UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Karl-Johan Pilve

Lane Centerline Detection from Orthophotos using Transformer Networks

Master's Thesis (30 ECTS)

Supervisors: Tambet Matiisen, MSc
Edgar Sepp, MSc

Tartu 2024

Lane Centerline Detection from Orthophotos using Transformer Networks

Abstract:
A high-definition (HD) map containing accurate lane network data is essential for autonomous driving. Yet, the creation of an HD map for a larger area is often a time-consuming and labor-intensive process. Recently, neural networks utilizing the transformer architecture have shown promising results in the field of computer vision. One such example is RNGDet, which uses aerial images to iteratively generate a road network graph. This thesis explores the viability of fine-tuning the RNGDet model with lane network data to generate a lane network graph for the entire city of Tartu based on high-resolution orthophotos. The obtained results show that RNGDet could be used in principle to generate a lane network graph. However, the model's architecture likely requires more extensive modifications to accommodate the differences between road and lane network data. Since the orthophotos may not contain all the necessary information for correct lane network generation, the best-performing model used orthophotos with rasterized raw GPS trajectories, which were collected by a survey vehicle. The obtained results also indicate that additional training data is needed to increase the quality of the generated lanes enough that they could be used in HD maps.

Keywords:
Computer vision, High-definition maps, Transformer, Lane network graph detection

CERCS:
T111 - Imaging, image processing; P176 - Artificial intelligence

Sõiduraja keskjoonte tuvastamine ortofotodelt transformeriga

Lühikokkuvõte:
Sõiduradade andmeid sisaldav täppiskaart on isejuhtivate sõidukte opereerimisel väga oluline komponent. Samas on täppiskaardi loomine suurema ala kohta sageli ajakulukas ning töömahukas protsess. Hiljuti on transformeri arhitektuuri kasutavad tehisnärvivõrgud näidanud paljulubavaid tulemusi masinnägemise valdkonnas. Üks selline näide on RNGDet, mis genereerib iteratiivselt teedevõrgu graafi aerofotode põhjal. Käesolevas töös uuritakse võimalust peenhäälestada RNGDet mudelit sõiduradade andmetega, et genereerida kogu Tartu linna kattev sõiduradade graaf kasutades kõrge resolutsiooniga ortofotosid. Töös saadud tulemused näitavad, et RNGDeti on põhimõtteliselt võimalik kasutada sõiduradade graafi genereerimiseks. Samas oleks mudeli arhitektuuris vaja tõenäoliselt teha suuremaid muudatusi, et võtta arvesse teedevõrgu ja sõiduradade andmete vahelisi erinevusi. Kuna ortofotodel ei pruugi alati olla kogu vajaliku informatsioon sõiduradade õigesti genereerimiseks, siis kõige paremaid tulemusi andis mudel, mis kasutas ortofotodele rasterdatud mõõdistussõiduki poolt kogutud töötlemata GPS trajektoore. Saadud tulemused näitavad veel, et täppiskaardi jaoks sobiva kvaliteediga sõiduradade genereerimiseks oleks vaja koguda täiendavaid treeningandmeid.

Võtmesõnad:
Masinnägemine, Täppiskaardid, Transformer, Sõiduradade graafi tuvastamine

CERCS:
T111 - Pilditehnika; P176 - Tehisintellekt

Contents

1 Introduction

High-definition maps (HD maps) are an integral part of autonomous driving. HD maps can be referred to as the most intelligent sensor of the autonomous vehicle, as they offer unlimited range and enable the vehicle to maintain adequate situation awareness in occluded areas and during poor weather conditions [18]. Additionally, HD maps are essential for localization, journey planning, and trajectory calculation [26]. HD maps can also be used to predict the trajectories of other vehicles and pedestrians [17] and to anticipate obstacles or other potential hazards. Common characteristics of an HD map are lane-level accuracy, at least centimetre-level precision, and highly detailed semantic information.

Creating high-definition maps can be costly and a labor-intensive process [18, 24]. Boa *et al.* [5] describes the general HD map creation pipeline. First, data must be collected with survey vehicles, which are usually equipped with GPS, IMU, LiDAR, and cameras. The following steps in HD map creation are point cloud map generation to create a geometric HD map layer used for localization, and the extraction of lane network and other semantic features for the semantic HD map layer, which is mainly used for journey planning and navigation. Finally, the extracted semantic features can be integrated into HD map frameworks, such as Lanelet2 [38] or OpenDRIVE [1], to ensure compatibility with existing autonomous driving software.

The feature extraction process usually involves a large amount of labor by human annotators, who use the data gathered by survey vehicles as well as aerial images. Traditionally, the task of lane network extraction was automated with segmentation-based methods, which can yield good pixel-level results but suffer from network connectivity and topological correctness issues. Recently, with advancements in deep learning algorithms, graph-based road and lane generation methods have shown comparable or even greater accuracy than segmentation-based methods. Some state-of-the-art graph generation models have successfully incorporated the transformer [46] into their architecture.

The transformer is a deep learning architecture which has shown exceptional results in natural language processing, computer vision, and multi-modal processing. Since transformers are intended for predicting sequential data with non-uniform length, they should be ideal for directly predicting road or lane network graphs, where the individual roads or lanes could be represented as a sequence of coordinates. Additionally, the attention mechanism should allow the transformer to focus on important areas of the aerial images and relevant past predictions. Therefore, the task of lane generation from aerial images is similar to an image captioning task, where visual transformers have dominated the benchmarks.

The Autonomous Driving Lab (ADL) at the University of Tartu has mapped the lanes on a few test roads in Tartu and Tallinn. Each lane on ADL's HD map is represented by its centerline, a sequence of two-dimensional points at the center of the lane's drivable area. Individual lane geometries are connected if they contain a point with exactly the

same coordinates. The mapping process at the ADL relies mostly on human annotators, although there have also been experiments with heuristics-based algorithms. The human mappers have drawn the lanes based on the survey vehicle's GPS trajectories and geometrically corrected aerial images, named orthophotos. As previously mentioned, many methods are available to automate the lane network creation process, and the ADL is actively researching which method could deliver the most accurate results.

This thesis explores the viability of generating the lane network from orthophotos with a deep-learning model utilizing the transformer architecture. Our main contributions are as follows:

- The creation of Estonian road and lane detection datasets. As part of creating the lane detection dataset, a large area in Tartu was mapped to supplement the ADL's existing lane data.
- Training a well-performing road detection model with the RNGDet [49, 51] deep-learning network.
- Modifying the RNGDet for the lane detection task and fine-tuning the road detection model with multiple lane detection datasets.

Excluding the introduction, conclusion, and acknowledgments, this thesis is divided into five sections. Section 2 overviews existing road or lane network extraction methods and the transformer architecture. The models used for training are described in Section 3. Section 4 gives an overview of the training datasets and their creation process. Section 5 presents the quantitative and qualitative results of the trained models. Section 6 discusses the overall results, limitations of aerial images, and post-processing steps required to make the model's output suitable for HD maps.

This thesis was written using the Overleaf[1] text editor. The text was checked with the Grammarly[2] writing assistant to catch typos and other grammatical errors.

[1]Overleaf: https://www.overleaf.com
[2]Grammarly: https://www.grammarly.com

2 Background

This section presents the background information related to detecting lane networks with a transformer. In the first subsection, past work on the task of road or lane extraction from aerial images is discussed. The second subsection describes the transformer architecture.

2.1 Road or lane network extraction

Road and lane detection from bird's-eye-view (BEV) images is an extensively researched topic, with the first attempts dating back to the 1970s [3]. Earlier works mainly focused on road network detection, as high-resolution aerial imagery was not widely available and usually employed a segmentation-based approach. However, with the adoption of deep learning algorithms in the 2010s, graph-based methods and methods extracting BEV features from data gathered by survey vehicles have also shown promising results.

2.1.1 Segmentation-based methods

Segmentation-based methods [4, 8, 13, 14, 19, 36, 37, 52] usually generate the road or lane network in two steps. First, the probabilistic segmentation map of the road or lane network is predicted. Secondly, the network graph is extracted with post-processing algorithms. Traditional segmentation algorithms used geometric-stochastic models [6], which predicted the segmentation mask based on the color intensity, texture, and assumption about the road width and trajectory.

Since the mid-2010s, convolutional neural networks (CNNs) have started to replace the stochastic models for the segmentation map prediction task. Models utilizing powerful CNNs, such as U-Net [41], ResNet [22], and DeepLabv3+ [12], have demonstrated excellent accuracy and robustness in predicting the segmentation map. However, extracting the road network graph from the predicted segmentation map is still challenging. Classical skeletonization algorithms have struggled to produce a connected road network in areas occluded by trees, buildings, or intense shadows or in areas of high road intensity [36, 49].

DeepRoadMapper, proposed by Máttyus *et al.* [36], utilizes the A* search algorithm to fill gaps in the network and connect disconnected segments. DeepRoadMapper uses a ResNet-inspired CNN with a custom loss function developed by the authors to generate the segmentation maps. Then, the skeletonization algorithm is applied, and small curves and loops are removed from the road graph. Next, segments connecting leaf nodes to other nodes are generated if their distance is less than 50m. Each connecting segment is weighted based on the likelihood of it being a road. Finally, the likeliest segment is chosen with the A* algorithm.

Batra *et al.* [8] proposed a connectivity task called Orientation Learning, inspired by how humans connect disjointed road segments. The Orientation Learning approach

calculates an orientation vector for each consecutive point pair in a road segment. A convolutional model was trained to fill in the gaps in the road network extracted with a segmentation model by tracing the connecting segment based on the calculated orientation vectors. The authors use a multi-task learning approach [27] to combine the orientation and the segmentation tasks into one CNN model. However, the model still struggled with complex intersections, minor roads, and parking lots, which prompted the authors to employ an additional connectivity refinement step in the final stage of their road detection approach.

SPIN Road Mapper, proposed by Bandara *et al.* [4], introduced a Spatial and Interaction Space Graph Reasoning (SPIN) module, which performs reasoning over graphs built on spatial space and a projected latent interaction space from feature maps. Graph reasoning on spatial space improves the connectivity between road segments, and reasoning on interaction space helps distinguish roads from other topographic elements. This improves the segmentation accuracy and increases the convergence rate of the network by half. SPIN module is intended to be easily added into a CNN model after a convolutional block. The authors employ a SPIN pyramid in their experiments to perform graph reasoning at multiple scales and increase the network's receptive field. The proposed SPIN Road Mapper is depicted in Figure 1.

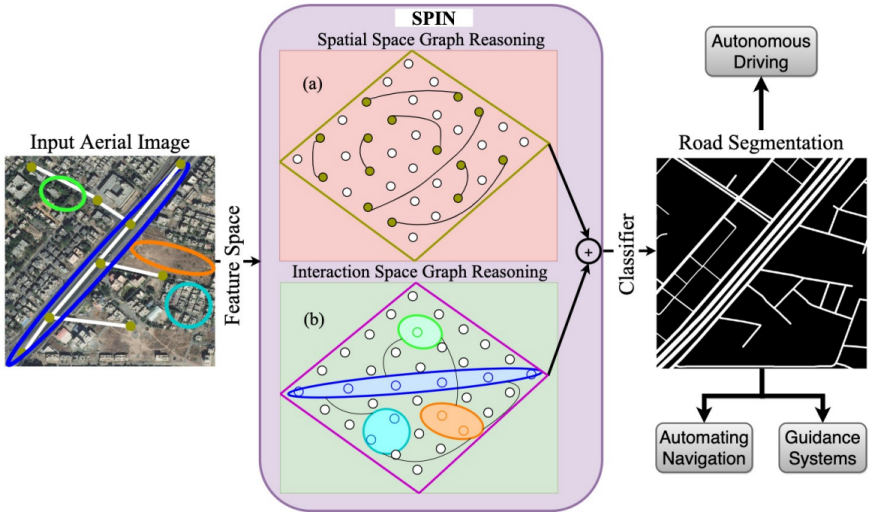


Figure 1. An overview of the SPIN Road Mapper. Graph reasoning is done (a) on spatial space to improve the road connectivity and (b) on a projected latent interaction space to delineate roads from other topographies [4].

2.1.2 Iterative graph-generation methods

Although there have been many proposals to increase the performance of the segmentation-based method in complex and occluded zones, none have produced robust results in these problematic areas. A fundamental problem with the segmentation-based method is that it only utilizes local information at the segmentation map generation stage. At the same time, the actual network graph is constructed in a post-processing stage, which often relies more on heuristics than machine learning [7].

Iterative graph-generation methods [7, 9, 31, 42, 49, 51] use deep learning models to generate a road or lane network graph directly from the aerial image. The method starts from a given vertex on the network and continuously predicts the following vertices based on the current vertex, the surrounding area, and past predictions. The algorithm stops after exploring the whole input image. Road segments are guaranteed to be connected in each network component, and the maximum number of disconnected components cannot be greater than the number of initial vertices. However, due to the iterative nature of the method, inference can be considerably slower compared to the segmentation-based methods. The method is also prone to drifting away from the ground truth during the generation of a single road segment.

