IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

AutoGraph: Predicting Lane Graphs from Traffic Observations

Jannik Zürn[1], Ingmar Posner[2], and Wolfram Burgard[3]

***Abstract*—Lane graph estimation is a long-standing problem in the context of autonomous driving. Previous works aimed at solving this problem by relying on large-scale, hand-annotated lane graphs, introducing a data bottleneck for training models to solve this task. To overcome this limitation, we propose to use the motion patterns of traffic participants as lane graph annotations. In our *AutoGraph* approach, we employ a pre-trained object tracker to collect the tracklets of traffic participants such as vehicles and trucks. Based on the location of these tracklets, we predict the successor lane graph from an initial position using overhead RGB images only, not requiring any human supervision. In a subsequent stage, we show how the individual successor predictions can be aggregated into a consistent lane graph. We demonstrate the efficacy of our approach on the UrbanLaneGraph dataset and perform extensive quantitative and qualitative evaluations, indicating that AutoGraph is on par with models trained on hand-annotated graph data. Model and dataset will be made available at http://autograph.cs.uni-freiburg.de/.**

***Index Terms*—Deep Learning for Visual Perception, Computer Vision for Transportation, Semantic Scene Understanding**

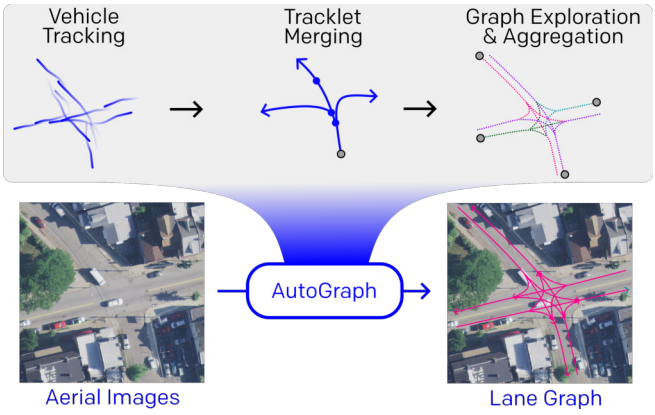

Fig. 1: Our approach AutoGraph leverages vehicle tracklets and predicts complex lane graphs from aerial images without requiring any hand-annotated lane graphs for supervision.

I. Introduction

Autonomous vehicles require detailed knowledge about their surroundings to safely and robustly navigate complex environments. Most approaches to automated driving follow one of the two major paradigms: *map-based* or *mapless* driving. Map-based approaches typically rely on HD maps entailing detailed geospatial information relevant to driving tasks, including the positions of traffic lights, lanes, or street crossings. In this context, the graph of lane centerlines (i.e. the lane graph) is a crucial component that encodes the position and connectivity of all lanes. A major bottleneck in deploying map-based autonomous driving approaches is the slow and expensive manual annotation process to generate HD maps for all regions where the vehicle is intended to operate. Methods capable of estimating the lane graphs robustly in an automated fashion are crucial for scaling up the areas covered by HD maps [15], [30], [3]. Mapless driving approaches, in contrast, solely rely on onboard sensor measurements to infer the position and layout of objects and surfaces relevant to the driving task, including the position and orientation of roads and lanes. For mapless driving, the accurate and robust estimation of the spatial and topological lane layout in the vicinity of the vehicle is paramount for safe and efficient navigation. Therefore, automatic lane graph estimation is a crucial task in map-based and mapless automated driving.

Prior work in lane graph estimation focuses on training models under full supervision [3], [30], [14], relying on large-scale ground-truth lane graph annotations, typically obtained from a large number of human annotators. The production of accurate annotations such as those available as part of the Argoverse2 [22] and NuScenes [4] datasets is, therefore, resource-intensive in both money and time.

Inspired by the success of vehicle tracking approaches [17], [9], [20] and by prior work in the context of automatic annotation from traffic participants [1], [31], in this work, we propose to leverage traffic participant tracklets as the only annotation source for lane graph estimation and do not require any manually obtained graph annotations. Most traffic participants follow their respective lanes with high accuracy. Aggregated over large numbers, the trajectories of traffic participants encode the overall structure of lane graphs well (see Fig. 1). We interpret this driving data as a data source for the annotation of lane graphs. In our approach, AutoGraph, we track traffic participants in challenging urban environments and propose a novel tracklet merging scheme, allowing us to formulate a supervised learning task in which we leverage aerial images as input and the merged tracklets serve as the learning target for our model. The overall approach is capable of accurately predicting lane graphs covering large areas with

Manuscript received: June 27, 2023; Revised: October 4, 2023; Accepted: November 3, 2023. This paper was recommended for publication by Editor Cesar Cadena Lerma upon evaluation of the Associate Editor and Reviewers' comments.

