RNGDet: Road Network Graph Detection by Transformer in Aerial Images

Zhenhua Xu, *Student Member, IEEE*, Yuxuan Liu, *Student Member, IEEE*, Lu Gan, *Student Member, IEEE*, Yuxiang Sun, *Member, IEEE*, Ming Liu, *Senior Member, IEEE* and Lujia Wang, *Member, IEEE*

***Abstract*—Road network graphs provide critical information for autonomous vehicle applications, such as motion planning on drivable areas. However, manually annotating road network graphs is inefficient and labor-intensive. Automatically detecting road network graphs could alleviate this issue, but existing works are either segmentation-based approaches that could not ensure satisfactory topology correctness, or graph-based approaches that could not present precise enough detection results. To provide a solution to these problems, we propose a novel approach based on transformer and imitation learning named RNGDet (Road Network Graph Detection by Transformer) in this paper. In view of that high-resolution aerial images could be easily accessed all over the world nowadays, we make use of aerial images in our approach. Taken as input an aerial image, our approach iteratively generates road network graphs vertex-by-vertex. Our approach can handle complicated intersection points of various numbers of road segments. We evaluate our approach on a publicly available dataset. The superiority of our approach is demonstrated through the comparative experiments.**

***Index Terms*—Road Network Graph Detection, Transformer, Imitation Learning, Aerial Images.**

I. Introduction

In recent years, road network graph has attracted increasing attention in the field of autonomous driving. It serves as a fundamental component for autonomous navigation of vehicles. The graph of road networks is a kind of vectorized data representation, which consists of vertices and edges [1]. Each road segment could be seen as a graph edge, and the intersection points of road segments are vertices. Manually annotating the road network graph is time-consuming and labor-intensive, especially when road networks cover a large area (e.g., a whole city). Therefore, how to automatically detect road network graphs using computer algorithms in large areas is of great interest to the research community.

To address this issue, past approaches on road network graph detection usually use aerial images obtained from unmanned aerial vehicles (UAVs) or satellites [2]. As aerial imaging technology evolves, high-resolution and high-quality aerial images can be easily accessed world-widely nowadays. Moreover, some aerial imaging datasets also provide extra channels besides RGB (Red-Green-Blue), such as the infrared channel [3], making them more informative for detection purposes. So, this work also uses aerial images for road network graph detection.

Zhenhua Xu, Yuxuan Liu, Lu Gan, Ming Liu and Lujia Wang are with The Hong Kong University of Science and Technology (email: {zxubg,yliuhb,lganaa}@connect.ust.hk,{eelium,eewang}@ust.hk).

Yuxiang Sun is with The Hong Kong Polytechnic University (email: yx.sun@polyu.edu.hk, sun.yuxiang@outlook.com).

Corresponding author: Lujia Wang.

(a) Ground-truth (b) RNGDet

Fig. 1: A sample result of RNGDet. (a) The ground-truth road network graph (cyan lines). (b) The graph predicted by RNGDet (orange lines for edges and yellow points for vertices). We can see that RNGDet effectively detects the road network graph with high quality. For better visualization, the lines are drawn with a thicker width while it is actually one-pixel width. The figure is best viewed in color. Please zoom in for details.

Existing works on road network graph detection can be generally classified into two categories: segmentation-based approaches [2], [4]–[7] and graph-based approaches [8]–[12]. The segmentation-based approaches first predict the probabilistic segmentation map of the road network graph, and then conduct a series of processing to obtain the graph structure of the road network, such as skeletonization and filtering. Most of the early works on road network graph detection in this field fall into this category. The segmentation-based approaches could present good results in the pixel-level evaluation (e.g., by F1 score) due to the use of existing powerful semantic segmentation networks, but they usually suffer from unsatisfactory topology correctness such as incorrect crossroad connectivity and false disconnection on the road. To address this issue, recent graph-based approaches resort to detecting the graph of road networks directly. They usually first pre-predict candidate initial vertices, then, starting from each candidate initial vertex, train a decision-making agent to predict adjacent vertices of the current vertex. In this way, road network graphs can be generated vertex-by-vertex in an iterative manner. Although these graph-based approaches could enhance the topology correctness, they are usually composed of two separate stages, making them hard to be optimized in an end-to-end way. The separate stages might accumulate errors and hence degrade their effectiveness and efficiency.

To provide a solution to these issues, in this paper, we

propose a graph-based end-to-end approach named RNGDet (**R**oad **N**etwork **G**raph **Det**ection by Transformer). Similar to previous graph-based approaches, RNGDet starts from pre-predicted candidate initial vertices to extract local visual features using a convolutional neural networks (CNN) backbone, and then sends the features to a transformer network inspired by the DETR structure [13]. Due to the use of deep candidate queries, RNGDet can directly predict any number of adjacent vertices of the current vertex at one time, so that it can handle any road networks, even those with complicated topology (e.g., road intersections of arbitrary numbers of road segments). Different from previous graph-based approaches, RNGDet can be optimized as a whole and trained end-to-end. We train RNGDet through imitation learning to enable it to take the most appropriate action under different circumstances. To generate the training label (i.e., expert demonstration from the imitation learning perspective), we propose an algorithm to supervise the agent to explore the whole road network. The algorithm can also correct the action of the agent if it is too far away from the right track, which guarantees the quality and efficiency of the sampling process.

The proposed RNGDet is trained and evaluated on a publicly available dataset released by RoadTracer [8]. With this dataset, we compare RNGDet with state-of-the-art works based on multiple evaluation metric scores. An example of RNGDet is visualized in Fig. 1. The contributions of our work are summarized as below:

- We propose an end-to-end trainable approach named RNGDet based on transformer and imitation learning to automatically detect the road network graph.
- We propose an algorithm to automatically generate training labels for RNGDet.
- We evaluate RNGDet and compare it with state-of-the-art works on a publicly available dataset to demonstrate the superiority of RNGDet.

The remainder of this article is organized as follows. Section II introduces related works. Section III describes the network structure and working pipeline of RNGDet. Section IV presents experimental results, discussions and limitations. Section V concludes this paper.

II. Related works

A. Segmentation-based approaches

Segmentation-based road network graph detection approaches [2], [4]–[7] mainly have two stages: (1) predict the segmentation map (i.e., probabilistic map of the road network) and (2) process the segmentation map and obtain graph structures by post-processing, such as skeletonization and filtering. It is believed that [2] proposed by Mnih *et al.* is the first work that implemented neural networks to detect the road network in aerial images. They first split the large aerial image into small patches, then predicted the road network within each patch and finally merged patches into the final predicted road network segmentation map. Most afterwards segmentation-based works followed a similar pipeline, but with more powerful segmentation networks, such as U-Net [14], DeepLab V3+ [15] and FPN [16]. Batra *et al.* [4] extended the aforementioned works by adding another refinement network to fix incorrect pixels in the predicted segmentation map, which effectively improved the final performance. Mattyus *et al.* [5] generated candidate connections to correct wrong disconnections in the road network and trained a new network to filter the candidates. Even though the topology correctness of the final road network graph was enhanced to some extent, it was still not sufficient. Since semantic segmentation only optimizes on pixel-level predictions while topology information is not effectively considered, the road network graph obtained by segmentation-based approaches tends to have poor topology correctness.

