Sidewalk Extraction Using Deep Learning and Cost-based Route Optimization with Mini-max Objective Function

by

Zhibin Bao

A thesis submitted to the
School of Graduate and Postdoctoral Studies in partial
fulfillment of the requirements for the degree of

Master of Applied Science in Mechanical Engineering

Faculty of Engineering and Applied Sciences

University of Ontario Institute of Technology (Ontario Tech University)

Oshawa, Ontario, Canada

December 2023

THESIS EXAMINATION INFORMATION

Submitted by: **Zhibin Bao**

Master of Applied Science in **Mechanical Engineering**

Thesis title: Sidewalk Extraction Using Deep Learning and Cost-Based Route Optimization with Mini-Max Objective Function

An oral defense of this thesis took place on November 29, 2023 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Zia Saadatnia
Research Supervisor	Dr. Xianke Lin
Research Co-supervisor	Dr. Haoxiang Lang
Examining Committee Member	Dr. Yuping He
Thesis Examiner	Dr. Akramul Azim

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

ABSTRACT

With the growing diversification of modern urban transportation options, such as small-scale autonomous delivery vehicles, autonomous patrol robots, e-bikes, and e-scooters, sidewalks have gained newfound importance as critical features of High-Definition (HD) Maps. Since these emerging modes of transportation are designed to operate on sidewalks to enhance public safety, there is an urgent need for efficient and precise sidewalk annotation methods for HD maps. This is crucial for accurate representation and the development of robust path-planning algorithms for autonomous vehicles to navigate urban environments safely. The following thesis proposes a semantic segmentation-based sidewalk extraction on aerial images method using an $A*$ path planning algorithm for sidewalk segmentation refinement. The $A*$ path planning algorithm with and without heuristic function was then applied to the extracted and refined sidewalk annotations to generate a safe and efficient route for autonomous navigation. An objective function considering travel distance and safety level is also proposed to determine the optimal route on the sidewalk and crosswalk. The results of this work show that the proposed sidewalk extraction method can precisely and efficiently predict sidewalks from aerial images, and it is feasible to navigate throughout the city using the predicted sidewalks.

Keywords: semantic segmentation; sidewalk extraction; route optimization; HD maps; autonomous vehicles

AUTHOR'S DECLARATION

I hereby declare that this thesis consists of original work of which I have authored. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize the University of Ontario Institute of Technology (Ontario Tech University) to lend this thesis to other institutions or individuals for the purpose of scholarly research. I further authorize University of Ontario Institute of Technology (Ontario Tech University) to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research. I understand that my thesis will be made electronically available to the public.

Zhibin Bao

Zhibin Bao

STATEMENT OF CONTRIBUTIONS

Part of the work described in Chapter 2 has been published as:

Z. Bao, S. Hossain, H. Lang, and X. Lin, "A review of high-definition map creation methods for autonomous driving," *Eng Appl Artif Intell*, vol. 122, p. 106125, Jun. 2023, doi: 10.1016/J.ENGAPPAI.2023.106125.

Part of the work described in Chapter 3 has been accepted for publication as:

Z. Bao, H. Lang, and X. Lin, "Deep Learning-Based Sidewalk Extraction on Aerial Image," Proceedings of the Canadian Society for Mechanical Engineering International Congress (CSME 2023)

Part of the work described in Chapter 4 is being reviewed for publication as:

Z. Bao, H. Lang, and X. Lin, "Sidewalk Extraction on Aerial Images with Deep Learning and Path Planning Algorithm," *Eng Appl Artif Intell*.

ACKNOWLEDGEMENTS

I sincerely thank my supervisor and co-supervisor, Dr. Xianke Lin and Dr. Haoxiang Lang for their continuous support and patience throughout my academic journey. The accomplishments I've achieved during my Masters' program would simply not have been possible without their guidance, insight, and kindness.

My sincere thanks to the AEM and the GRASP lab teams at Ontario Tech University. I am honoured to collaborate with a group of intelligent and kind researchers.

Finally, I would like to thank my friends and family for being supportive throughout this journey. I could not have completed this without them.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 2

CHAPTER 4

CHAPTER 5

LIST OF FIGURES

CHAPTER 1

CHAPTER 2

CHAPTER 3

CHAPTER 4

CHAPTER 5

LIST OF ABBREVIATIONS AND SYMBOLS

HD	High-Definition
DL	Deep Learning
GPUs	Graphics Processing Units
SOTA	State-of-the-art
CNN	Convolutional Neural Network
FPN	Feature Pyramid Network
AfA	Attention for Adjacency Net
c-DCGAN	Conditional Deep Convolutional Generative Adversarial Network
FCN	Fully Convolutional Network
CRF	Conditional Random Forest
QGIS	Quantum Geographic Information System
GSI	Google Satellite Image
PNG	Portable Network Graphics
GeoTIFF	Georeferenced Tag Image File Format
TIFF	Tag Image File Format
GIS	Geographic Information Systems
OSM	OpenStreetMap
PAN	Pyramid Attention Network
PSPNet	Pyramid Scene Parsing Network
MA-Net	Multi-scale Attention Network
BCE	Binary Cross Entropy
Adam	Adaptive Moment Estimation
AdaGrad	Adaptive Gradient
RMSProp	Root Mean Squared Propagation
IoU	Intersection over Union
ROS	Robot Operating System

Chapter 1. Introduction

1.1 Background and Motivation

Autonomous driving has been among the most popular yet challenging topics in recent years. Among all modules in an autonomous driving system, HD maps have drawn a noticeable amount of attention in recent years. HD maps have been known for their high precision, usually centimetre-level, and rich geometric and semantic information normally unavailable on traditional maps. An HD map contains all critical static properties (such as roads, road markings, traffic lights, buildings, and obstacles) of the environment necessary for autonomous driving, especially the objects that are impossible for sensors to appropriately detect due to occlusion. In an autonomous driving system, an HD map is used to constantly interact with different sensors, including camera, lidar, and radar, to construct a real-time perception module, which ultimately supports the mission and motion planning of the ego vehicle. This interaction is further demonstrated in Figure 1.1.

As a base layer of an HD map, the road network defines all drivable and non-drivable lanes, such as local ways, highways, pedestrian crossings, and sidewalks. It is a fundamental geofencing component binding the ego vehicle within drivable roads. It also works as a traditional map, allowing the autonomous driving system to plan and navigate the path. Extracting road networks from aerial images is also appealing since aerial photographs have extensive coverage of maps, usually at the city scale, and are constantly updated through satellites. This allows the autonomous driving system to understand the environment before planning the route. Traditionally, road network annotations/extractions were done by labour, which was costly, time-consuming, and low precision due to human error. In recent years, with the fast development of accelerated Graphics Processing Units

(GPUs) and excellent deep learning (DL) algorithms, road network extraction has become an automatic application with high precision and efficiency and low labour cost. Most of the current DL-based road network extraction research focuses on extracting motorways or highways from aerial images due to the increasing popularity of autonomous vehicles, resulting in a considerable lack of sidewalk extraction methods. However, with increasingly more small-scale vehicles and robots operating autonomously on the road, the significance of including a sidewalk network in an HD map cannot be ignored. Thus, accurate and automatic sidewalk extraction methods are in demand. Additionally, global routing on sidewalks is also inevitable for smooth navigation throughout the urban environment. The quality of the selected route for autonomous navigation also depends on factors, such as the total distance of the route and the safety concerns on the route. Thus, routing methods for generating high quality route on sidewalk networks are in demand.

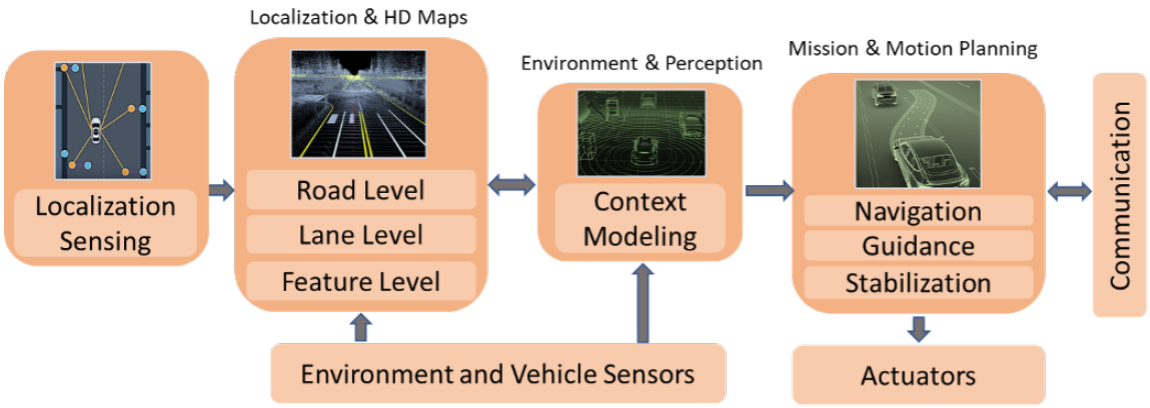

Figure 1.1 Interaction Between HD Maps and Other Autonomous Driving Modules

1.2 Scope and Objectives

The primary scope of this research is to develop an efficient and precise method for sidewalk extraction on aerial images utilizing semantic segmentation and segmentation

refinement. All extracted sidewalk images are to be concatenated together to construct a city-scale sidewalk network of a part of North Oshawa. A cost-based routing algorithm is then applied to the network to generate a smooth route using only sidewalks and crosswalks (if necessary). An objective function with the mini-max strategy is also proposed to select the efficient route considering the route distance and safety. The purpose of route planning is to show the feasibility of autonomous robots travelling through the city using sidewalks and crosswalks. The objective function can also be adapted to other global path planning applications with specific preferences.

The detailed objectives include:

- Developing an automatic sidewalk extraction method using the deep learning approach
- Developing a cost-based global routing algorithm incorporating terrain types (sidewalk vs crosswalk)
- Developing an objective function for global routing optimization

Tasks are required to be completed to achieve the objective above include:

- The design of a customized sidewalk dataset with precise sidewalk masks specifically for sidewalk extraction and semantic segmentation research
- The development of an automatic sidewalk extraction method on aerial images using semantic segmentation
- A comprehensive comparison of semantic segmentation performance on sidewalk extraction

- The development of a path-planning-based sidewalk segmentation refinement method
- The development of a cost-based routing algorithm for sidewalk and crosswalk networks
- The design of an objective function that determines the optimal route for different route planning applications or preferences considering travel distance and safety level

1.3 Contributions

To the knowledge of the author, the main contributions of this research are listed below:

- A customized sidewalk dataset is prepared for sidewalk extraction and semantic segmentation research, which can also be used as a benchmark dataset for future research.
- A DL-based sidewalk extraction method is proposed to annotate all sidewalks efficiently and precisely when given an aerial image, replacing the traditional manual labelling methods.
- A path planning-based segmentation refinement method is proposed to fix broken/incomplete sidewalks when segmentation algorithms fail to produce correct predictions.
- All extracted sidewalk images are concatenated to construct a city-scale sidewalk network map of a part of North Oshawa, including the Ontario Tech University, residential area, and urban traffic, which can be used as a map texture in an HD map for route planning and navigation.

