

Reconstructing Road Network Graphs from both Aerial Lidar and Images

Biswas Parajuli ^{*1}, Ahana Roy Choudhury ^{*1}, and Piyush Kumar¹

¹Department of Computer Science, Florida State University, Tallahassee, FL.

{parajuli, roychoud, piyush}@cs.fsu.edu

Abstract

We address the problem of reconstructing road networks as undirected graphs over large geographic regions in cold start scenarios where neither the preliminary graph nor any on-road trajectory information is available. The goal of this paper is to transform bimodal aerial data in the form of 3-dimensional Lidar scans and high resolution images into road network graphs. We use a fully convolutional architecture that fuses the two datasets by reducing the disparity in their modalities to segment out roads. We then apply a simple, disk-packing based algorithm that covers the segmented regions with a minimal set of variably sized disks, connect the intersecting disks and use a provable curve reconstruction algorithm to obtain the road network graph. We show that our method is better at removing outliers and gives improved connectivity and topological accuracy than the existing state of the art thinning based method.

1. Introduction

The task of accurately and efficiently detecting road networks has gained immense importance [30] due to the current trend towards providing mobile location-based services and the ongoing research for the deployment of self-driving cars. Applications like autonomous navigation and routing [35], self-driving [12], localization [19] and others use road networks in the form of graphs. These road network graphs are mostly created by volunteers with manual labeling [28] which is a slow and expensive process and is prone to errors. For example, one of the most widely used open source mapping systems, OpenStreetMap (OSM) is currently in need of improved correctness. Bastani et al. [8] report a discrepancy of 14% between OSM and the ground truth in TorontoCity Dataset. Thus, we consider the problem of automatically extracting accurate road network graphs for large geographic regions.

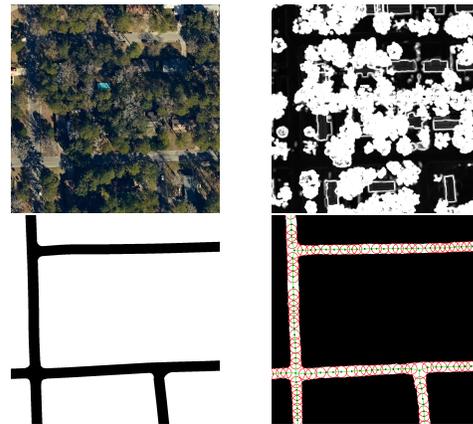


Figure 1: Given the aerial RGB (top-left) and the depth (top-right) images, we segment out roads and apply disk-packing (bottom-right) to get the road network graph. The red circles, their centers and the green links are the packed disks, the graph nodes and the graph edges respectively. The bottom-left image is the hand-curated road ground truth.

In recent days, a variety of datasets are currently being used for extracting road networks. One of the common datasets comprises GPS traces due to the proliferation of portable navigation devices on cell phones and vehicles [21, 11]. The collection and use of personal GPS traces raises privacy concerns [25] and may also result in discrepancy in coverage density because some roads in remote and rural areas will have negligible traffic compared to the roads in and around urban cores. Overall, to mitigate these shortcomings, we need a dataset which does not encroach upon users' personal privacy, provides wider road coverage and is also cost effective to collect. We use high-resolution multi-channeled aerial images, which are defined by pixels, as well as air-borne Lidar, which are defined by 3-dimensional point clouds over vast swathes of geographic regions, to extract road network graphs. At present, the current state of remote sensing technology has made the acquisition of aerial images and Lidar data cheaper, and thus, we consider these datasets for the extraction of road networks.

^{*}Equal contribution

In this paper, we present a pipeline which automatically builds road network graphs from publicly available bimodal dataset containing airborne Lidar and images. In the first step we use a deep learning architecture that fuses both Lidar and RGB images to obtain accurate road segmentation while reducing the disparity in the dataset modalities. We choose to supplement RGB images with depth information from Lidar because in case of roads that are occluded (for example beneath dense canopy), the last return of Lidar can still penetrate through the gaps among the branches and capture the surface geometry of roads [5]. We also ensure larger neighborhood context for road vs non-road classification by selecting a suitable encoder-decoder architecture that uses dilated convolution [39]. The output given by the trained segmentation model consists of per-pixel road predictions. Since these predictions are usually rasterized and noisy, we need to translate the raw road segmentation into planar road network graphs (Section 5) [22, 2].

We design a road network extraction algorithm that combines disk-packing and provable curve reconstruction in order to perform road center-line extraction. The disk-packing approach, which generates the disk intersection graph, samples points on the approximate medial axis of the road and connects these points based on the intersection of adjacent disks. However, due to imperfect segmentation, sparse and erroneous sampling of the medial axis, not all the edges in the road network graph can be obtained by using disk intersections. For this purpose, we use provable curve reconstruction to identify the missing edges and include them in our road network [22]. Finally, we run experiments on a real-world dataset and show that our method performs better than the current state of the art in various metrics.

The novelty of our technique lies in the use of a disk-packing based method for the extraction of road networks from the road segmentation masks. Our algorithm is implemented in a way that the algorithm works for reconstructing intersections, as well as handling outliers. To the best of our knowledge, deep learning-based segmentation, medial axis approximations, and provable curve reconstruction with outlier handling, have never been used together.

The paper is organized as follows. We discuss related work in the following section. We formulate the problem in Section 3. We explain our road segmentation methodology in Section 4. We present our disk-packing based algorithm in Section 5. In Section 6, we elaborate on the experiments that we conducted along with their results. We discuss the conclusion and our future plans in Section 7.