This work was supported under DFG grant number BU 865/10-2

[1]Jannik Zürn is with the Faculty of Engineering, University of Freiburg, 79115 Freiburg, Germany zuern@cs.uni-freiburg.de

[2]Ingmar Posner is with the Applied AI Lab, Oxford University, Oxford, UK, supported by a UKRI/EPSRC Programme Grant [EP/V000748/1]

[3]Wolfram Burgard is with the Department of Engineering, University of Technology Nuremberg, Nuremberg, Germany

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

high accuracy while the pipeline does not require any hand-annotated data.

To summarize, this work makes the following contributions:

- a novel tracklet aggregation scheme leveraging observed traffic participant tracklets as annotation sources for lane graph estimation models;
- the large-scale *UrbanTracklet* dataset with hundreds of thousands of vehicle and pedestrian tracklets generated from the Argoverse2 and UrbanLaneGraph datasets;
- and extensive qualitative and quantitative ablation studies on the UrbanLaneGraph dataset, demonstrating the efficacy of our approach.

II. Related Works

a) Lane Graph Estimation: Over the past years, the task of lane graph estimation has gained much attention in the autonomous driving research community. In contrast to road graph estimation, where the goal is to estimate the connectivity between roads [2], [19], lane graph estimation entails predicting the position of lanes and how they are connected. This task is much more challenging, in particular in areas with complex lane connections such as roundabouts or multi-arm intersections. Homayounfar *et al.* [15] predict the lane graph of highway scenes with an iterative RNN model from projected LiDAR data. Zürn *et al.* [30] proposed a Graph R-CNN-based model for lane graph estimation from aggregated LiDAR data in urban scenes. He *et al.* [14] leverage a multi-stage approach for lane graph extraction in which they first extract straight road sections between intersections and subsequently learn the connectivity between each incoming and outgoing lane arm. Büchner *et al.* [3] proposed a bottom-up approach for lane graph estimation. They first estimate the successor graph from a given starting position using a graph neural network and subsequently aggregate a full lane graph by iteratively merging each successor graph into a global one. Similar to our work in spirit, Karlsson *et al.* [16] infer maximum likelihood lane graphs from traffic observations with a directional soft lane probability model. They evaluate their model on the NuScenes dataset. However, they do not consider model inference from aerial images but from aggregated onboard sensor measurements. Crucially, and in contrast to our approach, their model is not capable of estimating large lane graphs due to the non-iterative nature of their approach.

While the aforementioned works show promising results in challenging environments, most of them require large-scale handcrafted graph annotations or cannot generate predictions for large-scale scenes. In the approach presented here, we do not require any manual annotations and instead leverage data encoded in the behavior of observed traffic participants.

b) Trajectory Prediction: Our successor lane graph prediction module is related to the task of trajectory prediction. From the large body of literature available in this field we briefly review the most relevant recent related works. Most approaches condition their models on rasterized or vectorized HD map representations [5], [27], [10], discrete graphs [18] or aerial images [28]. In Chai *et al.* [5], future vehicle positions are encoded by estimating the distribution over future trajectories for each agent while Zhao *et al.* [27] leverage a three-stage approach that finds prediction targets, estimates future motion for each, and scores each predicted trajectory to yield the final motion prediction. Our work also shares similarities with the line of work by Gilles *et al.* [11], [12], [13]. They frame the trajectory prediction task as a heatmap regression task, where an HD map representation is used for prediction conditioning. After subsequent post-processing of this heatmap, they sample future agent trajectories. In contrast to most existing works, we refrain from leveraging an HD map representation and instead solely rely on aerial images for our prediction task. In addition to regressing future possible agent positions, we also use this prediction block as input for a graph aggregation module to learn a complete lane graph of a given input image.

c) Automatic Annotations in Autonomous Driving: There exists a sizable body of work that considers the data encoded either in the driving behavior of the ego-vehicle or of other traffic participants. Barnes *et al.* [1] use the ego-trajectory of a vehicle to annotate drivable regions in an image. They project their own future positions into the current camera image to label pixels as drivable. Other works in self-supervised learning for navigation [21], [29] also use the ego-trajectory to label pixels for a vision-based ground classifier. Tracklets have been used by multiple previous works in the context of autonomous driving tasks. Zürn *et al.* [31] used the trajectories of other traffic participants such as vehicles and pedestrians, obtained from a LiDAR tracker, to annotate ground surfaces in urban environments. Other works also explored the benefits of inferring driving policies from the behavior of other traffic participants [23], [25], [6]. Chen *et al.* [6] leverage driving experiences collected from the ego vehicle and other vehicles jointly to train a driving policy from real-world data. Recent work by Collin *et al.* [8] proposes an automated system for aggregating observed traffic participant tracklets into a lane graph representation. While their work shows a good performance in dense traffic scenarios, it does not generalize to unseen areas since their approach does not involve training a model on this data.

III. Technical Approach

Our approach proceeds in three steps. In the first step, denoted *tracklet parsing and merging*, we track traffic participants through all scenes in the dataset and prepare the data for model training. In the subsequent *model training* step, we train the proposed model with data obtained in the first stage. In the third step, we perform inference with our trained model to perform *graph exploration and aggregation* into a globally consistent representation. In the following, we describe each step in detail.