- Using GPS coordinates, a cost-based routing algorithm is applied to the city-scale sidewalk network map for global route planning and navigation. Various routes between two locations on the map are generated when different cost values are given.
- An objective function is proposed for route optimization considering distance and safety level.
- A path follower simulation is created to mimic the real-life scenario of an autonomous robot following the generated route on the sidewalk network.

1.4 Outline

The thesis is organized as follows:

Chapter 1 introduces the research background, scope, objectives, contributions/outcomes, and thesis outline.

Chapter 2 provides an in-depth literature review regarding DL-based road network extraction on aerial images for HD maps, outlining the lack of current road network extraction research.

Chapter 3 describes the design process in detail on the sidewalk dataset preparation, including the aerial image and the sidewalk mask.

Chapter 4 describes the proposed DL-based sidewalk extraction method, including introducing semantic segmentation algorithms, model training and evaluating, and sidewalk extraction performance before refinement. The second part of this chapter describes the proposed sidewalk segmentation refinement method and the results after the refinement.

Chapter 5 describes the cost-based route planning algorithm on the extracted sidewalk and the objective function considering both route distance and safety level. The path follower simulation is carried out in detail in the second part of this chapter.

Chapter 6 concludes the current work and suggests future work to be done.

Chapter 2. Literature Review

2.1 Introduction

The "high-definition map" concept was first carried out in the Mercedes-Benz research in 2010 for the Bertha Drive Project. In the Bertha Drive Project, a Mercedes-Benz S500 completed the Bertha Drive Memorial Route fully autonomously, using a highly precise and informative 3D road map [1]. The road map was later named "High-Definition Live Map" by a mapping company called HERE [2]. In the HD map defined by HERE, a three-layer data structure, shown in Figure 2.1, is adopted, including a Road Model, a Lane Model, and a Localization Model as the first, second, and third layers, respectively. The three-layer structure is also adopted by other mapping companies and organizations, such as TomTom and Lanelet (Bertha Drive), with different layer names, as shown in Table 2.1.

Table 2.1 Examples of the three-layer structed HD maps

Layer number	**HERE**	**Lanelet (Bertha Drive)**	**TomTom**
1	HD road	Road network (OpenStreetMap)	Navigation data
2	HD lanes	Lane level map	Planning data
3	HD localization	Landmarks/Road marking map	Road DNA

In the three-layer structure, the first layer defines all road characteristics, including but not limited to road topology/network, direction of travel, travelling rules, intersections, curbs/boundaries, slopes/ramps, and elevation. Its role is to provide general navigation information. The second layer defines the lane-level features such as road types, widths, speed limits, and stop areas. This layer supports the perception module of the autonomous

driving system for decision-making based on real-time traffic or environment. The third layer localizes the autonomous vehicle by adding roadside furniture like trees, buildings, traffic signs and signals, and road markings. Those features allow the autonomous vehicle to quickly localize itself by matching the HD map environment to the real-time environment.

Road networks, as the base/first layer of an HD map, define the general topology of the map and are essential for an autonomous driving system to localize the ego vehicle and plan routing. Road network extraction, a process of manually or automatically annotating the road or road boundary from an aerial map, as shown in Fig. 2.2, is also commonly done on aerial or satellite images since they have high resolution, extensive map coverage, and up-to-date map information. As mentioned in Chapter 1, DL-based road network extraction methods have been actively proposed and intensively used to replace the traditional manual annotation methods. This chapter provides a comprehensive literature review on DL-based road network extraction methods, and their advantages and limitations are also discussed.

Figure 2.1 Road boundary extraction on NYC planimetric dataset [3]

2.2 Deep Learning-based Road Network Extraction Methods

Automatic road network extraction on aerial images can be classified into segmentation-based, iterative graph growing, and graph-generation methods.

2.2.1 Segmentation-based Methods

Semantic segmentation is a computer vision task that aims to correctly give the same colour code to pixels of an image that belongs to the same class/category. An example of image segmentation data is presented in Figure 2.2, where each class is labelled in a specific colour code, such as roads, pedestrians, vehicles, signs, and traffic lights. In segmentation-based methods, the probabilistic segmentation map is predicted from an aerial image, and the segmentation prediction is refined and extracted through post-processing.

Figure 2.2 Image segmentation example from the Cityscapes Dataset [4]

DeepRoadMapper, proposed by Mattyus et al., is a segmentation-based method that directly predicts the road topology and annotates the road network from aerial images. In

this approach, a variant of ResNet [5] was used to segment aerial images into interest categories, such as roads and buildings. A softmax activation function was used with a threshold of 0.5 probability to filter the road class, and the shinning [6] method was applied to extract the road's centerline. Additionally, the endpoints of the discontinued road were connected to alleviate the discontinuity issue of the road segmentation. In their work, shown in Figure 2.3, the yellow lines are considered potential roads, and the A^* algorithm was applied here to select the shortest path between every two discontinuities. This approach was evaluated on the TorontoCity dataset and achieved the state-of-the-art (SOTA) result in the year of the publication. Besides the excellent performance, it is noticeable that the heuristics (A^* algorithm) is not an ideal solution when the road or the surrounding environment complexity increases as the shortest path is not always the correct path.

Figure 2.3 DeepRoadMapper road network extraction [7]

Batra et al. proposed the Orientation Learning and Connectivity Refinement approaches to enhance the segmentation-based road network extraction performance and improve the road segmentation refinement method in the DeepRoadMapper. This proposed

method fixed the road network discontinuity issue by predicting the road topology's orientation and segmentation and using an n-stacked multi-branch convolutional neural network (CNN) to correct road segmentation results. This method was evaluated on the SpaceNet [8] and the DeepGlobe [9] dataset, and the results were compared with DeepRoadMapper and other methods [10]–[13] to show its SOTA performance. The evaluation results and the comparison are shown in Table 2.2, where the best results are bolded.

Additionally, a road segmentation refinement method was proposed by Ghandorh et al., adding an edge detection algorithm on top of the segmentation-based method [14]. The proposed method combined the encoder-decoder architecture with dilated convolutional layers and the attention mechanism to allow the neural network to segment large-scale objects and focus more on the crucial features. The road segmentation was then further refined by applying an edge detection algorithm to them.

Table 2.2 Comparison of the state-of-the-art road extraction methods on SpaceNet and DeepGlobe dataset. IOU^r and IoU^a refers to relaxed and accurate road IoU. APLS refers to average path length similarity

Method	**SpaceNet**						**DeepGlobe**					
	Precision	Recall	F1	IoU^r	IoU^a	*APLS*	Precision	Recall	F1	IoU^r	IoU^a	*APLS*
DeepRoadMapper (segmentation) [7]	60.61	60.80	60.71	43.58	59.99	54.25	79.82	80.31	80.07	66.76	62.58	65.56
DeepRoadMapper (full) [7]	57.57	58.29	57.93	40.77	N/A	50.59	77.15	77.48	77.32	63.02	N/A	61.66
Topology Loss (with BCE) [10]	50.35	50.32	50.34	33.63	56.29	49.00	76.69	75.76	76.22	61.58	64.95	56.91
Topology Loss (with Soft IoU) [10]	52.94	52.86	52.90	35.96	57.69	51.99	79.63	79.88	79.75	66.32	64.94	65.96
LinkNet34 [12]	61.30	61.45	61.39	44.27	60.33	55.69	78.34	78.85	78.59	64.73	62.75	65.33
LinkNet34 [12] + Orientation [30]	63.82	63.96	63.89	46.94	62.45	60.76	81.24	81.73	81.48	68.75	64.71	68.71
MAN [11]	49.84	50.16	50.01	33.34	52.86	46.44	57.59	56.96	57.28	40.13	46.88	47.15
RoadTracer [13]	62.82	63.09	62.95	45.94	62.34	58.41	82.85	83.73	83.29	71.36	**67.61**	69.65
OrientationRefine [30]	**64.65**	**64.77**	**64.71**	**47.83**	**63.75**	**63.65**	**83.79**	**84.14**	**83.97**	**72.37**	67.21	**73.12**

2.2.2 Iterative Graph Growing Methods

Iterative graph growing methods, as the name implies, generate the road network from aerial images by selecting several vertices along the road or road boundary and constructing the road network vertex by vertex until the entire road network is created.

The iterative graph growing method was proposed by Bastani et al. to solve the broken road segmentation issue. The author noticed that the performance of the heuristics dramatically drops when there is more uncertainty or complexity in the road segmentation, which can be caused by occlusion and complex topology, such as parallel roads [13]. The purely CNNs-based road segmentation performs poorly as the occlusion area increases, which rises from trees, shadows, buildings, and over-road bridges. Since prior approaches [7], [15] could not handle such problems very well, Bastani et al. proposed a new method, RoadTracer [13], to address the abovementioned issues and automatically extract the road networks from aerial images. The RoadTracer adopts an iterative graph construction process. It has a search algorithm guided by a CNNs-based decision function. The search algorithm starts from a known single vertex on the road or road boundary, and as the search algorithm explores the aerial map, it continuously adds vertices and edges to the road network. The CNNs-based decision function decides if a vertex or an edge can be added to the road network. As the search algorithm finishes exploring, all added vertices and edges are connected one after another to generate the road graph in an iterative growing manner. The growing process can be visualized in Figure 2.4. Additionally, RoadTracer was also evaluated on the SpaceNet and the DeepGlobe dataset, and the results are shown in Table 2.2.

The iterative graph growing method does solve the road segmentation discontinuity issue caused by occlusions; however, the limitation of this method is also apparent. The efficiency of this method heavily depends on the scale of the road network or the aerial map as well as the exploring speed of the search algorithm. Since it creates the road network vertex-by-vertex, the process will become time-consuming as the road network scale grows. To the best of the author's knowledge, RoadTracer is the first work that applies the iterative graph growing method in road network extraction research. Thus, further research on implementing more efficient strategies on large-scale maps or enhancing the speed of the search algorithm should be conducted.

