

Lane Segmentation Refinement with Diffusion Models

Antonio Ruiz^{1,2} ✉, Andrew Melnik^{1†,2}, Dong Wang^{1*}, and Helge Ritter²

¹ Riemann Lab, Huawei

² Bielefeld University

Abstract. The lane graph is a key component for building high-definition (HD) maps and crucial for downstream tasks such as autonomous driving or navigation planning. Previously, He et al. (2022) explored the extraction of the lane-level graph from aerial imagery utilizing a segmentation-based approach. However, segmentation networks struggle to achieve perfect segmentation masks resulting in inaccurate lane graph extraction. We explore additional enhancements to refine this segmentation-based approach and extend it with a diffusion probabilistic model (DPM) component. This combination further improves the GEO F1 and TOPO F1 scores, which are crucial indicators of the quality of a lane graph, in the undirected graph in non-intersection areas. We conduct experiments on a publicly available dataset, demonstrating that our method outperforms the previous approach, particularly in enhancing the connectivity of such a graph, as measured by the TOPO F1 score. Moreover, we perform ablation studies on the individual components of our method to understand their contribution and evaluate their effectiveness.

Keywords: Lane Segmentation · Lane Graph Connectivity · Aerial Imagery · Diffusion Models

1 Introduction

High-definition (HD) maps play an important role in the development of autonomous driving and advanced driver-assistance systems (ADAS) [11, 15]. Such maps can be constructed and updated by collecting data with an ego-vehicle, equipped with multiple sensors such as LiDARs, cameras, inertial measurement units (IMUs), etc. However, this way of building and updating HD maps is time-consuming and expensive, resulting in limited coverage. In contrast, utilizing aerial imagery to extract structured information for such maps offers a more efficient and scalable solution.

A crucial component of HD maps is the lane graph since its high accuracy can enable certain level of autonomous driving. Nevertheless, building an accurate lane graph from aerial imagery is quite hard, since this type of imagery presents

[†] This work was done while Andrew was working at Huawei.

* Project Leader.

✉ Corresponding author. Email: aruiz@techfak.uni-bielefeld.de

some challenges, such as low resolution of objects on the ground, occlusions caused by trees, queues of cars or bridges and shadowed sections due to tall buildings.

Approaches for extracting the lane graph from aerial imagery can be divided into two main categories: segmentation-based methods [14, 38] and iterative graph-based methods [5]. Segmentation-based methods start by predicting lane segmentation masks, which are then used to extract the lane graph with traditional graph algorithms. On the other hand, iterative graph-based methods utilize an agent to predict local lane graphs based on its position, which are then aggregated by navigating the agent across the image. Each approach offers distinct advantages and challenges. However, we will concentrate on segmentation-based approaches, as our method is based on one of them. Segmentation-based methods rely on segmentation networks. Yet, these networks are susceptible to the complexities inherent in aerial imagery, resulting in noisy segmentation mask predictions. Given that the algorithm for extracting the graph heavily relies on the quality of these segmentation masks, any inaccuracy present in them significantly affects the overall performance of these methods. Our method aims to rectify the inaccuracies in the lane segmentation masks produced by the segmentation network, thereby enhancing the overall quality of the lane graph extracted from them.

Since our method exclusively focuses on extracting the undirected lane graph in non-intersection areas, we will simply refer to this specific graph as the lane graph throughout the paper to avoid redundancy.

Recently, diffusion models [16, 30, 31] have emerged as an alternative to solve segmentation problems, demonstrating state-of-the-art results in specific domains [2, 34–36]. We employ diffusion models to enhance the quality of segmentation masks, which improves the overall quality of the extracted lane graph. Methods [2, 34–36] that leverage diffusion models for segmentation problems depend on ensembles of these models to improve the quality of segmentation mask predictions beyond what could be achieved with a single diffusion model. However, this approach falls short when segmenting thin lines, a critical challenge in the context of lane segmentation masks (refer to the discussion in Section 2.4). To overcome this limitation, our approach avoids the ensemble strategy and conditions the initial latent variable of the diffusion sampling process on the segmentation masks produced by the segmentation network instead. This results in a refinement of the segmentation masks. A remarkable aspect of this refinement process is the improvement in the connectivity of the lane graph, a highly desired attribute.

In summary, the main contributions of this paper are:

- We propose a novel yet simple approach that merges a segmentation network with a diffusion model to produce refined lane segmentation masks from aerial imagery. These refined masks are subsequently utilized to extract the lane graph with a traditional algorithm. This synergistic combination increases the performance beyond the capabilities of each model when used alone (see Table 2). Moreover, by conditioning the initial latent variable of

- the diffusion sampling process with the segmentation masks, we achieve a good refinement in just a few sampling steps (see first row of Table 3).
- Our experiments on a public dataset show that our method surpasses the previous approach (see Table 1). In particular, the lane graph connectivity is enhanced, which is critical for downstream tasks such as autonomous driving. Furthermore, to the best of our knowledge, our method is one of the first that explores diffusion models in the context of lane-level graph extraction from aerial imagery.
 - In addition, we perform multiple experiments to analyze the impact of different variants of our method on the overall lane graph metrics.

2 Related Work

2.1 Lane Graph Extraction from Onboard Sensors

There are previous works [17, 18, 39, 41, 42] that utilize either street-view images or 3D point clouds obtained from onboard sensors such as cameras and LiDAR to extract the lane graph. Homaounfar *et al.* [17] train a hierarchical recurrent neural network to predict lane boundaries with a regression loss. Zürn *et al.* [42] use a multi-modal BEV input to train a Graph R-CNN [37] with a regression loss and a post-processing algorithm to predict the lane graph of local BEV patches. In both works, the ground truth data is obtained by projecting 3D LiDAR point clouds into a BEV representation. DAGMapper [18] utilizes LiDAR intensity images as input for a recurrent convolutional network architecture with three heads that parameterize a directed acyclic graphical model (DAGM), which is used to build the lane graph in an iterative manner. Zhang *et al.* [39] combine sensing data from LiDAR and cameras to train a hierarchical fully convolutional network that predicts a lane-level road network. Zhou *et al.* employ an encoder-decoder architecture to derive semantic maps from camera and LiDAR data, accumulating frames into a LiDAR-synchronized BEV projection and add Open Street Map (OSM) data, aiding the model to determine lane connections at intersections. Can *et al.* [7, 8] employ images of a single onboard camera. In their first work [7], they use a Transformer [32] with two heads that predicts lane center-lines and bounding boxes of objects. In their second work [8], they refine their approach by replacing the object detection head with a minimal cycle head. This head is designed to identify the smallest cycles formed by directed curve segments between intersections.

2.2 Lane Graph Extraction from Aerial Imagery

There are very few works [5, 14, 38] that attempt to extract the lane graph from aerial imagery. This might be attributed to the lack of a popularized and standardized dataset for aerial imagery, similar to how nuScenes [6] is employed for onboard data. Büchner *et al.* [5] use centerline regressor models to iteratively predict the successor graph from a BEV patch. They identify potential nodes

within corridors likely to contain the successor graph, which is then predicted using a message passing network [4]. Yao et al. [38] implement a two-step method for predicting the lane graph. Initially, they utilize HRNet [33] to generate a segmentation map, a feature map, and a vertex map. Subsequently, a nearest-neighbor strategy is applied to the vertex map, in conjunction with the other outputs from the first stage. These are then used by a two-headed network to predict the vertex connections.