B. Graph-based approaches

Different from segmentation-based approaches, graph-based approaches for road network graph detection can directly output the graph structure. Most of the graph-based works detect the road network graph by iterative graph generation [8]–[12]. RoadTracer [8] is believed to be the first work that iteratively generates the road network graph. In this work, ground-truth initial vertices were used to start the iteration. From each initial vertex, RoadTracer predicted the direction of the adjacent vertices of the current vertex as a multi-class classification problem, then moved the agent in the predicted direction by a fixed length. RoadTracer presented much better topology performance than past segmentation-based approaches, but it failed to detect road intersections with high quality due to the fixed step length. Tan *et al.* [11] solved this problem by replacing the direction prediction with heatmap prediction, where heatmap demonstrated the probabilistic distribution of adjacent vertices of the current vertex. After obtaining the heatmap, the authors extracted local peaks as the predicted adjacent vertices. Although this approach had dynamic step length and presented superior performance than past works, it can not be optimized in an end-to-end way due to the post-processing of the heatmap, which degrades the final performance. In addition, this work cannot distinguish vertices that are too close to each other.

Different from aforementioned iterative graph-based approaches, Song *et al.* proposed Sat2Graph [10] to directly predict the road network graph. Taken as input an aerial image, Sat2Graph predicted a 19 dimension tensor. This high dimension tensor contained all the information of the road network graph so that the graph could be decoded from the feature tensor by proposed algorithms. However, Sat2Graph had the isomorphic encoding issue analyzed in [10], which made it difficult to supervise during training. Moreover, it cannot distinguish road segments whose intersection angle is small.

C. Graph detection of objects similar to road networks

Due to the shape characteristics of the road network, there are some works on the detection task of objects that are similar to the road network, such as road boundaries [17]–[19], road lane lines [20]–[22], road lane [23], [24] and road curbs [25], [26]. Even though these works do not work on road network detection problems, their task is similar to ours and some ideas or techniques are inspiring to us. Xu *et al.* [25] first proposed to analyze the graph detection problem from the perspective of imitation learning, and they designed a DAgger-based system for road curb detection following the DAgger algorithm [27]. Homayounfar *et al.* proposed DagMapper [21]

to detect road lane line graph in the point cloud map on the highway. DagMapper can predict the direction of adjacent vertices of the current vertex, and whether the agent should create a new lane line branch when lane line intersections were encountered. Although these works could be inspiring to our task, they cannot handle the road network graph detection task since road networks possess much more complicated topology (e.g., road split, road merge and crossroads).

D. Transformer-based detection

In recent years, transformer [28] has been receiving more and more attention since its powerful parallelization capacity and great ability to handle sequential tasks. Considering these properties, Carion *et al.* [13] proposed DETR (Detection by Transformer) for one-shot 2D object detection, which is anchor free and can be trained in an end-to-end way. After extracting image features by a CNN backbone, DETR sent the obtained features as well as multiple candidate object queries to a transformer, and then obtained the bounding box coordinates of objects. Each bounding box was encoded by a 4D embedding. Therefore, taken as input an image, DETR can directly output the coordinates of object bounding boxes. Xu *et al.* [29] adapted DETR to line segment detection task and named the new model as LETR (Line Segment Detection by Transformer). In this paper, the authors encoded each line segment as an embedding and predicted the encoding embedding by a DETR-based network. Similarly, Can *et al.* [30] modified the DETR network and pursued to detect lane centerlines. Each lane centerline was fitted by a B-spline and each B-spline was encoded by an embedding. In this way, the authors could directly detect all lane centerlines at one time by predicting the embedding that encoded B-spline information. Our work RNGDet is also inspired by DETR, while the output embedding encodes the information of adjacent vertices of the current vertex.

III. The Proposed Approach

A. Approach overview

This work aims to detect the road network graph from aerial images, and the road network graph can be used for real-world applications (e.g., autonomous vehicle navigation). Suppose the input is a large aerial image I, then the final output should be a graph $G = (V, E)$. $E = \{e_i\}$ is a set of graph edges and each edge e_i represents a road segment. $V = \{v_j\}$ is a set of graph vertices, and each vertex v_j is either one intersection point of some road segments or the endpoint of a broken road.

Based on the DETR structure, our proposed RNGDet detects the road network graph by iterations. Starting from a pre-predicted candidate initial vertex in $S = \{s_k\}_{k=1}^{K}$, RNGDet iteratively generates the road network graph by controlling an agent exploring the road network. During iterations, the history trajectory of the agent is recorded by a graph G_H. At each step, centering at the current position of the agent v_t, RNGDet crops an ROI $\in \mathbb{R}^{3\times L\times L}$ on I and rasterizes G_H within the ROI as $\mathcal{H} \in \mathbb{R}^{1\times L\times L}$. A CNN backbone network is utilized to extract the deep visual feature of the ROI as $\mathcal{F}_I$ which is sent the segmentation heads to predict the road segment segmentation map $\mathcal{S}$ and the road intersection segmentation map $\mathcal{I}$. The candidate initial vertices in S can be obtained by finding local peaks of $\mathcal{I}$. After concatenating $\mathcal{H}$ with $\mathcal{S}$ and $\mathcal{I}$, another CNN backbone network is used to extract the deep feature $\mathcal{F}_{\mathcal{H}}$. Then, $\mathcal{F}_{\mathcal{H}}$ and $\mathcal{F}_I$ are fused together as the input feature tensor of the transformer.

Taking N candidate vertex queries $Q = \{q_i\}_{i=1}^{N}$ as input, the transformer decoder directly predicts N vertex embeddings. Each embedding encodes the valid probability and coordinates of a vertex. After filtering out vertices with low probability, we obtain $M (M \leq N)$ valid vertices that are adjacent to v_t as a set $\mathcal{V} = \{v_{t+1}^i\}_{i=1}^{M}$. If $M = 0$, RNGDet pops a new candidate initial vertex from S and repeats the above process; if $M = 1$, the agent moves to the predicted coordinate and repeats the above process; if $M > 1$, the agent pushes all the predicted vertices into set S, pops one vertex from set S and then repeats the above process. When S is empty, RNGDet stops and outputs the final road network graph. The overview pseudocode of our system is shown in Alg. 1. The system diagram of RNGDet is visualized in Fig. 2.

Algorithm 1: RNGDet

Input: An aerial image I
Output: The road network graph $G = (V, E)$

```
begin
    S ← find_initial_vertex(I)
    while S not empty do
        t ← 0
        v_t ← S.pop()
        while true do
            F ← CNN(I, G_H, v_t)
            {v̂_{t+1}^i}_{i=1}^N ← Transformer(F|Q)
            V ← filter({v̂_{t+1}^i}_{i=1}^N)
            if |V| = 0 then
                break
            else if |V| = 1 then
                v_t ← v̂_{t+1}^1
                t ← t + 1
                update G and G_H
            else if |V| > 1 then
                S ← S ∪ V
                break
        end
    end
    return G
end
```

B. CNN backbone and segmentation

The input of RNGDet is a large RGB aerial image $I \in \mathbb{R}^{3\times 4096\times 4096}$. Since RNGDet requires the history information for iterative graph generation, we maintain a graph G_H recording the past trajectory of the agent. Due to the size of input images, RNGDet crops a ROI$\in \mathbb{R}^{3\times L\times L}$ (L is 256 in our experiment) on I. A multi-layer convolutional neural network (CNN) is utilized to extract the deep feature of the ROI as $\mathcal{F}_I$, and the CNN backbone in this paper is ResNet [31]. Based on the extracted deep feature $\mathcal{F}_I$, two feature paradigm network (FPN) [16] segmentation heads predict the segmentation map $\mathcal{S}$ and $\mathcal{I}$, where $\mathcal{S}$ demonstrates the distribution of road segments

