Lane-Level Street Map Extraction from Aerial Imagery

Songtao He
MIT CSAIL
songtao@mit.edu

Hari Balakrishnan
MIT CSAIL
hari@csail.mit.edu

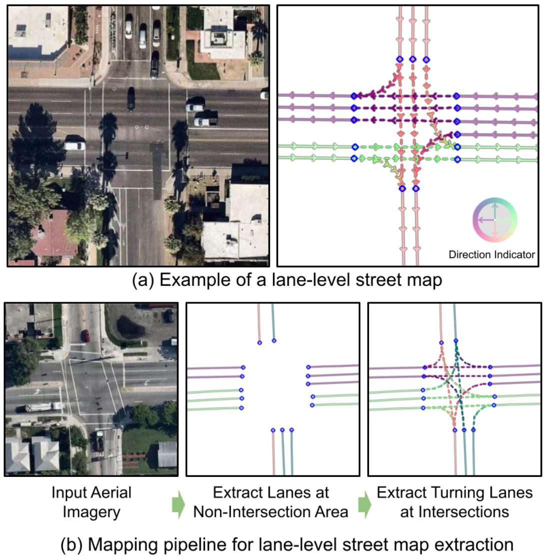
Abstract

Digital maps with lane-level details are the foundation of many applications. However, creating and maintaining digital maps especially maps with lane-level details, are labor-intensive and expensive. In this work, we propose a mapping pipeline to extract lane-level street maps from aerial imagery automatically. Our mapping pipeline first extracts lanes at non-intersection areas, then it enumerates all the possible turning lanes at intersections, validates the connectivity of them, and extracts the valid turning lanes to complete the map. We evaluate the accuracy of our mapping pipeline on a dataset consisting of four U.S. cities, demonstrating the effectiveness of our proposed mapping pipeline and the potential of scalable mapping solutions based on aerial imagery.

1. Introduction

Digital maps with lane-level details are the foundation of many applications, including lane-to-lane navigation, route planning for delivery fleets or autonomous vehicles, and lane-level localization. However, creating and maintaining digital street maps, especially with lane-level details, are time-consuming and labor-intensive. As a result, automatic mapping techniques have drawn much attention from both industry and academics, and they have proposed many solutions to automate the mapping process. Albeit great efforts devoted to this problem, given the complex nature of real-world scenarios, how to automatically produce digital maps at scale with low costs and an acceptable accuracy is still an open research question.

We can put automatic mapping techniques into two categories based on their data sources. (1) One type of mapping solution relies on the sensors mounted on vehicles, e.g., GPS receivers, IMUs, cameras, and lidars. This type of mapping solution can provide decent mapping accuracy. However, because it relies on sensor-equipped vehicles, the mapping cost is often high, especially when maintaining an up-to-date map covering a large region. (2) Another type of mapping solution relies on remote sensing data, i.e., imagery and radar data, collected from airplanes or satellites. This type of mapping solution can quickly scale up to a large region at a low cost. However, limited by the top-down view and the resolution of the data, prior works mainly focus on extracting maps at road level [30, 7, 44, 9, 40, 29, 14, 21, 35, 6, 8], extracting road curbs [37, 38, 39], segmenting visible lane markers [5, 18], and inferring lane count and lane position [22, 42]; none of them extract a complete and routable lane-level street map.


Direction Indicator
Input Aerial Imagery
Extract Lanes at Non-Intersection Area
Extract Turning Lanes at Intersections

(a) Example of a lane-level street map

(b) Mapping pipeline for lane-level street map extraction

Figure 1. Example of a lane-level street map and our proposed mapping pipeline. We show lanes at non-intersection areas with solid lines and show the turning lanes with dashed lines. The colors of lanes indicate their driving directions. We highlight the terminal nodes in blue.

This work takes one step forward toward a promising direction – extracting routable lane-level street maps from aerial imagery. In Figure 1 (a), we show an example of the lane-level street map at an intersection. Unlike road-level street maps, the lane-level street maps capture detailed road network features, including lane geometry such as lane merging and lane diverging, and rich semantic information such as driving directions, e.g., the one-way road in Figure 1 (a), and turning lane restrictions, e.g., the left-only and right-only lanes in Figure 1 (a,b). Because of the complexity of lane topology and semantic, extracting lane-level street maps from aerial imagery is a challenging vision task and existing road-level street map extraction solutions are incompetent to solve it.

In this work, to address the challenge, we propose a mapping pipeline for lane-level street map extraction. We have a key observation that, unlike road-level map extraction, extracting the entire lane-level maps in one shot can be very challenging. However, the lane extraction problem becomes solvable if we divide the task into sub-tasks and solve them separately. Hence, we split the lane extraction task into two sub-tasks. As shown in Figure 1(b), we first create lane-level maps at non-intersection areas with lane geometry and lane direction. Then, we enumerate all the possible turning lanes, i.e., a connection between two terminal nodes (we highlight the terminal nodes in blue in Figure 1), at the intersections. We check if the turning lanes are valid and extract the geometry of the valid turning lanes to complete the map. Here, we consider the lanes that go straight at intersections also turning lanes.

We evaluate our mapping pipeline on a dataset containing 400 km of lanes in four US cities, Boston, Seattle, Phoenix, and Miami. In the evaluation, we compare different neural network designs at each stage of the mapping pipeline, showing the effectiveness of our mapping pipeline and the potential of aerial-imagery-based lane-level street map extraction.

In this work, we make the following contributions,

- We proposed an automatic mapping pipeline to extract routable lane-level street maps from aerial imagery. To the best of our knowledge, this is the first work that extracts a fully-routable lane-level street maps from aerial imagery.
- Our evaluation demonstrates the effectiveness of our mapping pipeline and unveils challenges in the task, which can inspire future research work.