He *et al.* [14] splits the problem into two subtasks: lane extraction in non-intersection areas and turning lane extraction at intersections. In the first subtask, a D-LinkNet [40] is employed to extract segmentation and direction maps. Subsequently, in the second subtask, a classifier distinguishes between valid and invalid lane connections. These lane connections are chosen from a pool of candidates, which is constructed based on a distance threshold criterion between terminal nodes within the graph derived from non-intersection areas (first subtask). This process is followed by a prediction of the geometry segmentation for valid connections, using a D-LinkNet similar to the one previously employed. Finally, the lane graphs derived from both subtasks are integrated. Our method is an extension of this work, but it is specifically tailored to enhance lane extraction in non-intersection areas (first subtask).

2.3 Diffusion Probabilistic Models

Diffusion probabilistic models (DPM) [30], or simply referred as diffusion models, are generative models that leverage Markov chains and variational inference to synthesize new data from simple known distributions, such as Gaussian distributions. They comprise three principal components: the forward process, the reverse process, and the sampling procedure [9]. During the forward process, noise is gradually added to the data over a fixed time horizon, following a Gaussian distribution with a pre-defined variance schedule. Subsequently, a model is trained to learn the reverse process. The sampling procedure then employs this learned backward process to generate new data from a known Gaussian distribution. Denoising diffusion probabilistic models (DDPM) [16] finds a simple parametrization for the model that learns the reverse process, resulting in the generation of high-quality images, which follow-up works [10, 23] improve further. Denoising diffusion implicit models (DDIM) [31] accelerates the sampling procedure by generalizing DDPM to non-Markovian chains, resulting in a substantial reduction of the sampling steps relative to the forward steps.

2.4 Conditional Diffusion Probabilistic Models

Conditional diffusion probabilistic models are a type of DPM that condition the generation on some additional input, it could be class labels [10, 23], or embeddings [27] of the external input, which may include text, semantic maps, images, etc.

This type of diffusion models has been used for image segmentation, showing state-of-the-art performance in specific domains [2, 34–36]. SegDiff [2] employs a

conditional DPM for segmentation tasks from diverse domains, where the segmentation mask is diffused and the corresponding image conditions the diffusion process. They implement several instances of the sampling process, creating an ensemble of diffusion models, whose outputs are averaged to get the segmentation mask. Subsequent studies, MedSegDiff [35] and MedSegDiff-V2 [36], adapt and refine this methodology with architectural modifications for segmenting medical imagery.

Nonetheless, an ensemble of diffusion models is not adequate to segment thin lines, which is the case of the lane segmentation mask of a lane graph. This is due to slight lateral variations in the segmented pixels across the ensemble, resulting in a blurred average segmentation mask, which leads to poor results in the predicted lane graph. By contrast, this issue is much less significant for larger objects, where minor discrepancies along the contours have a negligible effect on the overall segmentation mask. Our method leverages the idea of segmentation using conditioned DPMs; however, we avoid the use of an ensemble and further condition the starting latent variable of the sampling process (see Section 3.2).

3 Method

Our method is divided into three stages: lane segmentation, lane segmentation refinement and lane graph extraction (see Figure 1). For the first stage, we train a modified version of a D-LinkNet [40]. We take the architecture of the segmentation network and training procedure from LaneExtraction [14] with slight changes. At the second stage, which is our extension, we train a diffusion model - similar to Improved DDPM [23] - conditioned on the aerial RGB patch. Its objective is to denoise the lane segmentation mask. After training the diffusion model, we implement DDIM [31] sampling, initiating the process by conditioning the starting latent variable with the unrefined segmentation mask obtained from the first stage. In the third stage, we employ the same extraction algorithm outlined in LaneExtraction [14], which generates the lane graph from the refined segmentation mask produced in the second stage. It is worth noticing that only stages one and two have training and inference phases whereas stage three does not require any training. Another noteworthy aspect is that the training phases for stages one and two are independent of each other. This independence allows for the selection of distinct architectures and training procedures for the segmentation and diffusion models as well as parallel training processes. However, at inference time, the sequential order of the stages mentioned above should be preserved (see Figure 1). In the following subsections we explain each of the stages in more detail.

3.1 First Stage: Lane Segmentation

We use the same architecture as LaneExtraction [14], which uses a D-LinkNet network [40] with two heads, one head outputs the segmentation mask while

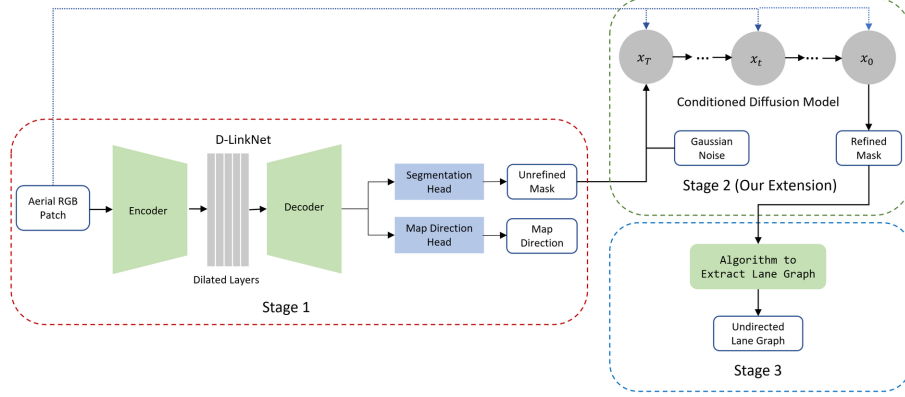
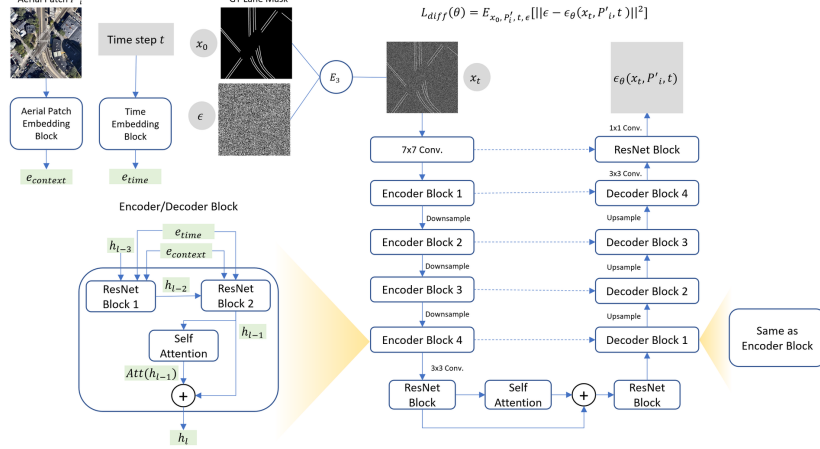


Fig. 1: Overall pipeline of our method. At inference time, first, the aerial RGB patch is fed into the segmentation model, which outputs an unrefined segmentation mask and a (unrefined) direction map. Then, Gaussian noise is added to the unrefined segmentation mask to form the starting latent variable x_T of a DDIM [31] sampling, which after several sampling steps is able to get a refined segmentation mask x_0 . In Stage 2, the blue dotted arrows specify conditioning whereas the solid black arrow indicates the initial input. Finally, the refined segmentation mask is used by the lane graph extraction algorithm to produce the final lane graph.