Figure 2.4 Iterative graph growing method for road network extraction [1]

2.2.3 Graph-generation Methods

Graph-generation methods extract road networks by directly predicting the road network graphs from aerial images. This method encodes the input aerial images into vector fields for prediction purposes by the neural network. The prediction is then decoded into segmentation graphs using up-sampling algorithms such as the transposed convolutional neural network. Besides road networks, this method has also been used to extract other features, including line segments[16], line-shaped objects [17], and polygon-shaped buildings [18].

Xu et al. adopted the graph-generation method and added a transformer [19] into the algorithm to propose a novel system named csBoundary [20] for automatic road network extraction. The system's first part contains a Feature Pyramid Network (FPN) [21] that inputs a 4-channel aerial image. The FPN processes the image and predicts a keypoint map and a segmentation map, as shown in Figure 2.5. A set of vertex coordinates with a length of m is then extracted from the keypoint map. For each set of the vertex coordinates, a size of $L \times L$ region of interest (ROI) is cropped and placed on the keypoint map, as shown in the red box in Figure 2.5. The system then concatenates the input image, the segmentation map, and the keypoint map to construct a 6-channel feature tensor. Inspired by the transformer algorithm, Xu et al. proposed the attention for adjacency net (AfANet), which builds the second part of the csBoundary system. The encoder of AfANet uses the 6-channel feature tensor with the ROI to compute the local and global feature vectors. The decoder of AfANet takes the local and global feature vectors as well as the extracted vertex to predict the adjacency matrix for generating the road network graph. All obtained graphs are then stitched together to construct the final city-scale road boundary graph, as shown in Figure 2.5.

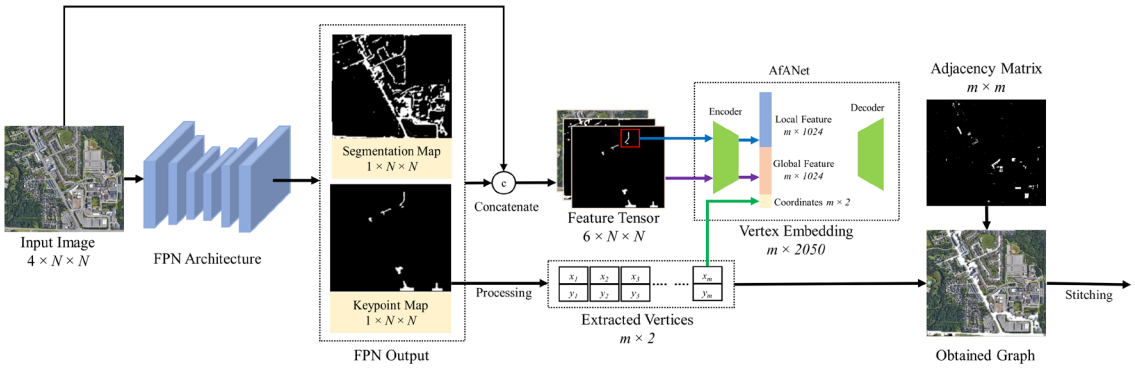

Figure 2.5 CsBoboundary system architecture

2.2.4 Other Methods

Besides the methods mentioned above, there are also different approaches in road network extraction using various data sources/formats instead of aerial images, such as vehicle dashcam images, lidar point clouds, and the fusion of both.

Schreiber et al. [22] used 3D reconstruction to reconstruct the road from camera images. Jang et al. [23] designed a fully convolutional network to detect and classify the road in camera images. Ibrahim et al. and Ding et al. utilized lidar point clouds to create 3D road maps. The former applied a loop detection algorithm and the 3D normal distribution transformation (NDT) algorithm to extract the road from point clouds, and the latter created a 3D HD map using the NDT algorithm with a digital 3D scene of the actual scene as a reference. Ma et al. proposed BoundaryNet [24], a novel DL framework adopting a modified U-Net [25] and a conditional deep convolutional generative adversarial network (c-DCGAN) [26], to extract road boundaries using both laser scanning point clouds and satellite imagery. Li et al. built the road map by fusing GPS trajectories and remote sensing images [27]. Yu et al. designed a fully convolutional network (FCN) and used a residual fusion strategy to merge feature maps learned from lidar-camera data for road detection [28]. Gu et al. created a mapping layer to project the features of lidar's imagery view onto the camera's perspective imagery view. They designed a conditional random forest (CRF) framework to extract both range and colour information from lidar-camera data. Since road network extraction using other data sources and data fusion are out of the scope of this thesis, such methods will not be further discussed. However, readers can refer to [1] for more details.

2.2.5 Summary

The segmentation-based method automatically extracts road networks from aerial images and can be directly applied to large-scale maps for fast road network extraction using CNN. However, the performance of this method heavily relies on the quality/clearance of the road network on aerial images. The segmentation-based method does not guarantee a correct prediction if the aerial image contains occlusion that blocks the road network. Segmentation refinement is usually applied after the segmentation prediction, such as path planning and edge detection, to fix the discontinuity issue. It still cannot completely address this issue, especially when the complexity of the road network increases.

On the other hand, the iterative graph growing method solves the discontinuity issue by adopting a search algorithm powered by a CNN-based decision function, but it still has its limitations. Since this method generates the road vertex-by-vertex, the extraction time dramatically increases as the size of the road map increases. This method also suffers from the drifting issue because it generates the road iteratively, making it challenging to extract large-scale road networks. Graph-generation methods are still limited to extracting certain shapes of the object as they depend on the ability of the decoding algorithm, which strains their generalization ability. More powerful decoding algorithms need to be developed to enhance the performance of graph-generation methods. Table 2.3 compares the evaluation results of the above three road network extraction methods on the Topo-Boundary [3] benchmark dataset using APLS and TLTS [29] evaluation metrics, including OrientationRefine [30] (segmentation-based), Enhanced-iCurb [3] (iterative graph growing), Sat2Graph [17] (graph generation), and csBoundary [20] (graph generation).

Table 2.3 Evaluation results of three road extraction methods on the Topo-Boundary benchmark dataset

Methods	Precision			Recall			F1			APLS	TLTS
	2.0	5.0	10.0	2.0	5.0	10.0	2.0	5.0	10.0		
OrientationRefine	**0.517**	**0.816**	**0.868**	0.352	0.551	0.589	0.408	0.637	0.678	0.235	0.219
Enhanced-iCurb	0.412	0.695	0.785	**0.412**	**0.671**	**0.749**	**0.410**	**0.678**	0.760	0.299	0.279
Sat2Graph	0.460	0.484	0.604	0.128	0.240	0.293	0.159	0.304	0.374	0.037	0.030
csBoundary	0.309	0.659	0.830	0.291	0.600	0.738	0.297	0.652	**0.772**	**0.376**	**0.343**

Furthermore, from reviewing the above road network extraction methods, it is easy to notice that all the proposed work solely focuses on extracting driveways/highways for autonomous vehicles. Since sidewalks have become an unavoidable feature on HD maps due to the growing diversification of modern transportation options, automatic sidewalk extraction methods are in demand.

Chapter 3. Aerial Image Dataset Preparation

3.1 Introduction

When training a deep learning algorithm/model for some specific application, such as object detection, object recognition, and semantic segmentation, one of the most important aspects is the dataset used for training. The quality and the size of the dataset have a direct impact on the final performance of the deep learning model. Unlike object detection/recognition datasets, where annotations are directly placed on the training image, a semantic segmentation dataset has two types of image data: images and masks. Images in sidewalk extraction refer to the original aerial images that will be the input of the DL algorithm. Masks here refer to the sidewalk annotation image, where sidewalk pixels from the original aerial image are assigned to a colour, and the background pixels are set to a different colour. Dataset preparation is a multi-step process, including sidewalk annotation on aerial images, image processing, and dataset splitting. This chapter will provide details on the dataset preparation process.

3.2 Sidewalk annotation on aerial images

The Quantum Geographic Information System (QGIS) software is the primary platform for collecting all aerial images and sidewalk annotations. In QGIS, a raster layer of Google Satellite Image (GSI) is loaded on the software as the base aerial map. The map is set to the EPSG:3857-WGS 84 coordinate system since this system has high-resolution aerial images and top-down views of the map. A scale of 1:1000 is selected to maintain the clarity of the aerial images while allowing them to have enough sidewalk features within each aerial image. With the coordinate system and the scale correctly set, a new Shapefile layer is created on top of the GSI map layer. The new Shapefile layer uses the same

coordinate system to avoid the mismatching issue with the GSI map layer when creating the sidewalk annotation. The Polygon is selected for the geometry type of this layer since this type best describes the general shape of the sidewalk. The newly created Shapefile layer will be the sidewalk layer that contains all sidewalk annotations on the GSI map layer. On the sidewalk layer, various polygon shapes can be created by defining the polygon's vertices. To create polygons that fit the sidewalk, vertices are placed along the edge of sidewalks. The more vertices selected, the better the polygon shape will fit the sidewalk. To maintain the annotation precision and reduce the time consumption, straight sidewalks have fewer vertices than curved sidewalks. The contour of the sidewalk polygon, along with the polygon itself, are assigned to the same colour code. The purpose of the same colour code is to help reduce the complexity of the image processing part. An example of the original and annotated aerial image is shown in Figure 3.1 and Figure 3.2, respectively, where the sidewalk annotation is assigned an RGB value of (255, 127, 0).