A. Tracklet Parsing and Merging

In the following, we describe our tracklet parsing and merging pipeline. We start our data processing pipeline by tracking traffic participants from ego-vehicle data in all available scenes of the Argoverse2 dataset [22] across all six available cities. Each scene in the dataset consists of approximately 20 seconds

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

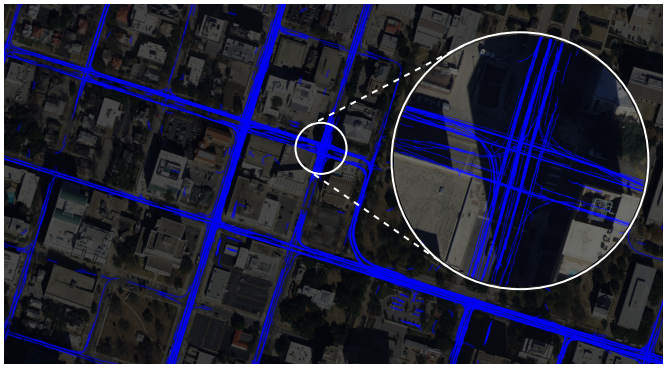

Fig. 2: Visualization of tracklets in the city of Austin, Texas, aligned with aerial imagery (darkened for better contrast).

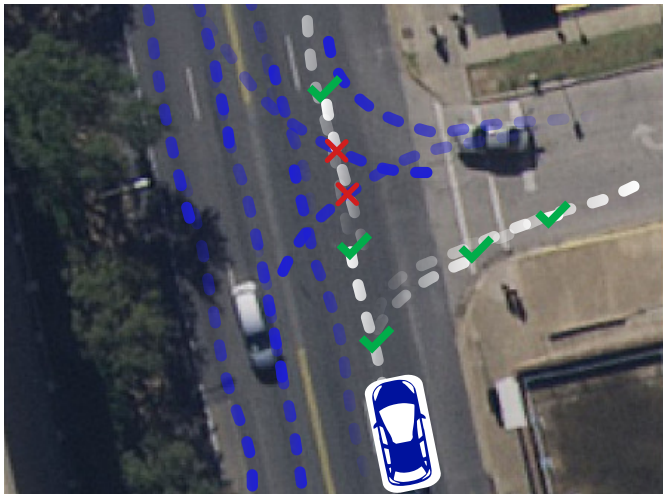

Fig. 3: A T-junction with vehicle tracklets (blue dashed lines). Merging points for tracklets are indicated with a green checkmark while exemplary failed merges are indicated with a red cross. The extracted successor graph is visualized in white.

of driving. For each scene, we track vehicles such as cars, trucks, motorcycles, and busses using a pre-trained LiDAR-based object detector [20] that processes the vehicle onboard LiDAR point clouds. We note that the performance of the tracker predictions may affect the quality of the annotations and thus the downstream performance of our approach. We transform all tracklets into a global coordinate frame. Subsequently, we smooth the tracklets with a moving average filter to minimize the amount of observation noise and the influence of erratic driving behavior (i.e., steering inaccuracies).

Fig. 2 illustrates an exemplary urban scene with the observed vehicle tracklets. Due to imperfect vehicle driving manoeuvres and the inherent observation noise, tracklets of observed traffic participants do not perfectly overlap with the ground-truth lane graph. Furthermore, and more importantly, each tracklet only covers a subset of the actual lane graph since the corresponding vehicle was only visible from the ego vehicle for a few seconds. Thus, the goal of the tracklet merging module is to merge tracklets that have significant overlap and follow the same underlying lane segment with a high likelihood.

In the following, we describe the tracklet merging procedure. We define a tracklet $\mathbf{T}$ as a list of points $\mathbf{p}_t \in \mathbb{R}^2$ describing the 2D position of a tracked object centroid at the tracking time step t and a list of heading vectors $\mathbf{h}_t \in \mathbb{R}^2$ describing the heading of the tracked object. The tracking frequency is constant for all tracklets and equals the LiDAR frequency. We define the set of all object positions in all tracklets as $\mathcal{P}$. Our goal is to merge multiple tracklets into a successor graph that encompasses all regions that have been traversed by tracked vehicles which passed through a given starting position. To this end, we define an Euclidean distance tracklet merging matrix $\mathbf{M}_D$ and an angle-based merging matrix $\mathbf{M}_A$. The Euclidean distance merging matrix $\mathbf{M}_D$ is defined as the element-wise Euclidean distance of two object centroids:

$$M_{D,ij} = ||\mathbf{p}_i - \mathbf{p}_j||_2^2, \tag{1}$$

where $\mathbf{M}_D \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{P}|}$. We also define an angle matrix $\mathbf{M}_A \in \mathbb{R}^{|\mathcal{P}|\times|\mathcal{P}|}$, indicating the absolute relative angle between object heading $\mathbf{h}_i$ and $\mathbf{h}_j$:

$$M_{A,ij} = \left| \arccos\left(\frac{\mathbf{h}_i \cdot \mathbf{h}_j}{|\mathbf{h}_i| \cdot |\mathbf{h}_j|} \right) \right|. \tag{2}$$