2. Related Work

Mapping with aerial or satellite imagery. Extracting digital maps from aerial or satellite imagery has been extensively studied. Most of the prior works focus on extracting road-level maps. A widely-used strategy to solve this problem is to turn the road extraction problem into a road segmentation problem, for examples, DeepRoadMapper [30], D-LinkNet [44], joint orientation learning [9], and many other works [40, 11, 36, 17] – they all adopt a segmentation-based map extraction strategy. Besides the segmentation-based approaches, RoadTracer [7] first proposes to extract the road network using an iterative graph construction approach that does not rely on the road segmentation but constructs the road network directly. After RoadTracer, several follow-up works [29, 35, 14] have proposed improvements upon the graph construction strategy. Recently, Sat2Graph [21] proposes the graph-tensor encoding to unify the segmentation strategy and the graph construction strategy, and shows promising results. However, all of those works focus on map extraction at road levels, and we cannot directly apply them to lane-level map extraction given the complexity of lane-level maps in terms of both topology and semantic.

Besides road topology extraction, many other works extract different map features from aerial or satellite imagery. For example, lane marker extractions [5, 18], lane curb extraction [37, 38, 39], lane count and position extraction [22, 42], road attribute inference [22], road safety assessment [23, 31], etc. These works extract important features about the road networks; however, they cannot produce a complete routable lane graph that is directly useful for downstream applications.

Mapping with vehicle sensors. Creating maps using sensor data collected on vehicles, including dedicated survey vehicles and other vehicles like taxis, has drawn great attention in industry and academics. Prior works have explored mapping solutions with different sensor data sources including GPS [3, 10, 12, 15, 16, 20, 34, 4], cameras [26, 25], lidars [24, 27, 32, 43], and the combinations of different data sources [45]. In this work, we focus on mapping with remote sensing data, i.e., aerial imagery, which complements the mapping solutions based on vehicle sensors.

3. Mapping Pipeline

In this work, we represent the lane-level street map as a directed graph $G = \{V, E\}$, where the vertices represent locations in a plane and the edges represent lane segments (centerlines). This graph captures both non-intersection lanes and the turning lanes (virtual lanes) at intersections. The direction of the edge encodes the driving direction of the lane. We call this directed graph a lane graph when we use it to describe a lane-level street map. While a lane-level street map contains many features, the lane graph is a basic but essential map representation capturing the core features that make the lane-level street map routable. In this work, we propose a mapping pipeline to extract the lane graph from aerial imagery.

Though prior works have proposed many solutions on road network extraction, we cannot directly use them to extract lane graphs because most of the existing works extract the road network as a planar graph (no intersecting edges). However, at road intersections, the lane graph is not planar – it needs to represent turning lanes that intersect but are not connected. Nevertheless, we need to extract the driving directions of the lanes, whereas this is not necessary for road-level map extraction. Therefore, we have to design a new solution to extract the lane graph.

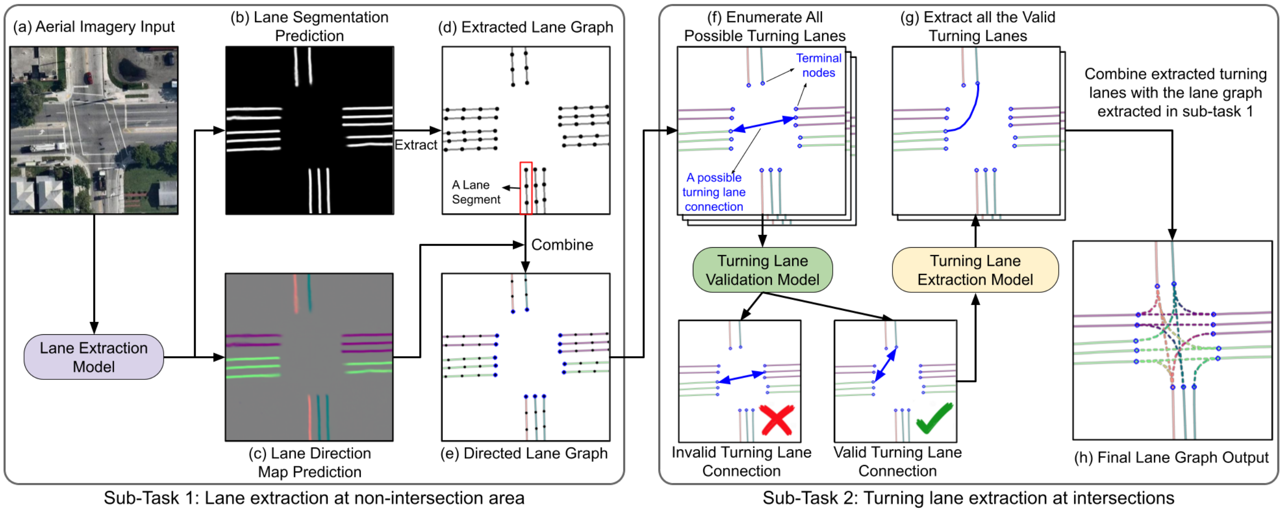

(a) Aerial Imagery Input
(b) Lane Segmentation Prediction
(d) Extracted Lane Graph
Extract
A Lane Segment
Combine
Lane Extraction Model
(c) Lane Direction Map Prediction
(e) Directed Lane Graph
Sub-Task 1: Lane extraction at non-intersection area
(f) Enumerate All Possible Turning Lanes
Terminal nodes
A possible turning lane connection
(g) Extract all the Valid Turning Lanes
Combine extracted turning lanes with the lane graph extracted in sub-task 1
Turning Lane Validation Model
Turning Lane Extraction Model
Invalid Turning Lane Connection
Valid Turning Lane Connection
(h) Final Lane Graph Output
Sub-Task 2: Turning lane extraction at intersections

Figure 2. Our proposed lane-level street map extraction pipeline.

We propose a mapping pipeline to extract lane graphs from aerial imagery. As shown in Figure 2, we split the lane graph extraction task into two sub-tasks. In the first sub-task, the goal is to extract lanes at non-intersection areas, where the lane graph is planar; therefore, we can adapt existing road extraction algorithms to extract the lanes. We show the details of this sub-task in Section 3.1.