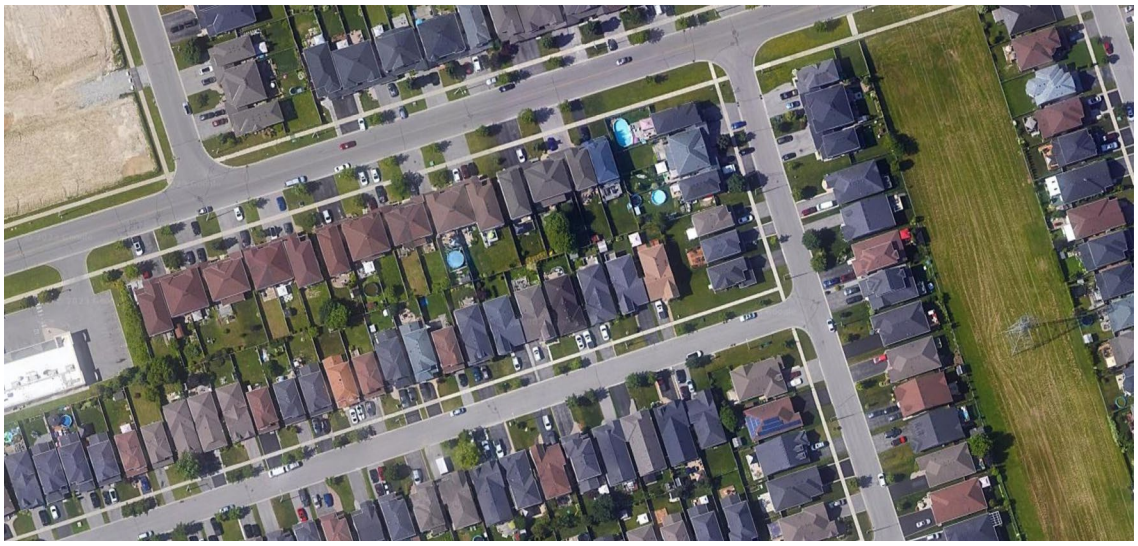

Figure 3.1 Original aerial image at 1:1000 scale

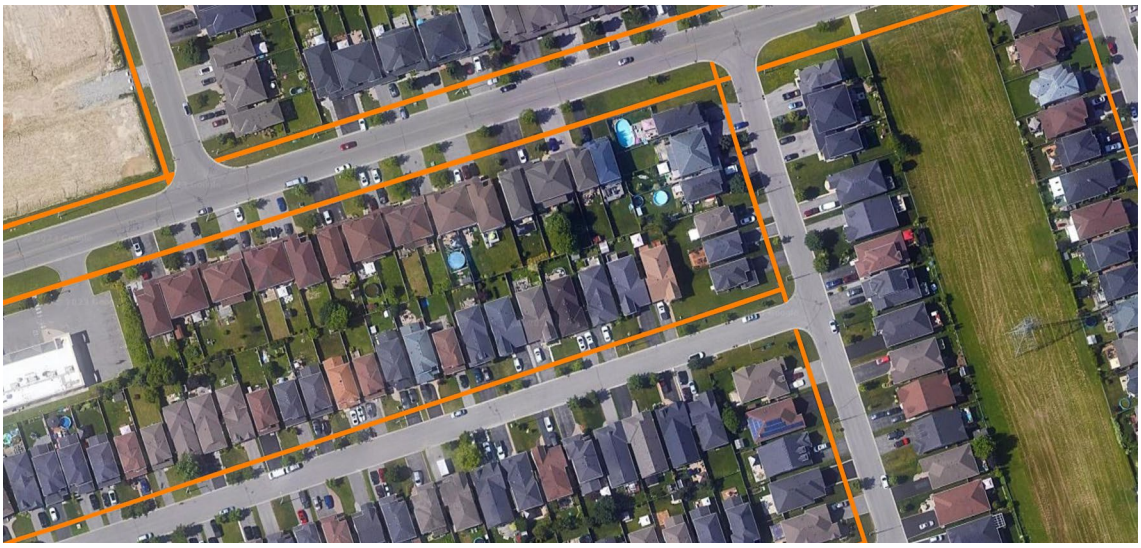

Figure 3.2 Annotated aerial image at 1:1000 scale

The sidewalk annotation covers Ontario Tech University's North campus, residential areas, and urban traffic of a part of North Oshawa. Once the sidewalk annotation has been completed, some image processing procedures will be conducted to prepare the sidewalk dataset.

3.3 Image Processing

3.3.1 Aerial and Sidewalk Annotation Image Exportation

The first procedure in the image processing step is to export the aerial image as well as the sidewalk annotation image. The sidewalk annotation image here differs slightly from Figure 3.2, as seen in Figure 3.3. Since the only feature that the DL algorithm will learn is the sidewalk, features other than sidewalks will all be treated as the background. This leads the sidewalk annotation image to only contain two types of pixels: white (255, 255, 255) and orange (255, 127, 0). Since the annotation is done on a large city-scale aerial map, it is necessary to crop the map into multiple same-sized images and export them into proper format in an efficient manner. To do so, the QGIS Python plugin is used to go through the aerial map automatically, initialize the image size, crop the image, and save all images in

a proper order for both aerial images and annotation images. The pseudocode for this process can be seen as follows:

Algorithm 1 Aerial Image and Annotation Image Export

Input: QGIS Project, vertical_num, horizontal_num
Output: Aerial and annotation images

```
Function render_and_convert():
    Initialize QGIS project instance
    Initialize QGIS map settings
    Set the destination CRS (Coordinate Reference System)
    Set layers to those in the current QGIS project
    Define output size
    Set the output size in map settings
    Get current extent from map canvas
    Extract x_min, y_min, x_max, y_max from current extent
    Store these in original_extent
    Define shift_distance_horizontal as width of original_extent
    Define shift_distance_vertical as height of original_extent
    Initialize image_count to 0

    For j from 0 to vertical_num:
        For i from 0 to horizontal_num:
            Compute new_extent by shifting original_extent by i * shift_distance_horizontal
                    horizontally and j * (-shift_distance_vertical) vertically
            Set the new_extent in map settings
            Initialize and start parallel map rendering job
            Wait for rendering job to finish
            Save rendered image to temporary PNG file
            Call convert_to_geotiff() to convert temporary PNG to GeoTIFF
            Increment image_count

Function convert_to_geotiff(src_filename, dst_filename, extent, img_size):
    Open source file using GDAL
    Initialize GDAL GeoTIFF format driver
    Create destination dataset as a copy of the source dataset
    Compute pixel width and height based on extent and image size
    Set GeoTransform using calculated pixel width, pixel height, and extent
    Close source and destination datasets

Call render_and_convert()
```

In the above Algorithm, the entire annotated map is essentially divided into a grid map. For each grid, an aerial image and its corresponding sidewalk annotation image are

cropped. *vertical_num* and *horizontal_num* are the height and width of the grid map which can be adjusted based on the coverage of the sidewalk annotation. The program starts from the first cell located on the top left corner and crops the image from left to right and top to bottom. The aerial images and sidewalk annotation images are saved in two different formats: Portable Network Graphics (PNG) and Georeferenced Tag Image File Format (GeoTIFF). PNG is an image format that commonly used in training DL models. The saved PNG files here will be used for sidewalk extraction model training. GeoTIFF, on the other hand, is an extension of the Tag Image File Format (TIFF). A GeoTIFF file contains additional metadata that allows it to be placed in the correct geographical location. It is specifically designed to be used by Geographic Information Systems (GIS) software, which requires information about the coordinate system and projection rules. The exported GeoTIFF files are useful for HD maps in GPS-based localization and route planning, and will be used for the route planning part of the thesis.

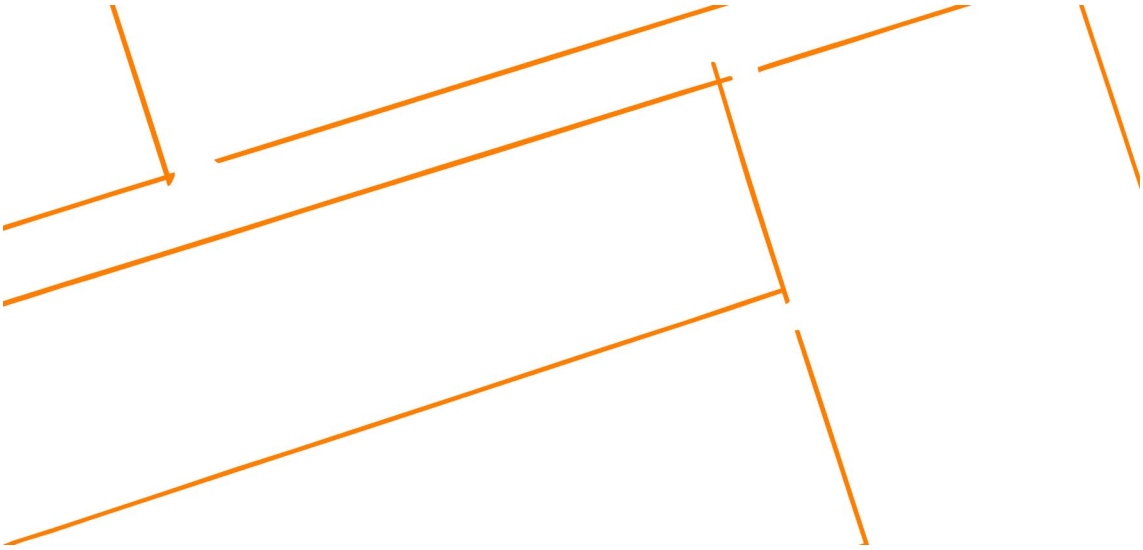

Figure 3.3 Sidewalk Annotation Image

3.3.2 Binary Sidewalk Masks Creation

A very common application in the DL field is the binary classification. The task for a DL algorithm in a binary classification is to learn and differentiate a specific object type from a give data source. Sidewalk extraction is a binary classification application where the DL model needs to learn from the sidewalk dataset and differentiate the sidewalk feature from all other background features. In binary classification for sidewalk extraction, the sidewalk annotation image, also known as mask, is required to be in a binary format, where the image contains only two values (colours): 0 (black) and 1 (white). In a mask image, shown in Figure 3.4, sidewalks are represented by 1 and background are represented by 0.