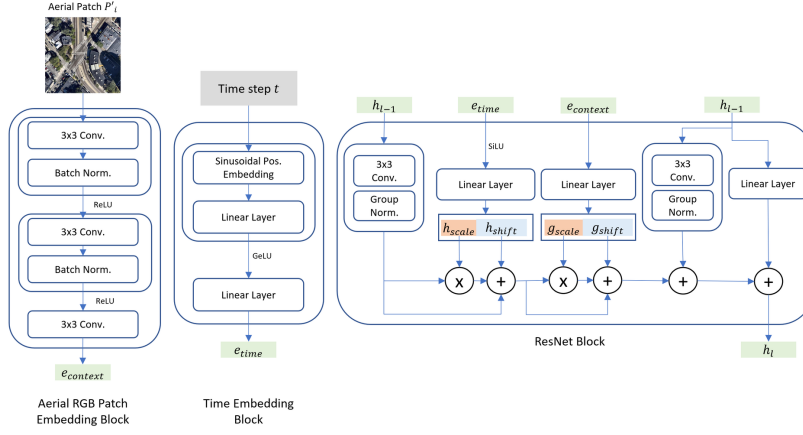
the other predicts the lane direction map. We also follow a similar preprocessing procedure, which consists in taking random patches of fixed size $N_s \times N_s$ ($N_s = 1024$) from the training set (see Section of dataset 4.1) and apply some augmentation techniques, including random rotations, as well as random adjustments to color and brightness. Let $\mathcal{T} = \{(P_i, s_i, d_i)\}_{i=1}^n$ be the patch training set of n triplets induced by the preprocessing procedure, where P_i is an aerial training patch, s_i is its ground truth (GT) segmentation mask and d_i is its ground truth direction map. The segmentation loss \mathcal{L}_{seg} is a combination of the cross-entropy loss and the dice loss for the segmentation head, and the \mathcal{L}_2 loss for the direction map head, formally defined as follows:

$$\mathcal{L}_{seg} = \mathcal{L}_2(d_i, \hat{d}_i) + \frac{1}{2}(\mathcal{L}_{ce}(s_i, \hat{s}_i) + \mathcal{L}_{dice}(s_i, \hat{s}_i)), \quad (1)$$

where \hat{s}_i and \hat{d}_i are the predictions for the segmentation and direction map, respectively, output by the segmentation network given the aerial patch P_i . The inference is carried out by using a sliding window (same size as training patches) on the full test tiles with the same horizontal and vertical stride (see Figure 4). Then, for each window we make a prediction with the trained segmentation network, and once the whole tile is covered, we take an average of the overlapping areas (see dashed squares in different colors in Figure 4) to get the final output.



(a) Architecture of the U-Net used to train the conditioned DDPM [16].



(b) Building blocks of the architecture from Figure 2a.

Fig. 2: Architecture employed to train our diffusion model, which is based on the framework utilized in Improved DDPM [23]. A level of noise is added to the segmentation mask via Equation 3 (forward process). After that, the U-Net is trained to learn the backward process with the loss from Equation 2. The aerial patch and time-step processed through embedding blocks. Subsequently, these embeddings are injected into all but the final ResNet [13] block, where only time embeddings are used.

3.2 Second Stage: Lane Segmentation Refinement

For this stage, which is the main contribution of the paper, we utilize a conditioned diffusion model by using a similar architecture as Improved DDPM [31]. First, we apply a similar preprocessing as the previous stage to the training set, which includes cropping, random rotations, random adjustments to color and brightness, and rescaling. Let $\mathcal{R} = \{(s'_i, P'_i)\}_{i=1}^n$ be the set induced by the preprocessing, where s'_i is a ground truth (GT) segmentation mask and P'_i is

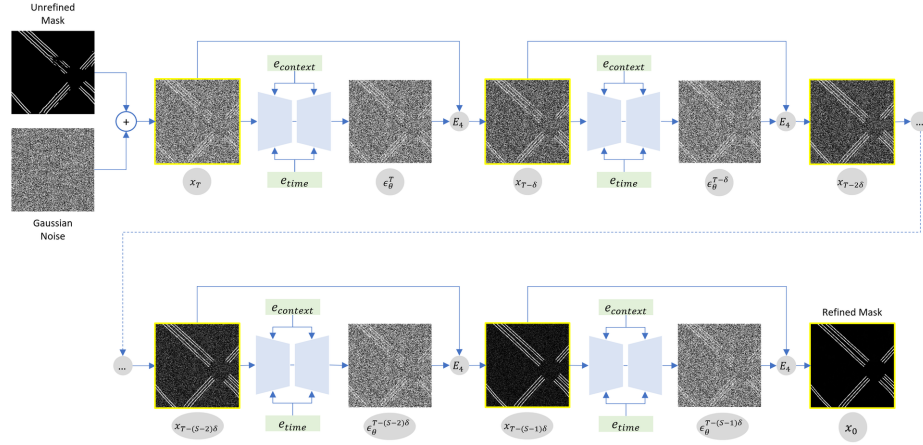


Fig. 3: DDIM [31] sampling process. Gaussian noise is added to the unrefined segmentation mask (from stage 1) to create the initial latent variable x_T . Then, several DDIM sampling steps, linked through Equation 4, are executed to obtain the refined segmentation mask x_0 .

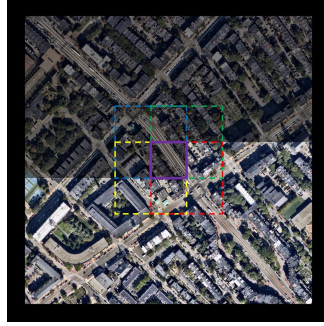


Fig. 4: Sliding windows during inference time. The shaded area represents the portion already predicted. The dotted squares, in different colors, depict the sliding windows. The small region framed in purple denotes the overlap of the four dotted sliding windows, where the final output is derived by averaging the outputs from these sliding windows. This inference process is employed in both stage 1 and stage 2. However, stage 2 also incorporates the unrefined segmentation mask as an additional input.

an aerial patch with dimensions $M_d \times M_d$, with $M_d = 256$. The rescaling is done to fit the patches into GPU memory. Then, we encode the rescaled aerial patches P'_i with a simple three-layer CNN and time-steps with an embedding block composed of sinusoidal positional embedding [32] and two linear layers (see embedding blocks in Figure 2b). After that, we utilize the rescaled GT segmentation masks s'_i to perform the forward process of a denoising probabilistic diffusion model (DDPM) [16] that generates noisy segmentation masks according to the sigmoid variance schedule [19]. Subsequently, we train a U-Net [28]

to learn the backward diffusion process using the noise parametrization utilized in DDPM [16]. This procedure involves injecting the aerial RGB patch and time-step embeddings into all but the final ResNet [13] block, where only time embeddings are injected. The architecture is based on the one used in Improved DDPM [23] (refer to Figure 2). The U-Net is trained with the following loss:

$$\mathcal{L}_{diff}(\theta) = \mathbb{E}_{x_0, P'_i, t, \epsilon} [\|\epsilon - \epsilon_\theta(x_t, P'_i, t)\|^2], \quad (2)$$

where $x_0 \sim q(x_0)$, $q(x_0)$ is the GT segmentation masks data distribution and P'_i is the corresponding rescaled aerial patch of the sample x_0 , with $x_0, P'_i \in \mathbb{R}^{M_d \times M_d}$, $t \sim U([1, T])$ is the time-step sampled from the uniform distribution U , T is the length of the forward diffusion process, $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and x_t is computed from x_0 via the following equation as performed in DDPM [16]:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad (3)$$

where $\bar{\alpha}_t := \prod_{s=1}^t \alpha_s$, with $\alpha_t := 1 - \beta_t$ and β_t 's are taken from the sigmoid variance schedule [19].