2. Related Work

Many road network graph extraction methods are related to curve reconstruction [22, 3, 15] where the goal is to retrieve the road centerlines as connected line segments from a point set sampled from the original road surface. GPS

traces provide such a sample [15, 26, 11]. Any method that uses this dataset can take advantage of the chronological ordering of points in the trajectories if it chooses to. Chen et al. [15] sample a set of points S with a $(b \cdot r_{min})$ -net such that all points in the input trajectories are within $b \cdot r_{min}$ of at least one point in S and all points in S are at least $b \cdot r_{min}$ apart. $b = 2\sqrt{6}$ is a constant chosen for algorithmic guarantees and r_{min} is a constant for the road width. In reality, road widths vary [29]. They construct a Voronoi diagram [4] with $p \in S$ as the sites and select the Delaunay edges [20] which correspond to the Voronoi edges that intersect at least one input GPS trace. In our case, we do not have the trajectory information because, CNN based road segmentation outputs binary pixel classification results. Moreover, we do not have the privilege of using principled sampling since we have to rely on the sampling provided to us by the trained road segmentation model.

In cases where aerial imagery and Lidar are used as the dataset, there are a few instances where researchers tweak and extend some parts of their road detection method to correct the output predictions [7, 29, 42]. Bajcsy et al. [7] use road-growing and thinning operators while detecting the road network which also simultaneously skeletonize the predictions to give a graph as output. Most often the corrections are done independently as a post processing step. Zhao et al. [42] use an adaptive, multi-step marching algorithm with voting to replace the detected road ribbons with centerlines. They infer the missing connections by modeling the type of intersection among the candidate road segments as a graph and determining the segment labels with an MRF framework.

Currently, use of deep convolutional neural networks (CNN) for road network detection and graph extraction from aerial images has become common [29, 17, 27, 8]. Mnih et al. [29] use a neural network to obtain road predictions and later they employ another neural network to exploit the structure present in the neighborhood of predicted patches to make corrections. Cheng et al. [17] use two CNNs to segment roads and get their centerlines. These methods address connectivity issues in the detected road networks, but they do not translate them into graphs.

Road network graphs can be obtained from the road segmentation or centerlines by executing the following steps: (1) apply morphological thinning [40] to the segmentation to get 1-pixel thick skeleton (2) do a generic breadth-first search along the skeleton to trace paths as piecewise linear curves (3) transform the curves into edges (4) post-process to either remove dangling segments or connect nearby degree-1 vertices. Instead of thinning, Voronoi diagrams can also be used to calculate the medial axis of the road segmentation polygons [32]. DeepRoadMapper [27], a state of the art, follows these steps to arrive at a preliminary graph which is prone to topological errors. To make cor-

rections, it samples candidate links within a certain neighborhood of each vertex. The bounding box of each such link is cropped from the aerial image and another CNN is used to predict if the link containing patch is a road. If it is, then the link is inserted into the predicted graph. Poor segmentation due to either occlusions or complexity of the road topology [8], has a negative impact on the accuracy.

The more recent state-of-the-art RoadTracer [8] skips the segmentation phase and directly extracts graphs from aerial images. It is more suited for warm-start scenarios where preliminary but incomplete graph is available. It does a guided search in a depth-first manner with the help of a decision CNN and the predicted partial graph by walking a predefined distance along a predicted direction. Since it requires starting locations that must lie on the road, for cold-start cases where no pre-existing map is available, we cannot do away with the road segmentation phase. Furthermore, in case the input image contains disconnected roads, each road fragment should be given its own starting location and the search algorithm should be applied as many times.

We try to address the weaknesses of the existing methods by using both aerial Lidar and images in conjunction to enhance the road segmentation accuracy. We also improve the output graph quality with our disk-packing based curve reconstruction algorithm.

3. Problem Definition

Given a feature representation of a geographic terrain as an input image, a trained road segmentation model, M_{seg} labels individual pixels as either road or non-road with pixel values 1 and 0 respectively. M_{seg} produces an image-like segmentation output I , where the pixel positions form a unitary integer grid that can be interpreted as a planar (two-dimensional, Euclidean) coordinate space. All the points in this coordinate space are grouped into two sets, $S_{rd} = \{(i, j) \mid I(i, j) = 1\}$ and $S_{nrd} = \{(i, j) \mid I(i, j) = 0\}$. In essence, M_{seg} samples points that are most likely to be on road surfaces and the global Intersection-Over-Union (IOU) metric gives a measure of its sampling quality.

The core problem this paper addresses is to translate S_{rd} into a reconstructed undirected graph $G = (V, E)$ that is embedded in the Euclidean plane. Points in S_{rd} do not form a well-behaved geometric object. Locally, given a neighborhood, the geometry is elongated that can be approximated with a rectangular strip (See Figure 2) [18]. Globally, S_{rd} can be regarded as a union of such strips. The widths of the strips vary, their perimeters may not be smooth, and they can be disconnected owing to occlusions such as trees. Going from S_{rd} to G involves replacing the local, approximate rectangles with their centerlines and adding them to E while maintaining connectivity to construct G .

4. Road Segmentation

Road Segmentation accuracy has a direct impact on the road network extraction problem. For this paper, we chose to improve one of the recent road segmentation CNN architectures, named TriSeg, that merges both Lidar and image data to solve the problem [31]. Parajuli et al. [31] attempt several methods for fusing aerial images and Lidar input and show that the TriSeg architecture performs the best. While architectures such as FuseNet allow the fusion of different types of data, the size of our GPU prevents us from exploring such architectures. We use the LPU algorithm proposed in [31] to convert the 3D point cloud to 2D image. We use this technique because our dataset has 1 Billion 3D points, and scaling 3D point cloud segmentation algorithms with noise is a challenge at this scale. TriSeg consists of 3 separate units. Unit 1 and Unit 2 process RGB and Depth images independently in parallel. Unit 3 performs late fusion by concatenating the outputs of the first two units and gives the final road segmentation.