To merge multiple tracklets into a successor graph, we define a binary tracklet merging matrix $\mathbf{M} \in \{0,1\}^{|\mathcal{P}|\times|\mathcal{P}|}$ as follows:

$$\mathbf{M} := [\mathbf{M}_D < d_{max}] \wedge [\mathbf{M}_A < \alpha_{max}], \tag{3}$$

where $M_{ij} = 1$ implies a merging of the tracklet containing $\mathbf{p}_i$ with the tracklet containing $\mathbf{p}_j$. So far, we only considered the relative heading angle and relative position of two tracked objects. In the final step, we update $\mathbf{M}$ and integrate all recorded tracklets into the connectivity matrix. Since each single tracklet consists of a list of observed object positions and headings, we add all pairs of consecutive object positions and headings in each tracklet to $\mathbf{M}$ as well. Note that a connection $p_i \rightarrow p_{i+1}$ encoded in a specific tracklet adds the respective connection in $\mathbf{M}[i, i+1]$ but does not add the connection in $\mathbf{M}[i+1, i]$ since each tracklet has a notion of direction. Thus, $\mathbf{M} \neq \mathbf{M}^T$ in the general case.

Using this formulation, we now have a mechanism for generating a successor graph $\mathcal{S}_q$ from a query point $\mathbf{q}$ by following all tracklets connected to $\mathbf{q}$ according to $\mathbf{M}$. In order to fit our model to this data, we randomly select a query point $\mathbf{q}$ from the aerial image and extract a small image crop around $\mathbf{q}$. We extract all tracklets visible in this crop and extract the successor graph $\mathcal{S}_q$. For an exemplary visualization of the merging procedure, please refer to Fig. 3.

Furthermore, we extract the *Drivable* map layer and the *Angles* map layer. In these layers, we collect all tracklets of the whole city and colorize all pixels that are covered by a tracklet as 1 for the *Drivable* map layer or as the tracklet angle α, for the *Angles* layer. The remaining pixels are assigned a value of 0. For visualization of these map layers, please refer to Fig. 4.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

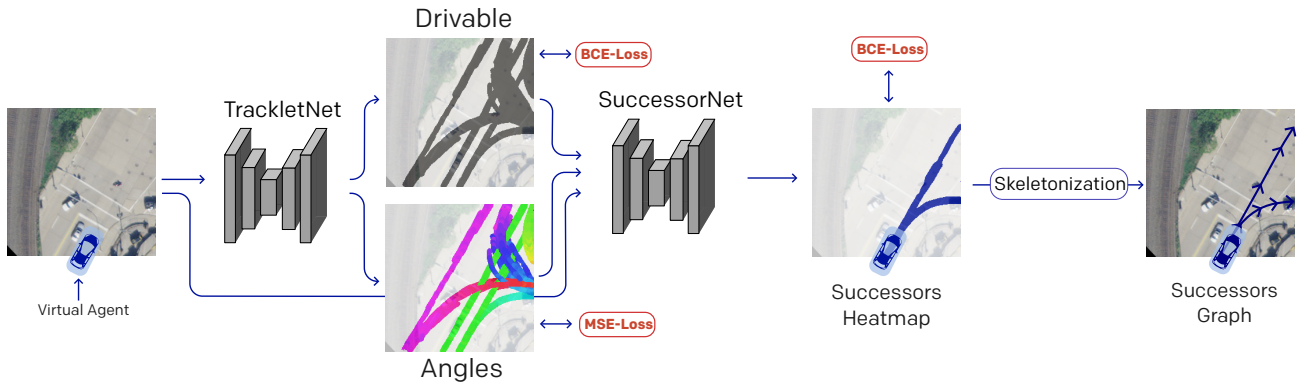

Fig. 4: Illustration of our successor graph prediction approach. We first place a virtual agent on a crop of an aerial image. We first predict the *Drivable* and *Angles* map layers from the aerial image crop with a fully convolutional neural network. We subsequently predict the successor lane heatmap from the aerial image crop, the predicted drivable surface, and lane angles. The successor lane heatmap is post-processed into a successor graph, encoding the location of successor lanes and lane split points that are reachable from the current agent's position.