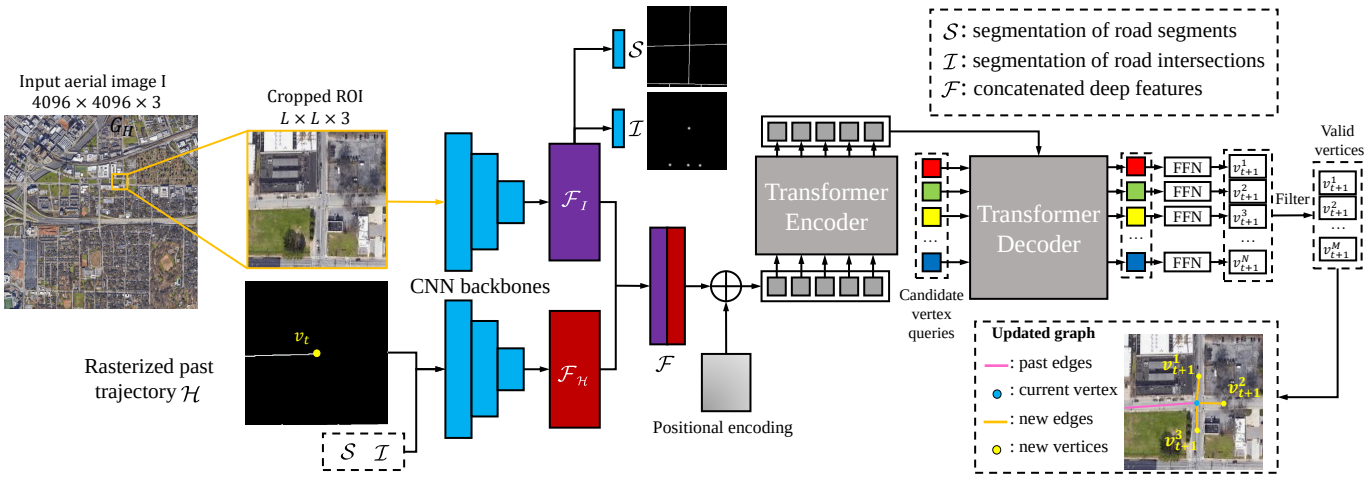


Fig. 2: The system overview of RNGDet. In this example, RNGDet processes a road intersection. Suppose the agent is at v_t at the current time step (denoted by a yellow node in $\mathcal{H}$), we crop an ROI on the input aerial image I and rasterized G_H within the ROI as $\mathcal{H}$. A CNN backbone network predicts the feature $\mathcal{F}_I$ of the ROI. Then $\mathcal{F}_I$ is sent to the segmentation heads to predict $\mathcal{S}$ and $\mathcal{I}$. Taken as input $\mathcal{S}$, $\mathcal{I}$ and $\mathcal{F}_I$, another CNN network predicts the deep feature tensor $\mathcal{F}_\mathcal{H}$ and concatenates it with $\mathcal{F}_I$ as the final feature tensor $\mathcal{F}$. The transformer predicts the adjacent vertices $\mathcal{V} = \{v_{t+1}^i\}_{i=1}^M$ of v_t. In the updated graph of this example, there are three predicted adjacent vertices $\{v_{t+1}^i\}_{i=1}^3$ (yellow nodes), and they are connected with the current vertex v_t (blue node) by three new edges (orange lines). The graph G is generated in this way iteratively. This figure is best viewed in color. Please zoom in for details.

and $\mathcal{I}$ shows the distribution of road intersection points as well as endpoints of broken roads. Both segmentation tasks are binary segmentation.

To obtain the information of past trajectories, we crop G_H in the same way as cropping the ROI and rasterize the cropped G_H as $\mathcal{H} \in \mathbb{R}^{1\times L\times L}$ for afterward concatenation. With $\mathcal{S}$, $\mathcal{I}$ and $\mathcal{H}$ as input, another CNN network outputs the feature tensor $\mathcal{F}_\mathcal{H}$. Two feature tensors are concatenated together as the final feature tensor $\mathcal{F}$, containing all the information required by the transformer.

C. Transformer architecture

After obtaining the deep feature tensor $\mathcal{F}$, the transformer predicts the adjacent vertices $\{v_{t+1}^i\}_{i=1}^M$ of the current vertex v_t. $\mathcal{F}$ is reduced to a sequence of smaller tensors, and then fixed positional encoding is fused due to the permutation-invariant characteristics of the transformer architecture. The input and output of the transformer encoder have the same length.

The decoder of the transformer takes in the output sequence of the encoder as well as a set of candidate vertex queries $Q = \{q_i\}_{i=1}^N$, and predicts N embeddings of the adjacent vertices. Each vertex query q_i is a learnable positional encoding and produces one predicted adjacent vertex. In fact, the transformer outputs the Maximum-a-Posterior (MAP) estimation of vertices in the next time step

$$\{\underset{v_{t+1}^i}{\operatorname{argmax}}\, p[v_{t+1}^i|\mathcal{F}, v_t, q_i]\}_{i=1}^N, \tag{1}$$

Each output embedding can be decoded into a probability p_i and a 2D vertex coordinate v_{t+1}^i by a feed-forward network (FFN). p_i demonstrates the probability that v_{t+1}^i is valid and should be added into the road network graph G. Suppose we have $M (M \leq N)$ valid predicted vertices as a set $\mathcal{V} = \{v_{t+1}^i\}_{i=1}^M$, then for each valid predicted vertex v_{t+1}^i, we update the road network graph G by adding v_{t+1}^i into V and a new edge connecting v_{t+1}^i with v_t into E.

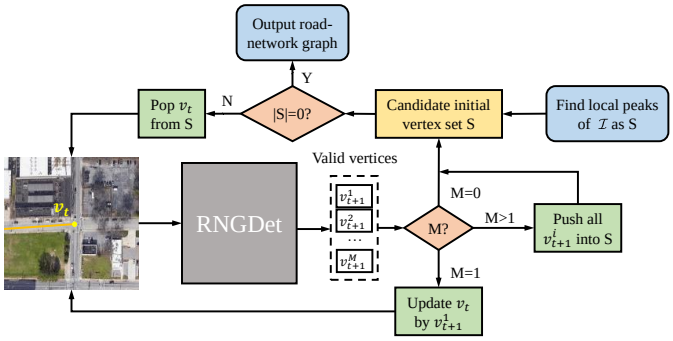
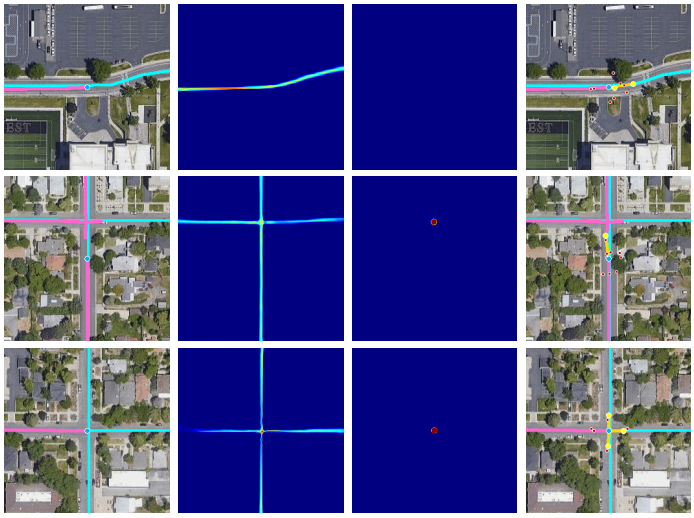


Fig. 3: The pipeline of road network graph generation by RNGDet. RNGDet iteratively generates the road network graph vertex-by-vertex. According to the number of valid vertices at each step, RNGDet takes different actions to update the graph. Please zoom in for details.