We make two changes to TriSeg. We replace SegNet [6] with Deeplabv3+ [16] since it increases the size of the neighborhood context by using atrous convolution. Instead of simply concatenating the output from the first two units of TriSeg, we average the input RGB image to get a grayscale channel and append as an additional channel to the input of the third TriSeg unit.

We show a few output examples along with their corresponding inputs in Figure 2. Our architecture, M_{seg} does well even at places where the roads are covered by trees and affirms the benefit of using aerial Lidar in conjunction with RGB images. Table 1 shows that we can achieve 5% improvement in the global IoU metric by making the above-mentioned changes.

Grid-shifting: Both random and deterministic shifting of grids has been used in the past for various problems [41, 14, 10]. We apply a deterministic grid-shifting technique to extract a small improvement in the road segmentation accuracy. It is applicable to any CNN problem where the input image has to be cut down to smaller sizes because CNN input sizes are restricted to GPU RAM sizes.

Our goal is to extract road network graphs for larger $K \times K$ resolution tiles from their corresponding road segmentations. To obtain the road segmentation in the original resolution, we partition the larger tile into smaller sub-tiles of size $k \times k$ where $k \ll K$ and is dependent on the memory constraints of M_{seg} . In our case, $K = 2500$ and $k = 500$. We feed each $k \times k$ sub-tile into M_{seg} independently. Given a sub-tile, M_{seg} cannot learn from the context present in the neighboring tiles while trying to segment roads along the tile edges. So, if there is an occlusion that covers the road along the sub-tile margin, M_{seg} can assume it to be the end of a road segment due to the absence of context. This cre-

Table 1: Comparison of TriSeg with M_{seg} on Parajuli et al.’s [31] dataset: G-TPR, G-TNR and G-IoU are the global true positive rate, true negative rate and intersection over union respectively.

Architecture	G-TPR	G-TNR	G-IoU
TriSeg	0.877	0.983	0.784
M_{seg}	0.93	0.984	0.834

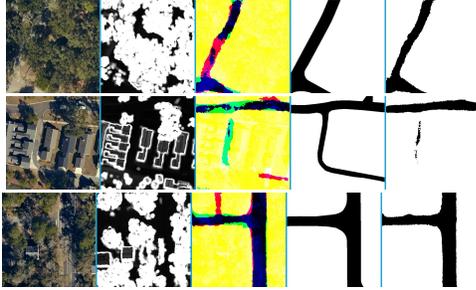


Figure 2: Sample output by M_{seg} . Each row consists of 5 images from left to right: RGB, depth, M_{seg} Unit 3 input, ground truth, output segmentation.

ates gaps in the segmentation output and results in missing links in the final road network graph. We can see a few of such examples in Figure 3. As a solution, we perform a second road segmentation but after shifting the original grid by $\frac{k}{2}$ pixels along both the directions. This forces the margins of the original sub-tiles to lie towards the middle of the new sub-tiles and M_{seg} gets more context while detecting roads.

We get the final road segmentation after combining the two segmentation outputs with and without grid-shift. For merging the segmentations, we simply take the union of the two where a pixel location is considered to be a road if at least one of the two segmentation outputs labels it as a road pixel. The Union method gives a global IoU metric of 81.2% whereas for the non-grid-shifted segmentation output, it was 79% for the dataset described in Section 6.1. We also experimented with a centrality-based approach where the segmentation is given higher confidence based on how far the pixel is from the center of the image. This method performs worse than the union method (80.6%). Thus, we use the union-merged output as our final road segmentation to extract the road network graph.

5. Disk-Packing for Graph Extraction

In this section we present our disk-packing based method to translate road segmentation into a road network graph. The vertices and the edges selected for such a graph should capture the road centerlines while preserving the road topology and connectivity. Since we only make assumptions about the directionality of roads and that degree > 2 vertices cannot be adjacent in the reconstructed graph, our algorithm is general and is applicable for other datasets.

To accomplish this goal, we greedily cover S_{rd} with a minimal set $\mathcal{B} = \{B_1, \dots, B_n\}$ of n closed disks where B_i ,

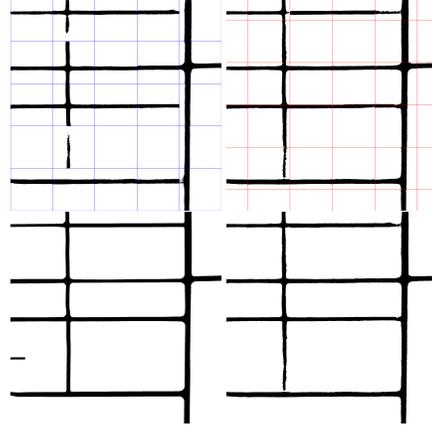


Figure 3: Effect of sub-tiling: Top left: Prediction using original tiling has gaps in road segmentation along the sub-tile margins. Bottom left: Ground truth segmentation. Top right: Grid-shifting covers some of the gaps. Bottom Right: Union-merging the two outputs improves the accuracy.