RoadTracer, proposed by Bastani *et al.* [7], is considered to be the first road network extraction model utilizing the iterative graph-generation method. RoadTracer uses a CNN decision function at each vertex to predict the next vertex, to stop, or to walk back along the already generated graph to some other vertex. The CNN makes its decision based on a 4-channel $n \times n$ window around the current vertex. The first three channels of the window are the RGB values from the aerial image, and the fourth channel contains information about the already generated graph. The model is initialized with a single vertex obtained from the ground truth. RoadTracer outperformed many segmentation-based methods and produced results comparable to those of DeepRoadMapper. RoadTracer showed strong topology performance while doing poorly with intersection detection.

VecRoad, proposed by Tan *et al.* [42], introduced a flexible step size while predicting the next vertex. This improves the detection accuracy of intersection points and other non-trivial vertices (see Figure 2). VecRoad uses segmentation-cues, i.e., segmentation maps predicting the location of road centerlines and intersections, to generate a probabilistic distribution of adjacent vertices of the current vertex. Also, the authors extracted starting vertices from the local peaks of these segmentation maps. VecRoad outperformed RoadTracer, but the prediction of the segmentation maps still requires post-processing algorithms, which make end-to-end usage hard.

Since the graph-generation models RNGDet [49] and RNGDet++ [51] are used for lane network detection in this thesis, their architecture is described in Section 3.

(a) (b) (c)

Figure 2. The flexible step size of VecRoad improves the accuracy of non-trivial vertices, such as (a) intersections, (b) road ends, and (c) points linking road segments [42].

2.1.3 Whole graph-generation methods

Whole graph-generation methods [23, 48] try to simultaneously predict the entire road or lane network graph. With this method, a neural network encodes aerial images into vector representations and then decodes these representations into a road or lane network graph. This method should have a much greater receptive field than the segmentation-based and iterative graph-generation methods. However, directly predicting the whole graph is a much more difficult task for the neural network because the relationships between vertices and the pixel values of the input image can be somewhat ambiguous.

Sat2Graph, presented by He *et al.* [23], is believed to be the first application of the whole graph-generation method on the road network extraction task. Sat2Graph employs a novel graph-tensor encoding scheme (GTE), which encodes the road network into a 19-dimension tensor. This enables the training of a simple neural network, which predicts graph structures from the input image. Sat2Graph can match the state-of-the-art methods, but the authors also discuss two problems regarding the Sat2Graph method. Firstly, the heuristics-based GTE might not be able to represent all road network graphs, and secondly, the GTE produces isomorphic encodings, which makes supervised training difficult.

csBoundary, proposed by Xu *et al.* [48], detects road boundaries from aerial images. The model outputs a continuous city-wide road-boundary graph based on the input image. csBoundary uses a novel adjacency matrix prediction network (AfANet) based on the transformer architecture [46]. According to the authors, csBoundar is the first model that uses the transformer for automatic HD-map annotation. Before the input image is sent to the AFANet, a feature pyramid network (FPN) [33] is used to predict the segmentation map and the keypoint segmentation map. These maps are added to the original aerial image as additional channels. Separately, a heuristics-based algorithm extracts the key-points from the keypoint map, which are also given to the AfANet. Finally, the adjacency

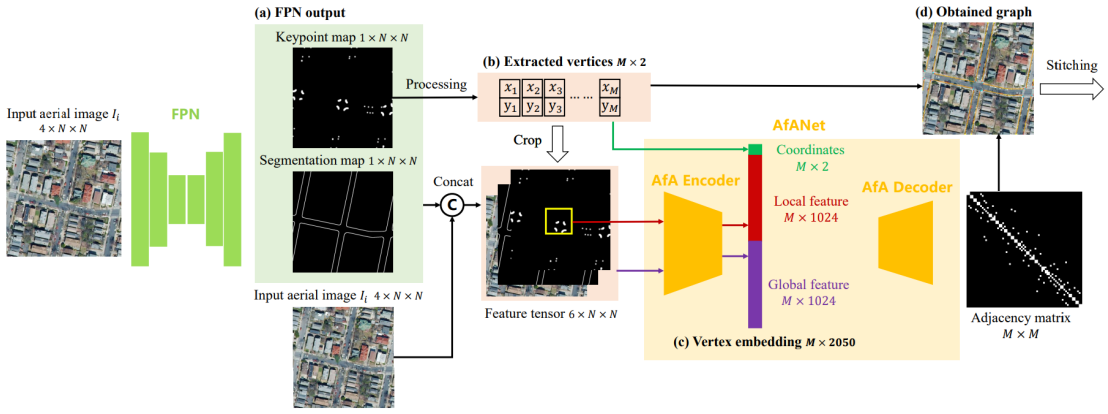


Figure 3. Overview of the csBoundary method [48]. (a) The keypoint and segmentation maps are predicted based on the 4-channel input aerial image. These maps are concatenated with the aerial images to form a 6-channel feature tensor. (b) Vertex coordinates of the resulting graph are extracted from the keypoint map. (c) In AfANet, a global feature vector is calculated based on the 6-channel tensor, and a local feature vector is calculated based on a small area around each extracted vertex. The vertex coordinate vector and the feature vectors are concatenated into the vertex embedding. (d) The adjacency matrix is predicted based on the vertex embeddings, and the road boundary graph is constructed from the extracted vertices and the adjacency matrix.

matrix of extracted vertices is obtained from the AfANet, and the road-boundary graph is constructed from the adjacency matrix and the extracted keypoints. Figure 3 gives an overview of the csBoundary pipeline. csBoundary showcased comparable performance to other state-of-the-art methods.

2.1.4 Vehicle sensors based methods

Many HD map creation methods use data gathered by survey vehicles instead of aerial images. The main advantage of survey vehicle data is the availability of rich multi-source geometric information with millimeter-level precision, making the method more suitable for lane network detection. Even if vehicle sensor data is not used by a road or a lane detection method, the literature on vehicle sensors-based methods contains many interesting approaches for converting BEV features into a road or a lane network graph. Some past works have tried to extract the road network from 3D LiDAR point clouds [25, 45], others have used images from vehicle cameras [10], cameras and point clouds [29, 32, 50] or fused the survey vehicle's data with aerial images to generate BEV features from which a road network graph is predicted [30, 35].

HDMapNet, proposed by Li *et al.* [29], is a semantic map learning method that predicts the BEV lane segmentation map from six vehicle camera images, from a point cloud, or both. The camera images are encoded into a perspective view feature map with a CNN and then transformed into BEV features with a natural view transformer. The 3D point cloud is divided into multiple pillars, and BEV feature maps are predicted from the pillar-wise features with a variant of the PointNet model [39]. Based on the features obtained from the camera images and point cloud, a fully convolutional BEV decoder predicts the segmentation map, instance embedding, and each lane's direction value. Finally, the resulting segmentation map is vectorized with a skeletonization algorithm using the instance embedding and direction values to produce the lane network graph.

BoundaryNet, presented by Ma *et al.* [35], is a road boundary detection model that makes its predictions based on vehicle trajectory data, point clouds, and satellite imagery. The model consists of four sections. In the first section, heuristics-based algorithms extract noisy road boundaries containing gaps from the point cloud. The noise is removed with a modified U-net network, and the gaps are closed with a CNN in the second section. The third section contains the D-LinkNet model [53], which extracts road centerlines from the satellite images. Road centerlines assist the further refinement of boundaries in the fourth section with a conditional deep convolutional generative adversarial network (c-DCGAN).

2.2 Transformer

The Transformer is a deep learning model architecture introduced in the paper "Attention is All You Need" by Vaswani *et al*. [46]. First developed for machine translation, it has quickly become the state-of-the-art approach in the natural language processing (NLP) domain. The best-performing large language models (LLMs), including BERT [15], OpenAI's GPT series [40], and Meta's LLaMA family [44], all use the Transformer in their architecture. The Transformer's ability to handle variable-length sequential data and its parallelization capabilities have led to its adoption in other machine learning domains, such as computer vision, audio, robotics, and multi-modal processing.

A vital component of the Transformer is the multi-head self-attention (MSA), which uses the attention mechanism to calculate the relationship between each element in the input sequence. MSA allows the model to selectively focus on different parts of the input when producing an output sequence. The amount each element should attend to some other element is represented with an attention score. The attention score is calculated with three vectors referred to as query (q), key (k), and value (v). These vectors are created by multiplying the embedding x_i of each element in the input sequence by weight matrices W_Q, W_K, W_V that are trained during the training process:

$$q_i = x_i \times W_Q \qquad k_i = x_i \times W_K \qquad v_i = x_i \times W_V \tag{1}$$

Attention can be calculated on the embedding matrix $X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$ of the whole input sequence with the following formula:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

where Q, K, and V are the matrices consisting of queries q_i, keys k_i, and values v_i of the whole sequence, and d_k denotes the dimension of the key vector. This particular attention is called scaled dot-product attention (Figure 4) because the attention weights are computed from the dot product of query and key vectors and scaled down by $\sqrt{d_k}$ to stabilize gradients during the training. Finally, the attention weights are normalized with softmax.

The original Transformer paper feeds the input embeddings into h different attention-heads that each have independently trained weight matrices for queries, keys, and values. The outputs from each head are concatenated and linearly projected into the output embedding of the MSA layer. The paper argues that multiple heads help the model focus on multiple positions simultaneously and capture different aspects or representations of the input data. Additionally, the calculations for each attention head can be performed in parallel, which speeds up training. Multi-head attention is depicted in Figure 4.

The original implementation of the Transformer uses an encoder-decoder structure without any convolutional or recurrent units (Figure 5). The task of the encoder is to take the vector representations of the input sequence, i.e., input embeddings, and transform them into encodings that can be passed to the decoder's encoder-decoder attention blocks.

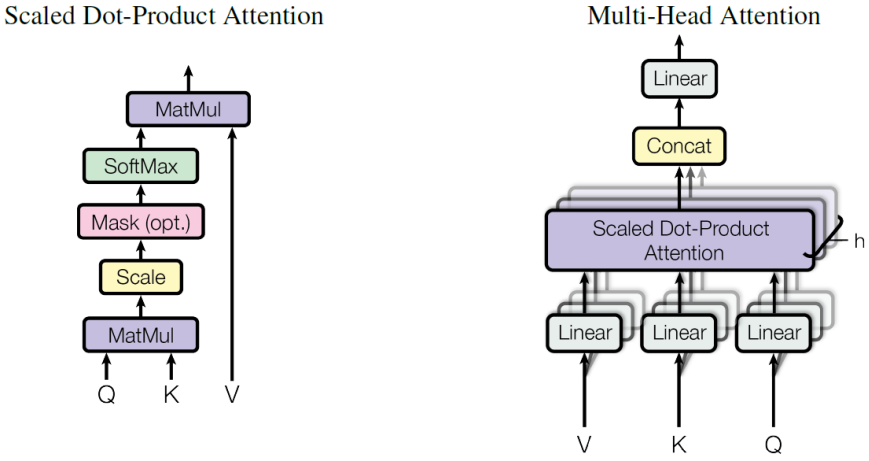


Figure 4. The calculation of scaled dot-product attention is depicted on the left, and the structure of multi-head attention is shown on the right [46].

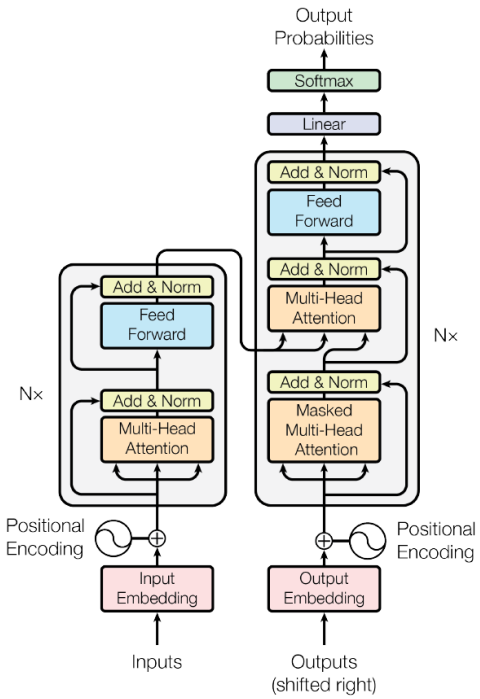


Figure 5. Architecture overview of the Transformer [46].

The encoder consists of N number of identical encoder layers. Each layer has two sub-layers – multi-head self-attention and a feed-forward network. A residual connection exists around both sub-layers, followed by layer normalization [2].

The decoder iteratively generates the output sequence elements based on the previous steps' output embeddings and encodings from the encoder. The decoder has N number of identical decoder layers, and each layer has three sub-layers – masked multi-head self-attention, encoder-decoder attention, and a feed-forward network. The masked multi-head self-attention is similar to the attention block in the encoder – it can only attend to the preceding elements in the input sequence. The encoder-decoder attention takes the queries from the previous sub-layer, and the keys and values come from the output of the encoder. Similar to the encoder, each decoder sub-layers has a residual connection and a layer normalization. Finally, the output of the decoder is passed through a linear and a softmax layer to obtain the probabilities for the next element in the sequence.

