields maximum performance across the metrics evaluated.

S.6.1.2 City-wise Test Set Results

In the following, we evaluate the used LaneGNN architecture for different training sets as well as different splits of the test set each consisting of different combinations of cities. For each respective city-split, we show the performance of

Table S.3. Additional ablations of the LaneGNN model $\mathcal{M}$, measured across the test sets of all 4 cities combined. <u>Underlined</u> parameter values denote our presented approach used in the main paper, whereas bold values represent respective **best** values for each distinct ablation. The tuples under feature dimensions represent the main LaneGNN architecture dimensions (`map_feat_dim, node_dim, edge_dim, msg_dim, edge_geo_dim`), please see Table S.1.

Parameter		TOPO P/R↑	GEO P/R↑	APLS↑	SDA$_{20}$↑	SDA$_{50}$↑	Graph IoU↑
Number of Nodes	200	**0.622**/0.561	**0.622**/0.560	0.077	0.094	0.162	0.172
	<u>400</u>	0.600/**0.699**	0.599/**0.695**	**0.202**	**0.227**	**0.377**	**0.347**
Node Loss Term	✗	0.502/**0.702**	0.501/**0.699**	0.144	0.149	0.288	0.335
	<u>✓</u>	**0.600**/0.699	**0.599**/0.695	**0.202**	**0.227**	**0.377**	**0.347**
Batch Size	1	0.421/0.557	0.421/0.560	0.170	0.110	0.222	0.296
	<u>2</u>	**0.600**/0.699	**0.599**/0.695	**0.202**	**0.227**	0.377	0.347
	4	0.581/**0.701**	0.579/**0.696**	0.162	0.220	**0.387**	**0.349**
GNN Depth	0	0.163/0.168	0.163/0.166	0.057	0.017	0.051	0.098
	2	0.517/0.688	0.517/0.684	0.144	0.152	0.310	0.331
	4	0.526/0.684	0.525/0.680	0.145	0.162	0.320	0.330
	<u>6</u>	**0.600**/0.699	**0.599**/0.695	**0.202**	**0.227**	**0.377**	0.347
	8	0.570/**0.710**	0.568/**0.704**	0.163	0.195	0.350	**0.349**
Feature Dimensions	(16, 8, 16, 16, 8)	0.599/0.655	0.598/0.650	0.114	0.072	0.178	0.234
	<u>(64, 16, 32, 32, 16)</u>	0.600/**0.699**	0.599/**0.695**	**0.202**	**0.227**	**0.377**	**0.347**
	(128, 32, 48, 48, 16)	**0.616**/0.695	**0.613**/0.688	0.162	0.180	0.310	0.332

Table S.4. Additional ablations of the LaneGNN model $\mathcal{M}$ for the *Successor-LGP* task trained on respective cities as detailed and evaluated on various combinations of city-respective test sets. PAO, ATX, MIA, and PIT represent the cities of Palo Alto, Austin, Miami, and Pittsburgh, respectively. Bold entries denote the maximum value for the given evaluation set under different training sets. Underlined values denote the numbers presented in the main paper.