B. Model Training

After our aggregation step, we are able to query all tracklets that are visible in an aerial image crop, starting from a given querying position $\mathbf{q}$. To obtain a training dataset for our models, for each query pose $\mathbf{q}$, we extract an aerial image crop $RGB_{\mathbf{q}}$ from the full aerial image, centered and oriented around the query pose. In the same way, we crop the drivable map, producing $D_{\mathbf{q}}$, and the angle map, producing $A_{\mathbf{q}}$. The whole training pipeline is visualized in Fig. 4. Our model consists of two sub-networks. We train a DeepLabv3+ model [7] to predict the pixel-wise drivable and angle maps from an RGB aerial image input, using $D_{\mathbf{q}}$ and $A_{\mathbf{q}}$ as the learning targets. We denote this model as TrackletNet. This initial task is identified as an auxiliary task, leveraging the vast amount of tracklets readily available for a given crop. For training, we use a binary cross-entropy loss to guide the prediction of the drivable map layer and a mean squared error loss for the prediction of the angle map. We encode the *Drivable* layer as a tensor $D_{ij} \in \{0,1\}^{H \times W}$. To circumvent the discontinuous angles at the singularity $\alpha = \pm\pi$, we encode the angle at pixel location (i,j) as a value pair $[\sin(\alpha), \cos(\alpha)]^T$, producing the *Angles* layer $A^k_{ij} \in \mathbb{R}^{H \times W \times 2}$. To summarize, during the TrackletNet training stage, we minimize the following loss:

$$\mathcal{L} = \frac{1}{HW} \sum_{i<H} \sum_{j<W} \left(D_{ij} \log \hat{D}_{ij} + \alpha \sum_{k} ||A^k_{ij} - \hat{A}^k_{ij}||^2_2 \right), \quad (4)$$

with a weighing factor α between the drivable surface classification and the angle regression. In our experiments, we set $\alpha = 1$.

Subsequently, we train a separate DeepLabv3+ model [7] to predict the successor graph from pose $\mathbf{q}$, which we parameterize as a heatmap $\mathbf{S_q}$. To account for the additional *Drivable* and *Angles* input layers, which we feed into this model in addition to the RGB aerial image crop, we adapt the number of input layers of the model. We denote this model as SuccessorNet. To obtain per-pixel labeling of the successor graph in the image crop, we render the successor graph $\mathcal{S}_q$ as a heatmap in the crop by drawing along the graph edges with a certain stroke width. This heatmap highlights all regions in the aerial image that are reachable by an agent placed at pose $\mathbf{q}$. We train our SuccessorNet model with a binary cross-entropy loss. Finally, we skeletonize the predicted heatmap $\hat{\mathcal{S}}_q$ using a morphological skinning process [26] and convert the skeleton into a graph representation.

C. Graph Exploration and Aggregation

The approach described in the previous sections is capable of inferring the graph structure of the successor graph from a given query position. In this section, we illustrate how a complete lane graph can be obtained by running our AutoGraph model iteratively on its own predictions and by subsequently aggregating these predictions into a globally consistent graph representation.

To this end, we leverage a depth-first exploration algorithm: We initialize our model by selecting start poses, which can either be chosen manually or obtained from our *TrackletNet* model. We predict the successor graph from this initial position and repeatedly query our model along the successor graph. In the case of a straight road section, for each forward pass of our model, we add a single future query pose to the list of query poses to process. If a lane split is encountered, for each of the successor subgraphs starting at lane splits, we add a query pose to the list. If a lane ends or no successor graphs are found, the respective branch of the aggregated lane graph terminates and we query the next pose in the list. The exploration terminates once the list of future query poses is empty. In contrast to prior work [3], where they aggregate the complete set of successor graphs according to an elaborate graph aggregation scheme, we instead only add graph nodes to the global graph where the virtual agent was placed at a given time. Therefore, we add edges between graph nodes according to the movement of the successor graph query position. This aggregation formulation simplifies the graph aggregation scheme since the number of nodes to integrate into the global graph is greatly reduced.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

TABLE I: Comparison of two baseline models with our AutoGraph approach for the Successor-LGP task. We evaluate on the test split of the UrbanLaneGraph dataset. Best model results are marked in **bold**.

Model	APLS ↑	IoU ↑	TOPO P/R ↑	GEO P/R ↑	SDA_{20} ↑	SDA_{50} ↑	Human supervision
LaneGraphNet [30]	0.179	0.063	0.0 / 0.0	0.0 / 0.0	0.0	0.0	✓
LaneGNN [3]	0.202	**0.347**	**0.600 / 0.699**	**0.599 / 0.695**	**0.227**	0.377	✓
AutoGraph (ours)	**0.310**	0.233	0.412 / 0.628	0.422 / 0.601	0.159	**0.678**	✗

TABLE II: Key statistics of our *UrbanTracklet* dataset

City	Number of tracklets	Total tracklet length
Austin, TX	287,306	3.642 km
Detroit, MI	73,232	1.099 km
Miami, FL	283,641	3.312 km
Palo Alto, CA	82,351	1.050 km
Pittsburgh, PA	34,505	1.390 km
Washington D.C.	121,557	1.469 km
All	882,592	11.962 km