Since the Transformer model does not have any information about the order of the elements in the input sequence, a vector representing the positional information of the elements is added to each input embedding before they are passed to the Transformer layers. These vectors are called positional encodings, and multiple options exist to create them [21]. The original Transformer used sine and cosine functions of different frequencies to calculate the positional encodings.

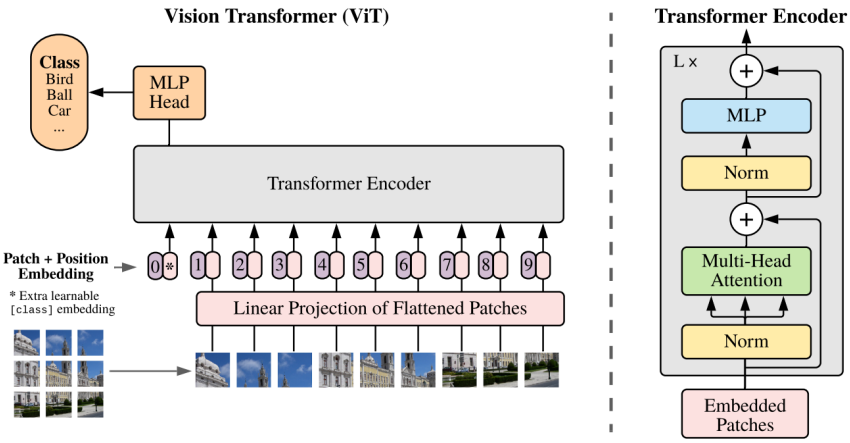


Figure 6. Architecture overview of the Vison Transformer (ViT) [16].

2.2.1 Vision transformer

Transformers designed for computer vision are named Vision Transformers (ViT). Many variants exist, but the original architecture was proposed in a 2020 paper by Dosovitskiy et al. [16]. It was the first transformer to achieve state-of-the-art performance on image classification tasks while not relying on CNNs to extract and process the relevant features from the input image. Since the release of the original ViT, several Vision Transformer architectures have been proposed, including Swin Transformer [34], DeiT [43], and Pyramid Vision Transformer (PVT) [47].

The original Vision Transformer follows closely the design of the original Transformer proposed by Vaswani *et al.* [46]. Since the transformer takes only 1-dimensional sequences as input, the input images are divided into non-overlapping square patches, which are then flattened to 1-dimensional vectors with a trainable linear projection. These flattened patches are called patch embeddings and serve as input tokens for the transformer. Learnable position embeddings are added to the patch embeddings to retain spatial information. Also, a special embedding, used for predicting the class label, is added to the input sequence.

The encoder layers of the ViT are the same as those in the original Transformer: multi-head attention, feed-forward network, and layer normalization. The only difference is that in ViT, the normalization is done before the attention and the feed-forward network. The output from the last encoder layer is passed through a classification head to predict the label of the input image. The architecture of the ViT is shown in Figure 6.

Although the original ViT matched the state-of-the-art on many image classification datasets, it performed poorly on other computer vision tasks, such as object detection and segmentation. One reason might be that ViT has much less image-specific inductive

bias than CNNs. The 2-dimensional structure of the image is implicitly included in CNN's architecture, which allows it to recognize spatial patterns much more easily. Additionally, the 16×16 pixel patch size of the original ViT makes it hard to learn pixel-level information. Many architectures developed from the original ViT, such as the Swin Transformer, aim to fix these shortcomings.

2.2.2 DETR

A popular approach for detecting objects from images is to combine the transformer with CNNs to leverage the strengths of both designs in tandem. DEtection TRansformer (DETR) proposed by Carion *et al.* [11] is a simple end-to-end object detection model that processes the input images with CNNs before using the transformer to predict the location and class label of the objects in the image.

DETR treats the object detection task as a set prediction problem and employs a unique loss function that performs bipartite matching between predicted and ground-truth objects. Since DETR predicts exactly N number of objects where N is larger than the set of ground truth objects, it can be hard to score the predictions with respect to the ground truth. The aforementioned loss ensures that each predicted object is associated with the most appropriate ground-truth object, and the matching can be efficiently calculated with the Hungarian algorithm [28].

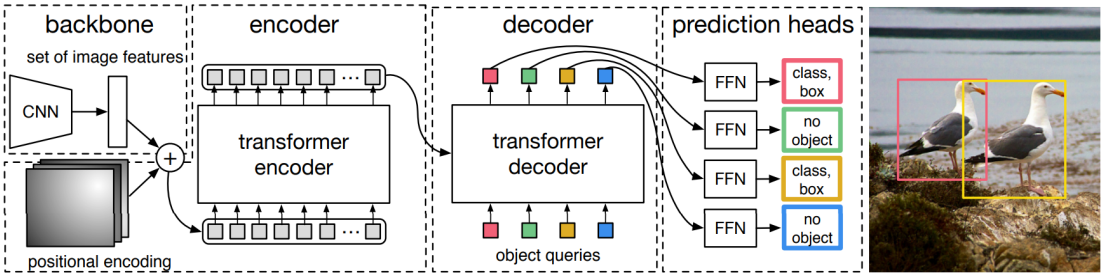


Figure 7. Architecture overview of the DEtection TRansformer (DETR) [11].

The overall architecture of DETR is relatively simple (Figure 7). It consists of a CNN, an encoder-decoder transformer, and a feed-forward network (FFN). First, the input image is passed through a conventional CNN backbone, which reduces the spatial information of the image into a lower-resolution activation map. Before the activation map is fed into the encoder, it is flattened, and fixed positional encodings are added. The encoder features a standard architecture with a multi-head self-attention block and a feed-forward network. The decoder is also similar to the design in the original transformer paper, the only difference being that it decodes N objects in parallel instead of predicting one element at each step. The decoder is also permutation-invariant, meaning the order of the

decoder input embeddings does not matter, and they have to be distinct from each other to produce different results. Thus, these input embeddings are learned positional encodings and are referred to as object queries. The decoder produces an output embedding from N number of object queries.

From the decoder output, an FFN predicts the bounding box's center coordinates, height, and width. A simple linear layer with a softmax function predicts the class label. Since the transformer always predicts N number of objects, often more than in the input image, a special class label is used to denote "empty" detections with no actual objects of interest.

In the original paper, DETR showed comparable results to the optimized Faster R-CNN baseline on the COCO dataset and outperformed the Faster R-CNN with larger objects. It has a simple architecture, meaning the individual components, like the CNN, transformer, and FNN, can easily be replaced with improved versions. Other advantages of DETR include parallel processing and the effective self-attention mechanism to capture complex spatial relationships. However, DETR's performance is weaker on small objects, and its training can be more computationally expensive.

3 Models

This thesis uses the RNGDet [49] and RNGDet++ [51] models to experiment with lane centerline detection from orthophotos. These models were developed to detect roads from aerial images, but the intention is to fine-tune them on lane network data. RNGDet and RNGDet++ models were selected for the following reasons:

- RNGDet and RNGDet++ are only among the few road detection models that incorporate the transformer into their architecture. One objective of this thesis is to assess the viability of the transformer architecture for HD map creation based on aerial images.
- RNGDet and RNGDet++ models make predictions only based on aerial images. Many previous works have additionally used LiDAR data or images from vehicle cameras. Collecting or purchasing this kind of data for large areas can be expensive, but up-to-date, high-quality orthophotos can be obtained from the Estonian Land Board free of cost.
- The models' code is open-source[3]. Therefore, there is no need to implement the model and the data processing pipeline from scratch.

This section gives a detailed description of RNGDet and RNGDet++. The first subsection gives an overview of the architecture of RNGDet. The second subsection discusses the additional features of the RNGDet++ model compared to the original RNGDet.

3.1 RNGDet

RNGDet was proposed by Xu *et al.* in the 2022 paper "RNGDet: Road Network Graph Detection by Transformer in Aerial Images" [49]. It uses an iterative graph approach to detect the road network from aerial images. It showed comparable results to the state-of-the-art on multiple public benchmarks.

3.1.1 Model architecture

RNGDet is developed on DETR's architecture. The model takes an aerial image I_A as an input and outputs a road network graph $G = (V, E)$, where V is a set of vertices representing the coordinate points on the road network, and E is a set of road segments as edges. RNGDet consists of two CNN backbones, two FPN heads, a transformer encoder, and a transformer decoder. The architecture of RNGDet is depicted in Figure 8.

[3]RNGDetPlusPlus: https://github.com/TonyXuQAQ/RNGDetPlusPlus

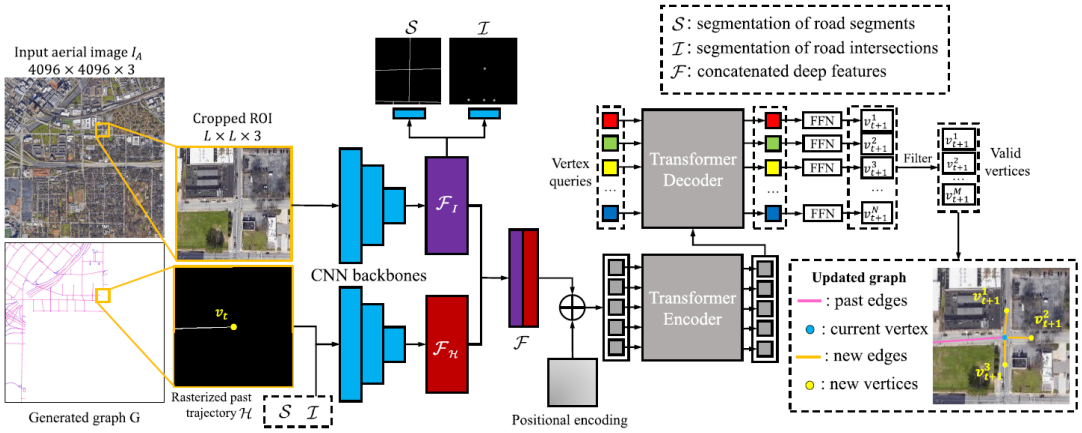


Figure 8. Architecture overview of RNGDet [49].

The model starts by padding the input image I_A with L number of black pixels and dividing I_A into $L \times L$ sized patches. Next, one CNN backbone predicts a feature tensor $\mathcal{F}_I$ from each patch. Each tensor $\mathcal{F}_I$ is sent to a segmentation head and an intersection segmentation head, both of which have the structure of an FPN [33]. The segmentation head predicts the road segmentation map $\mathcal{S}$, and the intersection segmentation head predicts the intersection segmentation map $\mathcal{I}$. Vertex $v_i \in V$ is considered an intersection if its degree is equal to 1, marking the endpoint of a road, or it has a degree greater than 2, marking an actual road intersection. A set of initial candidate vertices C is extracted from $\mathcal{I}$. The model uses those extracted vertices as starting points for road graph generation.

The model chooses a random vertex from C and starts iteratively generating the road network graph G. At each time step t, the model takes the current vertex v_t and crops out a region of interest (ROI) with the size $L \times L$ around the vertex v_t. One CNN backbone extracts the visual tensor $\mathcal{F}_I$ from the ROI. The other CNN backbone takes the rasterized past road trajectories within the ROI and predicts a deep feature tensor $\mathcal{F}_\mathcal{H}$. Tensors $\mathcal{F}_I$ and $\mathcal{F}_\mathcal{H}$ are then concatenated into $\mathcal{F}$.

The tensor $\mathcal{F}$ is split into a sequence of smaller tensors, and positional encodings are added before it is fed into the transformer encoder. Positional encodings are also added to the encoder's output sequence. The transformer decoder takes the encoder output and a set of N vertex queries $Q = \{q_i\}_{i=1}^{N}$ to produce N number of potential adjacent vertices and N probabilities. Based on each vertex query q_i, which itself is a learned embedding, the decoder predicts a coordinate tuple for a potential adjacent vertex v_{t+1}^i and a probability p_{t+1}^i of vertex v_{t+1}^i being a valid vertex. The invalid vertices are filtered out with some custom threshold p_{th} to obtain the valid adjacent vertices $\mathcal{V}$. Finally, the current graph G is updated by adding each vertex $v_{t+1}^j \in \mathcal{V}$ to V and edge $e_{t+1}^j = \{v_t, v_{t+1}^j\}$ to E.

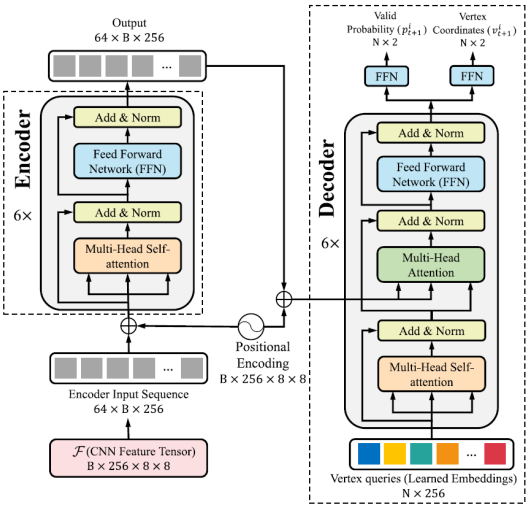


Figure 9. Architecture overview of the transformer in RNGDet [49].