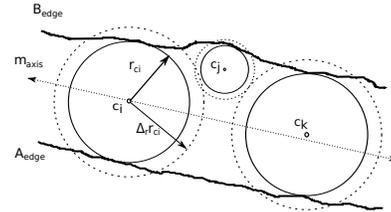


Figure 4: Detecting outlier disk: A_{edge} and B_{edge} belong to S_{edge} , the set containing pixels along the edges in the segmentation output I . m_{axis} is the ground truth road centerline. The dotted circles around each disk are the corresponding shells. The disk at c_j is an outlier disk because its shell contains only a portion of B_{edge} which translates to a single connected component. The other two are valid degree two nodes since their shells contain two connected components that lie almost opposite each other.

centered at $c_i = (x_i, y_i)$ has a radius $r_{c_i} \geq r_{min}$ and is equidistant to at least two points in S_{nrd} . If $dist(c_i, c_j)$ is the Euclidean distance between c_i and c_j and $r_{c_i} + r_{c_j} \geq dist(c_i, c_j)$, then B_i and B_j intersect. We create a planar disk intersection graph G_{disk} from a set of such intersecting disks [23]. The disk centers and the lines connecting the centers of the intersecting disks are the vertices and the edges of G_{disk} respectively. We take G_{disk} as a preliminary graph as there are still missing links because M_{seg} is not perfect and gives an IOU metric of < 1 . Next, we describe how we select \mathcal{B} and resolve connectivity issues in G_{disk} .

5.1. Disk Selection

We start with the road segmentation output I which may contain holes, corrugated edges and outliers. We partially address this issue by morphologically dilating I with a neighborhood of size r_{morph} which updates every pixel in I with the max value in that neighborhood. We apply Canny edge detection algorithm [13] on I , and find the set S_{edge}

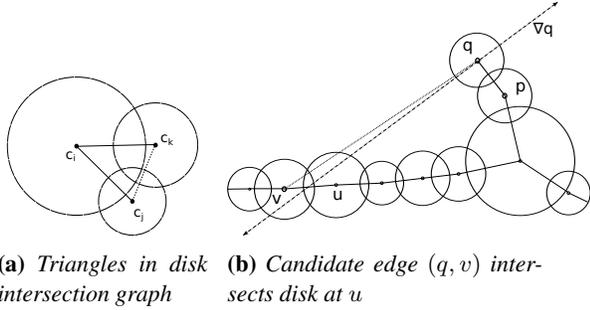


Figure 5: Disks and connections. Figure 5a: When three disks with centers c_i, c_j, c_k intersect, they create a triangle. If $r_{c_i} > r_{c_j}$ and $r_{c_i} > r_{c_k}$, we remove the shortest edge (c_j, c_k) and keep (c_i, c_k) and (c_i, c_j) . Figure 5b: q is a degree 1 node connected to p . (q, v) is a candidate edge because v is the nearest neighbor of q in the cone ∇_q with $\theta_{\nabla_q} = \pi$. Although (q, v) may satisfy the road vote count (*Condition 1*), it creates an incorrect topology and is discarded since it intersects an existing disk at u (*Condition 2*).

that contains points along the edges in I . For efficient nearest neighbor search, we construct a KD-tree KDT_{nrd} with points in S_{edge} . For every point $p \in S_{rd}$, we find the nearest $q \in S_{edge}$ at a distance $r_p = dist(p, q)$ with the help of KDT_{nrd} and then insert the tuple (p, r_p) into L_{disk} , a list that keeps track of candidate disks. Here, r_p is the radius of the disk centered at p . To proceed greedily, we iterate through L_{disk} in descending order with respect to the disk radius such that we select the largest disk first. For each p with disk radius r_p , we drop all points and disks that are covered by the disk centered at p with radius r_p . We also drop those disks whose radius is less than r_{min} .

Eventually, we want the selected disks to lie as close as possible to the medial axis [37] of actual roads. If the edges in I were perfectly smooth, each such disk would be equidistant to two (or more in case of junctions) points in S_{edge} . In reality, the edges still contain variably sized concavities. So even if a disk is equidistant to two points in S_{edge} , it can be far away from the road centerline. Figure 4 demonstrates an example. We want to automatically drop such disks that predominantly lie only on one of the half-spaces created by the closest centerline. Given a disk (c, r_c) , we first find all the points in S_{edge} which lie within $\Delta_r \cdot r_c$ of c . Here, $\Delta_r > 1$, is the radius expansion factor. We then count the number of connected components n_{comp} present in the $\Delta_r \cdot r_c$ shell of c . If $n_{comp} < 2$, it is highly likely that (c, r_c) is a noise disk and, thus, it can be dropped. Our method is similar to the shell neighborhood search described by Aanjaneya et al. [1].

5.2. Disk Intersection Graph

The disk-packing method discussed above gives a set \mathcal{B} of n disks. To obtain the disk intersection graph G_{disk} , we do a $k = 10$ nearest neighbor search for each c_i of $B_i \in \mathcal{B}$. We assume that the upper bound for the maximum degree

of any road junction is 10 and, moreover, on a random set of 20 tiles in our dataset, we find the maximum degree of any junction to be 4. If a pair of disks centered at c_i and c_j either touch or intersect, we add a new edge (c_i, c_j) to G_{disk} . Although this method takes $O(n \log n)$, since the maximum number of disks in a given square tile with the side length of 1250 feet is ≤ 400 , we do not see significant performance penalties. While polynomial time ($O(n)$) approximation schemes [23] can be used in cases where bigger tiles with larger number of disks are considered, we use smaller tiles and can compute the exact solution without having significantly longer execution times.

We should be careful while inserting edges based on disk intersection. In case there are three disks that intersect each other, c_i, c_j and c_k , as shown in Figure 5a, the graph will end up with a triangular loop. This loop translates to two U-turns within a single road width which is not a feasible scenario in road engineering [33]. So, we avoid a triangular loop by discarding the edge which has the smallest length.