IV. Dataset

We evaluate our proposed method on a large-scale dataset for lane graph estimation from traffic participants. We use the RGB aerial images and the ground-truth lane graph annotations from the UrbanLaneGraph dataset [3]. To obtain the traffic participant tracklets, we leverage the LiDAR dataset split of the Argoverse2 [22] dataset. The dataset contains consecutive LiDAR scans for hundreds of driving scenarios. A single driving scenario entails approx. 20 s of real-world driving. We leverage the OpenPCDet [20] detection and tracking suite for LiDAR point clouds with a CenterPoint [24] model, pre-trained on the NuScenes dataset [4]. We track the vehicle classes of *Car*, *Bus*, *Trailer*, and *Motorcycle*. Subsequently, we transform the respective LiDAR-centric tracklet coordinates to a global reference frame that is aligned with the aerial image coordinates. We smooth each tracklet with a mean filter approach to account for sensor noise and tracking inaccuracies. We call our tracklet dataset the *UrbanTracklet* dataset and make it publicly available as an addition to the UrbanLaneGraph dataset [3]. In Tab. II, we list all relevant metrics of our *UrbanTracklet* dataset. In total, our dataset entails tracklets with an accumulated total length of approximately 12 000 km.

V. Experimental Results

A. Implementation Details

The TrackletNet and SuccessorNet have identical DeepLabv3+ architectures. The TrackletNet receives an RGB input image of shape $H \times W \times 3$ and outputs the *Drivable* and *Angles* layer map output. We use two separate decoders to produce the outputs. The drivable area segmentation has a resolution of $H \times W$ while the lane angle output has the size of $H \times W \times 2$. The training data used to train the two models is obtained from the dataset described in Sec. IV. We crop image segments of size $256\,\mathrm{px} \times 256\,\mathrm{px}$ from the global aerial image. The crops are oriented along a randomly sampled tracklet at the bottom centre of each crop. To increase the efficacy of our aggregation method (see Sec. III-C), we require the successor graph prediction to be robust w.r.t. perturbations in the position of the virtual agent. To provide more diverse samples with different positional variations, we randomly rotate the crop with an angle $\Delta\phi \sim \mathcal{U}(-\pi/3, \pi/3)$.

Using this sampling method, a vast amount of samples can be generated since the aerial image can be cropped at arbitrary locations and orientations. For our experiments, we generate a total number of $1.5\,\mathrm{M}$ samples from all cities combined. We found that the lane graph complexity differs between different scenes, i.e., straight road sections have much simpler successor graphs than entries to roundabouts or multi-arm intersections. We found that a balanced mix between easy (successor graph has no splits) and hard (successor graph has one or more splits) samples is beneficial.

B. Tasks

Following Büchner *et al.* [3], we evaluate our approach on two complementary tasks: Successor Lane Graph Prediction (*Successor-LGP*), and Full Lane Graph Prediction (*Full-LGP*). In *Successor-LGP*, we aim at predicting a feasible ego-reachable successor lane graph from the current pose of the virtual agent. In the task of *Full-LGP*, we compare the complete lane graph in a local region to the ground-truth graph. We evaluate each task on the test images of the UrbanLaneGraph dataset [3], which are not used for model training at any stage. For model evaluation, we use the metrics proposed by Büchner *et al.* [3].

C. Baselines

To provide relevant comparisons and ablations demonstrating the efficacy of our AutoGraph approach, we compare it with a baseline model trained on ground-truth graph annotations, denoted as AutoGraph-GT. For this model, we use the ground-truth lane graph in places where we would otherwise query the recorded vehicle tracklets in our AutoGraph approach. This approach yields the ground truth lane graphs and successor lane graphs according to the graph annotations available in the dataset. The ground-truth lane graph annotations have none of the shortcomings of tracklet-based approaches, such as observation noise or erratic driving behaviour. We also compare to the previously proposed models LaneExtraction [14] and LaneGNN [3]. Note that other recent works by Colling *et al.* [8] and Karlsson *et al.* [16] are relevant works but do not aim at solving the task this work is concerned with and thus cannot be used for comparison.

D. Task Evaluation

We evaluate our model on two tasks for lane graph estimation: Successor Lane Graph Prediction and Full Lane Graph Prediction.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Fig. 5: Qualitative results of our models for the Successor-LGP task. We visualize the successor heatmap and the graph generated from it for our human-supervised model AutoGraph-GT and our tracklet-supervised model AutoGraph.

TABLE III: Ablation study with multiple model variants for the Successor-LGP task. We compare our AutoGraph model with our human-supervised model variant AutoGraph-GT and evaluate the influence of different map input layers on the model performance. We highlight the best-performing model with and without human supervision, respectively, in **bold**.