The transformer block of RNGDet has the same structure as DETR's transformer. It has 6 encoder layers and 6 decoder layers. As usual, the encoder comprises multi-head self-attention and FFN blocks, both followed by layer normalization. The decoder includes multi-head self-attention, encoder-decoder attention, and FFN blocks, each also followed by layer normalization. The output from the final decoder layer is passed to two FFN layers that predict the potential adjacent vertex coordinates and probabilities. The transformer's architecture is in Figure 9.

3.1.2 Agent

The RNGDet approach aims to train a decision-making agent that iteratively annotates every road on the input image by imitating an expert annotator. The agent starts by taking an initial candidate vertex from C and makes a prediction based on the ROI cropped out around the current vertex. Based on the number of predicted adjacent vertices $M = |\mathcal{V}|$, the agent can continue in three different ways:

1. If $M = 0$, then the agent has reached the end of the current road segment. It pops a new vertex from C and starts generating a new road segment. If C is empty, the algorithm stops and outputs the final road network graph.

2. If $M = 1$, then the generation of the current road segment continues. The agent moves to the predicted vertex v^1_{t+1}.

3. If $M > 1$, then the agent has reached an intersection. It pushes all the predicted

vertices $v_{t+1}^j \in \mathcal{V}$ into C. The agent then pops a random vertex from C and continues the road generation from that vertex.

The agent's algorithm is illustrated in Figure 10.

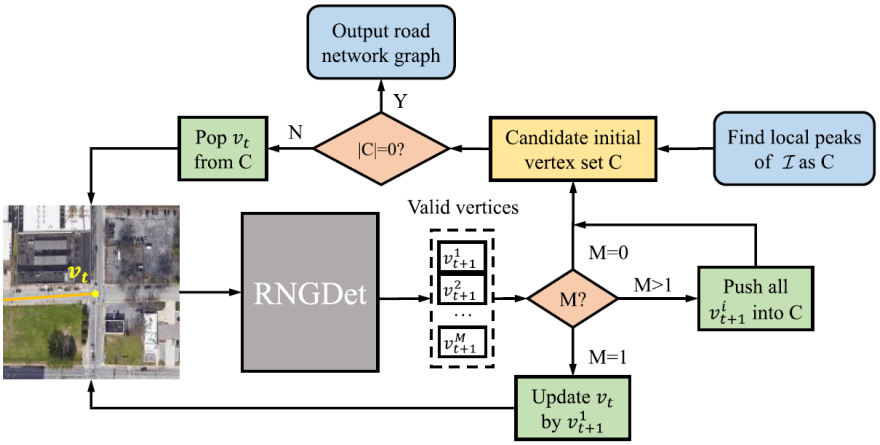


Figure 10. Algorithm of the RNGDet's agent [49].

3.1.3 Loss functions

Since RNGDet predicts a segmentation map, intersection segmentation map, vertex coordinates, and probabilities, the final training loss $\mathcal{L}$ is a weighted sum of the segmentation loss $\mathcal{L}_{seg}$, coordinate loss $\mathcal{L}_{coord}$, and probability loss $\mathcal{L}_{prob}$:

$$\mathcal{L} = \mathcal{L}_{seg} + \alpha\mathcal{L}_{coord} + \beta\mathcal{L}_{prob} \tag{3}$$

Here α and β are custom weights to balance the losses.

Binary cross-entropy loss is used for the segmentation maps $\mathcal{S}$ and $\mathcal{I}$. Let $\hat{\mathcal{S}}$ and $\hat{\mathcal{I}}$ be the predicted segmentation maps and $\mathcal{S}^*$ and $\mathcal{I}^*$ be the ground-truth segmentation maps. The segmentation loss is calculated as follows:

$$\mathcal{L}_{seg} = \text{BCELoss}(\hat{\mathcal{S}}, \mathcal{S}^*) + \text{BCELoss}(\hat{\mathcal{I}}, \mathcal{I}^*) \tag{4}$$

Here the BCELoss between a predicted map $\hat{\mathcal{Y}}$ and a ground-truth $\mathcal{Y}^*$ is defined as:

$$\text{BCELoss}(\hat{\mathcal{Y}}, \mathcal{Y}^*) = -\frac{w}{L^2}\sum_{i=1}^{L^2} w_p y_i^* \cdot \log(\hat{y}_i) + (1 - y_i^*) \cdot \log(1 - \hat{y}_i) \tag{5}$$

where $\hat{y}$ and y^* are the predicted and ground-truth pixel values, respectively. L is the width and height of the ROI, w is a rescaling weight, and w_p is a positive weight used to balance the positive and negative pixels.

Before the coordinate loss calculation, a bipartite matching is done between the predicted vertices $\{\hat{v}_i\}_{i=1}^{N}$ and the ground-truth vertices $\{v_j^*\}_{i=j}^{M}$. Given that $N \geq M$, some predicted vertices will be left unmatched. The optimal matching can be found by finding the minimal value of the following function:

$$\hat{\sigma} = \underset{\sigma}{\operatorname{argmin}} \sum_{i=1}^{N} \|(\hat{v}_i - v^*_{\sigma(i)})\| \tag{6}$$

where $\sigma(i)$ is the index of ground-truth vertex matched to the prediction $\hat{v}_i$. The Hungarian algorithm [28] is used to find the optimal assignment $\hat{\sigma}$. After obtaining the optimal matching, pairwise Euclidean distances can be calculated to get the coordinate loss:

$$\mathcal{L}_{coord} = \frac{1}{M} \sum_{i=1}^{M} \|(\hat{v}_i - v^*_{\hat{\sigma}(i)})\| \tag{7}$$

In addition to vertex coordinates, RNGDet also predicts valid probabilities $\{\hat{p}_i\}_{i=1}^{N}$ for each predicted vertex $\hat{v}_i$. The probability prediction is treated as a classification task - each vertex can be valid or invalid. The predicted vertex $\hat{v}_i$ is labeled as valid if it was matched with some ground-truth vertex v_j^*. Otherwise, it is labeled as invalid. The final vertex probability is the predicted probability score for the valid class. Cross-entropy loss is used to obtain the probability loss between the predicted valid probabilities $\hat{\mathcal{P}}$ and ground-truth $\mathcal{P}^*$:

$$\mathcal{L}_{prob} = \text{CELoss}(\hat{\mathcal{P}}, \mathcal{P}^*) = -\frac{1}{\sum_{i=1}^{N} w_{p_i^*}} \sum_{i=1}^{N} w_{p_i^*} \cdot \log \frac{e^{\hat{p}_{i,p_i^*}}}{e^{\hat{p}_i^+} + e^{\hat{p}_i^-}} \tag{8}$$

Here $\hat{p}_{i,p_i^*}$ is the predicted log probability of the ground-truth class for vertex $\hat{v}_i$. $\hat{p}_i^+$ and $\hat{p}_i^-$ are the log probabilities of $\hat{v}_i$ being valid or invalid, respectively. $w_{p_i^*}$ is a custom rescaling weight of the ground-truth class, which ensures that both classes remain balanced.

3.2 RNGDet++

The original RNGDet model performed well but still failed in some complicated areas, such as multi-level interchanges and very occluded streets. Therefore, the authors published a follow-up paper [51], which introduced an improved version of RNGDet, named RNGDet++. The proposed model is similar to the original RNGDet with two significant changes (Figure 11):

- RNGDet++ extracts deep visual features from all four layers of the backbone to better capture the visual information from the input image. RNGDet only took the feature from the fourth backbone layer.
- An instance segmentation head is added to RNGDet++ to better supervise the training. At each step, the network predicts an instance segmentation map for each predicted vertex together with the vertex coordinates and probabilities. The instance segmentation map depicts the shape of the road section between the current vertex v_t and the predicted vertex v_{t+1}^i.

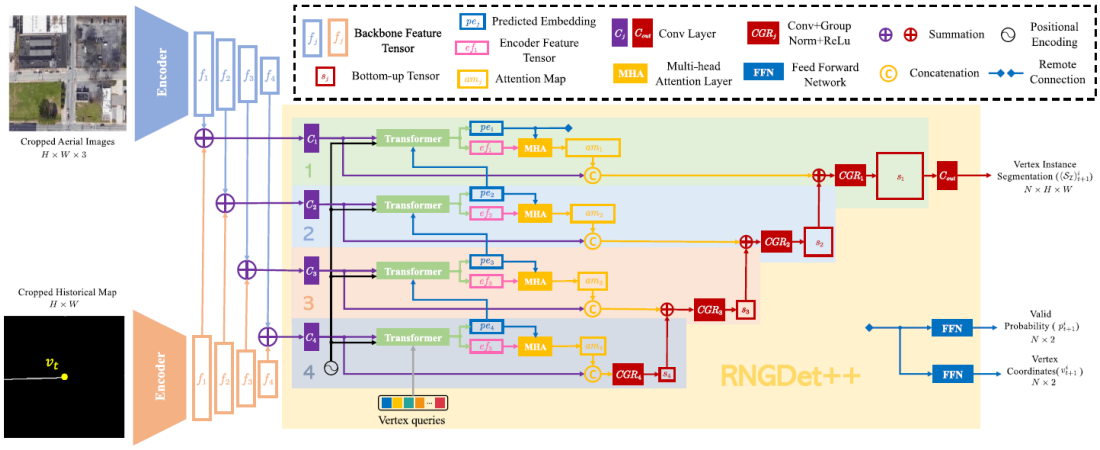


Figure 11. Architecture overview of RNGDet++ [51].

With the addition of instance segmentation head to the RNGDet++ network, an instance segmentation loss $\mathcal{L}_{ins}$ is added to the final training loss:

$$\mathcal{L} = \mathcal{L}_{seg} + \alpha\mathcal{L}_{coord} + \beta\mathcal{L}_{prob} + \gamma\mathcal{L}_{ins} \tag{9}$$

Here γ is the custom weight for the instance segmentation loss. All other variables are the same as in Equation 3. Instance segmentation loss itself is a binary cross-entropy loss that was defined in Equation 5:

$$\mathcal{L}_{ins} = \frac{1}{N}\sum_{i=1}^{N} \text{BCELoss}(\hat{\mathcal{S}}_{\mathcal{I}i}, \mathcal{S}^*_{\mathcal{I}i}) \tag{10}$$

where $\hat{\mathcal{S}}_{\mathcal{I}i}$ is the predicted instance segmentation map for the predicted adjacent vertex $\hat{v}_{t+1}^i$ and $\mathcal{S}^*_{\mathcal{I}i}$ is the ground-truth segmentation map of a road segment between the ground-truth of the current vertex v_t^* and the ground-truth of the adjacent vertex v_{t+1}^*.

4 Datasets

Since the Autonomous Driving Lab at the University of Tartu has mapped only a few areas in Estonia and, to the best of the author's knowledge, there are no publicly available lane detection datasets for Estonia, the RNGDet model was trained on a custom road detection dataset, and it was later fine-tuned with a lane detection dataset. This section gives an overview of both datasets and the training samples generation pipeline.

4.1 Road detection dataset

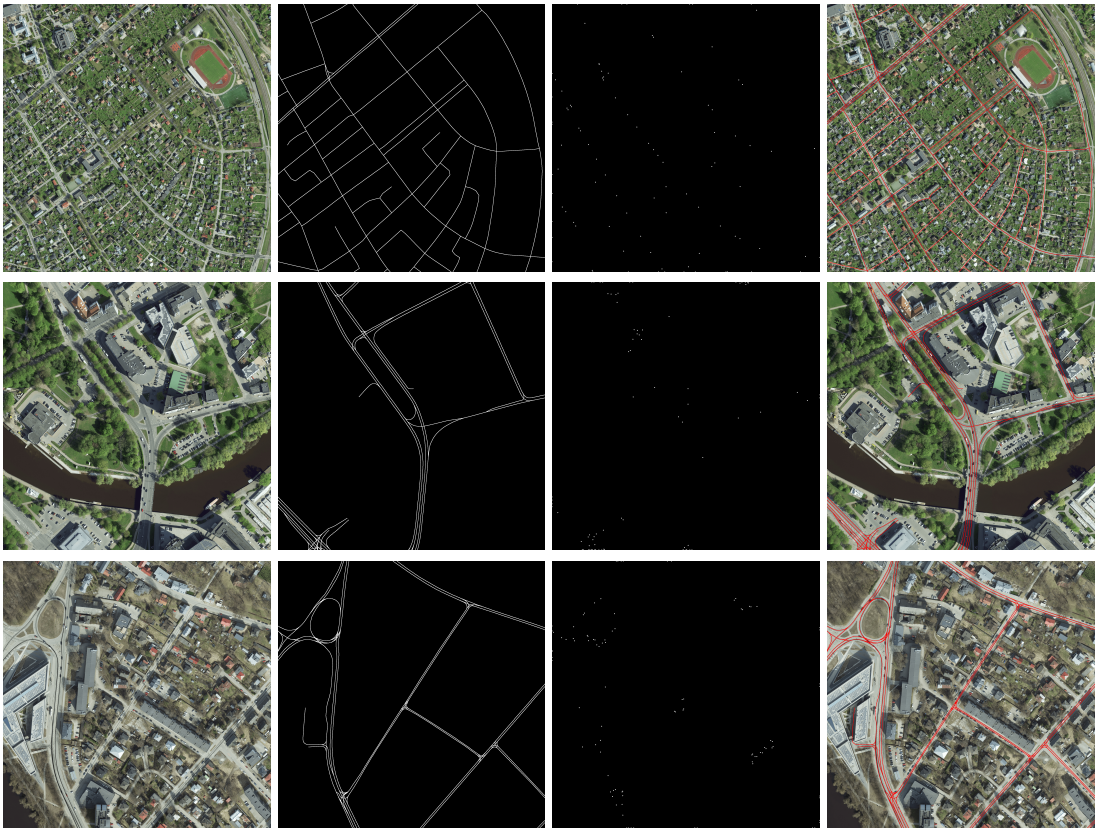
The first road detection models were trained with the dataset created by the authors of the Sat2Graph model [23], as the authors of RNGDet used this dataset for both training and evaluation. The Sat2Graph dataset consists of 180 satellite images from 20 cities in the United States. However, a custom road detection dataset was created for this thesis because the Sat2Graph dataset had a much smaller resolution than the imagery from the Estonian Land Board. Also, a model trained on an Estonian dataset will likely predict lanes from Estonian aerial images more accurately.