The sidewalk annotation image exported from the previous section is used to create the binary masks. This process is also made automatic by using Python programming language and the OpenCV library. Each sidewalk annotation image (Figure 3.3) was first read into a matrix and converted it into grayscale using functions *cv2.imread* and *cv2.IMREAD_GRAYSCALE* from OpenCV. This step essentially converted the 3-channel (RGB) image to a single channel image where pixel values range from 0 to 255. A binary thresholding method, Eq. 3.1, was applied on the grayscale image, where all pixel intensity values $I(x, y)$ that are greater than a specified threshold value are set to a maximum value maxVal and the remaining pixel values are set to zero.

$$dst(x, y) = \begin{cases} maxVal, & I(x, y) > threshold \\ 0, & otherwise \end{cases} \tag{3.1}$$

The threshold value is defined by using the Otsu's Thresholding method. Otsu's method, Eq. 3.2, is an advanced way to determine an optimal threshold *t* by maximizing the between-class variance of the thresholded black and white pixels.

$$\sigma_b^2(t) = w_1(t) \cdot w_2(t) \cdot (\mu_1(t) - \mu_2(t))^2 \tag{3.2}$$

where $w_1(t)$ and $w_2(t)$ are the sum of the probability of the two classes separated by some threshold *t*, Eq. 3.3 and 3.4, and $\mu_1(t)$ and $\mu_2(t)$ are the average intensity values for the two classes, Eq. 3.5 and Eq. 3.6.

$$w_1(t) = \sum_{i=0}^{t} p(i) \tag{3.3}$$

$$w_2(t) = \sum_{i=t+1}^{L-1} p(i) \tag{3.4}$$

$$\mu_1(t) = \frac{1}{w_1(t)} \sum_{i=0}^{t} i \cdot p(i) \tag{3.5}$$

$$\mu_2(t) = \frac{1}{w_2(t)} \sum_{i=t+1}^{L-1} i \cdot p(i) \tag{3.6}$$

where *i* is the intensity value of a selected pixel and *L* is the number of possible intensity levels in the matrix, which is 256 for an 8-bit grayscale image. The Algorithm for the binary mask creation process is shown below:

Algorithm 2 Binary Masks Creation

Input: Sidewalk annotation images, sidewalk_annotation_image_dir, binary_mask_dir
Output: Sidewalk binary masks

Initialize a list of PNG filenames from sidewalk_annotation_image_dir

For each png_name **in** list of PNG filenames:
  Read the image into a grayscale matrix im_gray using *cv2.imread*
  Apply thresholding to im_gray to obtain binary black and white image im_bw using *cv2.threshold*
  Apply Otsu's thresholding method to automatically determine the threshold value

  Invert the binary image using *cv2.bitwise_not()* to get invert
  Save the inverted image to binary_mask_dir

In Algorithm 2, the OpenCV function *cv2.bitwise_not* was used after the binary thresholding to invert the black and white pixel values. The program essentially turns all the orange sidewalk annotations into white pixels and all white backgrounds into black pixels. The examples of result masks along with their aerial images are shown in Figure 3.4, where the aerial images are on the left column and the masks are on the right column. Both aerial images and masks have a size of 1835×875 pixels.

3.3.3 Image Augmentation

Image augmentation is an image processing technique involving image transformations, such as horizontal flip, vertical flip, and random rotation. It is commonly used in deep learning, particularly for tasks like object detection, image classification, and semantic segmentation. Multiple work has shown that image augmentation is effective in increasing the size of the dataset and enhancing the model performance [25], [31]. When preparing the sidewalk dataset, image augmentation including horizontal and vertical flip are used to increase the size of the dataset. Two other image augmentation methods are also used during the training of the algorithm including random rotation and TrivialAugment [32], which will be discussed in Chapter 4. A sample of the original data

and the augmented data is shown in Figure 3.5 and Figure 3.6, where the original image and mask, the horizontal flipped image and mask, and the vertical flipped image and mask are on the first, the second, and the third row respectively.

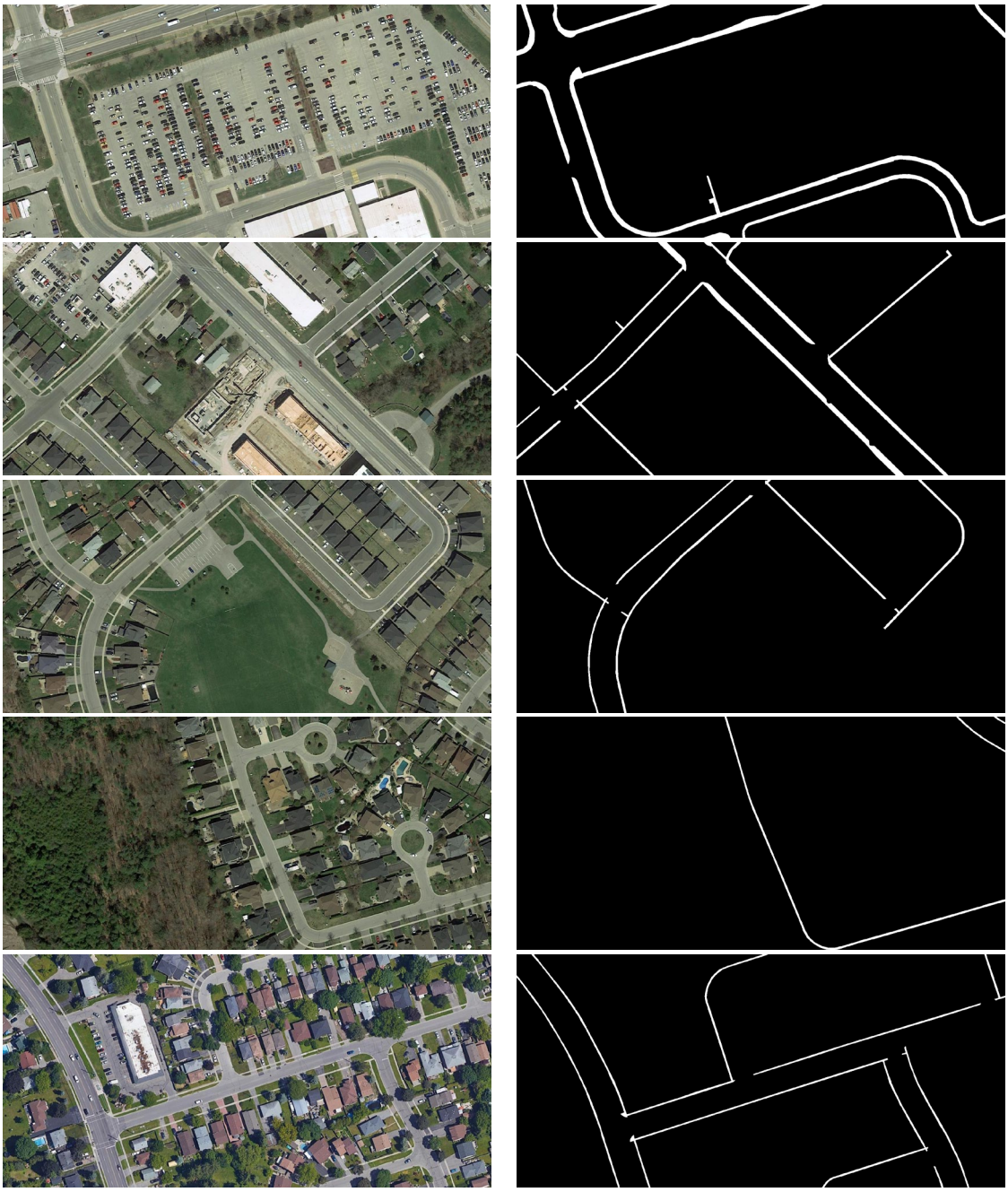

Figure 3.4 Aerial image and sidewalk mask examples

3.3.4 Dataset Split

A general process in preparing a dataset for model training in deep learning is to split the dataset into training, testing, and validating sets. A training set, seen by the neural network, is the dataset that the deep learning model will be trained on or learn features from. It contains the most portion of the dataset, ranging from 60% to 99%, depending on the size and the diversity. A testing and validating set, unseen by the neural network, is the dataset that the deep learning model will be tested and evaluated on after being trained on the training set. A testing or a validating set usually contains 1% to 20% of the entire dataset, also depending on the size and the diversity of the dataset. The testing set is usually used immediately after every training epoch to test the model performance. The validation set is used to validate the model performance again after the model is trained and tested on the training and testing sets. Sometimes the testing or the validating set is neglected to provide more training data for the deep learning model to learn from. When preparing the sidewalk dataset, the *train_test_split* function from the Python library *Scikit-learn* is used to split the dataset into train, test, and validation sets with a ratio of 8:1:1.

3.4 Summary

The finalized sidewalk dataset contains 636 image data pairs, covering the Ontario Tech University's North campus, some residential areas, and some busy traffic areas of Oshawa, Ontario, Canada. There are 508 data pairs in the training set, 65 data pairs in the testing set, and 63 data pairs in the validation set. Each image data pair contains one aerial image and one corresponding sidewalk mask in binary image format. Each image has a size of 1835×875 pixels which will be directly fed into the DL algorithm for training and evaluating.