In the second sub-task, the goal is to extract turning lanes at intersections. At intersections, the lane graph is no longer planar, and it contains many intersecting edges, making it very challenging to extract the lane graph. To overcome this challenge, we extract each turning lane separately instead of extracting the entire lane graph in one shot. As we have already extracted the lane graph at non-intersection areas, we can locate all the terminal vertices (the endpoints of lanes at intersections, highlighted in blue circles in Figure 2 (f)). Because each turning lane has to start from one terminal vertex and end at another terminal vertex, we can enumerate all the possible pairs of terminal vertices that may have valid turning lanes connecting them. In our mapping pipeline, we enumerate all the terminal vertex pairs whose distance (distance between two vertices in a pair) is below a threshold. Then, for each pair, we validate its connectivity using a classifier (a neural network). If the classifier indicates a turning lane connecting the pair of terminal vertices, we extract the vectors of the turning lanes (Figure 2 (g)). Finally, we merge the extracted turning lanes with the lane graph extracted in sub-task 1 to create a complete lane graph.

3.1. Lane Extraction at Non-Intersection Areas

In the first sub-task of the mapping pipeline, we extract the lane graph at non-intersection areas using a segmentation-based approach. In this approach, we represent the lane graph as a lane segmentation, e.g., Figure 2 (b), where each location in the segmentation has a value of either one or zero , indicating if that location has a lane or not. To extract the lane graph from aerial imagery, we first use a semantic segmentation model to extract the lane segmentation from aerial imagery and then extract the lane graph (Figure 2 (d)) from the lane segmentation. Different from road extraction tasks, we also need to extract the direction of the lanes. To do so, we extract a direction map (Figure 2 (c)) from the input aerial imagery and combine it with the extracted lane graph to create the final directed lane graph (Figure 2 (e)). Next, we discuss the details of the model architecture, and the training/inference processes.

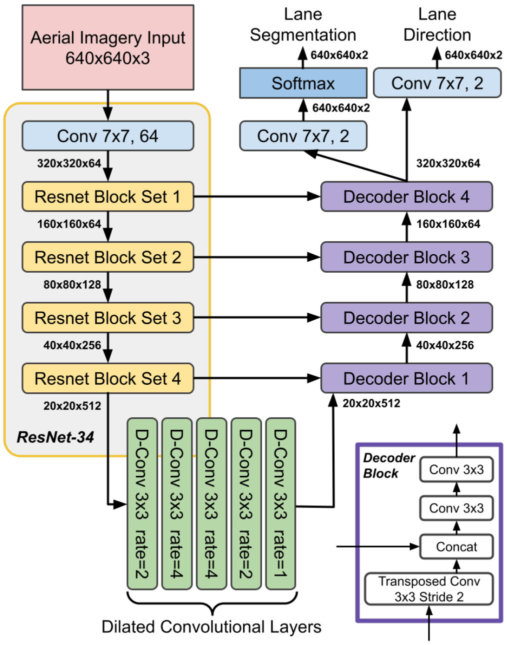


Figure 3. Lane Extraction Model

Lane extraction model. Suppose the input aerial image has a spatial dimension of $N \times N$ with three color channels. We use a convolutional neural network to extract the lane segmentation $s \in \mathbb{R}^{N \times N}, 0 \leq s_{i,j} \leq 1$ and the direction map $d \in \mathbb{R}^{N \times N \times 2}, -1 \leq d_{i,j,k} \leq 1$. In the direction map d, if the location (i,j) overlaps with a lane, we encode the lane direction as a normalized 2D vector in $d_{i,j}$; otherwise, we set $d_{i,j}$ to a zero vector.

In Figure 3, we show the architecture of our lane extraction model. Inspired by [44, 17], we adapts an UNet [33] structure with a ResNet [19] encoder for better feature extraction and dilated convolutional layers [41] for larger receptive field. To output the lane segmentation and the lane direction map, we use two branches after the last decoder block so that these two tasks can share most of the neural weights in the model. We use *softmax* as the activation function for the lane segmentation branch and use linear activation function for the direction map branch.

Ground truth. We create the ground truth lane segmentation $\hat{s}$ and the direction map $\hat{d}$ by rendering edges of the lane graph with a width of 5 pixels, which is 0.625-meter wide as the imagery in our evaluation dataset has a ground sampling distance (GSD) of 0.125 meters/pixel.

Training. We set the input window size to 640 × 640 during training. We intend to use this large window size so that the model can learn to use global information. For example, if trees occlude a short lane segment, the model should still extract it by using nearby information.

Inspired by the joint orientation learning work [9], we train the lane segmentation branch and the lane direction branch jointly as Batra et al. [9] have shown that learning road extraction jointly with road orientation can achieve better connectivity.

For the lane segmentation branch, inspired by the SpaceNet challenge [17], we use a linear combination of the cross-entropy loss ($\mathcal{L}_{ce}$) and the soft dice loss ($\mathcal{L}_{dice}$) as the loss function. For the lane direction branch, we use $\mathcal{L}_2$ loss. Therefore, the overall loss function is,

$$\mathcal{L} = \mathcal{L}_2(d, \hat{d}) + \frac{1}{2}(\mathcal{L}_{ce}(s, \hat{s}) + \mathcal{L}_{dice}(s, \hat{s})) \tag{1}$$

Inference. We need to apply our model to large aerial images whose size is often over thousands of pixels during inference. To run the lane extraction model on such large input images, we use a 2-dimension sliding window approach where the sliding window size is 640 × 640. Each time we move this sliding window by 256 pixels either vertically or horizontally to cover the entire input image. Because we move the sliding window 256 pixels each time, the inference results from different sliding window instances may overlap on each other. When this happens, we use the average result from different sliding window instances.

Lane graph extraction. After we extract the lane segmentation and the lane direction map from the input aerial image, we first extract the lane graph (undirected) from the lane segmentation. Similar to other segmentation-based road extraction work, we binarize the output lane segmentation with a threshold (e.g., 0.5), create a skeleton from this binary segmentation mask using morphology thining, and turn the skeleton into a graph.