5.3. Connectivity

Owing to the imperfect segmentation, G_{disk} still has missing links. The task that remains is to predict these links and insert them in G_{disk} . In the appendix, we provide figures to show that in case of imperfect segmentation, we are able to use degree 0 and degree 1 vertex connections to insert missing edges. Often there are neighboring regions in I which are partially segmented as roads but could not be covered by any disk in \mathcal{B} . This allows us to apply voting while predicting edges. Given a candidate edge $e = (u, v)$, we impose the following two conditions for it to be a valid edge:

Condition 1: Consider the strip of width β pixels centered on the segment (u, v) . If γ -fraction of the pixels in this strip are classified to be road then this predicate returns true. Both these parameters are computed using hyper-parameter tuning (Section 5.4).

Condition 2: If e intersects only $\{B_u, B_v\} \in \mathcal{B}$ then return true (A false example is shown in Figure 5b).

In our Algorithm, Condition 2 is checked only if Condition 1 returns true. Now we are ready to explain how we generate our candidate edges.

Choice of Nodes: For predicting the missing edges, we consider only those candidate edges that will be incident on degree 0 and degree 1 nodes in G_{disk} . This decision is based on the analysis of the nodes present in a randomly sampled set of 20 tiles each covering an area of $1250 \times 1250 ft^2$. Each of these tiles contains a node at every junction and at every significant bend as well as at every 100 (or less) feet. Since 91% of the nodes in the graphs are degree-2 nodes, there is a > 0.9 probability that, for a missing edge (u, v) ,

u and v are degree 2 nodes which appear as degree 1 or degree 0 nodes in G_{disk} . Out of the 258 junctions (nodes with degree > 2) present in the graphs, 236 are degree-3, 22 are degree-4 and no nodes have degree ≥ 5 . Moreover, only 7 junction pairs in the chosen tiles lie adjacent to each other and the rest have degree 2 nodes between two consecutive junctions. So, even for junctions, in most cases, missing edges result in adjacent degree 1 nodes, and inserting the missing edge for such degree 1 nodes will automatically insert the missing edge of the junctions. Thus, for predicting the missing links, we focus on degree 0 and degree 1 nodes in G_{disk} .

Establishing Connections: Using the sampling condition in Amenta et.al. [3], we can show that the smooth parts of the road network can be reconstructed correctly if we connect degree 0 vertices to its nearest neighbors (details are omitted to conserve space). To handle noise in the sampling, we connect each degree 0 vertex, q with its nearest neighbor, v if the new edge (q, v) satisfies the two conditions mentioned above. We drop any remaining degree 0 vertex as outliers. Next, we process the degree 1 vertices. A degree 1 vertex q has a direction because it is an end point of an existing edge (p, q) which is also a line segment on the plane. For reconstructing smooth parts of the road, with degree 1 vertices, we use a cone based nearest neighbor approach [22, 24]. The axis of this cone is obtained by extending the edge (p, q) beyond the point q . To give some room to wiggle while searching for u , we define a cone ∇ of angle θ_{∇} and length r_{∇} at q . If v is the nearest neighbor of q that falls within the cone ∇ , that is the angle at q made by the segments (p, q) and (q, u) is greater than $\pi - \frac{\theta_{\nabla}}{2}$, then (q, v) is a candidate edge. If the two conditions are satisfied, we insert (q, v) into G_{disk} (visual examples in Appendix).

Our algorithm is based on provable curve reconstruction algorithms [22, 24], where a set of points, P , which is an ϵ -sample of a smooth curve C , is taken as the input and used to generate a graph G . The graph G connects sample points which are adjacent to each other by connecting nearest neighbors and then considering the half-space opposite to a degree 1 node for candidate edges. However, we use a cone of angle θ_{∇} and tune it as a hyper-parameter by considering values from $\frac{\pi}{3}$ to $\frac{7\pi}{6}$ and using increments of $\frac{\pi}{6}$. Serendipitously, we get the best results by using $\theta_{\nabla} = \pi$, which matches with the theoretically provable algorithm. Hence, on the basis of the proofs provided in [22], for all parts of the road (smooth curve), where the ϵ -sampling property is satisfied, our algorithm is capable of correctly reconstructing the curve.

5.4. Hyper-parameters

There are a few hyper-parameters that govern our disk-packing based graph extraction method. We tune them with the Tree of Parzen Estimators algorithm [9] on a random

Table 2: Hyper-parameters for Disk-packing.

Hyper-parameter	Symbol	Value
Cone Angle	θ_{∇}	π
Cone radius	r_{∇}	225 feet
Shell Expansion Factor	Δ_r	1.3
Minimum Disk Radius	r_{min}	6.5 feet
Road Vote Count Width	β	11 feet
Road Vote Fraction	γ	0.2
Morphology Radius	r_{morph}	4.5 feet

Table 3: The per-tile and the global counts of the graph edges given by Disk- M_{seg} grouped based on how the edges were added.

Edges From	Per-tile Count	Global Count
Disk-intersection	357.532	149091
Degree 0	0.367	153
Degree 1	5.149	2147
Total	363.048	151391

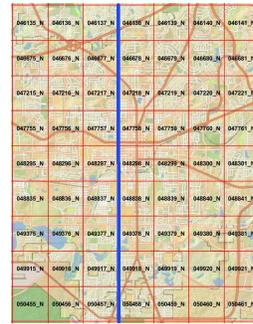


Figure 6: Tallahassee: train (right of blue margin)/test regions.

set of 20 tiles selected from the entire dataset of 1008 tiles and use the inverse of the Weighted Shortest Path metric (Section 6.3) as the loss (Table 2).