For inference, we follow DDIM [31] sampling, where samples are produced through a fixed process from the beginning x_T to the end x_0 , i.e., no noise is added to intermediate steps. DDIM also makes the sampling faster since much fewer sampling steps are needed to get good results. DDIM also guarantees flexibility to train a model using an arbitrary number of forward steps and sample from only a subset of these during the generative phase. We choose the subset $\{x_{\tau_1}, x_{\tau_2}, \dots, x_{\tau_S}\}$, where τ is an increasing sub-sequence of $[1, 2, \dots, T]$ of length S , and such that the difference between x_{τ_j} and $x_{\tau_{j-1}}$ is constant, so we end up with the sampling trajectory $\{x_T, x_{T-\delta}, x_{T-2\delta} \dots, x_{T-(S-1)\delta}, x_0\}$ (which is the reverse of τ), with $S\delta = T$, where S is the number of sampling steps. Then, $x_{t-\delta}$ and x_t are connected by the following equation, which is a special case of Equation 12 from DDIM [31] with $\sigma_t = 0$ (no noise added to intermediate steps) and our sub-sequence τ :

$$x_{t-\delta} = \sqrt{\bar{\alpha}_{t-\delta}} \left(\frac{x_t - \sqrt{1 - \bar{\alpha}_t} \epsilon_\theta^t}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1 - \bar{\alpha}_{t-\delta}} \epsilon_\theta^t, \quad (4)$$

where ϵ_θ^t is the output of the DDIM sampling process at time-step t . Contrary to typical diffusion models that start DDIM sampling with Gaussian noise, our approach conditions the starting latent variable x_T on the segmentation network's output, referred to as the unrefined segmentation mask (as depicted in Figure 3). This conditioned starting point allows the DDIM sampling to refine the segmentation mask since starting from pure noise leads to poor results (see last row of Table 4).

It should be noted that we explore this conditioning with various strategies, including: directly inputting the unrefined segmentation mask, adding Gaussian noise to it, and applying noise through several forward steps based on the sigmoid variance schedule [19] (utilized during training). We show experimentally that adding noise to the unrefined segmentation mask, directly or through the forward

process, enhances the metrics and stability of the model compared to directly using the unrefined mask in DDIM sampling (see Table 3 and Table 4). We present our analysis in the results section (see Section 4).

For covering the entire testing tiles, we adopt the same sliding window approach used in the segmentation stage, utilizing a fixed patch size (1024×1024) with identical vertical and horizontal strides to refine each sliding window. However, in this stage, the diffusion model receives two inputs: the aerial patch and its unrefined segmentation mask, both resized to 256×256 to fit the input size of the model. The refined mask output is then resized back to the original dimension of the input patches 1024×1024 . After the whole tile is covered, we take an average of the overlapping regions to get the final refined segmentation mask (see Figure 4).

3.3 Third Stage: Lane Extraction Algorithm

For the third stage, the lane graph extraction is carried out in the same way as in LaneExtraction [14]. To be more specific, a threshold $\alpha = 0.5$ is applied to the refined segmentation mask to get a binarized mask, followed by a morphological thinning algorithm [12] which produces an skeleton that is then turned into a graph. Subsequently, a post-processing algorithm prunes the graph from very small connected components and spurs³. Finally, the graph is simplified by applying the Douglas–Peucker algorithm [26].

4 Experiments and Results

4.1 Dataset

We use the dataset introduced by LaneExtraction [14]. The dataset consists of 35 aerial tiles of size 4096×4096 covering the area of four cities in the US: Miami, Boston, Seattle and Phoenix. In order to make a fair comparison, we also separate the dataset into 24 tiles for training and 11 tiles for testing like they do. For further details of the dataset we refer to Appendix A.

4.2 Experiments and Results

In this section, we present the experiments conducted on the publicly available dataset detailed in Section 4.1, along with the ablation studies performed. The results highlight the performance of our method in extracting the undirected lane graph in non-intersection areas. We use the GEO and TOPO metrics as defined in the previous work LaneExtraction [14]. Detailed explanations of the metrics are provided in Appendix B. We set $S = 25$ DDIM sampling steps as our default value. All experiments involving diffusion model components were performed 10

³ Spurs are edges that extend from a node but do not lead to another node that connects back to the main part of the graph.

times. We show the mean and standard deviation of these repetitions in Tables 2, 3 and 4. For experimental details we refer to Appendix C.

Table 1 shows the overall results of our experiments. Our method performs better than LaneExtraction [14] in the F_1 -scores in both the GEO and TOPO metrics, which are the main metrics considered to measure the quality of a lane graph. The best improvement happens in the TOPO F_1 -score, which is the metric that takes into consideration the connectivity of a lane graph, playing an important role in downstream tasks such as autonomous driving. Intuitively we can think of the diffusion model as filling the gaps left by the segmentation model. This is reflected by the significant increments in the recall metrics, achieved while maintaining virtually the same precision levels as prior to the application of the diffusion model, which improves the overall F_1 -scores.

Table 2 presents an ablation study of the key components of our method: the conditioned DDPM [16], where its U-Net architecture is conditioned with the aerial patches, and the conditioned DDIM [31], where the initial latent variable is conditioned on the unrefined segmentation mask produced by the segmentation network.

Table 3 presents an ablation study on the effect of introducing Gaussian noise to the unrefined segmentation mask with different number of sampling steps, which marginally enhances the GEO F_1 -score relative to initiating DDIM sampling directly from the unrefined mask. Conversely, the TOPO F_1 -score does not consistently improve with the addition of Gaussian noise, though the discrepancies in those instances are minimal. Although the improvements in F_1 -scores seems minor, adding Gaussian noise reduces the gap between precision and recall compared to not adding it, indicating a more stable model. This suggests the model balances avoiding false positives and negatives more effectively. We hypothesize that adding Gaussian noise to the unrefined segmentation mask makes x_T more closely resemble a sample from a Gaussian distribution, a requirement for x_T in DDIM sampling, compared to when such a noise is not added.