Next, we use the direction map output to assign directions to the lane graph. As shown in Figure 2 (d), we first decompose the lane graph into individual lane segments. Each lane segment consists of a sequence of edges $[e_1, ..., e_m]$ (e_i connects to e_{i+1}). Because the direction map output can be very noisy, we use the entire lane segment to decide its direction. Formally, we compute the following value for each lane segment.

$$c = \sum_{i=1}^{m} \sum_{(x,y) \in P(e_i)} \langle D(e_i), d_{x,y} \rangle \tag{2}$$

, where $P(e) = \{(x,y) | \text{Edge e intersects location (x,y)}\}$, $D(e)$ is the normalized direction vector of edge e, and $\langle\, , \rangle$ is the vector inner product operator. If c is greater than zero for a lane segment $[e_1, ..., e_m]$, that indicates the direction of the lane segment should be from e_1 to e_m; otherwise, the direction should be the opposite — from e_m to e_1.

3.2. Turning Lane Extraction

After the first stage of the mapping pipeline, we get a lane graph covering all the non-intersection areas, and this lane graph has many terminal nodes at intersections. In the second stage, we extract the turning lanes at intersections by examining all the terminal node pairs whose distances are below a threshold, i.e., 70 meters. We consider those terminal node pairs as candidate turning lanes. For each terminal node pair (v_A, v_B), we use neural network models to validate the turning lane going from node A to node B and extract the turning lane vectors. In Figure 4, we show the detailed workflow.

Input. For each terminal node pair (v_A, v_B), we consider a window of size $N \times N$ centered at the midpoint between node A and node B. We use the aerial image, the direction map (extracted in stage-1) in this window as inputs. Meanwhile, for each node, we create an auxiliary input $p \in \mathbb{R}^{N \times N \times 3}$ whose first channel is a binary mask indicating the position of the node, and the second and third channels encodes the offsets from the node. Formally, suppose the position of the node is at (n_x, n_y), we define p as,

$$\begin{gathered} p_{x,y,1} = 1 \text{ if } x = n_x, y = n_y, \text{ otherwise } 0 \\ p_{x,y,2} = \frac{1}{N}|x - n_x| \quad p_{x,y,3} = \frac{1}{N}|y - n_y| \end{gathered} \tag{3}$$

We use p to provide auxiliary position information for the neural network models so that the model can reason the relative distances effectively.

Turning lane validation. To validate a candidate turning lane (v_A, v_B), we first extract the segmentation of the

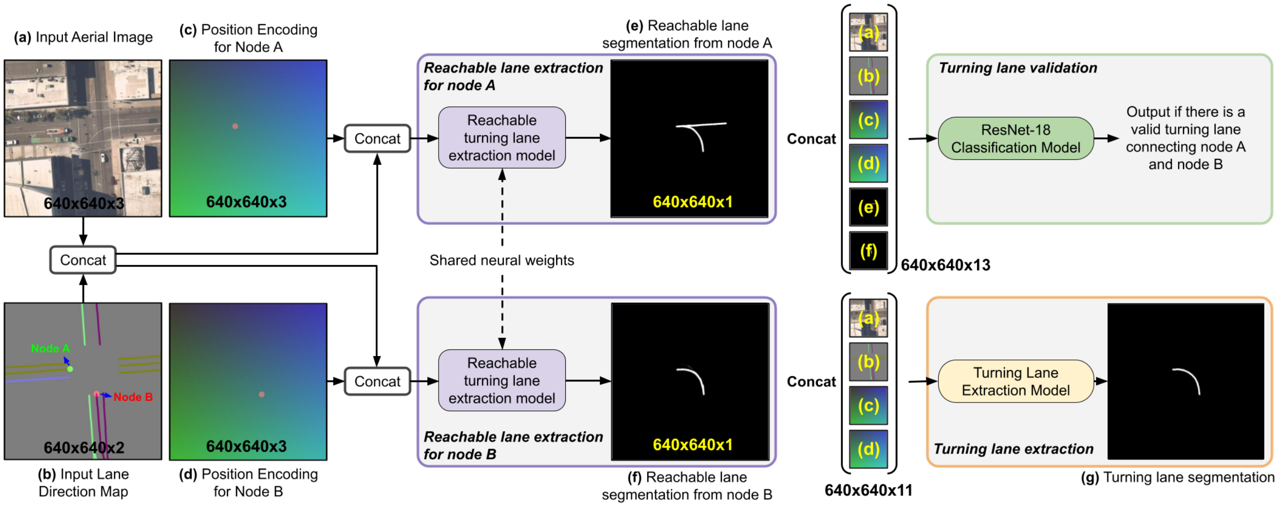

(a) Input Aerial Image
640x640x3
(c) Position Encoding for Node A
640x640x3
Concat
Reachable lane extraction for node A
Reachable turning lane extraction model
(e) Reachable lane segmentation from node A
640x640x1
Concat
(a)
(b)
(c)
(d)
(e)
(f)
640x640x13
Turning lane validation
ResNet-18 Classification Model
Output if there is a valid turning lane connecting node A and node B
Shared neural weights
Concat
Node A
Node B
640x640x2
(b) Input Lane Direction Map
640x640x3
(d) Position Encoding for Node B
Concat
Reachable turning lane extraction model
Reachable lane extraction for node B
640x640x1
(f) Reachable lane segmentation from node B
Concat
(a)
(b)
(c)
(d)
640x640x11
Turning Lane Extraction Model
Turning lane extraction
(g) Turning lane segmentation

Figure 4. Given a pair of terminal nodes A and B, we use the above architecture to infer if there is a valid turning lane connecting node A and node B. If so, we extract the turning lane vectors using a segmentation approach.

reachable lanes from node A and node B. Here the reachable lanes from a node are the lanes that can be reached on the lane graph through only turning lanes. As examples, we show the corresponding reachable lanes from node A and node B in Figure 4 (e) and (f), respectively. We extract the reachable turning lane segmentation using a neural network that has the same architecture (and the same loss function) as Figure 3 except the direction prediction branch.

After we extract the reachable lane segmentation, we concatenate it with all other input data and feed them into a binary classification model that predicts if the candidate turning lane is valid or not.