Model	APLS ↑	IoU ↑	SDA_{20} ↑	SDA_{50} ↑	Human supervision	Tracklet-Joining	Drivable map layer	Angles map layer
AutoGraph-no-join	0.344	0.232	0.052	0.498	✗	✓	✗	✗
AutoGraph	0.310	**0.233**	**0.159**	**0.678**	✗	✗	✗	✗
AutoGraph+D	**0.349**	0.230	0.063	0.510	✗	✓	✓	✗
AutoGraph+DA	0.346	0.211	0.080	0.447	✗	✓	✓	✓
AutoGraph-GT	0.377	**0.281**	0.048	0.416	✓	N/A	✗	✗
AutoGraph-GT+D	**0.418**	0.268	**0.072**	0.418	✓	N/A	✓	✗
AutoGraph-GT+DA	0.409	0.269	0.059	**0.463**	✓	N/A	✓	✓

1) Successor Lane Graph Prediction: We evaluate the performance of our AutoGraph model and compare it with the recently proposed LaneGNN [30] and LaneGraphNet [3] models. Tab. I lists the model performances on the test split of the UrbanLaneGraph dataset. Our experiments indicate that the performance of our AutoGraph model is superior to the LaneGraphNet model in all metrics and is mostly on par with the recently proposed LaneGNN model. While it performs much better in the APLS and SDA_{50} metric than the LaneGNN model, it is slightly inferior for the TOPO/GEO metrics and the Graph IoU metric. We hypothesize that the performance of our AutoGraph model could be further improved in scenes with road occlusions due to congested roads and overarching vegetation since our model struggles to predict accurate successor graphs in these regions. Specific treatment of such scenes in the model training schedule (i.e., active learning) might be beneficial.

Additionally, we perform ablation studies of multiple variants of our AutoGraph approach. The results are listed in Tab. III. In our AutoGraph-no-join variant, we do not join the tracklets (see Sec. III-A), ignoring their proximity and their relative angles. Instead, we follow tracklets until they end or until they leave the image crop. We also do not use the *Drivable* (D) and *Angles* (A) model outputs but feed the aerial image directly into the SuccessorNet model. For our AutoGraph model variant, we use joined tracklets as per Sec. III-A but omit the *TrackletNet* auxiliary network. For our AutoGraph+D and AutoGraph+DA model variants, we add the *Drivable* and *Angles* model outputs, respectively. The model variant AutoGraph-GT does not use the tracklets of other traffic participants but is trained on ground truth human graph annotations, where we encode the successor graph as a heatmap instead of the raw graph representation as in the LaneGNN or LaneGraphNet models.

Our ablation studies indicate that the AutoGraph-no-join method overall performs worse than our AutoGraph model variant. This indicates that joining tracklets to form more complete successor graphs helps produce higher-quality and more consistent annotations. Furthermore, the inclusion of the *Drivable* map layer on top of the RGB layer improves model performance for some metrics. Adding the *Angles* map layer in addition to the *Drivable* layer does not consistently improve

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

(a) Washington, D.C. (b) Miami, detail view

Fig. 6: Qualitative results on the Full-LGP task. We visualize predictions of our aggregated AutoGraph in pink color. Our aggregation scheme is capable of traversing challenging urban scenes featuring complex graph topologies with high accuracy.

our evaluation metrics any further. Despite the additional information that is available about the scene if the *Angles* map layer is included, the increased noise produced by imprecise angle estimates seems to outweigh the benefits of having additional information available. This result supports the results discussed in Zürn *et al.* [30], where additional input modalities did not significantly improve model performance. For our subsequent experiments, we opted to keep our *Drivable* and *Angles* model components despite the inconclusive results since drivable and angles maps are a useful model output for potential downstream tasks. Exemplarily, these outputs could also be aggregated and used to obtain a robust pixel-wise estimate for the prediction of drivable surface and lane orientation in large areas, similar to how we aggregate our successor graphs into a large graph structure.

For qualitative evaluation, we visualize predictions of our best-performing model in Fig. 5 and the reference model AutoGraph-GT. We observe that both models are capable of modeling the multimodal spatial distribution of successor lanes efficiently. However, the AutoGraph-GT model shows more accurate heatmap outputs, since the annotations used for training stem from the ground-truth successor lane graph.

To summarize, our experiments demonstrate that our AutoGraph model variants (trained on tracklets) perform overall similarly to our AutoGraph-GT model variants (trained on human lane graph annotations), indicating that vehicle tracklets recorded from a moving recording platform are suitable for training lane graph prediction models. In the APLS metric and the Graph IoU, the AutoGraph-GT model variant performs better than the AutoGraph model, presumably due to the higher annotation accuracy due to the better alignment of annotation with aerial images.

2) Full Lane Graph Prediction: For the Full Lane Graph Prediction task, we initialize our model on 10 initial poses per evaluation tile and run our aggregation scheme. We compare the performance of our best-performing model variant with the prediction results of LaneExtraction [14] and the aggregation module of LaneGNN [3]. Note that the number of initialization poses is much smaller compared to the number of initialization poses used for the LaneGNN model [3]. The results are listed in Tab. IV. We observe that for some metrics, our AutoGraph model achieves comparable or better performance than the human-supervised LaneGNN [3] or LaneGraphNet models [30]. Our model performs worse in the TOPO and GEO metrics. We note that our AutoGraph model struggles with road surface occlusions introduced by overarching vegetation. However, we emphasize that since our model uses fewer model initialization poses compared to LaneGNN [3], a degradation in graph connectivity may be expected since lane graph regions in occluded areas may not be reached with an iterative aggregation scheme when no successor graph is found in a given frame.

TABLE IV: Evaluation on the Full-LGP task.