In Table 4, we present an ablation study on the impact of introducing noise through varying numbers of forward steps (FS) with the sigmoid variance schedule [19] used during training, with the number of sampling steps fixed at $S = 25$. The best results are achieved with 50% of forward steps applied to the unrefined mask, with small variance observed up to 75%. Notably, the model performance declines sharply with 90% of forward steps, and at 100% (pure Gaussian noise), the model fails to obtain any meaningful results. This suggests the limitation of the model in reconstructing the segmentation mask purely from Gaussian noise (see underlined results in Table 4).

4.3 Qualitative Results

Figure 5 shows two visualizations of two different patches taken from two different testing tiles. They compare the results of the unrefined segmentation mask output by the segmentation network of LaneExtraction [14], and the refined segmentation mask output by our DDIM sampling (see middle column of Figure 5). In the visualizations depicting lane graph results (third column of Figure 5),

Table 1: Metrics of the extracted undirected lane graph in non-intersection areas with the baseline method LaneExtraction [14] (LE) and our method LSR-DM. We present the metrics from the LaneExtraction paper alongside our reproduced results. Results highlighted in bold represent the best results.

Method	Geo metrics			TOPO Metrics		
	F1 score	Prec.	Rec.	F1 score	Prec.	Rec.
LE [14]	0.828	0.835	0.821	0.748	0.774	0.724
LE (reproduced)	0.813	0.836	0.790	0.713	0.758	0.673
LSR-DM (ours)	0.841	0.833	0.849	0.774	0.759	0.789

Table 2: Ablation study on the components of our method: Cond. DDPM (conditioned DDPM [16]) uses aerial patches, while Cond. DDIM (conditioned DDIM [31]) conditions the initial latent variable on the unrefined segmentation mask. The format of the results indicates the mean \pm std. across 10 repetitions of each experiment. Results highlighted in bold represent the best results.

Cond. DDPM	Cond. DDIM	Geo metrics			TOPO Metrics		
		F1 score	Prec.	Rec.	F1 score	Prec.	Rec.
\times	\checkmark	$0.784 \pm 3e^{-3}$	$0.726 \pm 4e^{-3}$	$0.852 \pm 2e^{-3}$	$0.703 \pm 4e^{-3}$	$0.633 \pm 5e^{-3}$	$0.790 \pm 4e^{-3}$
\checkmark	\times	$0.666 \pm 4e^{-3}$	$0.819 \pm 4e^{-3}$	$0.561 \pm 3e^{-3}$	$0.562 \pm 5e^{-3}$	$0.752 \pm 6e^{-3}$	$0.449 \pm 5e^{-3}$
\checkmark	\checkmark	$0.841 \pm 2e^{-3}$	$0.833 \pm 2e^{-3}$	$0.849 \pm 2e^{-3}$	$0.774 \pm 2e^{-3}$	$0.759 \pm 2e^{-3}$	$0.789 \pm 3e^{-3}$

Table 3: Ablation study on adding Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ to the unrefined segmentation mask with different numbers of sampling steps S . The format of the results indicates the mean \pm std. across 10 repetitions of each experiment. Results without added Gaussian noise lack standard deviation indicators, as the input remains consistent. Results highlighted in bold represent the best results.

S	Noise	Geo metrics			TOPO Metrics		
		F1 score	Prec.	Rec.	F1 score	Prec.	Rec.
10	\checkmark	$0.841 \pm 2e^{-3}$	$0.835 \pm 2e^{-3}$	$0.847 \pm 2e^{-3}$	$0.774 \pm 3e^{-3}$	$0.762 \pm 3e^{-3}$	$0.786 \pm 3e^{-3}$
	\times	0.838	0.827	0.848	0.772	0.755	0.789
25	\checkmark	$0.841 \pm 2e^{-3}$	$0.833 \pm 2e^{-3}$	$0.849 \pm 2e^{-3}$	$0.774 \pm 2e^{-3}$	$0.759 \pm 2e^{-3}$	$0.789 \pm 3e^{-3}$
	\times	0.837	0.825	0.85	0.772	0.753	0.792
50	\checkmark	$0.84 \pm 2e^{-3}$	$0.831 \pm 2e^{-3}$	$0.849 \pm 2e^{-3}$	$0.773 \pm 3e^{-3}$	$0.756 \pm 3e^{-3}$	$0.79 \pm 2e^{-3}$
	\times	0.838	0.825	0.852	0.775	0.756	0.795
100	\checkmark	$0.84 \pm 3e^{-3}$	$0.831 \pm 4e^{-3}$	$0.849 \pm 3e^{-3}$	$0.773 \pm 4e^{-3}$	$0.756 \pm 4e^{-3}$	$0.79 \pm 3e^{-3}$
	\times	0.838	0.825	0.852	0.772	0.752	0.793
250	\checkmark	$0.839 \pm 1e^{-3}$	$0.829 \pm 2e^{-3}$	$0.849 \pm 1e^{-3}$	$0.772 \pm 2e^{-3}$	$0.755 \pm 2e^{-3}$	$0.79 \pm 1e^{-3}$
	\times	0.838	0.825	0.852	0.773	0.752	0.795
500	\checkmark	$0.841 \pm 1e^{-3}$	$0.831 \pm 2e^{-3}$	$0.851 \pm 1e^{-3}$	$0.773 \pm 2e^{-3}$	$0.756 \pm 2e^{-3}$	$0.79 \pm 2e^{-3}$
	\times	0.838	0.824	0.852	0.773	0.752	0.795
1000	\checkmark	$0.838 \pm 1e^{-3}$	$0.828 \pm 1e^{-3}$	$0.849 \pm 1e^{-3}$	$0.771 \pm 1e^{-3}$	$0.753 \pm 1e^{-3}$	$0.789 \pm 2e^{-3}$
	\times	0.838	0.824	0.852	0.771	0.751	0.792

Table 4: Ablation study on different noise levels added according to the forward process with the sigmoid schedule [19] (from training) to the unrefined mask, with sampling steps fixed at $S = 25$. FS means forward steps. The format of the results indicates the mean \pm std. across 10 repetitions of each experiment. Results highlighted in bold represent the best results. Results underlined denote the worst, occurring when the starting latent variable consists purely of Gaussian noise.

Noise Level	Geo metrics			TOPO Metrics		
	F1 score	Prec.	Rec.	F1 score	Prec.	Rec.
0% (0 FS)	0.837 \pm 0e $^{-3}$	0.825 \pm 0e $^{-3}$	0.85 \pm 0e $^{-3}$	0.772 \pm 0e $^{-3}$	0.753 \pm 0e $^{-3}$	0.792 \pm 0e $^{-3}$
10% (100 FS)	0.838 \pm 1e $^{-3}$	0.826 \pm 1e $^{-3}$	0.851 \pm 2e $^{-3}$	0.773 \pm 2e $^{-3}$	0.754 \pm 2e $^{-3}$	0.793 \pm 2e $^{-3}$
25% (250 FS)	0.837 \pm 1e $^{-3}$	0.825 \pm 1e $^{-3}$	0.85 \pm 1e $^{-3}$	0.772 \pm 2e $^{-3}$	0.753 \pm 2e $^{-3}$	0.792 \pm 2e $^{-3}$
50% (500 FS)	0.839\pm3e$^{-3}$	0.832 \pm 3e $^{-3}$	0.846 \pm 3e $^{-3}$	0.775\pm3e$^{-3}$	0.762 \pm 3e $^{-3}$	0.787 \pm 0e $^{-3}$
75% (750 FS)	0.826 \pm 2e $^{-3}$	0.842 \pm 2e $^{-3}$	0.811 \pm 3e $^{-3}$	0.757 \pm 2e $^{-3}$	0.774 \pm 3e $^{-3}$	0.741 \pm 3e $^{-3}$
90% (900 FS)	0.781 \pm 4e $^{-3}$	0.839 \pm 5e $^{-3}$	0.73 \pm 4e $^{-3}$	0.697 \pm 5e $^{-3}$	0.774 \pm 5e $^{-3}$	0.634 \pm 5e $^{-3}$
100% (1000 FS)	<u>0.666\pm4e$^{-3}$</u>	0.819 \pm 4e $^{-3}$	0.561 \pm 3e $^{-3}$	<u>0.562\pm5e$^{-3}$</u>	0.752 \pm 6e $^{-3}$	0.449 \pm 5e $^{-3}$