Alternative-1: An alternative design is to get rid of the reachable lane extraction modules and directly feed the input data to a binary classification model. Our evaluation finds that this design yields poor accuracy on the testing dataset because the turning lane validation task depends on spatial information such as the relative positions of lanes and lane markers. However, the encoder-only structure in the binary classification model makes it difficult to reason the spatial correlation effectively. In contrast, our design uses the reachable lane extraction model that adapts a UNet (encoder-decoder) structure to help reason the spatial correlation. Therefore, our design can achieve much higher accuracy on unseen data.

Turning lane extraction. To extract the turning lane, we first adopt the same neural network model as the reachable lane extraction model to extract the segmentation of the turning lane, and then extract the lane vector from the segmentation (similar to sub-task 1). In our evaluation, we find that this solution is sufficient to achieve very high accuracy.

Alternative-2: With this turning lane extraction model, we can derive an alternative design for turning lane validation. In this alternative design, we train a lane extraction model to produce the turning lane segmentation if the input node pair is valid and produce an empty segmentation map otherwise. We check if a path connects the two nodes in the segmentation output to validate the turning lane during testing. This alternative design is much simpler than ours. However, this alternative design also performs poorly because it often produces noisy and disconnected segmentation for valid turning lanes, making it hard to reach a clear decision by only checking the segmentation output.

Training. During training, we randomly sample terminal node pairs whose distances are shorter than 70 meters in the ground truth lane graph and use the ground truth labels to generate the corresponding direction maps. For the turning lane validation model, we train it together with the reachable lane extraction model end-to-end. For the turning lane extraction model, we train it with *only* valid terminal node pairs so that the model can always produce a sharp turning lane segmentation.

4. Evaluation

4.1. Dataset

We find there are very limited public datasets or resources for lane-level map extraction. One of the related datasets is the Argoverse dataset [13] where they provide two high-definition maps, but they do not provide the corresponding aerial imagery. Hence, to evaluate our proposed mapping pipeline, we create our dataset.

Our dataset covers about 400 km of lanes in four cities, Miami, Boston, Seattle, and Phoenix. For the aerial imagery, we collect them from MapBox [1] and resize the images to 0.125 meters per pixel. For the lane graph labels, we adapt the lane graph labels in the Argoverse dataset for Miami. We tweak the lane graph labels so that they match our aerial imagery. To improve the diversity of the dataset, we manually annotate the lane graphs in three more cities, Boston, Seattle, and Phoenix. We organize the dataset as 35 tiles. The dimension of each tile is 4096 by 4096 (pixels). We use 24 tiles as the training dataset and 11 tiles as the testing dataset. Our dataset is available on GitHub.

4.2. Metrics

To evaluate the quality of the extracted lane graphs, we adapt two widely used metrics in road extraction problems, the GEO metric and TOPO metric.

GEO metric. In the GEO metric, we interpolate (densify) the ground truth lane graph and the extracted lane graph so that the distances between any two connected vertices are 0.25 meters. After the interpolation, we got the ground truth lane graph $\hat{G} = \{\hat{V}, \hat{E}\}$, and the extracted lane graph $G = \{V, E\}$, where $\hat{V}, V$ are the sets of vertices and $\hat{E}, E$ are the sets of edges. We consider a pair of vertices $(v \in V, \hat{v} \in \hat{V})$ as a valid match if the distance between the two vertices is less than r meters. Then, we compute a maximal one-to-one matching between $\hat{V}$ and V and denote the matched vertices in the extracted lane graph as V_{match}. Finally, we report the precision $\frac{|V_{match}|}{|V|}$, recall $\frac{|V_{match}|}{|\hat{V}|}$ and F_1 score based on the matching result.

The threshold r determines the error tolerance of the metric. In our evaluation, we set it to 1 meter so that the metric can penalize minor errors.

TOPO metric. The GEO metric focuses on local correctness, but it does not take connectivity into account. For example, if there is a small missing gap in an extracted lane, the GEO metric will still report a very high recall even though the small missing gap makes the entire lane disconnected. In contrast, the TOPO metric takes connectivity into account. We implement the TOPO metric on top of the GEO metric. For each matched vertex pair $(v, \hat{v})$ in the GEO metric, we consider the sub-graphs S_v and $\hat{S}_{\hat{v}}$ on G and $\hat{G}$ where all the vertices in S_v and $\hat{S}_{\hat{v}}$ can be reached from v and $\hat{v}$ by walking on the graph for less than 50 meters, respectively. We then compute the GEO metric between the two sub-graphs S_v and $\hat{S}_{\hat{v}}$, denoted as $\text{Precision}_{\text{GEO}}(S_v, \hat{S}_{\hat{v}})$ and $\text{Recall}_{\text{GEO}}(S_v, \hat{S}_{\hat{v}})$. Finally, we report the TOPO precision and recall defined as,

$$\begin{aligned}\text{Precision}_{\text{TOPO}} &= \frac{\sum_{\text{matched }(v,\hat{v})} \text{Precision}_{\text{GEO}}(S_v, \hat{S}_{\hat{v}})}{|V|} \\ \text{Recall}_{\text{TOPO}} &= \frac{\sum_{\text{matched }(v,\hat{v})} \text{Recall}_{\text{GEO}}(S_v, \hat{S}_{\hat{v}})}{|\hat{V}|}\end{aligned} \tag{4}$$

Here, we can consider the TOPO metric as a weighted version of the GEO metric where each matched vertex pair only contributes a fraction to the precision and recall based on the correctness of the local connectivity.

Directed versions. To evaluate the directed lane graphs, we derive the directed versions of the GEO and TOPO metric. In the directed version, we assign a direction to each vertex based on its edges (we ignore vertices that have more than two neighbors). During matching, we only consider the vertex pairs whose angle differences are less than 60 degrees. We use these directed versions to evaluate lane graphs with driving direction information.

4.3. Implementation Details

We implement our mapping pipeline in Tensorflow [2] and use Adam optimizer [28] for training. For all the models, we train them for 500 epochs on our training dataset. We start with a learning rate of 0.001 and decrease it by ten at the 350-th epoch and 450-th epoch during training. We augment the input images with random rotations, cropping, color balances, and brightness. Our implementation is available on GitHub.