D. Policy for graph generation

RNGDet generates the road network graph by iterations. First, we initialize the candidate initial vertex set S by finding local peaks of the road intersection segmentation map $\mathcal{I}$. Then, RNGDet pops one vertex v_t from the set S. Centering at v_t, RNGDet crops input images and predicts valid vertices in the next time step as $\mathcal{V}$. Based on the number of valid vertices M, RNGDet takes different actions to update the road network graph: (1) $M = 0$. It means there is no road ahead, thus RNGDet should stop processing the current road and turns to work on other roads if S is not empty. (2) $M = 1$. This happens



(a) Step t (b) $\mathcal{S}$ (c) $\mathcal{I}$ (d) Step $t+1$

Fig. 4: Visualization of graph updating. Each row represents an example that shows how RNGDet iteratively generates the road network graph. (a) ROI of the current step t. The cyan lines are the ground-truth road network, the pink lines demonstrate history graph and the blue node is v_t. (b) The segmentation map of road segments (i.e., $\mathcal{S}$). (c) The segmentation map of road intersections (i.e., $\mathcal{I}$). (d) Updated graph at the next time step $t+1$. Yellow nodes are valid predicted adjacent vertices v_{t+1}^i while red nodes are invalid ones. Valid vertices will be used to update the road network graph. Orange lines are the predicted new edges connecting v_t with v_{t+1}^i. This figure is best viewed in color. Please zoom in for details.

when RNGDet travels along a single road. RNGDet adds v_{t+1}^1 into V and edge (v_t, v_{t+1}^1) into E, and then moves to v_{t+1}^1. RNGDet keeps updating the graph in this way until intersections or broken roads are met. (3) $M > 1$. This indicates that RNGDet encounters road intersections and needs to generate new vertices in multiple directions. RNGDet will update the graph, push all vertices in $\mathcal{V}$ to S, and pop one candidate initial vertex from S.

RNGDet keeps running the above iterations to generate the road network graph vertex-by-vertex. If and only if the candidate initial vertex set S is empty, RNGDet stops and outputs the generated road network graph $G = (V, E)$. The pipeline of the road network graph generation process of RNGDet is visualized in Fig. 3. Some example visualizations are shown in Fig. 4.

E. Loss functions

At each step, RNGDet predicts two segmentation maps as well as coordinates and valid probability of vertices in the next step. Thus, in our experiment three loss functions are utilized to train RNGDet. Suppose the predicted segmentation maps are $\hat{\mathcal{S}}$ and $\hat{\mathcal{I}}$ while the ground-truth segmentation masks are $\mathcal{S}^*$ and $\mathcal{I}^*$, then we have

$$\mathcal{L}_{seg} = L(\hat{\mathcal{S}}, \mathcal{S}^*) + L(\hat{\mathcal{I}}, \mathcal{I}^*), \tag{2}$$

where L is focal loss [32]. Similarly, suppose the ground-truth vertices in the next step $t+1$ are $\{v_j^*\}_{j=1}^M$ and the predictions are $\{\hat{v}_i\}_{i=1}^N$, where $M \leq N$. They can be matched by solving

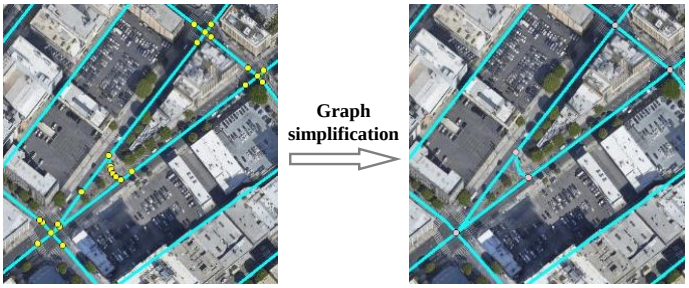


Fig. 5: Visualization of the simplicity of the ground-truth road network graph. In the raw ground-truth graph, there are various kinds of vertices with different degrees (yellow nodes in the left subfigure). For training label generation, we remove vertices whose degrees are two. After filtering, only intersection vertices (degrees larger than 2) and endpoint vertices (degrees are 1) remain (pink nodes in the right subfigure). The road connecting two adjacent vertices is defined as a road segment. The agent must finish exploring a road segment before switching to other road segments. In this way, we can guarantee the correctness of automatically generated labels. This figure is best viewed in color.

a bipartite matching problem through minimizing the following function:

$$\hat{\sigma} = \underset{\sigma}{\operatorname{argmin}} \sum_{i}^{N} \mathcal{L}_{match}(\hat{v}_i, v_\sigma^*), \tag{3}$$

where σ is the index of v^* matched with $\hat{v}_i$, and $\mathcal{L}_{match}$ calculates pair wise Euclidean distance. After the vertex matching, we have the L1 loss as

$$\mathcal{L}_{coord} = \frac{1}{N} \sum_{i}^{N} |\hat{v}_i - v_\sigma^*|. \tag{4}$$

RNGDet also predicts the valid probability p_i of each $\hat{v}_i$, and only vertices with high enough p_i will be used to update the road network graph. The ground-truth value of p_i is 1 if $\hat{v}_i$ is matched with a v_σ^*, otherwise the ground-truth value of p_i is 0 (i.e., $\hat{v}_i$ does not matched with any v^*). Binary cross entropy loss is utilized to optimize p_i:

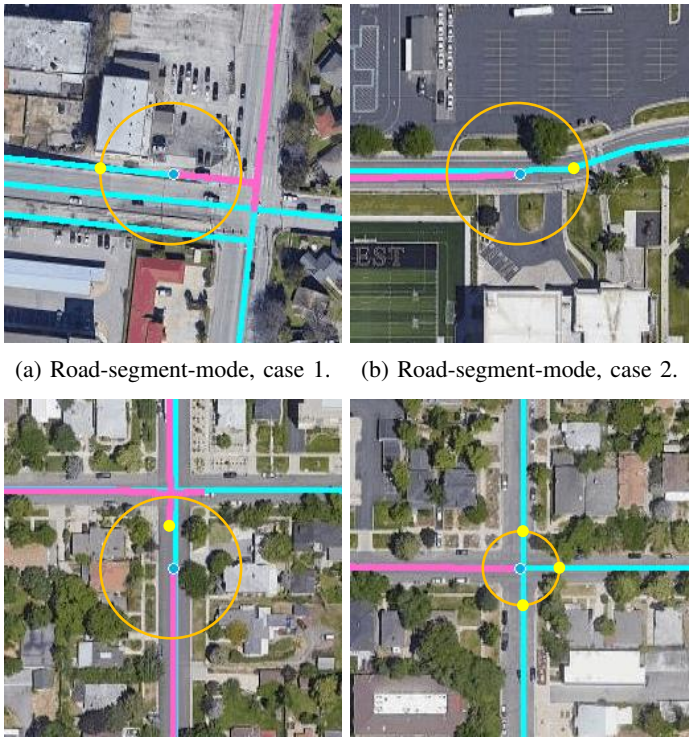
$$\mathcal{L}_{valid} = BCELoss(\hat{p}_i, p_i^*). \tag{5}$$

The final loss function training RNGDet is the weighted sum of the aforementioned loss functions:

$$\mathcal{L} = \mathcal{L}_{seg} + \alpha \mathcal{L}_{coord} + \beta \mathcal{L}_{valid}. \tag{6}$$

F. Training label generation

Based on the ground-truth road network graph and the current location of the agent v_t, we can automatically generate the training label for RNGDet (i.e., $\mathcal{S}^*$,$\mathcal{I}^*$,v_i^* and p_i^*). The segmentation labels can be simply cropped from the ground-truth segmentation masks centering at v_t. For v_i^* and p_i^*, we need to obtain coordinates of the ground-truth next step vertices. To achieve that, we (1) simplify the ground-truth road network graph by removing vertices whose degree is 2 and (2) supervise the agent to explore the graph.