we note a significant decrease in false positives and false negatives, particularly excelling at removing small segments of incorrect nodes. The output from the diffusion model can be viewed as a denoised version of the lane graph produced by the segmentation network. Some limitations of the diffusion model are also evident in the images, such as the incapacity of replacing relatively large segments of false negatives (see the large red segment at the bottom-left of the image depicting the lane graph produced by our model in Figure 5a) or generating segments of nodes that are sufficiently shifted such that they do not match anymore (see the overlapping blue and red segments located at the center-right of the same image in Figure 5a).

5 Conclusions

In this paper, we have presented a novel yet simple method that merges a segmentation network with a diffusion model to extract the undirected lane graph in non-intersection areas from aerial imagery. By utilizing a conditioned DDPM and conditioned DDIM sampling, we refined the segmentation masks produced by the segmentation network. Our experiments on a publicly accessible dataset revealed that these refined masks enhanced the quality of the resulting lane graph compared to the previous approach, with a marked improvement in graph connectivity as demonstrated by the boost in the TOPO F_1 -score. Moreover, we investigated the effects of applying varying levels of noise to the initial unrefined segmentation mask during the DDIM sampling process and its impact on the overall performance of the method.

Although our method concentrates on the undirected lane graph in non-intersection areas, we consider the integration of diffusion models into the directed lane graph, including in non-intersection areas, as promising areas for future research. For example, it has a potential to be connected with stroke-based rendering approaches [24] or value function guided segmentation [22].

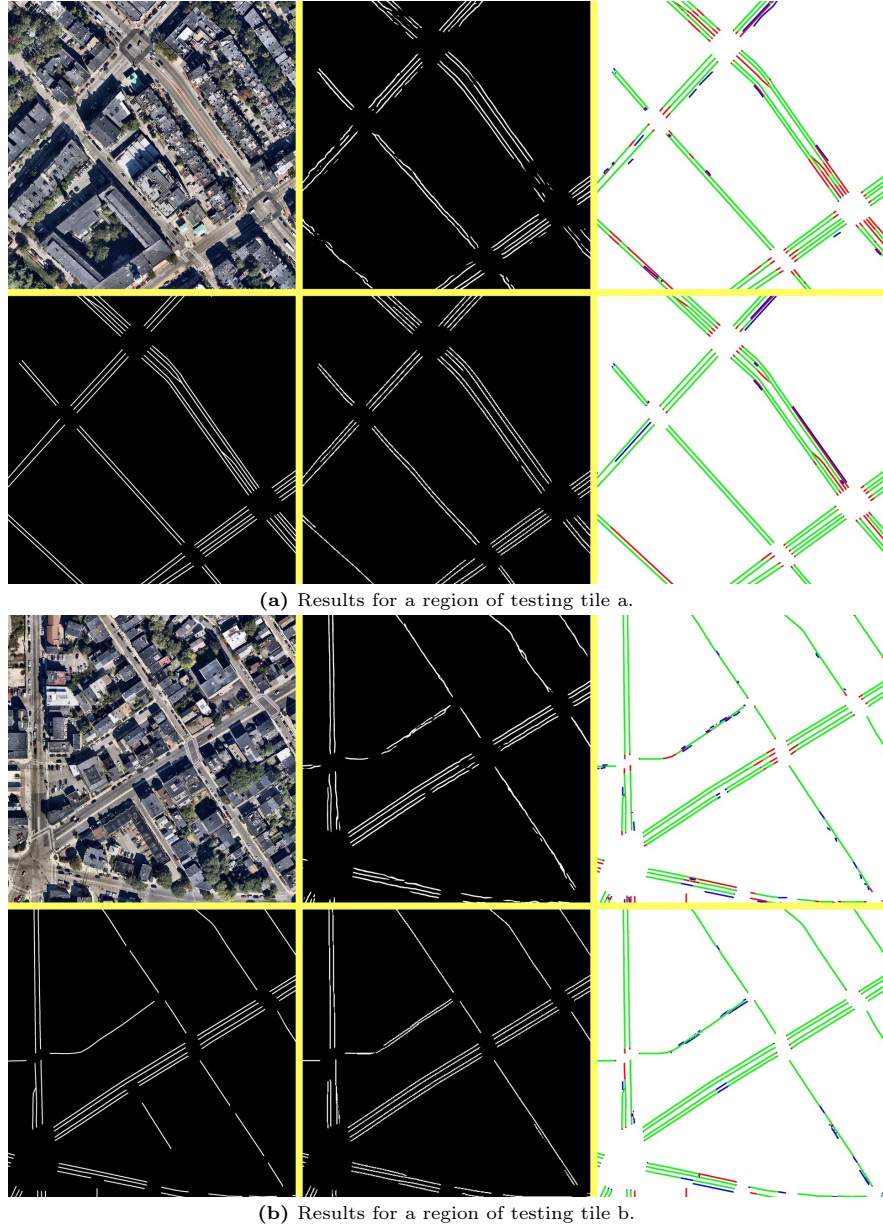


Fig. 5: Comparison of the segmentation model LaneExtraction [14] (first row of 5a and 5b) vs our model (second row of 5a and 5b), given two different RGB aerial tiles. The first row depicts the results of LaneExtraction and the second row the results of our model. The first column contains the aerial RGB (first row) and GT segmentation mask (second row). Second and third columns display the predicted lane segmentation masks and the lane graphs used to evaluate the graph metrics of the respective methods. In the lane graph, green nodes indicate matched nodes, blue nodes represent false positives, and red nodes denote false negatives. Nodes look like line segments because they are too close each other. For this visualization, both methods used patch sizes of 1024×1024 .