4.4. Performance of each component

In this section, we evaluate each component in our mapping pipeline separately. The components include,

(1) Undirected lane extraction in sub-task 1.
(2) Lane direction inference in sub-task 1.
(3) Turning lane validation in sub-task 2.
(4) Turning lane extraction in sub-task 2.

Undirected lane graph extraction. We report the evaluation result of the undirected lane graph extracted at non-intersection areas (see Figure 2(d)). We compare the extracted graph with the ground truth lane graph using GEO and TOPO metric (undirected). In Table 1, we report the precision, recall, and F_1-scores of our proposed model and several alternatives. As shown in Table 1, our proposed solution achieves decent F_1 scores in both the GEO metric and TOPO metric. Compared with the alternatives, adding dilated layers and using ResNet encoder can help improve the lane extraction accuracy. Overall, our model improves the GEO F_1-score by 4.2 points and improves the TOPO F_1-score by 6.9 points against the UNet baseline.

Lane direction inference. For each lane in the ground truth graph, we extract its direction from the lane direction map prediction (see Figure 2(c)) and compare the extracted direction with the actual direction. If the directions match, we consider the lane as a correct lane. In Table 2, we report the lane direction inference accuracy, which is the ratio between the total length of the correct lanes and the total length of all lanes.

As shown in Table 2, we find all the models can achieve high overall accuracy (around 94%). If we look at the accuracy on one-way and two-way roads separately, we can find that the accuracy on two-way roads is much higher than the accuracy on one-way roads because inferring the lane direction on one-way roads is very challenging. Usually, not many visible signs from aerial imagery (e.g., arrows) can explicitly tell the lane's direction. The neural network model often has to rely on weak indicators such as the head-

Method	GEO metric			TOPO metric		
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score
Basic UNet	0.811	0.762	0.786	0.747	0.622	0.679
Add Dilated Layers	0.818	0.792	0.805	0.753	0.680	0.715
With ResNet18 Encoder	0.828	0.812	0.820	0.768	0.708	0.737
With ResNet34 Encoder	**0.835**	**0.821**	**0.828**	**0.774**	**0.724**	**0.748**

Table 1. Lane extraction accuracy at non-intersection areas.

ing of cars and the positions of stop lines to speculate the direction.

Method	Lane direction inference accuracy		
	One-way	Two-way	All
Basic UNet	67.68%	97.81%	93.95%
+Dilated Layers	66.73%	98.22%	94.19%
+ResNet18	**71.40%**	98.15%	**94.72%**
+ResNet34	69.61%	**98.23%**	94.56%

Table 2. In our testing dataset, about 14.7% of lanes (in terms of lane length) belong to one-way roads. This table reports the lane direction inference accuracy for one-way roads, two-way roads, and all roads separately.

Turning lane validation. To evaluate the turning lane validation model, we enumerate all the turning lane candidates (a pair of terminal vertices whose distance is below 70 meters) at the intersections in the ground truth lane graph. Suppose we have N turning lane candidates. We check each of them using our lane validation model and produce a label indicating the valid turning lanes during the evaluation. Let V_{GT} be the set of ground truth valid turning lanes, and V_{Infer} be the set of predicted valid turning lanes. Because the counts of valid turning lanes and invalid turning lanes are imbalanced, we do not use accuracy as the metric. Instead, we report the precision, recall, F_1-score, and IoU defined as,

$$\text{Precision} = \frac{|V_{GT} \cap V_{Infer}|}{|V_{Infer}|} \quad \text{Recall} = \frac{|V_{GT} \cap V_{Infer}|}{|V_{GT}|}$$
$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{IoU} = \frac{|V_{GT} \cap V_{Infer}|}{|V_{GT} \cup V_{Infer}|} \tag{5}$$

Method	Prec.	Rec.	F_1	IoU
Classification only	0.391	0.443	0.415	0.262
Segmentation only	0.586	0.892	0.707	0.547
Ours full solution	**0.921**	**0.961**	**0.941**	**0.888**

Table 3. Turning lane validation performance.

We show the results of our proposed solution and two alternatives in Table 3. Our solution achieves a high F_1-score of 94.1%, which is much higher than the F_1-scores of the two alternative solutions (Section 3.2) – 41.5% for the classification only alternative and 70.7% for the segmentation only alternative.

Turning lane extraction. We compare each extracted turning lane (in graph format) with the ground truth turning lane using the TOPO metric. We find our proposed method has very high accuracy in this task; it achieves an average TOPO precision of 94.2% and an average TOPO recall of 95.8%.

4.5. Overall performance

We compare the complete lane graph extracted from our mapping pipeline with the ground truth lane graph using the GEO and TOPO metrics (directed). In Table 4, we report the precision, recall, and F_1-scores of the extracted lane graphs. We also show the results of the partial lane graphs from two earlier stages in the mapping pipeline. Overall, our mapping pipeline achieves a 76.5% F_1-score in the GEO metric and a 62.7% F_1-score in the TOPO metric. We consider this a decent result because the matching threshold in our GEO metric and TOPO metric is only one meter.

4.6. Qualitative results

We show examples of the extracted lane graphs in Figure 5 (a-d). In those examples, our mapping pipeline correctly extracts the lane graphs in many challenging scenarios including lane diverging, left-only and right-only turning lanes in Figure 5 (a), four-lane four-way intersection in Figure 5 (b), two-lane four-way intersections in Figure 5 (c), and three-way intersections in Figure 5 (d).

Failure example. In Figure 5 (e), we show an example of the output lane graph that has several errors. As shown in Figure 5 (e), our mapping pipeline fails to extract the correct lane geometry and direction at non-intersection areas in the red rectangles c,d, and e, which in turn messes up the turning lane extraction in the right two intersections. We observe similar errors on small roads where the lane markers are missing, unclear, or the entire roads are occluded.

Our model also fails to infer the u-turn lane (red rectangle a) and an *unusual* right turn lane (red rectangle b). These two failure cases are examples of a fundamental challenge in aerial-imagery-based lane graph extraction – it is often hard or even impossible to infer the correct turning lanes at some intersections without ground road sign information.