Table 1. Overview of orthophotos in the road detection dataset.

Country	No. of images	Date	Source
Estonia	243	2020-2023	Estonian Land Board[4]
Latvia	85	2016-2018	Latvian Geospatial Information Agency[5]
The Netherlands	70	2022	Beeldmateriaal Nederland[6]
Switzerland	70	2020-2022	Federal Office of Topography (Swisstopo)[7]
Total	**468**		

The road detection dataset used in this thesis consists of 243 orthophotos across all major towns and cities in Estonia. Since Estonia is a small country, some areas appear twice in the dataset, but in that case, one is a summer image, and the other is from spring before trees have leafed out. However, initial training results showed that there was not enough training samples in the Estonian data, so the dataset was augmented with orthophotos from Latvia, The Netherlands, and Switzerland. Those countries were chosen because the images are freely available and in the GeoTIFF format, which contains the necessary geospatial metadata for pre-processing. Table 1 gives an overview of the road detection dataset's final composition.

[4]https://geoportaal.maaamet.ee/eng/spatial-data/orthophotos/download-orthophotos-p662.html
[5]https://www.lgia.gov.lv/en/color-orthophoto-map-2016-2018-cycle-6
[6]https://www.beeldmateriaal.nl/data-room
[7]https://www.swisstopo.admin.ch/en/orthoimage-swissimage-10

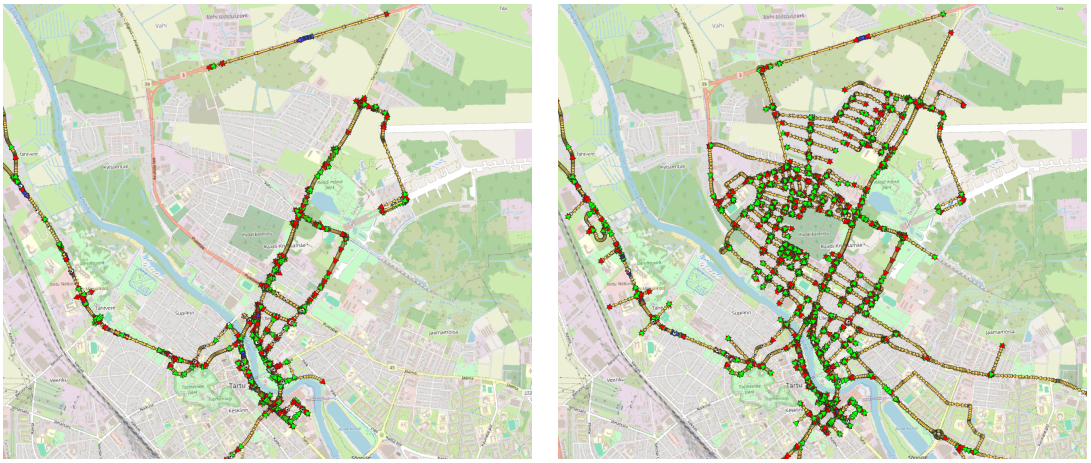


(a) Input orthophoto I_A (b) Segmentation map $\mathcal{S}^*$ (c) Intersection segmentation map $\mathcal{I}^*$ (d) Ground-truth visualization

Figure 12. Top row: images from road detection dataset. Middle row: images from lane detection dataset. Bottom row: images with trajectories from the lane detection dataset. Orthophotos are from the Estonian Land Board. The road network is from OpenStreetMap. Best viewed with digital zoom.

Each image in the dataset was cropped and rescaled to cover a square of 1 km $\times$ 1 km with the height and width of the image being 2048 pixels. Therefore, the images have a resolution of 48.8 cm/pixel. The gound-truth road network data was obtained from OpenStreetMap[8] (OSM). The imported roads from OSM did not include minor service roads and unpaved tracks. Based on the extracted road network, a ground-truth road segmentation map and intersection segmentation map were also generated. Example images from the road detection dataset are shown in Figure 12.

[8]https://www.openstreetmap.org/



(a) Before. (b) After.

Figure 13. ADL lane trajectory map of Tartu before and after the mapping effort. OpenStreetMap is used as the basemap.

4.2 Lane detection dataset

The lane detection dataset only contains data from Tartu. The source of the orthophotos is, again, the Estonian Land Board. Each original image had a resolution of 10 cm/pixel, but it was rescaled to 20 cm/pixel for the purpose of covering a larger area with ROI. The height and width of the orthophotos remained at 2048 pixels. Each area had spring and summer images, effectively doubling the size of the dataset without the need for additional lane trajectories. The dataset had 106 images in total.

The ground-truth lane trajectories came from the ADL's own HD map. Since the ADL had only mapped a few routes on larger roads, smaller streets intersecting with the mapped roads were annotated by hand based on the latest orthophoto and a GPS track recorded by a survey vehicle. However, during the experiments, it became clear that more lane data was needed, hence an effort was made to map the entire Raadi-Kruusamäe district of Tartu. The lane trajectories added as part of this thesis are visualized in Figure 13. The dataset with Raadi-Kruusmäe had 112 images.

At intersections, the model usually does not have any visual guide on how to generate the left turn lanes, and at some intersections, left turns are not allowed at all. In some experiments, raw GPS trajectories were added to the input orthophotos to give the model more information about the trajectories along which the vehicles are actually driving. Each trajectory was rasterized to the orthophoto with a black color at 20% opacity. The opacity should make it easier for the model to ignore trajectories that deviate from the average trajectory too much. Unfortunately, most of the existing trajectories were recorded on the same few routes on larger streets, and smaller streets were only covered during lane data collection for this thesis. The data was balanced a bit by the deletion of

some trajectories, but several streets and turns remained without trajectories. Also, the trajectories from smaller streets were usually collected only once, making them more susceptible to noise, often caused by bypassing parking cars and other obstacles.

Example images from the lane detection datasets are depicted in Figure 12.

4.3 Generation of training samples

The training dataset is obtained by extracting samples from each 2048 $\times$ 2048 pixel image and the associated road or lane network. The authors of the RNGDet [49] proposed to generate the training samples with two steps:

1. All vertices with degree 2 are removed from the road or lane network graph, but the shapes of the individual lane segments are preserved.

2. The road or lane network is iteratively traversed by the sampling agent, and one training sample is generated at each time step.

Figure 14. Generation of training samples [49]. (a) Road-segment mode, straight segment. (b) Road-segment mode, curved segment. (c) Road-segment mode, connecting with other candidate vertex. (d) Intersection mode.

The agent has two modes during the graph traversal - road-segment and intersection modes. The road-segment mode is used when the agent travels along one continuous road segment. If the agent is in road segment mode, a radius τ is defined around the current vertex v_t. If there is no other candidate vertex or a large curve within τ, then the ground-truth adjacent vertex v_{t+1}^1 is added τ distance away from v_t (Figure 14 (a)). If there is a large curvature within τ then v_{t+1}^1 is added on that curvature (Figure 14 (b)). If another candidate vertex $v_i \in C$ is within τ then the agent connects v_i with v_t and moves to some other candidate vertex from C (Figure 14 (c)). If there are multiple curves or other vertices within τ then the closest to v_t is chosen.

If the agent reaches an intersection, then a smaller radius τ' is defined around the current vertex v_t (Figure 14 (d)). All adjacent ground-truth vertices $v_{t+1}^i \in \mathcal{V}$ are added τ' distance away from v_t.

At each timestep t, ROI with size $L \times L$ is cropped around the current vertex v_t from the orthophoto. Random rotations and color augmentation are applied to the ROI image. Based on the already traversed ground-truth graph, image $\mathcal{H}$, depicting the past rasterized trajectories, is generated at the location of ROI. The ground-truth segmentation map $\mathcal{S}^*$ and intersection segmentation map $\mathcal{I}^*$ are cropped from the large segmentation maps in the dataset. The agent uses the previously described algorithm to generate the ground-truth adjacent vertices $\mathcal{V}^*$ as well as the valid probabilities $\mathcal{P}^*$. After one training sample $(\text{ROI}, \mathcal{H}, \mathcal{S}^*, \mathcal{I}^*, \mathcal{V}^*, \mathcal{P}^*)$ is obtained, Gaussian noise is added to the trajectory between v_t and $v_{t+1}^i \in \mathcal{V}$ to make the model more robust.

In experiments conducted for this thesis, the size of τ was set to 30 or 40 pixels, depending on whether the size of ROI was 128 or 256 pixels. Because of the larger agent step size, datasets where ROI $= 256$ have fewer training samples than those where ROI $= 128$. τ' was always half of τ ($\tau' = \frac{\tau}{2}$). The custom road detection dataset had over 300K samples, similar to the amount in the original RNGDet paper. The lane detection dataset had over 30K training samples. With the addition of images from Raadi-Kruusamäe, the number of training samples increased beyond 40K. The exact sizes of the training datasets are given in Table 2.

Table 2. Training set sizes

Dataset	**No. of training images**	**ROI size**	**No. of training samples**
Road detection dataset	438	128	390 357
		256	308 129
Lane detection dataset	97	128	40 453
		256	33 602
Lane detection dataset with Raadi-Kruusamäe	112	128	54 171
		256	42 589

5 Experiments and results

Several experiments were conducted to train a well-performing road detection model, which could then be fine-tuned on the lane detection dataset. This section discusses the evaluation metrics and general training details, describes the experiments, and presents the quantitative and qualitative analysis of the results.

5.1 Evaluation metrics

This thesis uses the same seven evaluation metrics used in the original RNGDet paper [49]. Pixel-precision (P-P), pixel-recall (P-R), and pixel-F1-score (P-F1) evaluate accuracy on the pixel level. Intersection-precision (I-P), intersection-recall (I-R), and intersection-F1-score (I-F1) evaluate the accuracy of intersections. Finally, average path length similarity (APLS) [20] is used to evaluate topology correctness.

Pixel-level metrics are calculated by rasterizing the ground-truth into a binary image $\mathcal{B}^*$. Similarly, the predicted graph is rasterized into $\hat{\mathcal{B}}$. Pixel-precision (P-P) is calculated by finding how many pixels $p \in \hat{\mathcal{B}}$ have a corresponding pixel $q \in \mathcal{B}^*$ that the Euclidean distance between p and q is smaller than some threshold δ_P:

$$\text{P-P} = \frac{|\{p | \|p, q\| < \delta_P, \exists q \in \mathcal{B}^* \ \ \forall p \in \hat{\mathcal{B}}\}|}{|\hat{\mathcal{B}}|} \tag{11}$$

Pixel-recall (P-R) is calculated by finding how many pixels $q \in \mathcal{B}^*$ have a pixel $p \in \hat{\mathcal{B}}$ within the distance of δ_p:

$$\text{P-R} = \frac{|\{q | \|q, p\| < \delta_P, \exists p \in \hat{\mathcal{B}} \ \ \forall q \in \mathcal{B}^*\}|}{|\mathcal{B}^*|} \tag{12}$$

After obtaining P-P and P-R, the F1-score can be calculated:

$$\text{P-F1} = \frac{2 \cdot \text{P-P} \cdot \text{P-R}}{\text{P-P} + \text{P-R}} \tag{13}$$

Intersection-level metrics I-P, I-R, and I-F1 are calculated similarly to P-P, P-R, and P-F1. The only differences are that the intersection points are rasterized on the binary images instead of the whole graph, and threshold δ_I can have some different value from δ_P.