(a) Road-segment-mode, case 1. (b) Road-segment-mode, case 2.

(c) Road-segment-mode, case 3. (d) Intersection-mode.

Fig. 6: Examples of training label generation. Cyan lines are ground-truth road network, pink lines are history road network, blue nodes represents v_t and v_i^* is denoted by yellow nodes. The radiant of the orange circle in (a)-(c) is τ while the radiant of the orange circle in (d) is τ'. τ' is smaller than τ in our experiment. (a) Road-segment-mode. The road ahead is straight. (b) Road-segment-mode. The road ahead has turning points with large enough curvature. (c) Road-segment-mode. The agent connects v_t with a previously generated candidate initial vertex. (d) Intersection-mode. There are three new road segments that are incident to the current intersection point. This figure is best viewed in color.

The raw ground-truth graph consists of various kinds of vertices, such as endpoint vertices (degrees are 1), vertices in the middle of roads (degrees are 2) and intersection vertices (degrees are larger than 2). Among them, vertices whose degrees are 2 are not uniquely defined and could be removed without harming the road network topology. Therefore, for simplicity, we remove these vertices from the ground-truth graph. This process is visualized in Fig. 5. After the graph simplification, there are only endpoint vertices and intersection vertices remaining.

To facilitate the calculation of labels, we define every road connecting adjacent vertices as a road segment. The agent will be either in road-segment-mode or intersection-mode. The road-segment-mode indicates that the agent is currently traveling along a road segment, and there will be only one ground-truth valid vertex at each step (i.e., $\mathcal{V}^* = \{v_1^*\}$). Three examples are visualized in subfigure (a), (b) and (c) in Fig. 6. To prevent the agent from being trapped in an infinite loop, the training label should encourage the agent to move forward (i.e., v_1^* should be far enough from v_t and move to the unexplored part of the current road segment). If the road ahead is straight (i.e., no turning points with large curvature within τ distance to v_t), we select the point whose distance is τ away from v_t as v_1^* (subfigure (a) in Fig. 6). If there are some turning points ahead, we find the turning point that is closest to v_t as v_1^* (subfigure (b) in Fig. 6). If there is some candidate initial vertices within τ distance to v_t, the closest candidate initial vertex is treated as v_1^* (subfigure (c) in Fig. 6). Note that the above algorithm is only utilized to generate the training label and does not affect the graph updating process.

To prevent the agent from being too far from the ground-truth position, we make force corrections by using $\mathcal{V}^*$ instead of $\hat{\mathcal{V}}$ to update the graph if the distance between v_1^* and $\hat{v}$ is larger than a threshold ξ, where ξ could be seen as the level of error tolerance. In this way, the agent has enough flexibility to explore the road network graph without being too far from the right track, which guarantees the quality and efficiency of the sampling process.

When the agent finishes exploring the current road segment, it switches to the intersection-mode and finds incident road segments. For each incident road segment, the point whose distance is τ' away from v_t is defined as the label vertex (subfigure (d) in Fig. 6). Usually there will be multiple v_i^* when RNGDet is in the intersection-mode (i.e., $\mathcal{V}^* = \{v_i^*\}_{i=1}^M$ and $M > 1$).

IV. Experimental Results and Discussions

A. Dataset

In this paper, all the experiments are conducted on the RoadTracer dataset [8]. The dataset contains 300 high-resolution aerial images (60cm/pixel) obtained from Google map and ground-truth road network graphs from OSM (OpenStreetMap). All the data has been converted to the image coordinate system. This dataset covers 40 cities (e.g., Los Angeles and Boston). Each aerial image has 3 channels and is 4096×4096-sized. A city may be composed of multiple aerial images.

B. Implementation

To obtain the dataset to train RNGDet, we run the random walk algorithm to explore the ground-truth road network graphs of training aerial images and generate training samples. During this process, we randomly rotate training samples for data augmentation. At each step, Gaussian noise is added for graph updating in order to make RNGDet more robust. Finally, the training dataset has around 300k samples from different aerial images. We split 10K training samples as the validation data.

In our experiment, we set the crop size of ROI as 256 (i.e., $L = 256$) for the trade-off between effectiveness and efficiency. When we generate the training labels, τ is set as 40 pixels and τ' as 20 pixels. For the transformer, the number of input candidate vertex queries is 10 (i.e., $|Q| = N = 10$). RNGDet is trained with a learning rate as 10^{-4} and a decay rate as 10^{-5} for 50 epochs. We evaluate the performance of RNGDet on the validation set at the end of each epoch. All the experiments are conducted on 4 RTX-3090 GPUs.

C. Baselines

We compare our proposed RNGDet with 2 segmentation-based approaches and 2 graph-based approaches.

- ImprovedRoad [4] (CVPR 2019): ImprovedRoad is one of the state-of-the-art segmentation-based approaches in the past. Orientation information is used to enhance the road segmentation, and it trains an extra refine network to fix incorrect road segmentation predictions.
- SPIN RoadMapper [7] (ICRA 2022): Based on ImprovedRoad, SPIN RoadMapper proposes a graph reasoning scheme to further capture spatial information of the aerial image. Both segmentation-based approaches are trained for 120 epochs. These two approaches usually suffer from poor topology correctness.
- RoadTracer [8] (CVPR 2018): RoadTracer is believed to be the first graph-based approach. It predicts the directions of the vertices in the next step as a multi-class classification problem. However, it has a fixed step size and the training label generation algorithm may produce inappropriate labels.
- VecRoad [11] (CVPR 2020): VecRoad is an improved version of RoadTracer, and is the state-of-the-art graph-based approach. It predicts the distribution of the vertices in the next step, which allows flexible step size. But it is still two-stage and cannot be optimized in an end-to-end manner. Moreover, it may not be able to distinguish vertices that are close to each other.

D. Evaluation metrics

In our experiments, we use three metrics pixel-precision (P-P), pixel-recall (P-R) and pixel-F1-score (P-F) for pixel-level evaluation, three metrics intersection-precision (I-P), intersection-recall (I-R) and intersection-F1-score (I-F) for intersection point evaluation and one metric average path length similarity (APLS) [33] for topology correctness evaluation.