Limitation. The above failures unveil a major limitation of our mapping pipeline — the quality of the extracted lane graph depends on the visibility of important road features such as lane markers. This limitation motivates a future work to estimate a confidence score for the extracted lane graph based on the visibility of roads. With this confidence estimator, we can still use our mapping pipeline to extract

Lane graphs at different pipeline stages	GEO metric			TOPO metric		
	Precision	Recall	F_1 Score	Precision	Recall	F_1 Score
Lane graph at non-intersection areas (undirected)	0.835	0.821	0.828	0.774	0.724	0.748
Lane graph at non-intersection areas (directed)	0.800	0.787	0.793	0.745	0.697	0.720
Final lane graph (directed)	0.770	0.760	0.765	0.612	0.642	0.627

Table 4. Accuracy of the lane graphs at different pipeline stages.

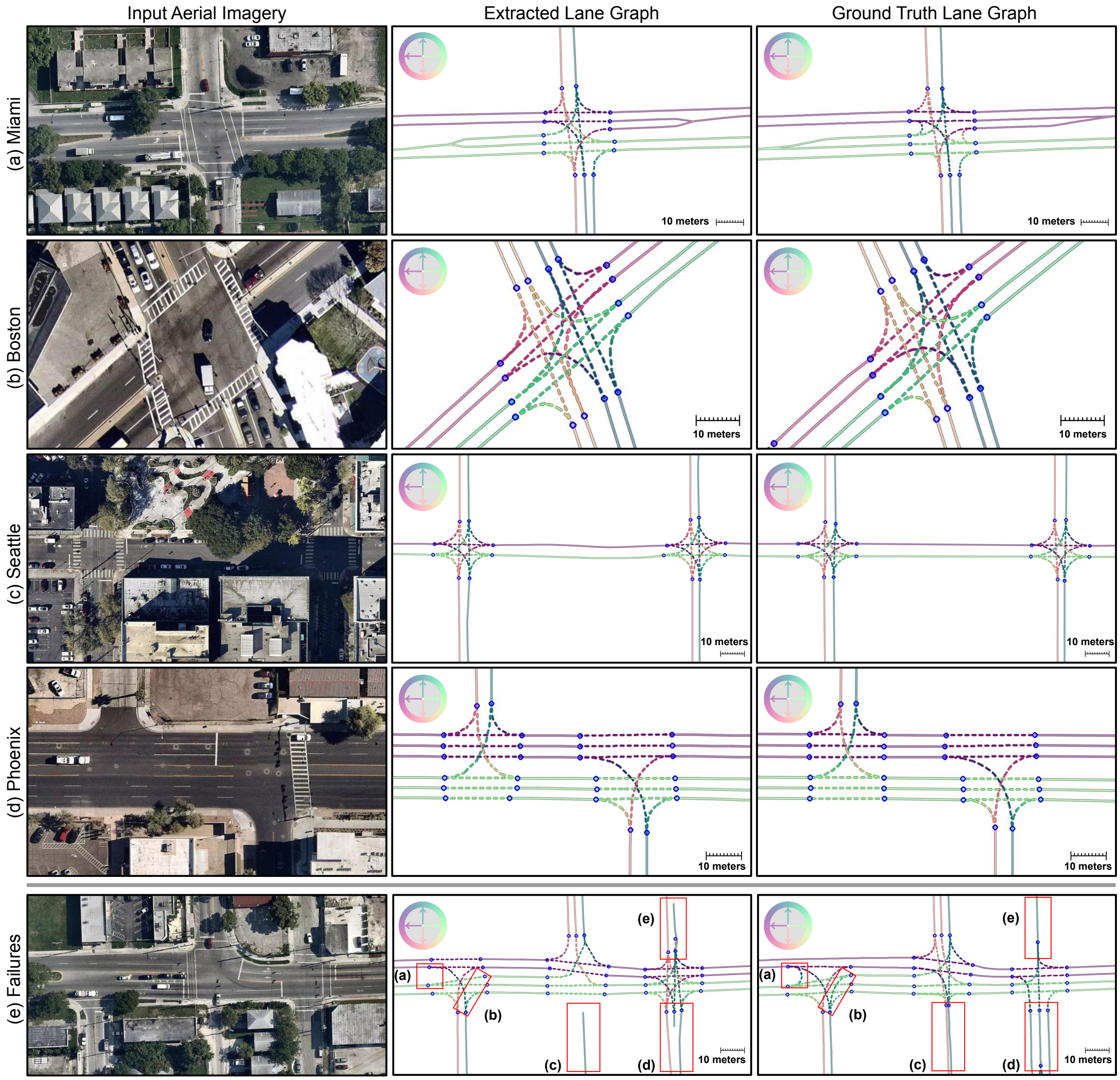


Figure 5. Examples of the extracted lane graph from our testing dataset.

lane graphs on high-confidence areas, e.g., the places in Figure 5 (a-d), and complement other mapping solutions.

5. Conclusion

In this work, we propose a mapping pipeline to extract routable lane-level street maps from aerial imagery. To the best of our knowledge, this is the first work that proposes a complete mapping pipeline for routable lane-level street map extraction from aerial imagery. We evaluate our solution on a dataset consisting of 400 km of lanes, showing the effectiveness of our solution and unveiling several challenges in the problem. Overall, we show that extracting lane-level street maps from aerial imagery is a promising direction toward scalable data-driven map making.

References

[1] Mapbox. www.mapbox.com. Accessed: 2021-03-01.

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632, 2015.

[4] Mohammad Ali Arman and Chris MJ Tampère. Lane-level routable digital map reconstruction for motorway networks using low-precision gps data. *Transportation Research Part C: Emerging Technologies*, 129:103234, 2021.

[5] Seyed Majid Azimi, Peter Fischer, Marco Körner, and Peter Reinartz. Aerial lanenet: Lane-marking semantic segmentation in aerial imagery using wavelet-enhanced cost-sensitive symmetric fully convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 57(5):2920–2938, 2018.

[6] Favyen Bastani, Songtao He, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. Machine-assisted map editing. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 23–32, 2018.