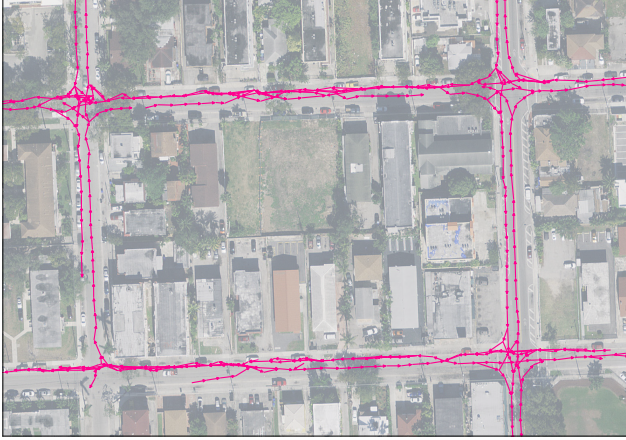| Train Set				Eval Set				TOPO P/R↑	GEO P/R↑	APLS↑	SDA$_{20}$↑	SDA$_{50}$↑	Graph IoU↑
PAO	ATX	MIA	PIT	PAO	ATX	MIA	PIT						
✓				✓				0.584/**0.744**	0.582/**0.739**	0.177	**0.220**	**0.367**	**0.378**
✓	✓	✓	✓	✓				**0.666**/0.702	**0.663**/0.696	**0.206**	0.199	0.337	0.366
	✓				✓			0.468/**0.726**	0.468/**0.727**	0.123	**0.227**	0.304	**0.327**
✓	✓	✓	✓		✓			**0.579**/0.660	**0.578**/0.660	**0.194**	0.206	**0.361**	0.301
		✓				✓		0.534/**0.687**	0.532/**0.683**	0.145	**0.217**	0.354	**0.330**
✓	✓	✓	✓			✓		**0.643**/0.672	**0.638**/0.666	**0.193**	0.198	**0.371**	0.315
			✓				✓	0.530/**0.722**	0.532/**0.714**	0.143	**0.151**	**0.307**	**0.336**
✓	✓	✓	✓				✓	**0.667**/0.674	**0.667**/0.665	**0.191**	0.115	0.243	0.333
✓				✓	✓	✓	✓	**0.603**/0.675	**0.601**/0.670	**0.203**	0.206	0.376	0.339
	✓			✓	✓	✓	✓	0.549/0.686	0.548/0.681	0.200	0.185	0.338	0.338
		✓		✓	✓	✓	✓	0.578/0.663	0.577/0.658	0.202	0.174	0.348	0.330
			✓	✓	✓	✓	✓	0.577/0.643	0.574/0.635	0.171	0.073	0.179	0.281
✓	✓	✓	✓	✓	✓	✓	✓	<u>0.600</u>/**0.699**	<u>0.599</u>/**0.695**	<u>0.202</u>	<u>**0.227**</u>	<u>**0.377**</u>	<u>**0.347**</u>

a LaneGNN instance trained on either only the respective city or on all cities (see Tab. S.4, upper half). In general, we observe that the specifically trained models do not necessarily perform better on the test-sets of their respective cities. In comparison, the LaneGNN model trained on all cities consistently produces higher precision while showing comparably small reductions in recall. This allows us to state that larger training sets across all cities do not necessarily harm the performance on specific cities when evaluating.

In addition to these findings, we also evaluate different LaneGNN instances trained on city-wise training sets and measure their performance across all cities combined. We observe no major perf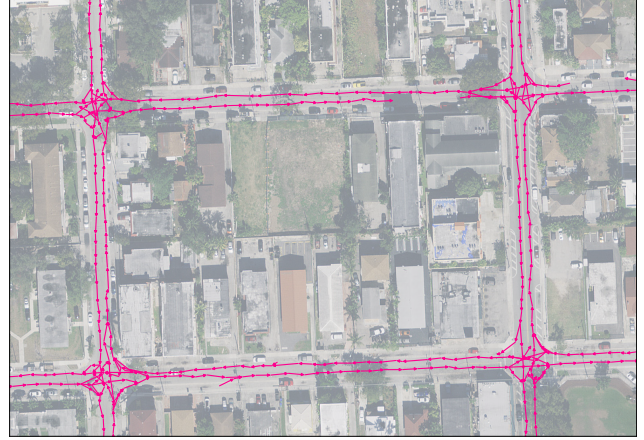ormance drops when testing a city-specific model on all cities except for the model trained on Pittsburgh (see Tab. S.4, lower half). This essentially means that we can use a model trained on one city and still perform reasonably well in other cities. Nonetheless, the model trained on all cities still shows the average best performance across all cities.
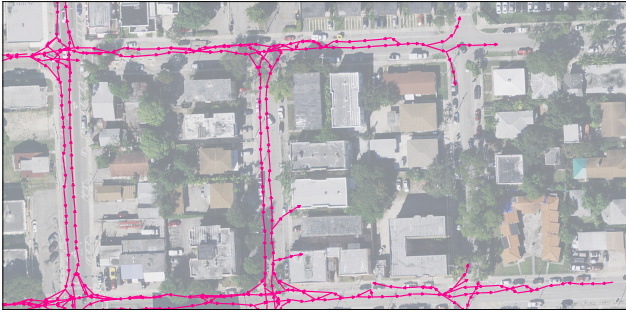
S.6.2. Qualitative Results