To calculate the pixel-level metric scores, we rasterize the ground-truth graph and the predicted graph as binary images B^* and $\hat{B}$, respectively. For a pixel in B^*, if there has a pixel in $\hat{B}$ and the Euclidean distance between them is smaller than δ, then this pixel is treated as correctly retrieved. Similarly, if a pixel in $\hat{B}$ can find a pixel in B^* within δ distance, then we say this pixel is correctly detected. In this way, P-P, P-R and P-F can be obtained by the following equations:

$$\begin{aligned} \text{P-P} &= \frac{|\{p|\|p,q\| < \delta, \exists q \in B^*, \forall p \in \hat{B}\}|}{|\hat{B}|}, \\ \text{P-R} &= \frac{|\{p|\|p,q\| < \delta, \exists q \in \hat{B}, \forall p \in B^*\}|}{|B^*|}, \\ \text{P-F} &= \frac{2\text{P-P} \cdot \text{P-R}}{\text{P-P} + \text{P-R}}, \end{aligned} \tag{7}$$

where $\|\cdot\|$ calculates the Euclidean distance and $|\cdot|$ is the cardinality of a set. Threshold δ measures the level of error tolerance, and we show the metric scores with δ as 2, 5 and 10 pixels for more comprehensive evaluation.

I-P, I-R and I-F are calculated in a similar way as the aforementioned pixels-level metrics. The only difference is that instead of rasterizing the whole graph into binary images B^* and $\hat{B}$, these three metrics only care about intersection points (i.e., rasterized binary images only contain intersection points). These three metrics evaluate the ability of approaches to detect road intersections. We also show these metric scores in our experimental results with different δ.

APLS measures the similarity of the ground-truth graph G^* and the predicted graph $\hat{G}$. It samples multiple vertex pairs on both graphs and compares the difference between the shortest distance of vertex pairs. APLS is calculated by the following equation:

$$APLS = 1 - \frac{1}{N}\sum min(1, \frac{|L(a,b) - L(a',b')|}{L(a,b)}), \tag{8}$$

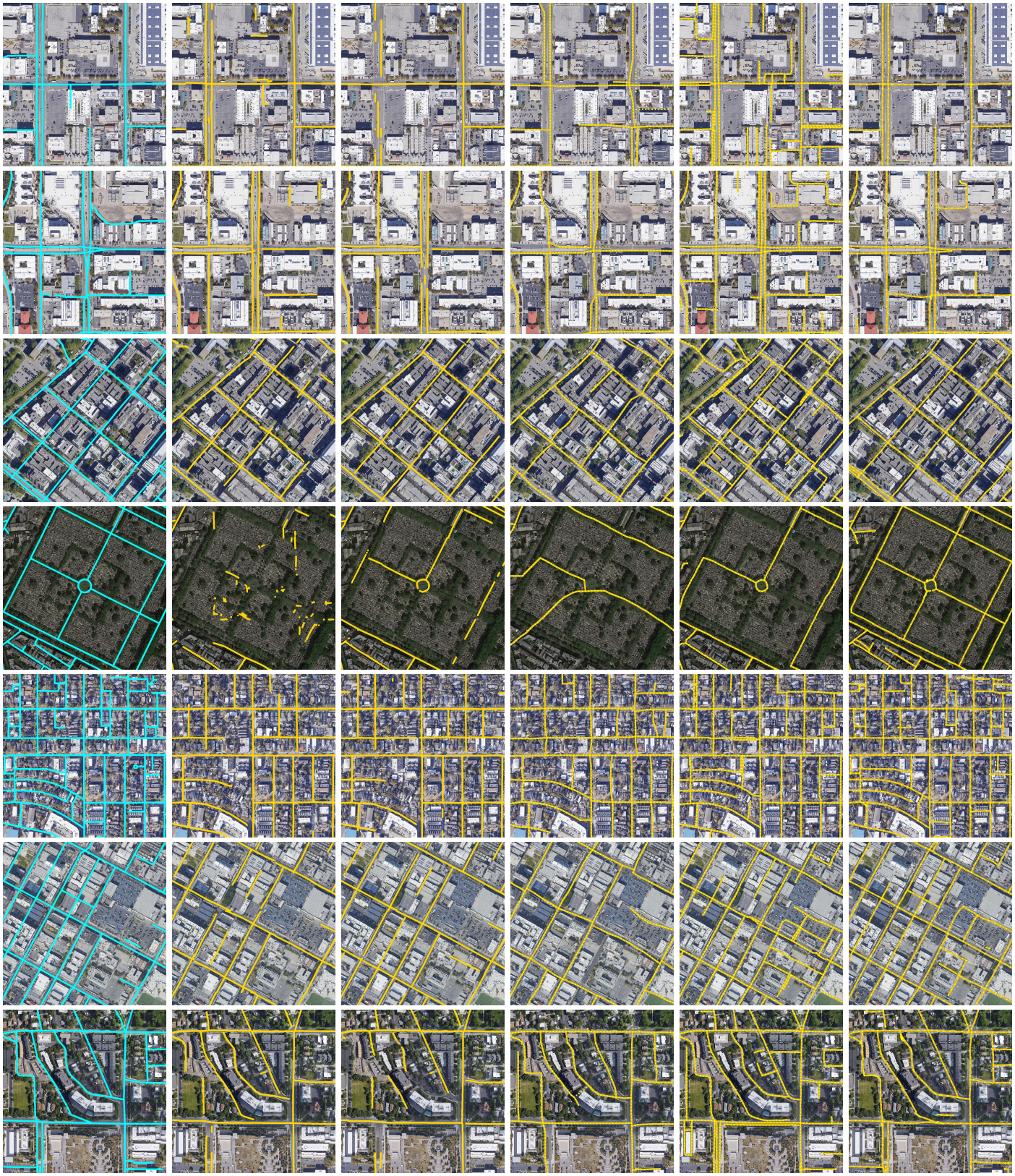
where (a, b) demonstrates a vertex pair sampled from G^* and (a', b') is the corresponding vertex pair sampled from $\hat{G}$. $L(\cdot,\cdot)$ calculates the length of the shortest path between two vertices. Higher APLS indicates better graph similarity and topology correctness.

E. Comparative results

RNGDet is compared with 4 baseline approaches, including two segmentation-based baselines and two graph-based baselines. The quantitative comparison results are shown in Tab. I. In Tab. I, besides baseline approaches, we also evaluate RNGDet with different backbones (i.e., ResNet-34, ResNet-50 and ResNet-101) for more fair and comprehensive comparison. Qualitative visualizations are provided in Fig. 7.

For segmentation results, we first predict the segmentation map, and then binaries it by thresholding. Finally, we run skeletonization algorithms to extract the graph of predicted segmentation maps. From the comparison results, we find that these approaches have good pixel-level results since they directly optimize pixel-level segmentation. However, since they cannot fully utilize spatial and geometric information, they have poor performance on topology correctness. Thus segmentation-based approaches have relatively inferior intersection-level and topology-level performance. Therefore, segmentation-based approaches are not sufficient for our road network graph detection task.

Graph-based approaches directly optimize the graph, thus they present much better results from the topology perspective. RoadTracer has a fixed step length, which makes it unable to handle some scenarios very well, especially when the agent is near road intersections. VecRoad is more powerful due to the use of Res2Net [34] backbone network and flexible step size. However, VecRoad predicts vertices by finding local peaks on heatmaps, so when vertices are very close to each other, it may not be able to correctly detect them. Therefore, it may fail to correctly detect some precise graph structures. Based on the aforementioned approaches and DETR, our proposed RNGDet can directly output the coordinates of vertices, therefore RNGDet can handle more complicated situations and has more superior performance. In this way, RNGDet with ResNet-101 backbone has the best performance during evaluation, gaining 2-5% improvement on almost all metrics scores compared with past state-of-the-art approaches. RNGDet with ResNet-50 backbone has a little bit inferior performance, but it still outperforms VecRoad except APLS. RNGDet with ResNet-34 backbone presents unsatisfactory results, which may be caused by the shallow backbone network and FPN segmentation heads.