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), <https://www.tensorflow.org/>, software available from tensorflow.org
2. Amit, T., Shaharbany, T., Nachmani, E., Wolf, L.: Segdiff: Image segmentation with diffusion probabilistic models. arXiv preprint arXiv:2112.00390 (2021)
3. Biagioni, J., Eriksson, J.: Inferring road maps from global positioning system traces: Survey and comparative evaluation. *Transportation research record* **2291**(1), 61–71 (2012)
4. Brasó, G., Leal-Taixé, L.: Learning a neural solver for multiple object tracking. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 6247–6257 (2020)
5. Büchner, M., Zürn, J., Todoran, I.G., Valada, A., Burgard, W.: Learning and aggregating lane graphs for urban automated driving. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 13415–13424 (2023)
6. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. pp. 11621–11631 (2020)
7. Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Structured bird’s-eye-view traffic scene understanding from onboard images. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 15661–15670 (2021)
8. Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Topology preserving local road network estimation from single onboard camera image. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 17263–17272 (2022)
9. Chang, Z., Koulteris, G.A., Shum, H.P.: On the design fundamentals of diffusion models: A survey. arXiv preprint arXiv:2306.04542 (2023)
10. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. *Advances in neural information processing systems* **34**, 8780–8794 (2021)
11. Eichenberger, C., Neun, M., Martin, H., Herruzo, P., Spanring, M., Lu, Y., Choi, S., Konyakhin, V., Lukashina, N., Shpilman, A., et al.: Traffic4cast at neurips 2021-temporal and spatial few-shot transfer learning in gridded geo-spatial processes. In: *NeurIPS 2021 Competitions and Demonstrations Track*. pp. 97–112. PMLR (2022)
12. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. *Communications of the ACM* **32**(3), 359–373 (1989)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 770–778 (2016)
14. He, S., Balakrishnan, H.: Lane-level street map extraction from aerial imagery. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. pp. 2080–2089 (2022)

15. Hermes, L., Hammer, B., Melnik, A., Velioglu, R., Vieth, M., Schilling, M.: A graph-based u-net model for predicting traffic in unseen cities. In: 2022 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2022)
16. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. *Advances in neural information processing systems* **33**, 6840–6851 (2020)
17. Homayounfar, N., Ma, W.C., Lakshmikanth, S.K., Urtasun, R.: Hierarchical recurrent attention networks for structured online maps. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3417–3426 (2018)
18. Homayounfar, N., Ma, W.C., Liang, J., Wu, X., Fan, J., Urtasun, R.: Dagmapper: Learning to map by discovering lane topology. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2911–2920 (2019)
19. Jabri, A., Fleet, D., Chen, T.: Scalable adaptive computation for iterative generation. *arXiv preprint arXiv:2212.11972* (2022)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014)
21. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017)
22. Melnik, A., Harter, A., Limberg, C., Rana, K., Sünderhauf, N., Ritter, H.: Critic guided segmentation of rewarding objects in first-person views. In: KI 2021: Advances in Artificial Intelligence: 44th German Conference on AI, Virtual Event, September 27–October 1, 2021, Proceedings 44. pp. 338–348. Springer (2021)
23. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: International Conference on Machine Learning. pp. 8162–8171. PMLR (2021)
24. Nolte, F., Melnik, A., Ritter, H.: Stroke-based rendering: From heuristics to deep learning. *arXiv preprint arXiv:2302.00595* (2022)
25. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703* (2019)
26. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. *Computer graphics and image processing* **1**(3), 244–256 (1972)
27. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., Ommer, B.: High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10684–10695 (2022)
28. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5–9, 2015, Proceedings, Part III 18. pp. 234–241. Springer (2015)
29. Salimans, T., Ho, J.: Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512* (2022)
30. Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., Ganguli, S.: Deep unsupervised learning using nonequilibrium thermodynamics. In: International conference on machine learning. pp. 2256–2265. PMLR (2015)
31. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2020)
32. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* **30** (2017)

33. Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., et al.: Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence* **43**(10), 3349–3364 (2020)
34. Wolleb, J., Sandkühler, R., Bieder, F., Valmaggia, P., Cattin, P.C.: Diffusion models for implicit image segmentation ensembles. In: *International Conference on Medical Imaging with Deep Learning*. pp. 1336–1348. PMLR (2022)
35. Wu, J., Fang, H., Zhang, Y., Yang, Y., Xu, Y.: Medsegdiff: Medical image segmentation with diffusion probabilistic model. *arXiv preprint arXiv:2211.00611* (2022)
36. Wu, J., Fu, R., Fang, H., Zhang, Y., Xu, Y.: Medsegdiff-v2: Diffusion based medical image segmentation with transformer. *arXiv preprint arXiv:2301.11798* (2023)
37. Yang, J., Lu, J., Lee, S., Batra, D., Parikh, D.: Graph r-cnn for scene graph generation. In: *Proceedings of the European conference on computer vision (ECCV)*. pp. 670–685 (2018)
38. Yao, J., Pan, X., Wu, T., Zhang, X.: Building lane-level maps from aerial images. *arXiv preprint arXiv:2312.13449* (2023)
39. Zhang, L., Tafazzoli, F., Krehl, G., Xu, R., Rehfeld, T., Schier, M., Seal, A.: Hierarchical road topology learning for urban mapless driving. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 3816–3823. IEEE (2022)
40. Zhou, L., Zhang, C., Wu, M.: 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*. pp. 182–186 (2018)
41. Zhou, Y., Takeda, Y., Tomizuka, M., Zhan, W.: Automatic construction of lane-level hd maps for urban scenes. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. pp. 6649–6656. IEEE (2021)
42. Zürn, J., Vertens, J., Burgard, W.: Lane graph estimation for scene understanding in urban driving. *IEEE Robotics and Automation Letters* **6**(4), 8615–8622 (2021)

A Dataset Details

The dataset covers 400 km of lanes in four US cities, namely: Miami, Boston, Seattle and Phoenix. The ground sample distance (GSD) of the aerial images is 12.5 cm per pixel.

For each tile, the ground truth lane graph is known. The ground truth segmentation masks are produced by rendering 5 pixel wide white lines on black images following the directed edges of the ground truth graph. The ground truth direction maps are created in a similar fashion, but colored lines in BGR (Blue-Green-Red) format are used to encode the direction of the edges, the first and second components of the color encode the x and y normalized directions respectively, whereas the third component is kept constant, pixels without lanes are set to the zero vector. Then, a training tile triple is formed, consisting of the aerial RGB tile, the ground truth segmentation mask, and the ground truth direction map. After the training tile triples are formed, we create the training set that will be used for the segmentation network (stage 1) and the diffusion model (stage 2) by using a sliding window of size 2048×2048 and vertical and horizontal stride of 1024 on each training tile; this resulted in 216 training samples. As mentioned in 3, the testing tiles are not cropped for inference, but used in full size. This methodology is also employed in LaneExtraction [14].

B Metrics

We employ the GEO and TOPO metrics to evaluate our method, these metrics are adapted from the problem of road extraction [3]. To ensure a fair comparison, we adopted the methodology and hyperparameters (used by the metrics) described in LaneExtraction [14] across all experiments.