The APLS metric samples N_S number of vertex pairs (u^*, v^*) from the ground-truth graph $\mathcal{G}^*$ and finds the length of the shortest path between them. Next, the corresponding vertex pair $(\hat{u}, \hat{v})$ from the predicted graph $\hat{\mathcal{G}}$ is found, and the length of the optimal path is again calculated. The APLS metric takes the average difference between the shortest paths of the prediction and ground truth over all N_S samples:

$$\text{APLS} = 1 - \frac{1}{N_S} \sum^{N_S} \min(1, \frac{|L(u^*, v^*) - L(\hat{u}, \hat{v})|}{L(u^*, v^*)}) \tag{14}$$

where $L(a, b)$ is the length of the shortest path between some vertex pair (a, b). If no corresponding vertices $(\hat{u}, \hat{v})$ exist, then the difference is equal to 1.0.

In the carried out experiments, the thresholds δ_P and δ_I were both set to 5 pixels. The number of samples N_S varied as the ground-truth graphs had different numbers of vertices. However, short paths with less than 100 pixels were filtered out.

5.2 General training details

The experiments mostly used the hyperparameters defined in the original RNGDet and RNGDet++ training scripts. Each model was trained for 50 epochs with a batch size of 10 or 20, depending on the machine used for training. AdamW was used as an optimizer. The initial learning rate for RNGDet was 10^{-4} and $9 \cdot 10^{-5}$ for RNGDet++ if ROI $= 128$. The learning rate was always 10^{-4} if ROI $= 256$. The initial learning rate was 10^{-5} during fine-tuning. The decay rate was always set to 10^{-5}.

The parameters α, β, γ in the loss function were set to 10, 1, and 1, respectively. The weight w in BCELoss was set to 1, and the positive weight w_p to 3. The only exception was intersection segmentation map loss, where $w = 3$ and $w_p = 6$. The class weights in probability loss were 1 for the positive class and 0.2 for the negative class. The probability threshold p_{th} to consider a predicted vertex as valid was 0.75, and the threshold for extracting a candidate vertex from the intersection segmentation map $\hat{\mathcal{I}}$ was set to 0.55.

When the model was trained on Estonian datasets, gradient clipping was added to the training pipeline to combat the exploding gradients problem. Gradient clipping had been used in the original DETR models but was missing from RNGDet. In all experiments with datasets created for this thesis, the max norm of the gradients was set to 0.5.

Resnet101 architecture is used as backbones. The backbones are initialized with default pre-trained weights from the PyTorch `torchvision.models` package. Models were trained on 4 GPU cores. Depending on the memory requirements, Nvidia GeForce RTX 2080, Nvidia V100-SXM2-32GB, or Nvidia A100-SXM4-80GB GPUs were used.

5.3 Experiments

This subsection presents the results of a selection of key experiments. For a fair comparison of results, only models trained for 50 epochs are listed here. Experiments conducted for hyper-parameter tuning or testing some new idea were usually trained for 10 or 30 epochs to reduce computational costs.

5.3.1 Retraining the RNGDet model

The first experiment was retraining the RNGDet model on Sat2Graph's dataset [23], which was included with the model's source code. This experiment would test if the

model would work as advertised and if it is suitable for lane detection. The first experiment's results are in Table 3.

The models trained as part of this thesis showed better pixel-level accuracy but worse intersection accuracy than the RNGDet model in the original article. The APLS score was very similar. The reason for these differences might be slight modifications in the architecture between the model used in the original article and the model released to the public, different values for hyper-parameters, and changes in the training dataset. The performance of RNGDet++ on intersections was disappointing, but as it still showed better pixel-level accuracy, it was decided to focus more on RNGDet++ in future experiments.

Overall, the experiment showed that the RNGDet and RNGDet++ models could predict road network graphs from aerial imagery with high accuracy, and fine-tuning the model on lane detection tasks should be possible.

Table 3. Road network detection results on Sat2Graph's dataset [23]. The best score for each metric is highlighted in bold.

	P-P	P-R	P-F1	I-P	I-R	I-F1	APLS
RNGDet in the original article	72.31	75.08	73.67	**65.30**	**66.50**	**65.89**	**67.88**
RNGDet trained for this thesis	80.13	87.44	83.48	56.29	58.74	57.30	65.74
RNGDet++ trained for this thesis	**84.28**	**88.04**	**86.02**	56.19	58.13	56.96	67.64

5.3.2 Road network extraction with Estonia dataset

The first experiments on the custom road detection dataset used only data from Estonia. However, it became apparent that additional data from other countries is required to achieve satisfactory performance. Except the experiment marked with EST, the results reported in Table 4 and visualized in Figure 15 were achieved with the augmented dataset.

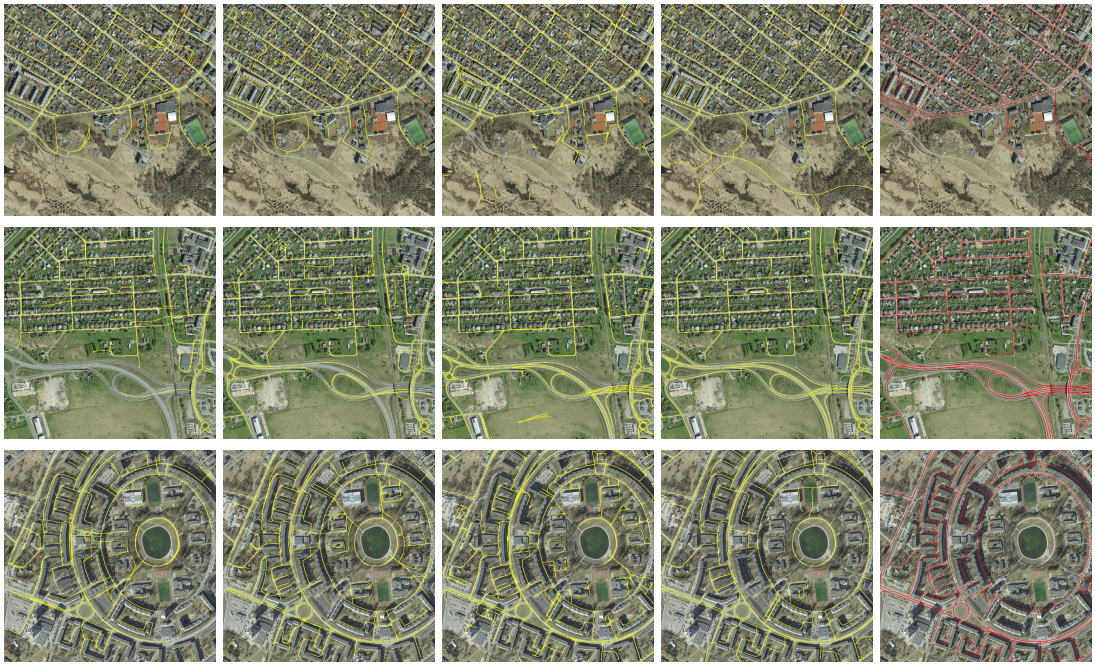
Even with the augmented dataset, the accuracy of the plain RNGDet++ model remained significantly lower than that of the Sat2Graph's dataset. This was expected as the RNGDet model was likely specifically tuned to show its best performance on Sat2Graph's dataset. Another reason for the reduced performance might be that road detection in Europe is more challenging than in the United States because in Europe, roads tend to be narrower, occlusion due to buildings tends to be greater, roads are not arranged in a uniform grid pattern, and intersections might not be in a 90° angle.

5.3.3 Additional backbone

The initial road detection models with ROI $= 128$ had a satisfactory pixel-level performance but struggled with initial intersection detection. Several intersections were not detected, so there were fewer initial candidate vertices. This meant fewer overall road detections. The issue became especially severe during initial lane detection experiments.

Table 4. Road network detection results on the custom road dataset. Number 128 or 256 denotes the ROI size. EST indicates that the model was trained only on Estonian data. Models with letters AB had an additional backbone for intersection detection. The best score for each metric is highlighted in bold.

	P-P	P-R	P-F1	I-P	I-R	I-F1	APLS
RNGDet 128 EST	54.14	45.34	48.07	20.33	10.18	12.88	47.90
RNGDet 128	62.04	60.57	58.41	28.16	26.73	24.24	37.16
RNGDet++ 128	57.03	82.07	66.54	24.37	**42.03**	30.16	45.33
RNGDet++ 128 AB	51.63	89.09	64.44	23.44	40.57	28.89	44.60
RNGDet++ 256 AB	**64.48**	**89.64**	**74.18**	**36.17**	41.00	**37.78**	**54.47**



(a) RNGDet 128 (b) RNGDet++ 128 (c) RNGDet++ 128 AB (d) RNGDet++ 256 AB (e) Ground-truth

Figure 15. Sample predictions from different road detection models. Number 128 or 256 denotes the ROI size. Models with letters AB had an additional backbone for intersection detection. Orthophotos are from the Estonian Land Board. Ground-truth road network is from OpenStreetMap. Best viewed with digital zoom.

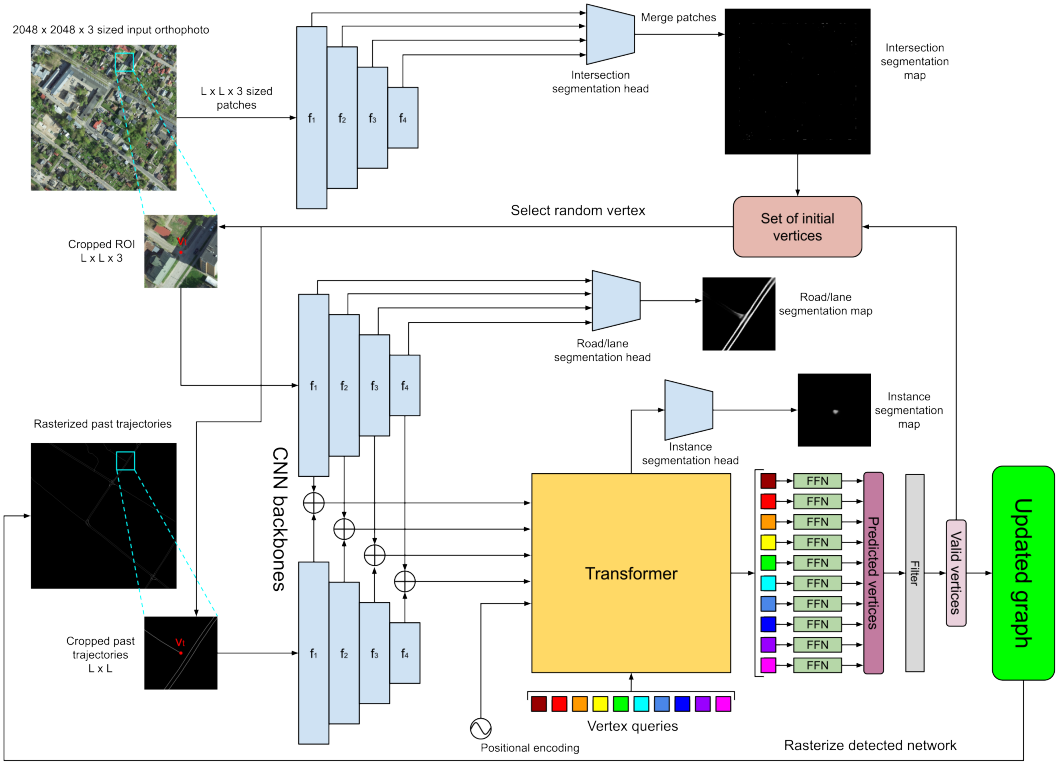


Figure 16. The architecture of RNGDet++ with additional backbone. Best viewed with digital zoom.

A third backbone was added to the model to improve intersection detection, which would be supervised with intersection segmentation maps $\mathcal{I}^*$. The features obtained from the third backbone would only contain the visual information about the intersection locations and not be diluted by the information from segmentation maps $\mathcal{S}^*$. Intersection backbone features were completely separated from the features of the segmentation backbone to make the intersection loss only dependent on the accuracy of the predicted intersection map $\hat{\mathcal{I}}$. Therefore, the transformer would receive the intersection information only from $\hat{\mathcal{S}}$. However, this was not a big concern as the ablation studies in the original RNGDet paper showed that removing the intersection segmentation would have only a minuscule effect on the model's performance. The architecture of RNGDet with the additional backbone is depicted in Figure 16.

However, the quantitative results of the road detection model with an additional backbone show a slight decrease in performance. Only the pixel-recall score increased compared to the model without the additional backbone. The decrease in pixel-precision can be explained by an increase in the number of roads being generated due to improved detection of initial candidate intersections. Thus, a larger share of false positives was

observed. Because some false positives are small service roads around shopping centers or industrial areas, detecting more roads than was in the ground truth is not as big of a problem as not detecting an actual street. The slight decrease in intersection-level accuracy score can be explained by the removal of intersection features from the transformer.

As an additional measure, the size of the ROI was increased to 256 pixels to capture more spatial information from the surrounding area. The RNGDet++ model with additional backbone and ROI $= 256$ achieved the best overall performance on the custom road dataset but still did not match the model's performance on Sat2Graph's dataset. With some hyper-parameter tuning, the model's performance could probably be increased. However, that was not done due to the model's large memory consumption and the roughly five-day training time.