Model	APLS	IoU	TOPO P/R	GEO P/R
LaneExtraction [14]	0.072	0.213	0.405 / 0.507	0.491 / 0.454
LaneGNN [3]	0.103	**0.384**	0.481 / **0.670**	**0.649** / **0.689**
AutoGraph	**0.258**	0.189	**0.503** / 0.529	0.503 / 0.351

Our qualitative evaluations show a high graph fidelity, recognizing most of the visible lanes and modelling their connectivity accurately. Fig 6 illustrates two exemplary lane graphs for the cities of Washington, D.C., and Miami. We observe that our approach is capable of accurately reconstructing the lane graph in visually challenging environments. Large scenes with multiple blocks are handled well and clearly reflect the underlying lane graph topology. The detailed view of a complex intersection in Miami illustrates that almost all major intersection arms are covered even in the presence of visual clutter such as water, boats, parking lots, and asphalt-colored roofs of buildings. Minor inaccuracies are produced at the five-armed intersection at the bottom of the aerial image, where not all connections between intersection arms are present in the inferred lane graph.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

VI. Conclusion

In this work, we presented a novel method for lane graph estimation in urban environments from traffic participant tracklets. We showed that our model, which is trained solely on data from tracked vehicles, is capable of predicting highly accurate lane graphs. We presented a tracklet processing scheme that allows us to use the observed tracklets of traffic participants as an annotation source to train our model. We demonstrated the efficacy of our approach on a large-scale lane graph dataset for which our approach demonstrated performance close to a ground-truth supervised baseline model. Future work will address adding pedestrians and bicycle tracklets to the approach for capturing more diverse annotations. Additionally, the improved handling of occluded roads appears to be a promising direction for future research.

References

[1] Dan Barnes, Will Maddern, and Ingmar Posner. Find your own way: Weakly-supervised segmentation of path proposals for urban autonomy. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 203–210. IEEE, 2017.

[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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4720–4728, 2018.

[3] Martin Büchner, Jannik Zürn, Ion-George Todoran, Abhinav Valada, and Wolfram Burgard. Learning and aggregating lane graphs for urban automated driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13415–13424, 2023.

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019.

[6] 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.

[7] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[8] Pascal Colling, Dennis Müller, and Matthias Rottmann. Hd lane map generation based on trail map aggregation. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 600–606. IEEE, 2022.

[9] Julie Dequaire, Peter Ondrúška, Dushyant Rao, Dominic Wang, and Ingmar Posner. Deep tracking in the wild: End-to-end tracking using recurrent neural networks. *The International Journal of Robotics Research*, 37(4-5):492–512, 2018.

[10] Jiyang Gao, Chen Sun, Hang Zhao, Yi Shen, Dragomir Anguelov, Congcong Li, and Cordelia Schmid. Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11525–11533, 2020.

[11] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Home: Heatmap output for future motion estimation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 500–507. IEEE, 2021.

[12] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Thomas: Trajectory heatmap output with learned multi-agent sampling. *arXiv preprint arXiv:2110.06607*, 2021.

[13] Thomas Gilles, Stefano Sabatini, Dzmitry Tsishkou, Bogdan Stanciulescu, and Fabien Moutarde. Gohome: Graph-oriented heatmap output for future motion estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 9107–9114. IEEE, 2022.

[14] 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.

[15] 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/CVF International Conference on Computer Vision*, pages 2911–2920, 2019.

[16] Robin Karlsson, Alexander Carballo, Francisco Lepe-Salazar, Keisuke Fujii, Kento Ohtani, and Kazuya Takeda. Learning to predict navigational patterns from partial observations. *arXiv preprint arXiv:2304.13242*, 2023.

[17] Peter Ondruska and Ingmar Posner. Deep tracking: Seeing beyond seeing using recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[18] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 683–700. Springer, 2020.

[19] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8910–8918, 2020.

[20] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020.

[21] Lorenz Wellhausen, Alexey Dosovitskiy, René Ranftl, Krzysztof Walas, Cesar Cadena, and Marco Hutter. Where should i walk? predicting terrain properties from images via self-supervised learning. *IEEE Robotics and Automation Letters*, 4(2):1509–1516, 2019.

[22] 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. *arXiv preprint arXiv:2301.00493*, 2023.

[23] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. End-to-end learning of driving models from large-scale video datasets. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2174–2182, 2017.

[24] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[25] Jimuyang Zhang and Eshed Ohn-Bar. Learning by watching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12711–12721, 2021.

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

[27] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR, 2021.

[28] Tianyang Zhao, Yifei Xu, Mathew Monfort, Wongun Choi, Chris Baker, Yibiao Zhao, Yizhou Wang, and Ying Nian Wu. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12126–12134, 2019.

[29] Jannik Zürn, Wolfram Burgard, and Abhinav Valada. Self-supervised visual terrain classification from unsupervised acoustic feature learning. *IEEE Transactions on Robotics*, 37(2):466–481, 2020.

[30] 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.

[31] 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.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.