6. Experiments

We run multiple experiments on the road segmentation produced by M_{seg} from the aerial dataset of Tallahassee. Each experiment used a single 11 GB GeForce GTX 1080 card on an Intel i7-based workstation with 32 GB RAM.

6.1. Dataset

Graph extraction proceeds in two stages, namely, road segmentation and the translation of output segmentation into graphs. Since we use the Lidar Data and the aerial images, we cannot use any of the traditional datasets that only have aerial images. We use the same dataset described in [31] but we split the tiles differently as shown in Figure 6. We chose this split so as to get larger contiguous regions instead of scattered, non-adjacent tiles for train and test. This ensures road network continuity over a larger region. The dataset has 36 and 27 10000×10000 tiles for train and test respectively and we split each into 500×500 pieces. Thus, the training and test sets include 14400 and 10800 500×500 tiles respectively. Instead of subsampling, we use the whole

training set to train M_{seg} .

For graph extraction, we stack 25 adjacent 500×500 tiles to get larger 2500×2500 tiles because larger tile sizes improve the accuracy of graph extraction. The smaller the tile, the higher the chances of the road regions getting split at the edges at various angles which, in turn, gives road segments of incorrect widths and shapes.

Ground Truth Graphs: We consider two ground truth graphs for training and evaluation (a) graph obtained from OpenStreetMap, OSM-GT (b) graph generated from the ground truth road segmentation by applying thinning, TLH-GT. On visual inspection we find differences between TLH-GT and OSM-GT. OSM-GT considers parking lots as part of the road network whereas TLH-GT does not. In case of multi-lane roads that do not have well defined medians, TLH-GT ends up with a single edge owing to the thinning algorithm whereas OSM-GT usually has an edge for every lane. Due to these reasons there are more road edges in OSM-GT than TLH-GT. Besides, TLH-GT is more suitable for the extraction of road network graphs, and this is evident from the fact that all the methods except RoadTracer trained on OSM-GT perform better when compared to TLH-GT rather than OSM-GT. If we compute the Weighted Shortest Path metric (Section 6.3) between OSM-GT and TLH-GT with OSM-GT as the ground truth and measure the number of correct paths in TLH-GT compared to OSM-GT, the *Correct* percentage is only 22%, whereas, with TLH-GT as the ground truth and the OSM-GT as the predicted graph, the same metric is 76%. This indicates that OSM-GT has several extra edges that are not present in TLH-GT and this matches with our intuition based on the visual inspection. So, the evaluation metrics will also differ when we compare the predicted graphs with the two ground truth graphs.

6.2. Baselines

We compare our disk-packing method (Disk- M_{seg}) described in Section 5 with two baselines. The first is RoadTracer [8] for which we train two different models, namely, RoadTracer-OSM and RoadTracer-TLH with OSM-GT and TLH-GT as the respective ground truths. For the second DeepRoadMapper [27] baseline, we use the implementation provided by Bastani et al [8]. We compare with the graphs produced after directly applying thinning on M_{seg} 's output I (Thin- M_{seg}). Additionally, we disk-pack DeepRoadMapper's segmentation output to get Disk-DeepRoadMapper.

RoadTracer is highly affected by the choice of starting locations. To select at least one starting location per road segment during prediction, we: (1) randomly select a node from each connected component in the ground truth graph ($start = random$) (2) apply disk-packing on I and pick all the centers of the packed disks ($start = \mathcal{B}$).

6.3. Evaluation Metrics

We evaluate the predicted graph $G = (V, E)$ with the corresponding ground truth graph $G^* = (V^*, E^*)$ with respect to vertex reachability, topology and path distance between registered vertices using three metrics. While Junction Metric [8] measures the number of *Correct* and *Extra* junctions, TOPO metric [11] evaluates the graphs in terms of topology using Spurious and Missing and utilizes the values of Spurious and Missing to compute the F -score, which lies in the closed interval $[0, 1]$.

The original Shortest Path metric [38] gives equal weights to all source/destination pairs and keeps track of the *Correct*, Short, Long and NoPath counts but the longer shortest paths in the ground truth are more valuable for the metric. Thus, we modify the metric by multiplying each count with $\frac{l_{u,v}}{l_{diag}}$, where l_{diag} is the length of the longer diagonal of the tile. In our case, $l_{diag} = 2500\sqrt{2}$ since we have square tiles with side length 2500 pixels. Finally, we normalize the values by dividing each score by the sum of all the weights and report the percentage of *Correct* edges in our Weighted Shortest Path metric (W-SP).

6.4. Vertex Registration and Distribution

Most of the graph similarity metrics require the registration of the ground truth vertex set with the predicted set. To register $u^* \in V^*$ with one of the vertices in V , we first find the edge $(u, v) \in E$ which is nearest to u^* and within a pre-defined threshold r_{max} . We set $r_{max} = 30$ feet which is the maximum lane width for US highways [36]. We project u^* onto (u, v) and get a new vertex p between u and v . If p is within r_{max} of one of the end points of (u, v) , say u , then u^* is registered with u . Otherwise, we replace the edge $(u, v) \in E$ with two new edges, (u, p) and (p, v) and then establish correspondence between u^* and p . If there is no edge in E that lies within r_{max} of u^* , u^* remains unregistered. This procedure does not affect the topology of the predicted graph G . The graph similarity metrics are affected by the vertex distribution which depends on the local road curvature [3, 34]. To make the distribution uniform, we insert new, evenly spaced vertices at a distance of 50 feet into existing edges of the graph without altering its topology.