5.3.4 Lane network extraction

The results of lane detection experiments are reported in Table 5 and visualized in Figure 17. Lane network extraction models gave sensible results when the ROI size was set to 256 pixels, corresponding to 51.2 meters in real life. This implies that using input images with a higher resolution (lower pixel size) than 20 cm/pixel is difficult in RNGDet since that would require an increase in ROI size and thus make the training even more expensive. Currently, training models with ROI $= 256$ requires more than 40 GB of GPU memory.

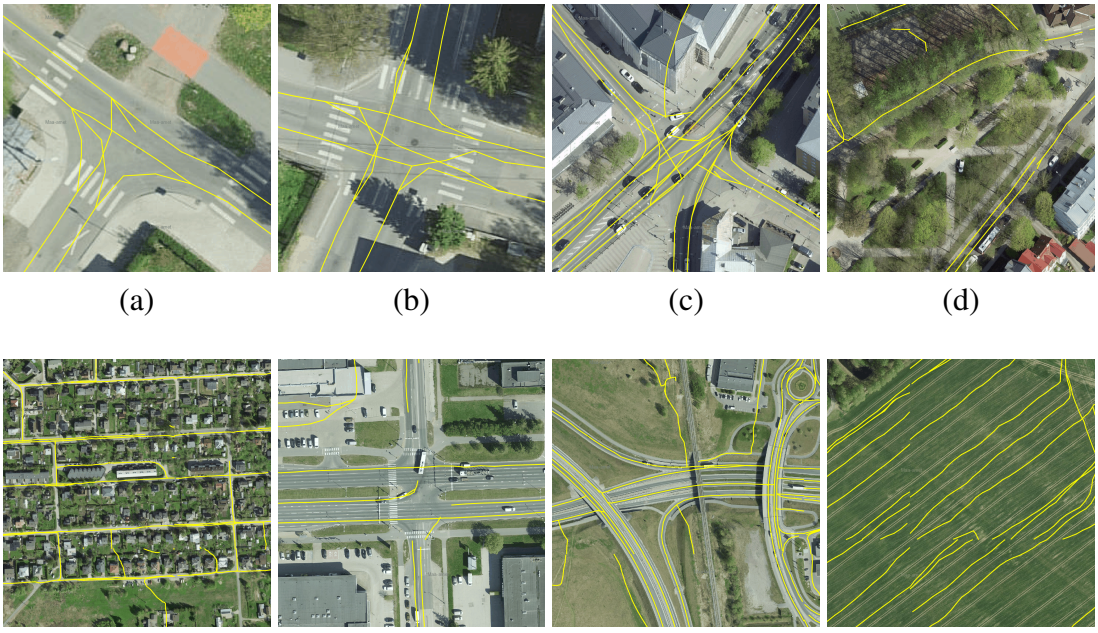
The intersection-level metrics are all very low. Models with ROI $= 128$ did not predict any intersection within δ_I pixels of the ground-truth, thus the intersection accuracy for those models was 0. Models with ROI $= 128$ and without the additional backbone struggled to predict any intersections at all. Intersection prediction is hard for the model

Table 5. Number 128 or 256 denotes the ROI size. Models with letters AB had an additional backbone for intersection detection. FT indicates that the models were fine-tuned on the road detection model. TRAJ denotes the model that used raw lane trajectories on input images.

	P-P	P-R	P-F1	I-P	I-R	I-F1	APLS
RNGDet++ 128	24.72	5.06	8.12	0.00	0.00	0.00	4.25
RNGDet++ 128 AB	20.35	8.61	11.76	0.00	0.00	0.00	1.90
RNGDet++ 128 FT	15.09	18.16	16.14	0.00	0.00	0.00	1.27
RNGDet++ 128 AB FT	21.68	22.90	21.70	0.00	0.00	0.00	1.18
RNGDet++ 256 AB	**91.66**	34.86	48.74	3.89	1.59	2.15	**44.44**
RNGDet++ 256 AB FT	63.63	65.91	64.11	5.66	2.05	2.99	23.62
RNGDet++ 256 AB FT TRAJ	69.28	**77.50**	**72.30**	**10.10**	**4.27**	**5.95**	40.96

(a) RNGDet++ 256 AB | (b) RNGDet++ 256 AB FT | (c) RNGDet++ 256 FT AB TRAJ | (d) Ground-truth

Figure 17. Sample predictions from lane detection models with ROI $= 256$. Models with letters AB had an additional backbone for intersection detection. FT indicates that the models were fine-tuned on the road detection model. TRAJ denotes the model that used raw lane trajectories on input images. Orthophotos are from the Estonian Land Board. Best viewed with digital zoom.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 18. Sample predictions of RNGDet++ 256 AB FT around the city of Tartu. Orthophotos are from the Estonian Land Board. Best viewed with digital zoom.

because, in the lane network graph, intersection nodes are only at places where the lanes split or merge but not where they intersect each other. The human annotator somewhat arbitrarily chooses the location of the split or merge depending on whether the divergence of lanes is drawn more gently or sharply. This causes the model to predict larger intersection areas where individual pixels have lower probability scores than the threshold for intersection extraction. Even when the model can recognize the approximate location of the intersection with high enough confidence, the detected intersection often falls outside of the 5-pixel evaluation threshold δ_I.

Out of the models with ROI $= 256$, the model that was not pre-trained achieved the best pixel-precision, but its pixel-recall was the worst. This discrepancy is because the model detected relatively few lanes, but the lanes detected were very accurate. The accurate lane detections meant this model had the highest APLS score.

The fine-tuned RNGDet++ model with ROI $= 256$ had similar P-P and P-R scores, indicating that the model was well-balanced. The overall pixel-F1 score of 64.11 can be considered satisfactory, given the amount of training data available. However, the APLS score of 23.62 was relatively low. To analyze the model's performance and identify common failure cases, lanes from 612 aerial images were extracted, encompassing the entire city of Tartu.

The fine-tuned model can generally detect both lanes from an ordinary two-way

street. The model is considerably less accurate at intersections. At simple intersections, the model can predict right turns with acceptable accuracy but still struggles with left turns (Figure 18 (a)). Predicting right turns is easier because the model can use the edge of the road as guidance, but there are usually no guiding features for left turns. The lack of guidance for left turns also makes the ground-truth trajectories varied in shape. Sometimes, the driver of the survey vehicle has cut the corner too much, making it even harder for the model to "understand" how left turns should be predicted. If the intersection has more incoming lanes and not many guiding features, then the model's performance degrades quickly (Figure 18 (b) and Figure 18 (c)).

Because the model does not predict the direction of the generated lanes, it often connects turning lanes to the lane in the opposite direction. The lane detection model needs a direction prediction module, but that can cause issues with pre-training on the road network dataset. Another problem at large intersections is that there are multiple lanes in a small, cramped area right at the center. This makes the prediction of segmentation maps hard and also confuses the agent, which does not know which lane it should connect to. This results in very messy and inaccurate intersections (Figure 18 (c)), and therefore, the usefulness of the model is even further reduced as drawing intersections is the most time-consuming part for a human in the HD map creation pipeline.

The model can predict lanes at an occluded street, but the prediction's accuracy depends on the severity of the occlusion. In Figure 18 (d), it can be seen that the street with minimal occlusion has a more accurate prediction than the street that has substantial occlusion by trees. The gaps in the lane segments are caused by the border between two input images. The post-processing required to remove these gaps is discussed in Section 6.2.

In areas with single-family houses, the streets tend to be relatively narrow, which often results in the detection of only one lane (Figure 18 (e)). The detected lanes are also more wobbly, and intersections do not have turning lanes. The ground-truth trajectories in these areas are often very near or even on top of each other, making the prediction much more complicated. The lack of turning lanes indicates that the model could not extract initial intersection vertices, possibly because the intersection area is small and turning lanes are short and very close to each other.

There were also issues with lane detection on large streets (Figure 18 (f)), even though these streets have clearly marked lanes. The problem might be the lack of training data from large multi-lane streets. The only street sections with a similar number of lanes shown in the figure were relatively short ones near the Riia-Turu intersection. The model also failed at two-level interchanges (Figure 18 (g)), but this was to be expected as the lane detection dataset did not contain any orthophotos of grade-separated interchanges.

As indicated by the low pixel-precision, the model produced many false-positive detections. Some of these detections were caused by parallel straight lines not related to any roads (Figure 18 (h)) or by the model just hallucinating. However, false-positive

detections are not a big concern as they could be easily filtered out using the road surface area data from the Estonian Land Board.

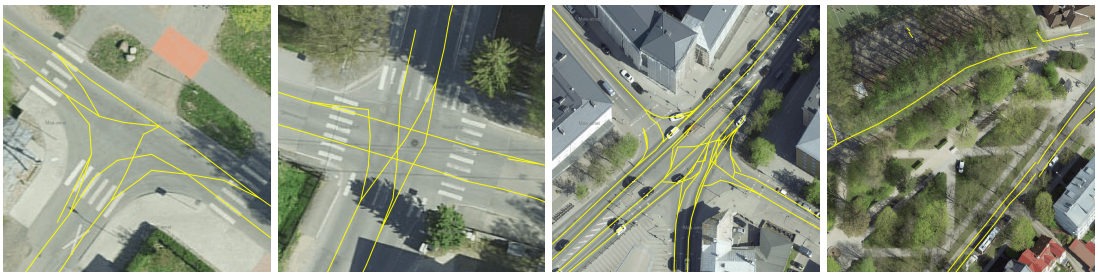
5.3.5 Images with trajectories

The model fine-tuned with images that included raw lane trajectories had the best pixel-level and intersection-level performance, although the intersection accuracy was still quite poor (Table 5). However, the APLS score was much higher than the fine-tuned model without trajectories. Again, the lane network of the entire Tartu was generated to analyze the model's performance. Since raw trajectories were available for less than half of Tartu streets, improvements were not expected in every area, but examples shown in Figure 19 had at least partial lane trajectory coverage.

The lane trajectories did not significantly improve intersection areas. Even if the lane splitting points were correctly detected, the agent tended to drift off course and connect to the opposite driving direction (Figure 19 (a)). Also, the model still struggled with detecting all the turning lanes at an intersection (Figure 19 (b)). One reason for the undetected turning lanes might be that some did not have a corresponding raw trajectory. This can mean the model is more confident in not predicting a splitting point on the existing trajectory.

Another problem is again the high density of lanes at the center of the intersections (Figure 19 (c)). The model has a hard time deciding if it should connect to a lane or if it should go over it. A higher resolution input image would separate the lanes more, but then the ROI covers a smaller area, meaning the model has less spatial information at each step.

On visual inspection, the model using raw lane trajectories performed similarly to the model without trajectories at occluded areas (Figure 19 (d)). This was not very reassuring, but the accuracy in occluded areas could possibly be increased by adding more trajectories to the input images during the training and inference phases.



(a) (b) (c) (d)

Figure 19. Sample predictions of RNGDet++ 256 AB FT TRAJ around the city of Tartu. Orthophotos are from the Estonian Land Board. Best viewed with digital zoom.

6 Discussion

The experiments did not yield results that were accurate enough for HD maps. A more extensive training dataset would undoubtedly improve the performance of the lane network prediction model, similar to how the road detection model improved after adding data from other countries. With enough lane data, fine-tuning would not be necessary, and some lane data-specific modifications to the RNGDet model could be made. Adding a lane direction prediction module would be the most useful change in the architecture of RNGDet. Further modifications to improve the initial vertex detection and mitigate the drifting issue at intersections are also essential for achieving high enough accuracy for HD maps. However, even with a well-performing model, there are still many cases where the model cannot get enough information from the aerial image to generate the lanes correctly. Also, generated lanes would still need post-processing before adding them to an HD map.

6.1 Limitations of aerial images

Aerial images offer limited information about the traffic rules of each road or intersection. Restricted left turns and one-way streets are two of the most common cases that are almost impossible for the model to figure out. Larger intersections with multiple turning lanes are generally challenging because the only clues the model would have are the arrows, ruts, and tire marks on the road surface and the number of lanes leaving the intersection. However, the road surface could have been freshly replaced, or the arrows could be worn out or covered by waiting vehicles. Lanes allowing cars to drive in multiple directions are tricky, even for a human mapper. Often, in these cases, the human annotator would need to use images taken from the street level to double-check the correctness of the turning lanes. Figure 20 depicts an intersection, which would be nearly impossible for humans to figure out only based on the orthophoto.

Another problem is the occlusion of roads by trees and especially buildings. Satellite images are less susceptible to this problem than orthophotos, but the problem still exists. Mid-rise buildings, common in European cities, can cover the right-most lane almost entirely for tens of meters. As the ROI is limited in size, it might not see both ends of the building and decide to stop generating the lane. Even when the lane can be seen, the dark shadow cast by a high building can make the lane difficult for the model to recognize. Narrow streets with parking vehicles can also cause problems. Sometimes, parking vehicles take up one lane of the road for a long stretch. In areas with primarily single-family houses, roads are often narrow and do not have any markings, meaning parking vehicles can make it hard for the model to assess the width of the street.