(a) Ground truth (b) ImprovedRoad [4] (c) SPIN Road [7] (d) RoadTracer [8] (e) VecRoad [11] (f) RNGDet

Fig. 7: Qualitative demonstrations. We visualize the road network detection results on aerial images. The size of each image is 512×512. (a) Ground-truth road network graph (cyan lines). (b)-(c) The road network graph predicted by segmentation-based approaches (orange lines). These two approaches have poor topology performance such as incorrect disconnections. (d)-(f) The road network graph predicted by graph-based approaches (orange lines as edges and yellow points as vertices). Compared with RoadTracer and VecRoad, RNGDet presents more precise graph structures. For better visualization, lines are widened but they are actually of one-pixel width. This figure is best viewed in color. Please zoom in for details.

TABLE I: The quantitative comparison results. The best results are highlighted in bold font. For all the metrics, larger values indicate better performance.

Approaches	P-P ↑			P-R ↑			P-F ↑			I-P ↑			I-R ↑			I-F ↑			APLS ↑
	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	
ImprovedRoad [4]	68.22	75.70	78.72	47.50	54.09	57.80	56.01	63.10	66.66	29.00	40.62	43.10	23.25	32.60	34.52	23.44	32.75	34.70	41.71
SPIN RoadMapper [7]	**78.53**	**85.82**	**89.02**	54.02	60.28	63.71	64.01	70.82	74.27	**45.27**	59.66	62.08	28.87	37.56	38.93	33.67	43.89	45.56	40.62
RoadTracer [8]	57.49	68.26	74.53	35.09	41.81	46.28	43.58	51.86	57.10	22.79	55.50	**78.93**	13.18	32.11	44.16	16.59	40.45	56.13	49.34
VecRoad [11]	60.87	69.33	73.97	64.91	74.00	78.88	62.83	71.59	76.35	37.49	63.87	68.77	37.51	63.70	68.43	37.12	63.12	67.87	65.69
RNGDet (ResNet-34)	58.33	69.07	73.81	63.80	**75.15**	79.43	60.94	71.98	76.52	32.83	57.02	69.22	34.19	55.96	72.16	35.86	56.39	69.47	55.64
RNGDet (ResNet-50)	62.11	69.54	74.39	66.07	74.92	80.05	64.03	72.12	77.13	36.51	63.97	75.20	39.98	65.48	62.84	38.17	64.72	70.66	63.76
RNGDet (ResNet-101)	65.63	72.31	77.08	**66.42**	75.08	**82.13**	**66.02**	**73.67**	**79.52**	42.37	**65.30**	72.18	**40.40**	**66.50**	**73.23**	**41.23**	**65.68**	**72.47**	**67.88**

TABLE II: The quantitative results for the ablation study. The best results are highlighted in bold font. For all the metrics, larger values indicate better performance. We assess the road segment segmentation ($\mathcal{S}$), the road intersection segmentation ($\mathcal{I}$) and the history graph ($\mathcal{H}$). "-" means there is no valid output.

			P-P ↑			P-R ↑			P-F ↑			I-P ↑			I-R ↑			I-F ↑			APLS ↑
$\mathcal{S}$	$\mathcal{I}$	$\mathcal{H}$	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0	
		✓	37.63	56.80	61.79	55.24	60.12	64.77	44.77	58.41	63.24	25.40	47.03	69.62	20.70	40.93	53.26	24.57	43.68	60.94	50.32
	✓	✓	40.09	55.83	65.97	57.41	63.28	66.21	47.21	59.32	66.09	36.95	56.38	73.72	33.39	58.64	64.30	35.32	57.18	67.35	61.22
✓		✓	59.80	65.34	72.79	62.19	73.83	79.02	60.97	69.33	75.78	32.14	50.86	68.44	32.62	55.04	62.47	32.70	53.83	65.78	59.90
✓	✓		-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
✓	✓	✓	**65.63**	**72.31**	**77.08**	**66.42**	**75.08**	**82.13**	**66.02**	**73.67**	**79.52**	**42.37**	**65.30**	**72.18**	**40.40**	**66.50**	**73.23**	**41.23**	**65.68**	**72.47**	**67.88**

F. Ablation studies

In this section, we study the significance of some components of our network design, including the road segment segmentation $\mathcal{S}$, road intersection segmentation $\mathcal{I}$ and the history map $\mathcal{H}$. All ablation studies are conducted on RNGDet with ResNet-101 backbone. The quantitative results of our ablation studies are shown in Tab. II.

First, we completely remove the segmentation heads from RNGDet. From the evaluation results, we notice that the performance of RNGDet drops a lot because the segmentation heads can provide strong supervision to assist the CNN backbone to extract features of the input. Then, we evaluate RNGDet by only removing the road segment segmentation $\mathcal{S}$. All metrics scores degrade, especially pixel-level metrics scores. After this, we remove the road intersection segmentation $\mathcal{I}$ only, and the intersection-level metrics scores and APLS are severely harmed. Because $\mathcal{I}$ is critical to make RNGDet aware of road intersections, removing $\mathcal{I}$ will lead to incorrect connections near road intersections and make the topology correctness much worse. In this way, the necessity of the segmentation maps including $\mathcal{S}$ and $\mathcal{I}$ is verified.

The history graph $\mathcal{H}$ records the past generated trajectories of RNGDet, which is critical for the agent to make correct decisions. After removing $\mathcal{H}$ from RNGDet, the agent cannot obtain any information of past trajectories, thus it may arbitrarily move forward or backward, making the agent hang around within a small region. Therefore, RNGDet cannot obtain any reasonable results without $\mathcal{H}$, and the importance of $\mathcal{H}$ is proved.

G. Number of candidate queries

Under normal circumstances, the number of queries $|Q|$ should be obviously larger than the maximum number of vertices in the next time step. Usually, the road intersections have at most 5 roads incident with each other. Thus, we try to train RNGDet with 5, 10 and 20 queries and observe the obtained performance. We use RNGDet with ResNet-101 for the experiments. During the experiments, we use metrics P-F ($\delta = 5$), I-F ($\delta = 5$) and APLS to evaluate models.

Based on the results shown in Tab. III, RNGDet with 10 input queries presents the best performance, thus the number of queries is set to 10.

TABLE III: The quantitative results of RNGDet with different numbers of queries ($|Q|$).