6.5. Results

Our method (Disk- M_{seg}) achieves the best performance for all the metrics when compared to TLH-GT. We present our results in Table 4 and identify the best performing methods for each metric. In case of junction metric, we highlight the methods that perform well for the *Correct* and *Extra* scores individually. Besides, we identify the method (in italics) that achieves the best balance between *Extra* and *Correct*, indicating the ability to detect junctions correctly without detecting too many spurious junctions. For TOPO

Table 4: Comparison of graph extraction methods with two ground truths: (a) OpenStreetMap (OSM-GT) (b) Map obtained after applying thinning to Tallahassee road masks (TLH-GT).

Method	OSM-GT			TLH-GT		
	Junction (Correct, Extra)	TOPO F-Score	W-SP Correct	Junction (Correct, Extra)	TOPO F-Score	W-SP Correct
RoadTracer-OSM [8] (start=random)	(0.34, 0.23)	0.45	0.09	(0.39, 0.73)	0.42	0.15
RoadTracer-TLH (start=random) [8]	(0.15, 0.20)	0.34	0.04	(0.37, 0.29)	0.45	0.14
RoadTracer-OSM (start= \mathcal{B}) [8]	(0.39 , 0.26)	0.56	0.14	(0.48, 0.70)	0.54	0.24
RoadTracer-TLH (start= \mathcal{B}) [8]	(0.17, 0.27)	0.50	0.06	(0.42, 0.37)	0.61	0.20
DeepRoadMapper [27]	(0.28, 0.60)	0.60	0.13	(0.57, 0.75)	0.64	0.40
Disk-DeepRoadMapper (DeepRoadMapper CNN [27] with our graph extraction)	(0.34, 0.46)	0.62	0.18	(0.69, 0.65)	0.68	0.56
Thin- M_{seg} (M_{seg} segmentation with DeepRoadMapper thinning [27])	(0.31, 0.86)	0.52	0.20	(0.81 , 0.90)	0.62	0.75
Disk- M_{seg} (M_{seg} segmentation process with our graph extraction)	(0.26, 0.32)	0.61	0.16	(0.72, 0.39)	0.76	0.75

and W-SP too, we highlight the best performing methods.

While our method produces the best performance for TLH-GT, it is also competitive in terms of TOPO and Weighted Shortest Path metric for OSM-GT. In fact, the evaluation metrics for all the reported methods are far better when we use TLH-GT instead of OSM-GT as the ground truth because OSM-GT has more road edges (Section 6.1) compared to TLH-GT. So, during metric computation with OSM-GT, a major fraction of the sampled vertices can belong to these extra edges and reduce the metric scores.

RoadTracer performs poorly for almost all the metrics in case of TLH-GT and only achieves good performance for junction metric in case of OSM-GT. However, RoadTracer has the advantage of utilizing existing graphs, and can be trained separately for OSM-GT and TLH-GT. Our dataset covers a vegetation rich geographic region with a small urban core because of which it is not guaranteed that the selected starting locations or the locations selected by its decision CNN will fall exactly on the roads. If such a location happens to fall on the canopy, RoadTracer directly terminates its search missing out on detecting significant portions of the roads. It also excludes regions around the tile edges because its decision CNN uses a window of a fixed size. Table 4 shows that RoadTracer’s performance is affected by the choice and the number of starting locations. The metrics are better if we use all disk centers in \mathcal{B} instead of one randomly selected point per connected component because number of connected components $\ll |\mathcal{B}|$ and points in \mathcal{B} are more likely to be on visibly cleaner road regions. DeepRoadMapper and M_{seg} can only be trained on segmentation masks, which are available for TLH-GT only and causes the segmentation to be performed in accordance with TLH-GT.

From the results in Table 4, we observe that the quality of the output graph depends on the IoU metric given by the road segmenter. M_{seg} with grid-shifting (81.2% IoU) is better than DeepRoadMapper (68.3% IoU). So, irrespective of which graph extraction method we use, the majority of the metrics are higher for M_{seg} ’s output.

Our Disk-packing based graph extraction approach performs better than thinning and has several advantages over thinning. Between Thinning and Disk-packing for the same M_{seg} road segmentation, both give almost equal accuracy

in terms of connectivity and correct paths. As we see in Table 3, Disk- M_{seg} on average adds at least 5 missing edges per tile. Thin- M_{seg} , by design, does not add any new edges but still performs as good as Disk- M_{seg} in the Weighted Shortest Path metric. This can be attributed to the fact that Thin- M_{seg} preserves the road continuity even when there is just a single road pixel along the path whereas Disk- M_{seg} does not place disks on regions that are thinner than r_{min} which results in missing links. Since Thin- M_{seg} is better in this regard, we can borrow connections into the disk-packed graph from the thinned graph where possible. As thinning allows outliers and unwanted branching, it predicts spurious edges which adversely affect the F -score in case of TOPO and increase the number of extra junctions in Junction metrics. Our disk-packing method captures significantly fewer number of erroneous junctions (a margin of 51%) and performs 14% better in TOPO’s F-score.

7. Conclusion & Future Work

The aim of this paper was to use both aerial Lidar and images to extract roads as graphs. We presented an algorithm based on disk-packing to transform road segmentation output to a road network graph where the edges of the graph lie along the road centerlines. We observed that our method is better at discarding outliers and is able to capture the network topology more accurately while being close to the existing method that uses thinning in terms of connectivity. As our approach decouples segmentation from graph reconstruction and since better segmentation always translates to a more accurate graph, we can aim for optimizing road segmentation with the best available CNN architecture. M_{seg} and our graph extraction method can be incorporated with existing road graph extraction methods like DeepRoadMapper and RoadTracer, in order to improve their performance.