Figure 3.5 Original aerial image and augmented aerial image

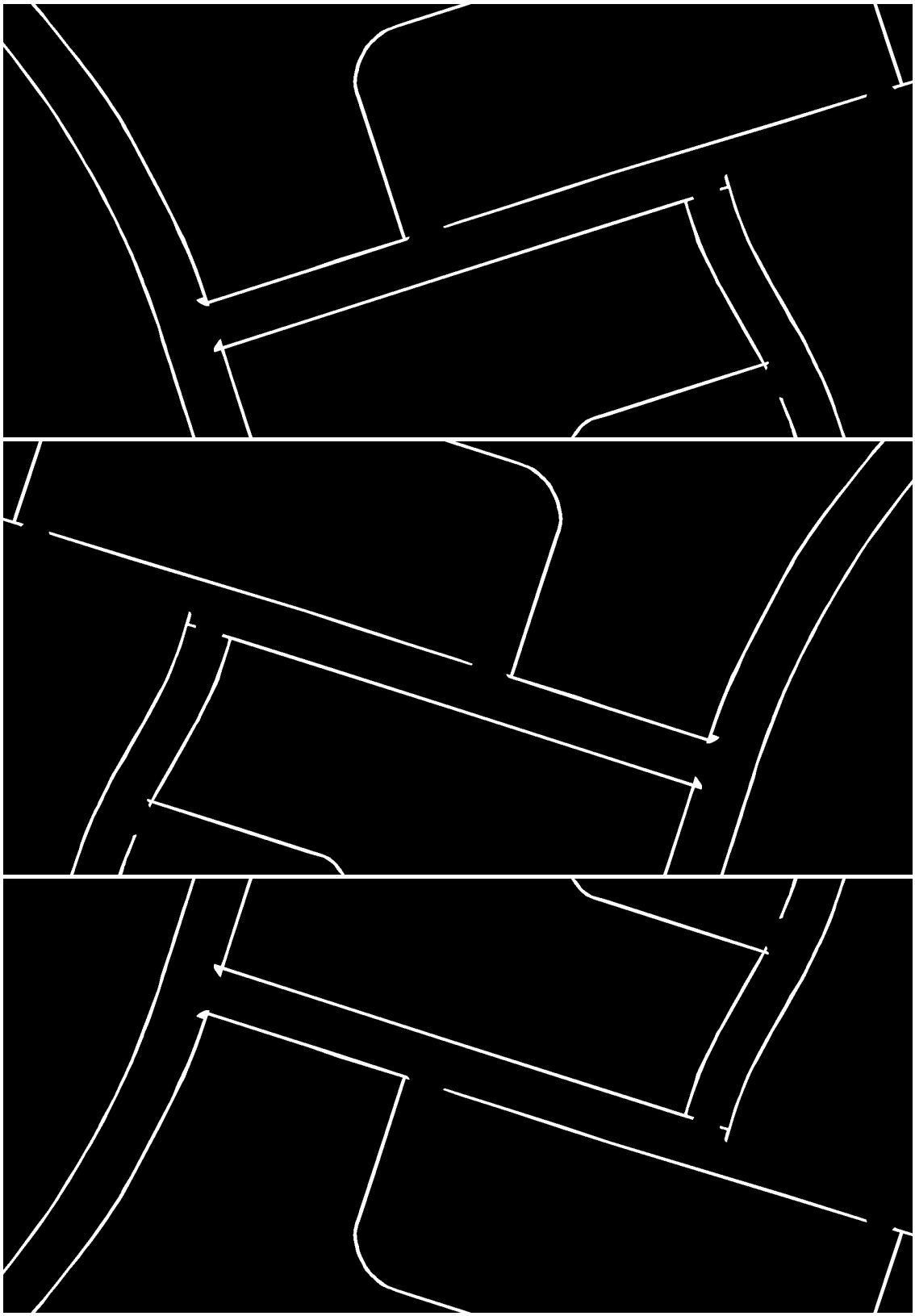

Figure 3.6 Original mask and augmented mask

Chapter 4. Sidewalk Extraction on Aerial Images

4.1 Introduction

Sidewalk extraction is a task to either manually or automatically annotate all sidewalk features on aerial/satellite images. It is a branch of the road network extraction research. Traditionally, road network is manually labelled using GIS software such as OpenStreetMap (OSM), QGIS, and ArcGIS. This is an exhausting and inefficient process which requires a large amount of labor work. The quality and the precision of labelled networks are also not guaranteed. In recent years, with the fast development of powerful GPUs, computing power and speed of computers have dramatically increased. This makes it possible to train bigger and deeper neural networks. Researchers have also proposed various DL-based methods to automate the road network extraction process. This Chapter presents a DL-based automatic sidewalk extraction on aerial images method using semantic segmentation with image augmentation and transfer learning techniques. Various segmentation algorithms will be used to train on the sidewalk dataset and the sidewalk extraction performance of each model will also be compared. The extraction result with the best performance will be used for the segmentation refinement stage. In the refinement stage, a path planning algorithm is used to fix any discontinuity in the sidewalk segmentation. Finally, the extracted and refined sidewalk extraction results will be presented.

4.2 Sidewalk Extraction with Semantic Segmentation (Part I)

4.2.1 Semantic Segmentation

The goal of semantic segmentation is to correctly give the same color code to pixels of an image that belong to the same class/category. It can be either a binary or a multi-class

classification DL task. Sidewalk extraction is a binary classification task using semantic segmentation since sidewalk and background are the only two classes on the aerial image. When training a segmentation DL model, the inputs are aerial images, and the outputs are sidewalk segmentation predictions. The predictions are compared with the ground truth labels (mask images) to compute the training loss and other evaluation metrics.

Several SOTA semantic segmentation models are selected to train and compare on the sidewalk dataset, including U-Net [25], LinkNet [12], FPN [21], MA-Net [33], UNet++ [34], PSPNet [35], PAN [36], and DeepLabV3+ [37]. U-Net [25] was originally designed for biomedical image segmentation. It utilizes the famous encoder-decoder architecture to down sample and up sample the image tensor to learn features and make predictions. U-Net was a significant advancement in the field of semantic segmentation due to its introduction of skip connections between the encoder and decoder pathways. When input image tensors are down sampled in the encoder pathway, the spatial information are lost. The skip connections concatenate the image tensor from the encoder and the decoder together to help recover the spatial information, thus enabling the model to produce better segmentation results. It is also known for handling small datasets. UNet++ [34] is a modified and advanced variant of U-Net. It introduced nested skip pathways, dense skip connections, and deep supervision to the original U-Net architecture to improve performance and reduce the chance of false positives and false negatives in the segmentation maps. Instead of having a single skip connection between each encoder and decoder pathway, UNet++ utilizes multiple nested skip pathways between all encoders and decoders to capture and propagate more low-level features to the decoder. The dense skip connections help improve gradient flow and feature reuse to capture intricate details in

images by facilitating the flow of features across the network. Deep supervision ensures the model to perform well not only at the final layer but also at intermediate layers. LinkNet [12], similar to U-Net, also adopts the encoder-decoder architecture. LinkNet employs a more efficient decoder with fewer parameters and uses the residual blocks [5] to mitigate the vanishing gradient problem. It is designed to perform segmentation in a computationally efficient manner while still maintaining high accuracy, making it suitable for real-time segmentation applications. FPN [21] was primarily designed for object detection applications and later was also used in semantic segmentation. It consists of a bottom-up pathway, a top-down pathway, and lateral connections. The bottom-up pathway usually uses a pretrained model like ResNet [5]or VGG [38] to extract features at different scales as the input image tensor passes through the layers. The top-down pathway upsamples the coarse feature spatially and enhances them using features learned from the bottom-up pathway. The lateral connections merge features from both pathways at the same spatial size using techniques like element-wise addition or concatenation. The above three key components allow the network to make predictions at multiple scales, which is useful for detecting objects or features that vary in scales. PAN (Pyramid Attention Network) [36] was primarily designed for tasks like semantic segmentation. It adopts the attention mechanism and feature fusion techniques to capture features with fine-grained details. Additionally, the pyramid pooling module in the architecture performances feature pooling at different scales, making the network more robust to variations in scale and position. PSPNet (Pyramid Scene Parsing Network) [35], similar to PAN, also contains the pyramid pooling module. The network generally starts with a pre-trained backbone model, such as ResNet [5], to learn low-level features from the input image tensor. The pyramid

pooling module takes those low-level features and pools them at various grid scales. The pooled features are then upsampled to the original size and concatenated to the original feature map. The network then applies a convolutional layer and a softmax layer after the pyramid pooling to refine the results and make segmentation prediction. PSPNet is capable of capturing scenes at different scales and fusing the features to improve segmentation performance. MA-Net (Multi-scale Attention Network) [33] was also primarily designed for semantic segmentation tasks, specifically in liver and tumor segmentation. Similar to PAN, it also utilizes the attention mechanism to capture rich contextual dependencies. In addition, it incorporates the MFAB (Multi-scale Feature Aggregation Block) to capture the channel dependencies between any feature map by multi-scale semantic feature fusion [33]. Multiple variants based on the MA-Net, such as STDC-MA network [39] and Multi-Attention UNet [40], have been proposed for semantic segmentation tasks in different fields, such as remote sensing, autonomous driving, and intelligent transportation. Lastly, DeepLabV3+ [37] is an extension of the DeepLabV3 [41] model designed for semantic segmentation. The model introduces the atrous convolution (also known as dilated convolution) to prevent the number of parameters or the computational complexity from increasing when having a larger field of view. Similar to PSPNet, DeepLabV3+ also uses the spatial pyramid pooling to capture multi-scale information from feature maps. In addition, the model proposed an encoder-decoder architecture to refine the segmentation results, especially along the object edges/boundaries. The model is known for achieving high segmentation accuracy while retaining speed and computational efficiency.

4.2.2 Model Training and Testing

The process of training a DL model for semantic segmentation tasks can be viewed in Figure 4.1. The input of the neural network is the aerial image. Since the training dataset is relatively small, the transfer learning technique is used during the training process. In deep learning, transfer learning refers to the practice of taking a pre-trained model, usually a previously SOTA model trained on a large dataset, and adopting it for a different but related task. It involves two stages: pre-training and fine-tuning. In pre-training, a neural network model, such as ResNet [5] and VGG [38], is trained on a large-scale dataset. The dataset is often much larger and more general than datasets for some specific tasks (e.g., ImageNet [42] dataset contains over 14 million images belonging to approximately 22000 classes). This step is also known as the feature extraction step as the pre-trained model has learned meaningful features which can be repurposed for the segmentation model to learn a specific feature. This stage omits the process of training a neural network completely from scratch, which is generally more time-consuming and lower in performance. In the fine-tuning stage, the pre-trained model is fine-tuned on the new and specific task, such as the sidewalk feature. This stage usually involves replacing the final layer of the pre-trained model and adding a new layer or model to continue training on the new dataset. This step updates the weights from the pre-trained model, so they fit the new feature/task. In the transfer learning step of training the sidewalk segmentation model, the ResNet152 [5] model is used as the backbone for the initial feature extractor and ImageNet [42] is selected as the pre-trained weights. The semantic segmentation model discussed in the previous section comes in after the transfer learning stage to train on the sidewalk dataset and learn

the sidewalk feature. The last layer of the segmentation model is set to a Sigmoid function, Eq. 4.1, as the activation function for binary classification purposes.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.1}$$

where x is the input to the activation function, often referred to as the pre-activated value. The Sigmoid function maps the input x to a value between 0 and 1, which can be interpreted as the probability p of the specific class. The output is defined in Eq. 4.2:

$$\begin{cases} p \geq 0.5: Classify\ as\ the\ positive\ class \\ p < 0.5: Classify\ as\ the\ negative\ class \end{cases} \tag{4.2}$$

where the positive class represents the sidewalk, and the negative class represents the background. In the sidewalk extraction task, after the image tensors going through the segmentation model with Sigmoid function, the model will produce a probability distribution map, also known as the prediction. The probability distribution map defines the probability of each pixel being the sidewalk pixel. The sidewalk prediction from the Sigmoid function is then compared with the ground truth label (mask) to compute the Binary Cross Entropy (BCE) loss, Eq. 4.3:

$$L(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{4.3}$$

where $L(y, \hat{y})$ is the BCE loss, y is the ground truth label, $\hat{y}$ is the prediction, N is the total number of observed images during training, and i is the index of an observation in the training set. Considering two sample predictions from the segmentation model with Sigmoid function, Sample 1: ground truth label $y_1 = 1$ (sidewalk), predicted probability $\hat{y}_1 = 0.9$; Sample 2: ground truth label $y_2 = 0$ (background), predicted probability $\hat{y}_2 = 0.3$. The BCE loss for sample 1 and 2 are $L(y_1, \hat{y}_1) \approx 0.105$ and $L(y_2, \hat{y}_2) \approx 0.357$,

respectively. The average BCE loss of two samples is approximately 0.231. The ultimate goal of training the segmentation model for sidewalk extraction is minimizing the BCE loss calculated above. The process until computing the BCE loss is generally called the forward pass, where the neural network uses current weights and bias to make predictions and compares with the ground truth label to calculate the loss. The BCE loss is then used to update the training weights (and bias) through backpropagation to further lower the loss by using the Adaptive Moment Estimation (Adam) [43] optimizer. The Adam optimizer is an extension of the Stochastic Gradient Descent (SGD) optimization algorithm, combining the advantages of two other popular optimization algorithms: AdaGrad (Adaptive Gradient) and RMSprop (Root Mean Squared Propagation). It uses the first and the second moments to adaptively change the learning rate for each parameter (weights and biases), Eq. 4.4, 4.5, 4.6, 4.7, and 4.8, eventually minimizing the BCE loss.