B.1 GEO Metrics

This methodology consists in interpolating the predicted lane graph and the ground truth lane graph such that the distance between any two connected nodes is 25 cm in real-world distance. The interpolation induces a predicted lane graph $\hat{G} = \{\hat{V}, \hat{E}\}$ and a ground truth lane graph $G = \{V, E\}$, where \hat{V} and V are sets of vertices and \hat{E} and E are sets of edges of the respective graphs, (\hat{v}, v) , with $\hat{v} \in \hat{V}$ and $v \in V$, is considered a valid match if $\|\hat{v} - v\|_2 < r$ (as in Section 3, presence or absence of a caret ^ on a symbol indicates prediction or ground truth values, resp.), where r determines the tolerance of the metric. We set $r = 1m$, in real-world distance. After that, a maximal one-to-one match between \hat{V} and V is computed and the matched vertices in the predicted lane graph are denoted as \hat{V}_{match} . Then, three GEO metrics: precision, recall and F1 score, are defined as follows:

$$pre_{GEO} = \frac{|\hat{V}_{match}|}{|\hat{V}|}, \quad rec_{GEO} = \frac{|\hat{V}_{match}|}{|V|}, \quad F1_{GEO} = 2 \cdot \frac{pre_{GEO} \cdot rec_{GEO}}{pre_{GEO} + rec_{GEO}}. \quad (5)$$

B.2 TOPO Metrics

GEO metrics do not take into account connectivity, so TOPO metrics try to correct that shortcoming. TOPO metrics are built on top of GEO metrics. For each matched vertex pair (\hat{v}, v) in the GEO metrics, the subgraphs $S_{\hat{v}}$ and S_v are defined in the following way: $S_{\hat{v}} = \{\hat{u} \in \hat{G} : d(\hat{v}, \hat{u}) < 50m\}$ and $S_v = \{u \in G : d(v, u) < 50m\}$, where $d(a, b)$ is the length of the shortest path from node a to node b , if there is any, otherwise it is ∞ . Then, the GEO metrics between the two subgraphs $S_{\hat{v}}$ and S_v – denoted as $pre_{GEO}(S_{\hat{v}}, S_v)$ and $rec_{GEO}(S_{\hat{v}}, S_v)$ – are computed for each vertex pair (\hat{v}, v) , and the final TOPO metrics are defined as follows:

$$pre_{TOPO} = \frac{\sum_{(\hat{v}, v)} pre_{GEO}(S_{\hat{v}}, S_v)}{|\hat{V}|}, \quad (6)$$

$$rec_{TOPO} = \frac{\sum_{(\hat{v}, v)} rec_{GEO}(S_{\hat{v}}, S_v)}{|V|}, \quad (7)$$

$$F_{1TOPO} = 2 \cdot \frac{pre_{TOPO} \cdot rec_{TOPO}}{pre_{TOPO} + rec_{TOPO}}. \quad (8)$$

C Experimental Details

The architecture for the segmentation network in Stage 1 was implemented in TensorFlow 2 [1]. We trained the model for 500 epochs, starting with a learning rate of $1e^{-3}$ and reducing it by a factor of 10 at epochs 350 and 450, following the schedule used in LaneExtraction [14], and employed the AdamW optimizer [21]. The architecture of the segmentation network was adopted from prior work (see Figure 3 in LaneExtraction [14]). Data augmentation techniques, including random cropping, rotations, and random modifications to color and brightness, were employed. For experimentation, we used patch sizes of 640×640 to reproduce previous results and 1024×1024 for the pipeline of our method. The batch size was set to 8.

The architecture for the diffusion model in Stage 2 was implemented in Pytorch 2 [25]. We used the sigmoid variance schedule [19] with hyperparameters set to $start = -3$, $end = 3$, and $\tau = 1$. The length of the forward process was set to $T = 1000$. The model was trained for $200K$ steps, utilizing a learning rate of $8e^{-5}$ with the Adam optimizer [20] and applying an exponential moving average (EMA) with decay factor of 0.995. The dimensions for both time and aerial patch embeddings were set to 256. Segmentation masks utilized one channel, while aerial RGB patches employed three channels. The batch size was set to 8. Data augmentation strategies included random cropping, horizontal and vertical flipping, integer rotations within $[-15, 15]$ degrees, and color jitter for the aerial RGB patches, which were sized at 1024×1024 . As outlined in Section 3.2, the patches were resized to 256×256 to fit into the GPU memory. Our implementation for the diffusion model is based on the following git repository:

Table 5: Average number of pixels changed from the unrefined to the refined segmentation masks on the test dataset. S represents the number of DDIM sampling steps. The third column shows the absolute difference between the first and second columns. The format of the results indicates the mean \pm std. across 11 testing tiles in one repetition of each experiment. The testing tiles contain 4096×4096 pixels.

S	Noise	Mean of # of changed pixels $\times 1K$		Abs. Diff. ($\times 1K$)
		White \rightarrow Black	Black \rightarrow White	
10	✓	128.106 ± 14.974	103.657 ± 25.401	28.322 ± 13.898
	✗	121.853 ± 13.809	102.733 ± 27.849	27.377 ± 12.36
50	✓	126.879 ± 14.657	106.552 ± 27.17	26.223 ± 13.973
	✗	121.543 ± 13.628	105.143 ± 29.365	27.18 ± 12.526
100	✓	125.656 ± 13.925	105.945 ± 26.776	25.594 ± 12.923
	✗	121.481 ± 13.483	105.466 ± 29.429	27.037 ± 12.245
500	✓	126.712 ± 13.816	106.884 ± 28.135	26.648 ± 12.113
	✗	121.491 ± 13.447	105.796 ± 29.63	27.199 ± 11.909

<https://github.com/lucidrains/denoising-diffusion-pytorch>. This implementation uses the v-parametrization (as explained in Appendix D in [29]) to train the diffusion model, we left it as it is.

The training and inference of both models were carried out on a single NVIDIA GeForce RTX 3090 GPU, using CUDA Version 12.2.

D Pixel Statistics

In Table 5, we detail the number of pixels changed from white to black and vice versa, varying the number of sampling steps and comparing scenarios with and without noise. This table demonstrates the capability of the diffusion model to perform changes in both directions, highlighting its utility not just in filling gaps between lane segments but also in rectifying misplaced white pixels in the unrefined segmentation mask. The third column, which indicates absolute difference between the changes from white to black pixels and vice versa, reveals that the model maintains a relative balance in both scenarios, where the differences represent about $\frac{1}{5}$ of the total in the first column and approximately $\frac{1}{4}$ in the second column. Additionally, Table 6 displays the percentages of relative pixel changes from white to black and vice versa, using the same settings as Table 5. This table reinforces the idea illustrated in Table 5 that the diffusion model changes pixels in both directions. The percentages of relative changes from white to black are significantly higher than their counterparts, reflecting the overall scarcity of white pixels, which are restricted to the 5-pixel-wide lines representing the lanes.

Table 6: Average percentage of relative pixel changes from the unrefined to the refined segmentation masks on the test dataset. S represents the number of DDIM sampling steps. The format of the results indicates the mean \pm std. across 11 testing tiles in one repetition of each experiment.

S	Noise	White \rightarrow Black	Black \rightarrow White
		Mean (%)	Mean (%)
10	✓	25.693 ± 3.287	0.637 ± 0.157
	✗	24.43 ± 3.001	0.631 ± 0.172
50	✓	25.456 ± 3.31	0.655 ± 0.168
	✗	24.369 ± 2.979	0.646 ± 0.181
100	✓	25.196 ± 2.962	0.648 ± 0.182
	✗	24.35 ± 2.495	0.826 ± 0.244
500	✓	25.41 ± 3.033	0.657 ± 0.174
	✗	24.361 ± 2.967	0.65 ± 0.183