A solution for the ambiguity of turning lanes at the intersections and the occluded lanes is to include raw vehicle trajectories with the input images. However, obtaining the vehicle trajectory data in large quantities with good coverage is difficult. GPS

(a) Plain orthophoto. (b) Correct lane network.

Figure 20. An example of a challenging intersection to map only based on the plain orthophoto. In subfigure (b), note the one-way street with multiple turning lanes at the top-right and the straight bus lane going over the right turning lane on the left of the image. Orthophotos are from the Estonian Land Board. Best viewed with digital zoom.

receivers would need to be installed on multiple vehicles, and these vehicles would have to drive around the city for an extended period. A smaller, Tartu-sized city can be covered once with one survey vehicle relatively quickly, but then the model would have only one guidance trajectory for each street. In that case, the model would predict the same trajectory that already has been collected. Alternatively, the trajectories that were collected only once could be cleaned and prepared with heuristics-based algorithms and added to HD maps as lanes with relatively low human effort. That would mean using a computationally expensive neural network, which produces very similar trajectories that would still likely require human adjustments, is not justified.

6.2 Post-processing

The post-processing of the predicted lane network was out of the scope of this thesis, but to make the lanes suitable for deployment on an HD map, a post-processing pipeline is necessary.

The first post-processing step would be the removal of all false positives. This could be quite easy as the road network data from OpenStreetMap can be employed to generate approximate road areas, and these areas could be used to filter out lanes that are far away from any roads. Estonian Land Board has published a road surface area dataset, which could also be used for the filtering task.

The current model works with images sized 2048×2048 pixels. With high-resolution orthophotos, this size covers a relatively small area. The lanes generated based on

different input images must be stitched together with heuristics-based algorithms to generate a lane network graph for a large area. Images forming a non-overlapping grid covering the entire Tartu were used during the qualitative performance evaluation of the trained models. This method left noticeable gaps in the resulting lane network. Hence, some overlapping between images should be introduced. Then, in the overlapping area, an algorithm could match the lane segments and vertices in one image with the segments and vertices in the other and average them. However, a grid-based method may not be the best, as intersection areas, which are the most difficult to predict, are often divided between multiple images. This can degrade the model's performance a lot. Therefore, a better idea might be to take the intersection locations from the road network graph and crop the orthophotos around the intersection first so that intersections would always be entirely in one image. Then, start cropping input orthophotos along roads between intersections with some overlap, given that the input images around intersections do not overlap already. Only lanes on the road along which the cropping was done need to be stitched together afterward.

Another critical step is determining if the lane segment goes straight or turns left or right because currently, the ADL's autonomous vehicle uses turn signals based on the information it gets from the HD map. A simple method that measures the angle between the lines defined by the first two points and the last two points of short lane segments could be employed. If the angle is close to 90°, the lane is classified as a right or a left turning lane.

However, no neural network or post-processing method would produce a completely correct lane network, as any road or intersection can have a unique design or be an edge-case situation. Therefore, every automatically generated lane network requires human validation and modifications.

7 Conclusion

Autonomous vehicles need accurate maps for journey planning and localization and to increase situational awareness. As creating accurate HD maps requires substantial human labor, this thesis explored the possibility of predicting the lane network graph, an integral part of any HD map, from orthophotos with deep-learning neural networks. RNGDet, a DEtection TRansformer (DETR) inspired road network detection model, was used to predict a lane network graph for the city of Tartu. For this, a custom road detection dataset was created first to train a road network detection model. The road detection model was then fine-tuned on ADL's lane data to create a lane detection model. ADL's Tartu HD map had to be significantly expanded to generate enough training samples for fine-tuning. An additional CNN backbone was added to the original RNGDet network to increase intersection detections. Finally, raw GPS trajectories were added to the input orthophotos to give the model more guidance in occluded areas and intersections.

The lane network obtained with the best-performing models was not accurate enough for use on HD maps. One reason for this was the difficulty in predicting initial intersection vertices and generating lanes correctly in a small area. Another limiting factor was still the shortage of lane training data. The results showed that the model could detect lanes from simple two-way streets and right turns on easier intersections but struggles with large roads, long occluded areas, and intersections in general. The model which used raw GPS trajectories achieved better quantitative results than the model which did not.

All in all, with additional lane data, the RNGDet model could yield better results. Therefore, a potential future experiment could be to train the lane detection model again once the lanes in all of Tartu have been mapped. Future work should also focus on making the overall design of RNGDet more suitable for the lane detection task. Refining the initial candidate vertex extraction pipeline and adding a lane direction prediction module are essential tasks in possible future work. In addition, establishing a post-processing pipeline will also be needed before importing the generated lane network onto the HD map. In the future, the RNGDet model could also be tested with other road elements that can be represented by a linestring, such as road borders, curbs, and road markings.

8 Acknowledgments

This thesis was funded by the Autonomous Driving Lab, a collaboration project between the University of Tartu and Bolt. I would like to express my sincere gratitude to my supervisors, Tambet Matiisen and Edgar Sepp, for their guidance and support throughout this project. Special thanks to Kertu Toompea for collecting the GPS lane trajectories from the streets of Tartu.

References

[1] ASAM OpenDRIVE. [Online]. Available: https://www.asam.net/standards/detail/opendrive/. Accessed: 2024-02-17.

[2] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *ArXiv*, abs/1607.06450, 2016.

[3] Ruzena Bajcsy and Mohamad Tavakoli. Computer recognition of roads from satellite pictures. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-6(9):623–637, 1976.

[4] W. G. C. Bandara, Jeya Maria Jose Valanarasu, and Vishal M. Patel. Spin road mapper: Extracting roads from aerial images via spatial and interaction space graph reasoning for autonomous driving. *2022 International Conference on Robotics and Automation (ICRA)*, pages 343–350, 2021.

[5] Zhibin Bao, Sabir Hossain, Haoxiang Lang, and Xianke Lin. A review of high-definition map creation methods for autonomous driving. *Engineering Applications of Artificial Intelligence*, 122:106125, 2023.

[6] M. Barzohar and D.B. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):707–721, 1996.

[7] 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 *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4720–4728, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

[8] Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, C.V. Jawahar, and Manohar Paluri. Improved road connectivity by joint learning of orientation and segmentation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10377–10385, 2019.

[9] Davide Belli and Thomas Kipf. Image-conditioned graph generation for road network extraction. *ArXiv*, abs/1910.14388, 2019.

[10] Yigit Baran Can, Alexander Liniger, Danda Pani Paudel, and Luc Van Gool. Topology preserving local road network estimation from single onboard camera image. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17242–17251, 2021.

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.

[12] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *arXiv:1802.02611*, 2018.

[13] Guangliang Cheng, Feiyun Zhu, Shiming Xiang, and Chunhong Pan. Road centerline extraction via semisupervised segmentation and multidirection nonmaximum suppression. *IEEE Geoscience and Remote Sensing Letters*, 13(4):545–549, 2016.

[14] Dragos Costea and Marius Leordeanu. Aerial image geolocalization from recognition and matching of roads and intersections, 2016.

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

[17] Babak Ebrahimi Soorchaei, Mahdi Razzaghpour, Rodolfo Valiente, Arash Raftari, and Yaser Pourmohammadi Fallah. High-definition map representation techniques for automated vehicles. *Electronics*, 11(20), 2022.

[18] Gamal Elghazaly, Raphaël Frank, Scott Harvey, and Stefan Safko. High-definition maps: Comprehensive survey, challenges, and future perspectives. *IEEE Open Journal of Intelligent Transportation Systems*, 4:527–550, 2023.

[19] Adam Van Etten. City-scale road extraction from satellite imagery v2: Road speeds and travel times. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[20] Adam Van Etten, David Lindenbaum, and Todd M. Bacastow. Spacenet: A remote sensing dataset and challenge series. *ArXiv*, abs/1807.01232, 2018.

[21] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann Dauphin. Convolutional sequence to sequence learning. *ArXiv*, abs/1705.03122, 2017.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[23] Songtao He, Favyen Bastani, Satvat Jagwani, Mohammad Alizadeh, H. Balakrishnan, Sanjay Chawla, Mohamed Mokhtar Elshrif, Samuel Madden, and Mohammad Amin Sadeghi. Sat2graph: Road graph extraction through graph-tensor encoding. In *European Conference on Computer Vision*, 2020.

[24] Namdar Homayounfar, Wei-Chiu Ma, Justin Liang, Xinyu Wu, Jack Fan, and Raquel Urtasun. Dagmapper: Learning to map by discovering lane topology, 2020.

[25] Muhammad Ibrahim, Naveed Akhtar, Mohammad A. A. K. Jalwana, Michael Wise, and Ajmal Mian. High definition lidar mapping of perth cbd. In *2021 Digital Image Computing: Techniques and Applications (DICTA)*, pages 01–08, 2021.

[26] Jinseop Jeong, Jun Yong Yoon, Hwanhong Lee, Hatem Darweesh, and Woosuk Sung. Tutorial on high-definition map generation for automated driving in urban environments. *Sensors*, 22(18), 2022.

[27] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2017.

[28] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[29] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework. *2022 International Conference on Robotics and Automation (ICRA)*, pages 4628–4634, 2021.

[30] Yali Li, Longgang Xiang, Caili Zhang, and Huayi Wu. Fusing taxi trajectories and rs images to build road map via dcnn. *IEEE Access*, 7:161487–161498, 2019.

[31] Zuoyue Li, Jan Dirk Wegner, and Aurélien Lucchi. Topological map extraction from overhead images. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1715–1724, 2018.

[32] Bencheng Liao, Shaoyu Chen, Yunchi Zhang, Bo Jiang, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Maptrv2: An end-to-end framework for online vectorized hd map construction. *ArXiv*, abs/2308.05736, 2023.

[33] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 936–944, 2017.

[34] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9992–10002, Los Alamitos, CA, USA, oct 2021. IEEE Computer Society.

[35] Lingfei Ma, Ying Li, Jonathan Li, José Marcato Junior, Wesley Nunes Gonçalves, and Michael A. Chapman. Boundarynet: Extraction and completion of road boundaries with deep learning using mobile laser scanning point clouds and satellite imagery. *IEEE Transactions on Intelligent Transportation Systems*, 23(6):5638–5654, 2022.

[36] Gellert Mattyus, Wenjie Luo, and Raquel Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[37] Volodymyr Mnih and Geoffrey E. Hinton. Learning to detect roads in high-resolution aerial images. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision – ECCV 2010*, pages 210–223, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[38] Fabian Poggenhans, Jan-Hendrik Pauls, Johannes Janosovits, Stefan Orf, Maximilian Naumann, Florian Kuhnt, and Matthias Mayr. Lanelet2: A high-definition map framework for the future of automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1672–1679, 2018.

[39] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017.

[40] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018.

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[42] Yong-Qiang Tan, Shang-Hua Gao, Xuan-Yi Li, Ming-Ming Cheng, and Bo Ren. Vecroad: Point-based iterative graph exploration for road graphs extraction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[43] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021.

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971, 2023.

[45] F. Tsushima, N. Kishimoto, Y. Okada, and W. Che. Creation of high definition map for autonomous driving. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B4-2020:415–420, 2020.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[47] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 568–578, October 2021.

[48] Zhenhua Xu, Yuxuan Liu, Lu Gan, Xiangcheng Hu, Yuxiang Sun, Lujia Wang, and Ming Liu. csboundary: City-scale road-boundary detection in aerial images for high-definition maps. *IEEE Robotics and Automation Letters*, 7:5063–5070, 2021.

[49] Zhenhua Xu, Yuxuan Liu, Lu Gan, Yuxiang Sun, Xinyu Wu, Ming Liu, and Lujia Wang. Rngdet: Road network graph detection by transformer in aerial images. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–12, 2022.

[50] Zhenhua Xu, Yuxuan Liu, Yuxiang Sun, Meilin Liu, and Lujia Wang. Centerlinedet: Centerline graph detection for road lanes with vehicle-mounted sensors by transformer for hd map generation. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3553–3559, 2022.

[51] Zhenhua Xu, Yuxuan Liu, Yuxiang Sun, Ming Liu, and Lujia Wang. Rngdet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement. *IEEE Robotics and Automation Letters*, 8(5):2991–2998, 2023.

[52] Gaodian Zhou, Weitao Chen, Qianshan Gui, Xianju Li, and Lizhe Wang. Split depth-wise separable graph-convolution network for road extraction in complex environments from high-resolution remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–15, 2022.

[53] Lichen Zhou, Chuang Zhang, and Ming Wu. D-linknet: Linknet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 192–1924, 2018.

Appendix

I. Source code and training data

The source code is available at `https://github.com/Karljohan99/lane_centerline_detection`. The training data can be obtained on request from the author (e-mail: karljohan30@gmail.com).

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Karl-Johan Pilve**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Lane Centerline Detection from Orthophotos using Transformer Networks,
   *(*title of thesis)

   supervised by Tambet Matiisen and Edgar Sepp.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Karl-Johan Pilve
15/05/2024