[7] 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.

[8] Favyen Bastani, Songtao He, Satvat Jagwani, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Sam Madden, and Mohammad Amin Sadeghi. Updating street maps using changes detected in satellite imagery. *arXiv preprint arXiv:2110.06456*, 2021.

[9] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, CV Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10385–10393, 2019.

[10] James Biagioni and Jakob Eriksson. Map inference in the face of noise and disparity. In *ACM SIGSPATIAL 2012*, 2012.

[11] Alexander Buslaev, Selim Seferbekov, Vladimir Iglovikov, and Alexey Shvets. Fully convolutional network for automatic road extraction from satellite imagery. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 207–210, 2018.

[12] Lili Cao and John Krumm. From GPS traces to a routable road map. In *ACM SIGSPATIAL*, pages 3–12, 2009.

[13] Ming-Fang Chang, John W Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[14] Hang Chu, Daiqing Li, David Acuna, Amlan Kar, Maria Shugrina, Xinkai Wei, Ming-Yu Liu, Antonio Torralba, and Sanja Fidler. Neural turtle graphics for modeling city road layouts. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4522–4530, 2019.

[15] Jonathan J Davies, Alastair R Beresford, and Andy Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4), 2006.

[16] Stefan Edelkamp and Stefan Schrödl. Route planning and map inference with global positioning traces. In *Computer Science in Perspective*. Springer, 2003.

[17] Adam Van Etten. City-scale road extraction from satellite imagery v2: Road speeds and travel times. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1786–1795, 2020.

[18] Peter Fischer, Seyed Majid Azimi, Robert Roschlaub, and Thomas Krauß. Towards hd maps from aerial imagery: Robust lane marking segmentation using country-scale imagery. *ISPRS International Journal of Geo-Information*, 7(12):458, 2018.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[20] Songtao He, Favyen Bastani, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. RoadRunner: Improving the precision of road network inference from gps trajectories. In *ACM SIGSPATIAL*, 2018.

[21] Songtao He, Favyen Bastani, Satvat Jagwani, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Mohamed M Elshrif, Samuel Madden, and Mohammad Amin Sadeghi. Sat2graph: road graph extraction through graph-tensor encoding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*, pages 51–67. Springer, 2020.

[22] Songtao He, Favyen Bastani, Satvat Jagwani, Edward Park, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Samuel Madden, and Mohammad Amin Sadeghi. Roadtagger: Robust road attribute inference with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10965–10972, 2020.

[23] Songtao He, Mohammad Amin Sadeghi, Sanjay Chawla, Mohammad Alizadeh, Hari Balakrishnan, and Samuel Madden. Inferring high-resolution traffic accident risk maps based on satellite imagery and gps trajectories. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11977–11985, 2021.

[24] 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.

[25] Wonje Jang, Jhonghyun An, Sangyun Lee, Minho Cho, Myungki Sun, and Euntai Kim. Road lane semantic segmentation for high definition map. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1001–1006. IEEE, 2018.

[26] Jinyong Jeong, Younggun Cho, and Ayoung Kim. Road-slam: Road marking based slam with lane-level accuracy. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1736–1473. IEEE, 2017.

[27] Soren Kammel and Benjamin Pitzer. Lidar-based lane marker detection and mapping. In *2008 IEEE Intelligent Vehicles Symposium*, pages 1137–1142. IEEE, 2008.

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[29] Zuoyue Li, Jan Dirk Wegner, and Aurélien Lucchi. Topological map extraction from overhead images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1715–1724, 2019.

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

[31] Alameen Najjar, Shun'ichi Kaneko, and Yoshikazu Miyanaga. Combining satellite imagery and open data to map road safety. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[32] Baoxing Qin, ZJ Chong, Tirthankar Bandyopadhyay, Marcelo H Ang, Emilio Frazzoli, and Daniela Rus. Road detection and mapping using 3d rolling window. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1239–1246. IEEE, 2013.

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[34] Rade Stanojevic, Sofiane Abbar, Saravanan Thirumuruganathan, Sanjay Chawla, Fethi Filali, and Ahid Aleimat. Robust road map inference through network alignment of trajectories. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 2018.

[35] 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.

[36] Adam Van Etten. City-scale road extraction from satellite imagery. *arXiv preprint arXiv:1904.09901*, 2019.

[37] Zhenhua Xu, Yuxiang Sun, and Ming Liu. icurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving. *IEEE Robotics and Automation Letters*, 6(2):1097–1104, 2021.

[38] Zhenhua Xu, Yuxiang Sun, and Ming Liu. Topo-boundary: A benchmark dataset on topological road-boundary detection using aerial images for autonomous driving. *arXiv preprint arXiv:2103.17119*, 2021.

[39] Zhenhua Xu, Yuxiang Sun, Lujia Wang, and Ming Liu. Cp-loss: Connectivity-preserving loss for road curb detection in autonomous driving with aerial images. *arXiv preprint arXiv:2107.11920*, 2021.

[40] Xiaofei Yang, Xutao Li, Yunming Ye, Raymond YK Lau, Xiaofeng Zhang, and Xiaohui Huang. Road detection and centerline extraction via deep recurrent convolutional neural network u-net. *IEEE Transactions on Geoscience and Remote Sensing*, 57(9):7209–7220, 2019.

[41] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[42] Andi Zang, Runsheng Xu, Zichen Li, and David Doria. Lane boundary extraction from satellite imagery. In *Proceedings of the 1st ACM SIGSPATIAL Workshop on High-Precision Maps and Intelligent Applications for Autonomous Vehicles*, pages 1–8, 2017.

[43] Junqiao Zhao, Xudong He, Jun Li, Tiantian Feng, Chen Ye, and Lu Xiong. Automatic vector-based road structure mapping using multibeam lidar. *Remote Sensing*, 11(14):1726, 2019.

[44] 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.

[45] Yiyang Zhou, Yuichi Takeda, Masayoshi Tomizuka, and Wei Zhan. Automatic construction of lane-level hd maps for urban scenes. *arXiv preprint arXiv:2107.10972*, 2021.