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t \tag{4.4}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2 \tag{4.5}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{4.6}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4.7}$$

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{4.8}$$

where g_t is gradient of the parameter θ_t (weights and biases) at time t, m_t and v_t are estimates of the first and the second moment of the gradients respectively, β_1 and β_2 are exponential decay rates for the moment estimates, $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected versions of m_t and v_t, α is the learning rate, ϵ is a small scalar added to the denominator to prevent

division by zero, and θ_{t+1} is the updated parameter (updated weights and biases). Adam optimizer is often considered an advanced form of gradient descent to find the local minima. Imagine a person is driving down a valley and the goal is to reach the lowest point (which represents the minimum loss in machine learning tasks). Adam optimizer provides a smarter way to drive down the valley than basic gradient. First, it can adjust the moving speed based on how steep or gentle the slope has been recently. If the person has been going down steep slopes, the person might speed up (this is similar to how Adam adapts the learning rate based on the gradients). Second, if the road is uneven (representing the noisy gradients), Adam helps smooth out these bumps and allows for a more consistent descent. Lastly, Adam remembers the past slopes (gradients) and adjusts the current direction and speed accordingly, like having a memory of the journey that helps guide the future steps. The optimization process is also known as the backward pass, where the weight and bias are updated through backpropagation. After every training epoch, dice loss and intersection over union (IoU) are used as evaluation metrics, defined in Eq. 4.9 and 4.10, to measure the model performance.

$$Dice\ loss = 1 - \frac{2TP}{2TP + FN + FP} \tag{4.9}$$

$$IoU = \frac{TP}{TP + FP + FN} \tag{4.10}$$

where TP, FP, and FN are true positive, false positive, and false negative, respectively, Eq. 4.11, 4.12, and 4.13.

$$TP = \sum_{i=1}^{N} (prediction[i] \times groundTruth[i]) \tag{4.11}$$

$$FP = \sum_{i=1}^{N} (prediction[i] \times (1 - groundTruth[i])) \tag{4.12}$$

$$FN = \sum_{i=1}^{N} ((1 - prediction[i]) \times groundTruth[i]) \tag{4.13}$$

where N is the number of pixels in the image, $prediction[i]$ is the predicted label (either 0 or 1) of the i^{th} pixel, and $groundTruth[i]$ is the actual label (either 0 or 1) of the same pixel. The entire training process repeats until either the given repeating time (number of epochs) reached, or the loss cannot be lowered any more. The training process is also presented in Algorithm 3 and Figure 4.1.

Within each training epoch, a testing loop is called after the training loop. The testing loop is essentially a training loop without the backward pass; thus, parameters will not be updated. It takes the image from the test set (unseen by the neural network) and perform the forward pass on the image to compute the BCE loss, dice loss, and IoU score. The testing results are compared with the training results to observe the model performance and determine if the model is underfitting or overfitting. In the field of deep learning, underfitting refers to model performing badly on both training and test sets. Overfitting means that, instead of learning from the training set, the model simply memorizes the training set and only performs well on the training set but badly on the testing set. The testing loop is essential since a good deep learning model should perform well on both seen and unseen data. It is also helpful for model fine-tuning based on the feedback from both training and testing results.

Algorithm 3 Train Semantic Segmentation Model for Sidewalk Extraction

Input:

SidewalkDataset: Dataset containing images and corresponding ground truth
ResNet152_Pretrained_Weights: Pre-trained weights from ImageNet
Num_Epochs: Number of epochs of training
Learning_Rate: Learning rate (α)

Output:

Trained_model: A semantic segmentation model trained for sidewalk extraction

Initialize ResNet152 with pre-trained ImageNet weights
Feature_Extractor = Initialize_ResNet152(ResNet_Pretrained_Weights)
Initialize Semantic Segmentation Model
Segmentation_Model = Initialize_Segmentation(Feature_Extractor)
Initialize Transformation
Transformation = Initialize_Transformation(Random_Rotation, TrivialAugment)
Initialize Optimizer with Learning Rate
Optimizer = Initialize_Optimizer(Segmentation_Model, Learning_Rate)
Initialize BCE Loss Function
Loss_Function = Initialize_BCE_Loss()
for epoch = i to Num_Epochs **do**:
for each (Batch_Images, Batch_Labels) in SidewalkDataset **do**:
Predictions = Segmentation_Model.Forward_Pass(Batch_Images)
Loss = Loss_Function(Predictions, Batch_labels)
Compute_Gradients(Loss)
Optimizer.Step()
Optimizer.Zero_Grad()
end for
Validate and log metrics
end for

Save Trained_Model

Return Trained_Model

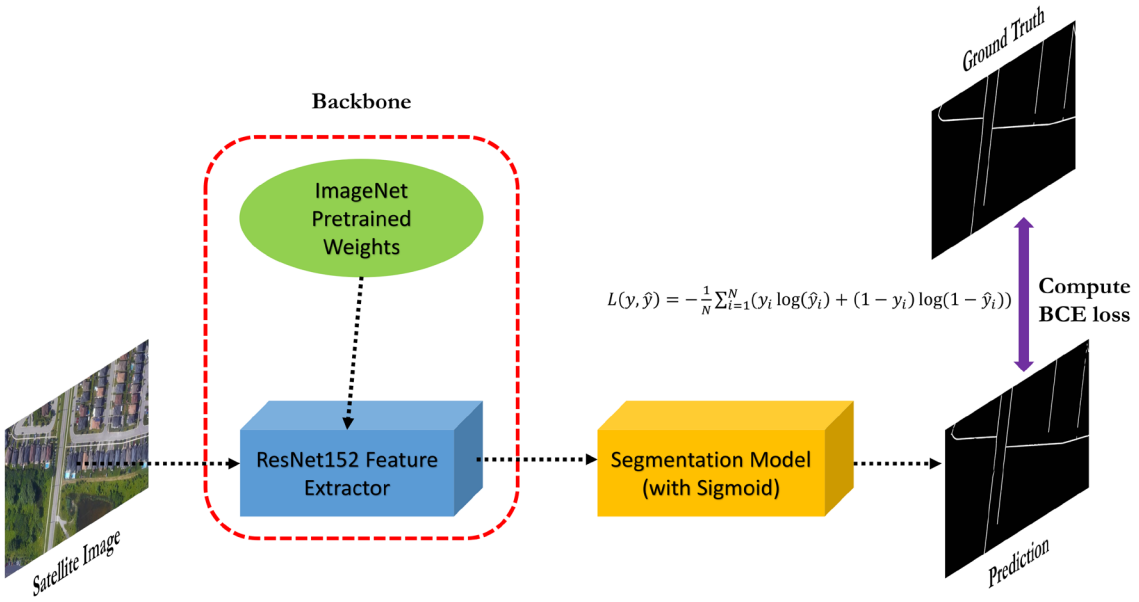

Figure 4.1 Semantic Segmentation with Transfer Learning

The training and testing of the semantic segmentation model is conducted on the Segmentation Model's framework [44] using Python programming language with the PyTorch machine learning framework. During training, two image augmentation techniques are used on the training data, including random rotation and TrivialAugment [32]. The *RandomRotation* transformation function from PyTorch rotates the input image randomly within the a given angle range and fills the corner with black pixels after the rotation. Examples of random rotation augmentation is shown in Figure 4.2, Figure 4.3, Figure 4.4, Figure 4.5, and Figure 4.6, where the Figure 4.2 is the original aerial image, and the rest four are the randomly rotated images. TrivialAugment, on the other hand, is a mix of random transformations, such as rotation, flipping, changing the brightness, and changing the contrast. Instead of transforming an image multiple times, it transforms an image only once using a random transform from a given list with a random strength number. Examples of TrivialAugment on Figure 4.2 is shown in Figure 4.7, Figure 4.8, Figure 4.9, and Figure 4.10.