In Fig. S.9, we visualize the success and failure cases of our LaneGNN model for the Successor-LGP task. Challenging scenes typically entail complex topological structures, such as roundabouts, multi-lane streets, and high-contrast illumination scenarios. In the depicted high-contrast scene,
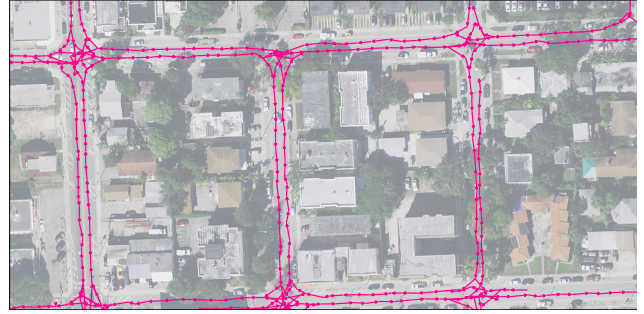
(a) Region 1, naive aggregation

(b) Region 1, our aggregation

(c) Region 2, naive aggregation

(d) Region 2, our aggregation

Figure S.10. Visualizations of the naive and our aggregation scheme for two large-scale areas within the testing region of the city of Miami.

the lane turning right is not detected, leading to a missing branch in the predicted lane graph. The roundabout depicted in the bottom center leads to a topologically correct predicted successor graph, but the predicted waypoints of the left-turning lane do not align with the position of the roundabout center. Finally, in the bottom right scene, an additional left-turning lane going in the opposite direction is predicted by our LaneGNN model. Our aggregation scheme can remove these false-positive graph branches if they are not consistently predicted for multiple successive LaneGNN forward passes.

S.7. Extended Results: Full-LGP

In order to generate the test-set results provided in the main paper, we initialized our drive($\mathbf{p}_{init}$) function (Algo. 1) starting at the predicted lane start points as provided by the yaw-segmentation output of LaneExtraction [15]. This results in 178 initial poses parametrized by $\mathbf{p}_i = (x_i, y_i, \gamma_i)$ used for parallel execution of drive() as described in Algo. 1. Our numbers provided in the main paper are generated using the parameters detailed below.

We make use of thresholding $\mathbf{S}_{lane}^{ego}$ at a value of 0.15, which produces relatively high recalls and thus more con-

Table S.5. Additional ablations of the aggregation scheme used in the Full-LGP task. We compare the presented naïve and full aggregation scheme with the reduced variants of the full pipeline (no smoothing of G_{pred}, no removal of invalidated splits and merges, and a smaller lateral aggregation threshold).

Model	APLS ↑	TOPO P/R ↑	GEO P/R ↑	Graph IoU ↑
w/o smoothing	0.105	0.452/0.671	0.631/0.726	0.377
w/o remove s/m	0.102	0.480/0.658	0.634/0.698	0.366
$a_{thresh} = 10$	**0.108**	0.458/**0.677**	0.581/**0.738**	0.354
Naïve	0.101	0.366/0.654	0.523/0.727	0.376
Ours	0.103	**0.481**/0.670	**0.649**/0.689	**0.384**