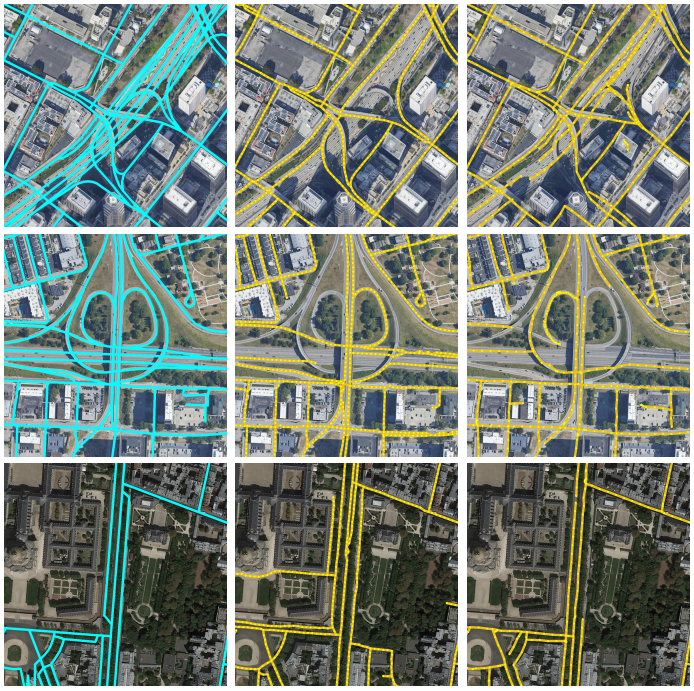
	P-F ↑	I-F ↑	APLS ↑
RNGDet with 5 queries	73.10	61.39	62.78
RNGDet with 10 queries	73.67	65.68	67.88
RNGDet with 20 queries	73.90	64.55	65.31

H. Failure cases

Although RNGDet presents superiority against past approaches, it still cannot handle some very complicated cases, such as occluded overpasses, very well yet. Moreover, RNGDet also suffers from the drifting issue of imitation learning (i.e., when the agent is away from the right track, it may not be able to get back to the correct state) similar to other graph-based approaches, even though it can relieve this problem to some extent. Some example visualizations of failure cases of RNGDet are shown in Fig. 8. These cases could be handled in the future with more powerful backbone networks or training strategies.

V. Conclusion and future works

We proposed here RNGDet, a novel approach to automatically detect the road network graph from aerial images by



(a) Ground truth (b) VecRoad (c) RNGDet

Fig. 8: Qualitative demonstrations of the failure cases. (a) The ground-truth (cyan lines); (b) The result of VecRoad (orange lines for edges and yellow points for vertices); (c) The result of RNGDet (orange lines for edges and yellow points for vertices). The first and second rows show complicated road intersections and overpasses with occlusion. The last row is an example containing severe tree occlusion. RNGDet at this stage cannot handle these cases very well yet. This problem could be relieved in the future by using more powerful backbone networks or training strategies for RNGDet. The figure is best viewed in color. Please zoom in for details

iterations. Taken as input an aerial image, RNGDet could directly output the road network graph with vertices and edges. First, RNGDet predicted a set of candidate initial vertices, and then iteratively generated the road network graph vertex-by-vertex starting from each candidate initial vertex. Due to the use of transformer and deep queries, RNGDet could handle complicated intersection points with arbitrary number of road segments. RNGDet was evaluated on a publicly available dataset and presented the state-of-the-art performance on all evaluation metrics, including pixel-level metrics, intersection-level metrics and topology-level metric APLS. The experimental results demonstrated the superiority of our proposed RNGDet. In the future, we plan to further improve RNGDet by using more powerful backbone networks and training strategies. Besides, we plan to adapt RNGDet to other graph detection tasks, such as road laneline detection and road lane centerline detection.

REFERENCES

[1] Y.-Y. Chiang and C. A. Knoblock, "Extracting road vector data from raster maps," in *International Workshop on Graphics Recognition*. Springer, 2009, pp. 93–105.

[2] V. Mnih and G. E. Hinton, "Learning to detect roads in high-resolution aerial images," in *European Conference on Computer Vision*. Springer, 2010, pp. 210–223.

[3] N. O. Department of Information Technology & Telecommunications (DoITT), "NYC-Planimetrics Database," https://github.com/CityOfNewYork/nyc-planimetrics, 2019.

[4] A. Batra, S. Singh, G. Pang, S. Basu, C. Jawahar, and M. Paluri, "Improved road connectivity by joint learning of orientation and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 385–10 393.

[5] G. Máttyus, W. Luo, and R. Urtasun, "Deeproadmapper: Extracting road topology from aerial images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3438–3446.

[6] A. V. Etten, "City-scale road extraction from satellite imagery v2: Road speeds and travel times," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1786–1795.

[7] W. Gedara Chaminda Bandara, J. M. J. Valanarasu, and V. M. Patel, "Spin road mapper: Extracting roads from aerial images via spatial and interaction space graph reasoning for autonomous driving," *arXiv e-prints*, pp. arXiv–2109, 2021.

[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*, 2018, pp. 4720–4728.

[9] Z. Li, J. D. Wegner, and A. Lucchi, "Topological map extraction from overhead images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1715–1724.

[10] S. He, F. Bastani, S. Jagwani, M. Alizadeh, H. Balakrishnan, S. Chawla, M. M. Elshrif, S. Madden, and M. A. Sadeghi, "Sat2graph: road graph extraction through graph-tensor encoding," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIV 16*. Springer, 2020, pp. 51–67.

[11] Y.-Q. Tan, S.-H. Gao, X.-Y. Li, M.-M. Cheng, and B. Ren, "Vecroad: Point-based iterative graph exploration for road graphs extraction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8910–8918.

[12] D. Belli and T. Kipf, "Image-conditioned graph generation for road network extraction," *NeurIPS 2019 workshop on Graph Representation Learning*, 2019.

[13] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision*. Springer, 2020, pp. 213–229.

[14] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.

[15] 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)*, 2018, pp. 801–818.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[17] J. Liang, N. Homayounfar, W.-C. Ma, S. Wang, and R. Urtasun, "Convolutional recurrent network for road boundary extraction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9512–9521.

[18] Z. Xu, Y. Sun, and M. Liu, "Topo-boundary: A benchmark dataset on topological road-boundary detection using aerial images for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7248–7255, 2021.

[19] Z. Xu, Y. Liu, L. Gan, X. Hu, Y. Sun, L. Wang, and M. Liu, "csboundary: City-scale road-boundary detection in aerial images for high-definition maps," *arXiv preprint arXiv:2111.06020*, 2021.

[20] N. Homayounfar, W.-C. Ma, S. Kowshika Lakshmikanth, and R. Urtasun, "Hierarchical recurrent attention networks for structured online maps," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3417–3426.

[21] N. Homayounfar, W.-C. Ma, J. Liang, X. Wu, J. Fan, and R. Urtasun, "Dagmapper: Learning to map by discovering lane topology," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2911–2920.

[22] Q. Li, Y. Wang, Y. Wang, and H. Zhao, "Hdmapnet: A local semantic map learning and evaluation framework," 2021.

[23] S. He and H. Balakrishnan, "Lane-level street map extraction from aerial imagery," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2080–2089.

[24] Y. Zhou, Y. Takeda, M. Tomizuka, and W. Zhan, "Automatic construction of lane-level hd maps for urban scenes," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6649–6656.

[25] Z. Xu, Y. Sun, and M. Liu, "icurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1097–1104, 2021.

[26] Z. Xu, Y. Sun, L. Wang, and M. Liu, "Cp-loss: Connectivity-preserving loss for road curb detection in autonomous driving with aerial images," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1117–1123.

[27] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[28] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[29] Y. Xu, W. Xu, D. Cheung, and Z. Tu, "Line segment detection using transformers without edges," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4257–4266.

[30] Y. B. Can, A. Liniger, D. P. Paudel, and L. Van Gool, "Structured bird's-eye-view traffic scene understanding from onboard images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 661–15 670.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[32] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[33] A. Van Etten, D. Lindenbaum, and T. M. Bacastow, "Spacenet: A remote sensing dataset and challenge series," *arXiv preprint arXiv:1807.01232*, 2018.

[34] S. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. H. Torr, "Res2net: A new multi-scale backbone architecture," *IEEE transactions on pattern analysis and machine intelligence*, 2019.