There are still ambiguities related to the ground truth graphs due to parking lots, medians and lane discrimination. In future, we plan to address these issues by segmenting each ambiguous object separately and building the graph in a fine-grained fashion. Another aspect that we will explore is the reconstruction of directed graphs, especially using a reduction from problems in TSP, linear programming or convex quadratic programming.

References

- [1] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. Guibas, and D. Morozov. Metric graph reconstruction from noisy data. *International Journal of Computational Geometry & Applications*, 22(04):305–325, 2012.
- [2] E. Althaus and K. Mehlhorn. Tsp-based curve reconstruction in polynomial time. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '00, pages 686–695, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.
- [3] N. Amenta, M. Bern, and D. Eppstein. The crust and the β -skeleton: Combinatorial curve reconstruction. *Graphical models and image processing*, 60(2):125–135, 1998.
- [4] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.
- [5] Z. Azizi, A. Najafi, and S. Sadeghian. Forest road detection using lidar data. *Journal of forestry research*, 25(4):975–980, 2014.
- [6] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.
- [7] R. Bajcsy and M. Tavakoli. Computer recognition of roads from satellite pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, (9):623–637, 1976.
- [8] F. Bastani, S. He, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, S. Madden, and D. 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.
- [9] J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International Conference on Machine Learning*, pages 115–123, 2013.
- [10] M. Bern, D. Eppstein, and J. Gilbert. Provably good mesh generation. *J. Comput. Syst. Sci.*, 48(3):384–409, June 1994.
- [11] J. Biagioni and J. Eriksson. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *Transportation research record*, 2291(1):61–71, 2012.
- [12] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.
- [13] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.
- [14] T. M. Chan. Closest-point problems simplified on the ram. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '02, pages 472–473, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.
- [15] D. Chen, L. J. Guibas, J. Hershberger, and J. Sun. Road network reconstruction for organizing paths. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1309–1320. Society for Industrial and Applied Mathematics, 2010.
- [16] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818, 2018.
- [17] G. Cheng, Y. Wang, S. Xu, H. Wang, S. Xiang, and C. Pan. Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing*, 55(6):3322–3337, 2017.
- [18] S.-W. Cheng, S. Funke, M. Golin, P. Kumar, S.-H. Poon, and E. Ramos. Curve reconstruction from noisy samples. In *Proceedings of the Nineteenth Annual Symposium on Computational Geometry*, SCG '03, pages 302–311, New York, NY, USA, 2003. ACM.
- [19] D. Costea and M. Leordeanu. Aerial image geolocalization from recognition and matching of roads and intersections. *arXiv preprint arXiv:1605.08323*, 2016.
- [20] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer Publishing Company, Incorporated, 3rd edition, 2010.
- [21] D. Dellling, A. V. Goldberg, M. Goldszmidt, J. Krumm, K. Talwar, and R. F. Werneck. Navigation made personal: Inferring driving preferences from gps traces. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 31. ACM, 2015.
- [22] T. K. Dey and P. Kumar. A simple provable algorithm for curve reconstruction. In *In Proc. 10th ACM-SIAM Sympos. Discrete Algorithms*. Citeseer, 1999.
- [23] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.
- [24] S. Funke and E. A. Ramos. Reconstructing a collection of curves with corners and endpoints. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, pages 344–353, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.
- [25] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 161–171. ACM, 2007.
- [26] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 89–98. ACM, 2012.
- [27] G. Mátyus, W. Luo, and R. Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *International Conference on Computer Vision*, volume 2, 2017.
- [28] G. Miller. The huge, unseen operation behind the accuracy of google maps. <https://www.wired.com/2014/12/google-maps-ground-truth/>.
- [29] V. Mnih and G. Hinton. Learning to detect roads in high-resolution aerial images. *Computer Vision—ECCV 2010*, pages 210–223, 2010.

- [30] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.
- [31] B. Parajuli, P. Kumar, T. Mukherjee, E. Pasiliao, and S. Jambawalikar. Fusion of aerial lidar and images for road segmentation with deep cnn. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 548–551. ACM, 2018.
- [32] R. Ramamurthy and R. T. Farouki. Voronoi diagram and medial axis algorithm for planar domains with curved boundaries i. theoretical foundations. *Journal of Computational and Applied Mathematics*, 102(1):119–141, 1999.
- [33] J. Reid, L. Sutherland, B. Ray, A. Daleiden, P. Jenior, J. Knudsen, et al. Median u-turn intersection: informational guide. Technical report, United States. Federal Highway Administration. Office of Safety, 2014.
- [34] A. Saalfeld. Topologically consistent line simplification with the douglas-peucker algorithm. *Cartography and Geographic Information Science*, 26(1):7–18, 1999.
- [35] G. Siegle, J. Geisler, F. Laubenstein, H.-H. Nagel, and G. Struck. Autonomous driving on a road network. In *Intelligent Vehicles' 92 Symposium., Proceedings of the*, pages 403–408. IEEE, 1992.
- [36] W. J. Stein and T. R. Neuman. Mitigation strategies for design exceptions. Technical Report FHWA-SA-07-011, CH2M HILL, Inc., Mendota Heights, Minnesota, July 2007.
- [37] S. Tsogkas and S. Dickinson. Amat: Medial axis transform for natural images. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [38] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler. A higher-order crf model for road network extraction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1698–1705, 2013.
- [39] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.
- [40] T. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984.
- [41] G. Zhao, J. Wang, and Z. Zhang. Random shifting for cnn: a solution to reduce information loss in down-sampling layers. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 3476–3482, 2017.
- [42] J. Zhao and S. You. Road network extraction from airborne lidar data using scene context. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 9–16. IEEE, 2012.