nections at intersections. This is complemented by only considering edges with an edge score of 0.5 or higher for graph traversal. For aggregation, we use a radius and distance threshold of 80 px and 0.5 radians to filter close nodes and edges of G_{agg} for constructing $LocalAggGraph(\mathbf{A})$ instances to speed up the aggregation process. For the actual merging of nodes (Eq. 6 and Eq. 7), we choose a lateral distance $a_{thresh} = 20$ that is used for aggregate() within one drive() instance as well as aggregation of multiple drive() outputs (see Algo. 2). Regarding the drive() function, the maximum overall number of steps is 36 over a
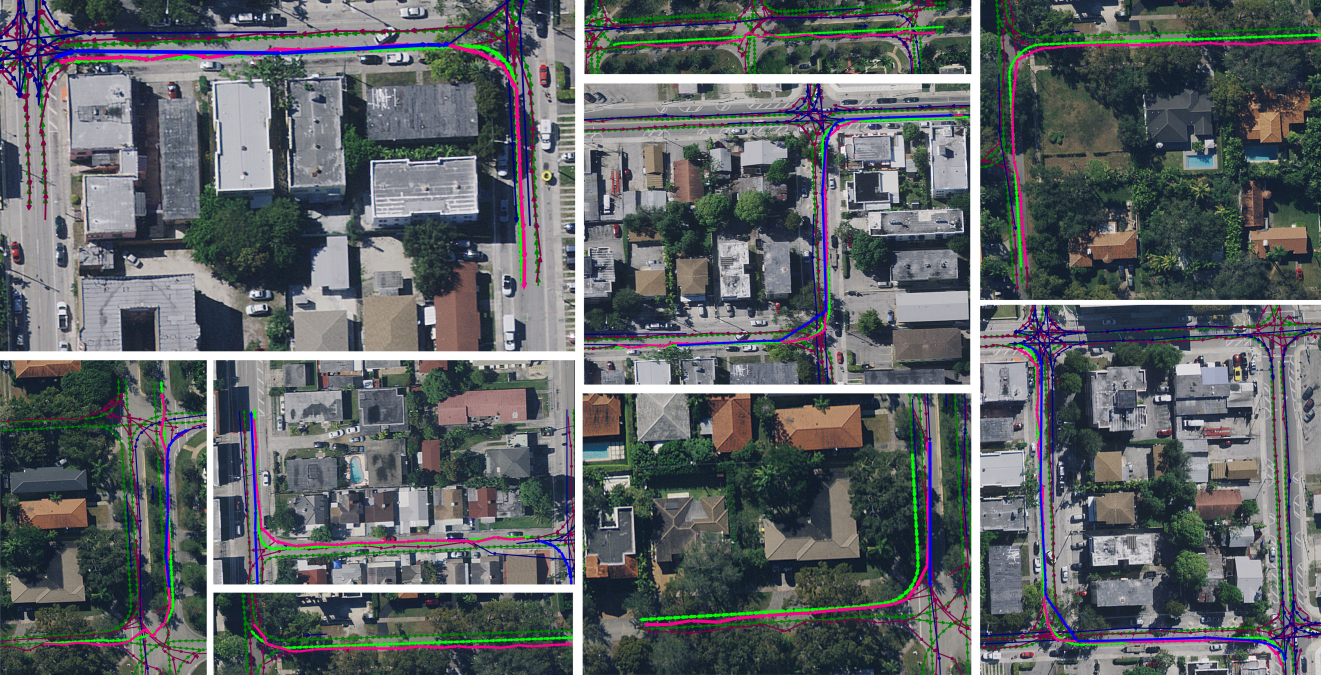
Figure S.11. Illustrative path planning experiment results. We visualize the ground-truth graph in thin green and the path planned on the ground-truth graph in bold **green**. We employ the same visualization scheme for the graph from LaneExtraction, and our aggregation scheme.

maximum of 4 branches with a maximum branch-age of 12 each. As described before we smooth the predicted graphs and also remove splits and merges.

In addition, we present results on three additional parameter settings for the *Full-LGP* task in Tab. S.5. We observe slight performance decreases in precision while the recall is similar or higher when not smoothing predicted graphs. Without removing unvalidated splits and merges, our performance shows slight decreases for GraphIoU, GEO precision, and TOPO P/R. Finally, we lower a_{thresh} to 10, which increases recall but lowers precision, specifically the GEO metric, while the Graph IoU also drops by three percent. As in the main paper, we also list our naïve aggregation scheme for comparison. The results are further illustrated in Fig. S.10.

S.8. Extended Results: Planning

As described in the main manuscript, we evaluate the quality of the generated lane graph on a planning task. Fig S.11 illustrates exemplary planned paths obtained on the graphs from the ground-truth annotations, LaneExtraction, and our aggregation scheme, respectively. We observe that for most scenarios, the planned paths from our aggregation scheme closely match the paths planned on the ground-truth graph. Even for long routes with multiple turns, we report a close match between our path and the ground-truth path. We notice some inaccuracies in our planned path where the tra-

jectory deviates from the correct lane and contains slightly misaligned path waypoint positions.

While some of the paths from LaneExtraction show encouraging results, we observe that for many routes, no close match between the LaneExtraction path and the ground-truth path can be observed, also mirrored in the quantitative evaluation of the planning task in the main manuscript. We hypothesize that the LaneExtraction graphs are not always complete and have missing links for occluded or topologically complex regions, resulting in the nonexistence of a path through the graph from start to goal nodes.