Figure 4.2 Original aerial image

Figure 4.3 Random Rotated Image #1

Figure 4.4 Random Rotated Image #2

Figure 4.5 Random Rotated Image #3

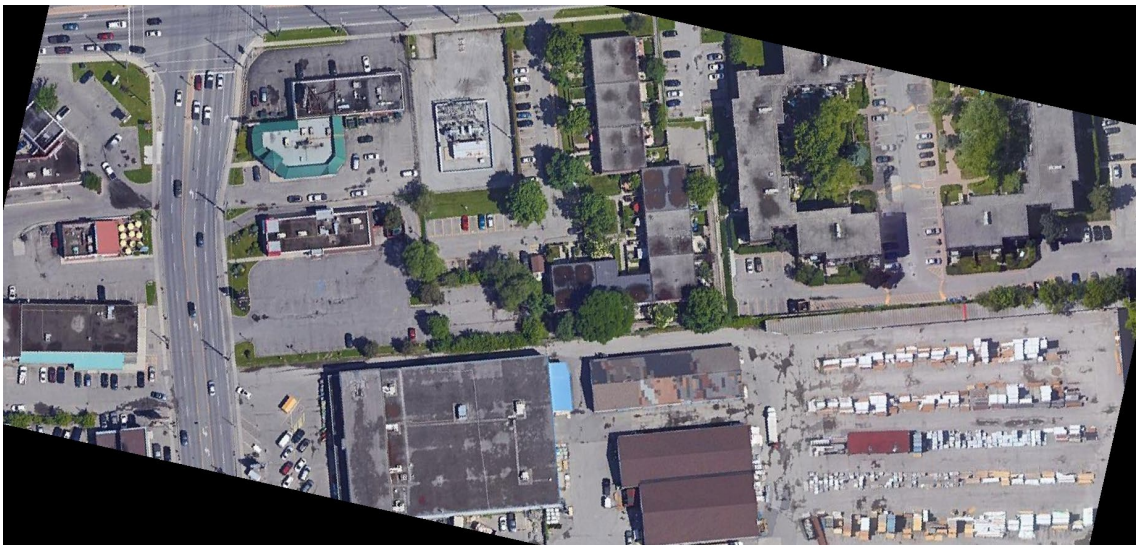

Figure 4.6 Random Rotated Image # 4

Figure 4.7 TrivialAugment Image #1

Figure 4.8 TrivialAugment Image #2

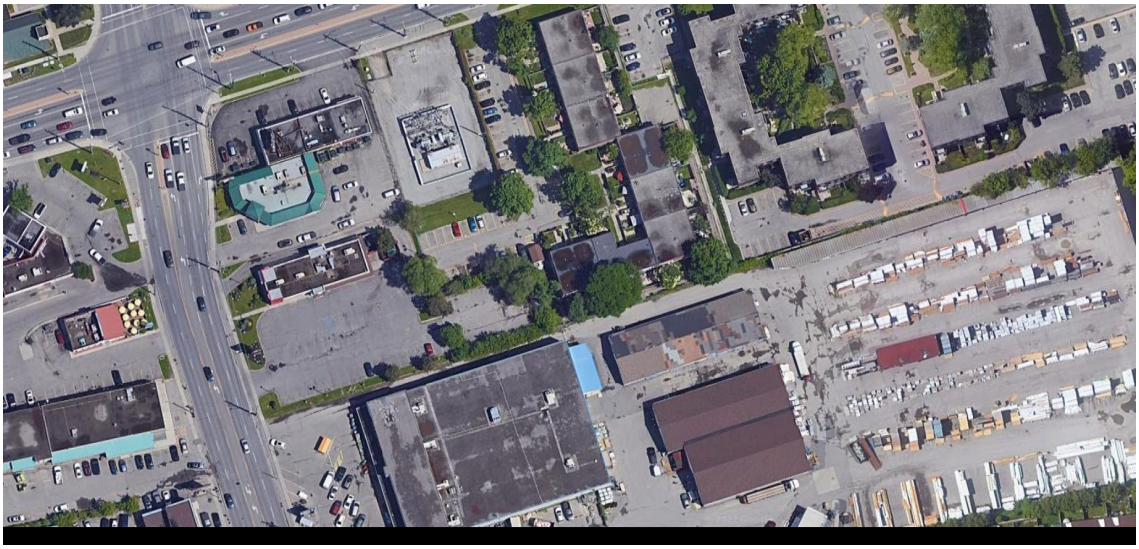

Figure 4.9 TrivialAugment Image #3

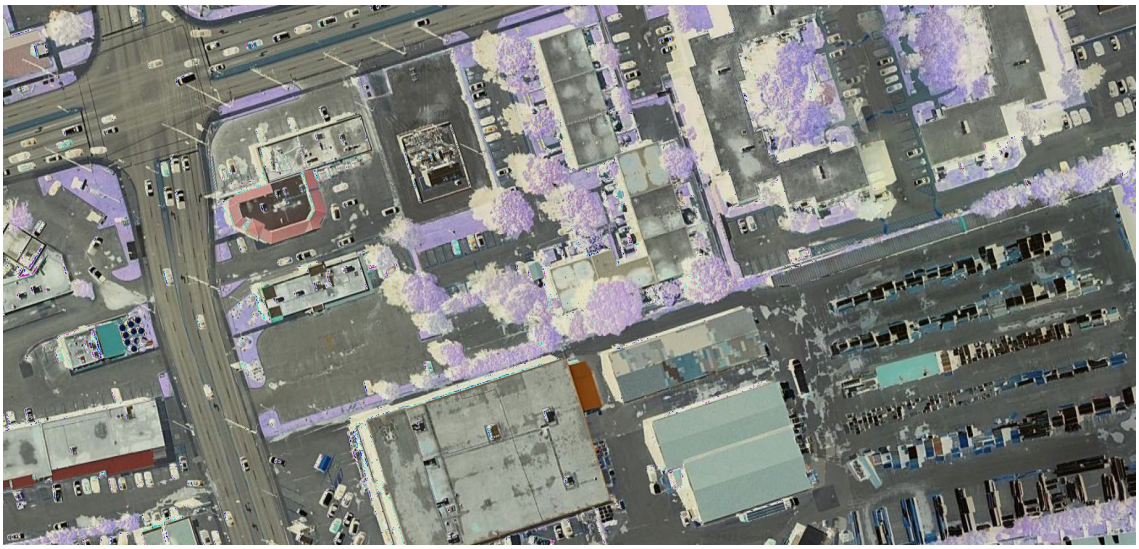

Figure 4.10 TrivialAugment Image #4

The two image augmentation techniques applied during training artificially increase the size of the training dataset, which is beneficial when the training set has a limited amount of training data. They also introduce variability in the training data, acting as a form of regularization, to help prevent overfitting. This is particularly important in deep learning model training when the model has a large number of parameters and is prone to overfitting on small datasets. Models trained with augmentation also have a better generalization ability, which enhances the model performance on unseen data. Furthermore, since all aerial images are taken by satellite, there will be difference in brightness, clearness, and contrast between images due to shadow and weather condition. Involving TrivialAugment particularly strengths the generalization ability of the model on those special images. The Adam optimizer is used with a learning rate (α) of 0.0001, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. All segmentation models are trained on the NVIDIA RTX 3090 GPU for 20 epochs. A callback function is implemented in the testing loop to only keep the best testing results, which will be used for the segmentation refinement stage.

4.2.3 Segmentation Testing Results and Evaluation Comparison

The training and testing process are conducted in two rounds. During the first round, TrivialAugment is not used. With the same training parameters for all segmentation models, the UNet++ is observed to achieve the best overall performance. The testing results of the UNet++ model have a dice loss of 0.004037 (the lower the better between 0 and 1) and an IoU score of 0.9922 (the higher the better between 0 and 1). The testing results of the UNet++ model is shown in Table 4.1 (bolded), along with other models' results. In the second round of training, the TrivialAugment is added to the training of the UNet++ model. It is observed that adding the TrivialAugment slightly improves the model performance in both dice loss and IoU score, as shown in Table 4.1 (bolded).

Table 4.1 Segmentation Model Testing Results

Model	Dice Loss	IoU
U-Net	0.004161	0.9918
LinkNet	0.004215	0.9919
FPN	0.02056	0.9599
MA-Net	0.005137	0.99
PSPNet	0.02056	0.9599
PAN	0.01788	0.9662
DeepLabV3+	0.02099	0.9594
UNet++	**0.004037**	**0.9922**
UNet++ w/TrivialAugment	**0.003955**	**0.9923**

The dice loss and IoU score testing curves are also plotted in Figure 4.11.

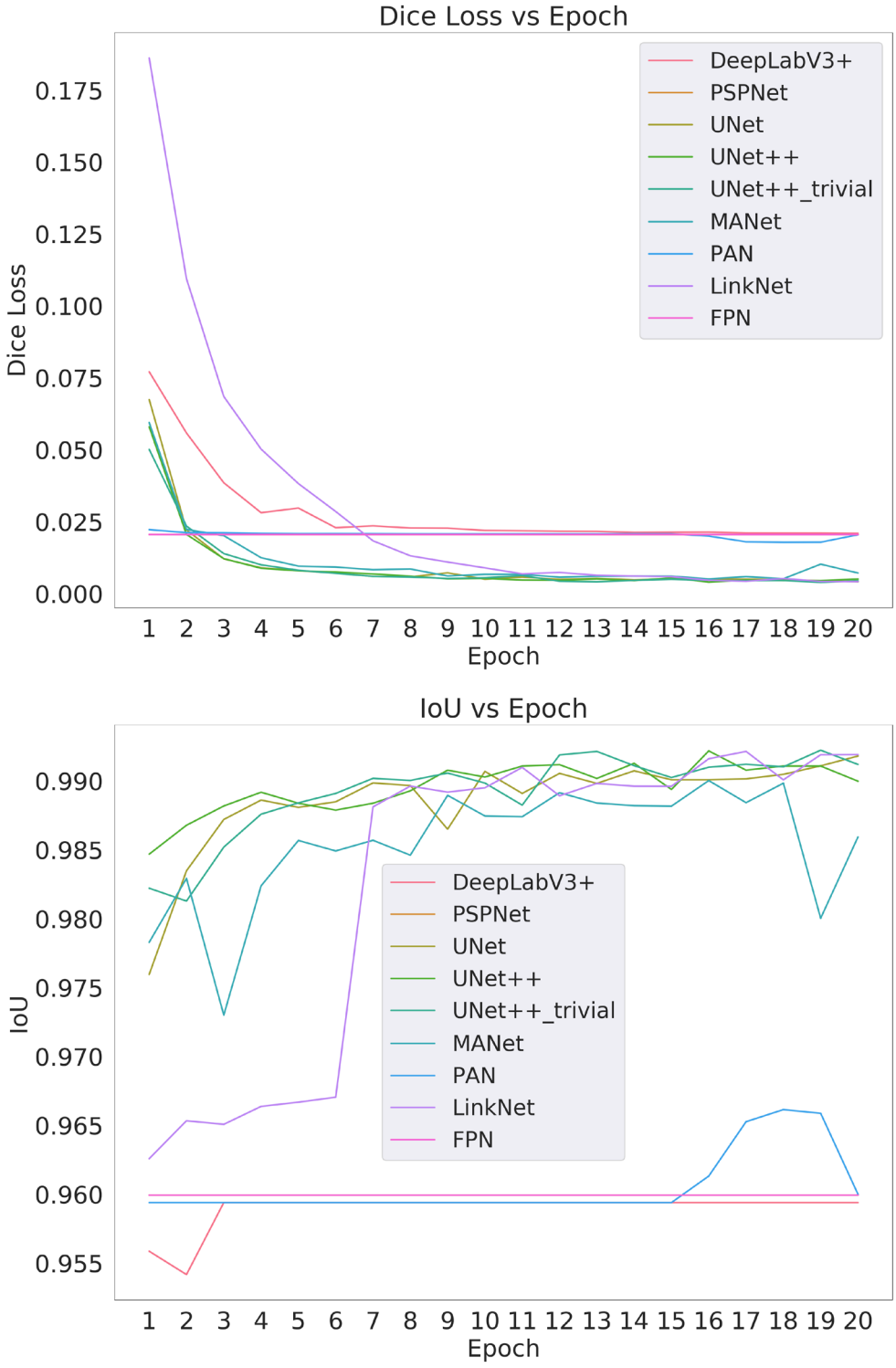

Figure 4.11 Dice loss and IoU score curves

Once the model with the best results is saved, the validation dataset is used for inferencing and evaluating the generalization ability of the model on unseen aerial images. Similar to the testing loop, the validation step is also a forward pass process where the validation images are sent to the trained model. The model simply makes predictions on the image and uses the Sigmoid function to classify every pixel. The probability map will then be converted into a binary image that has the same format as the ground truth where sidewalks are white pixels (255), and backgrounds are black pixels (0). Some of the inferencing results from the model on the validation dataset are shown below, where the original image is on the first row, the ground truth label is on the second row, and the prediction is on the third row for all figures from Figure 4.12 to Figure 4.21:

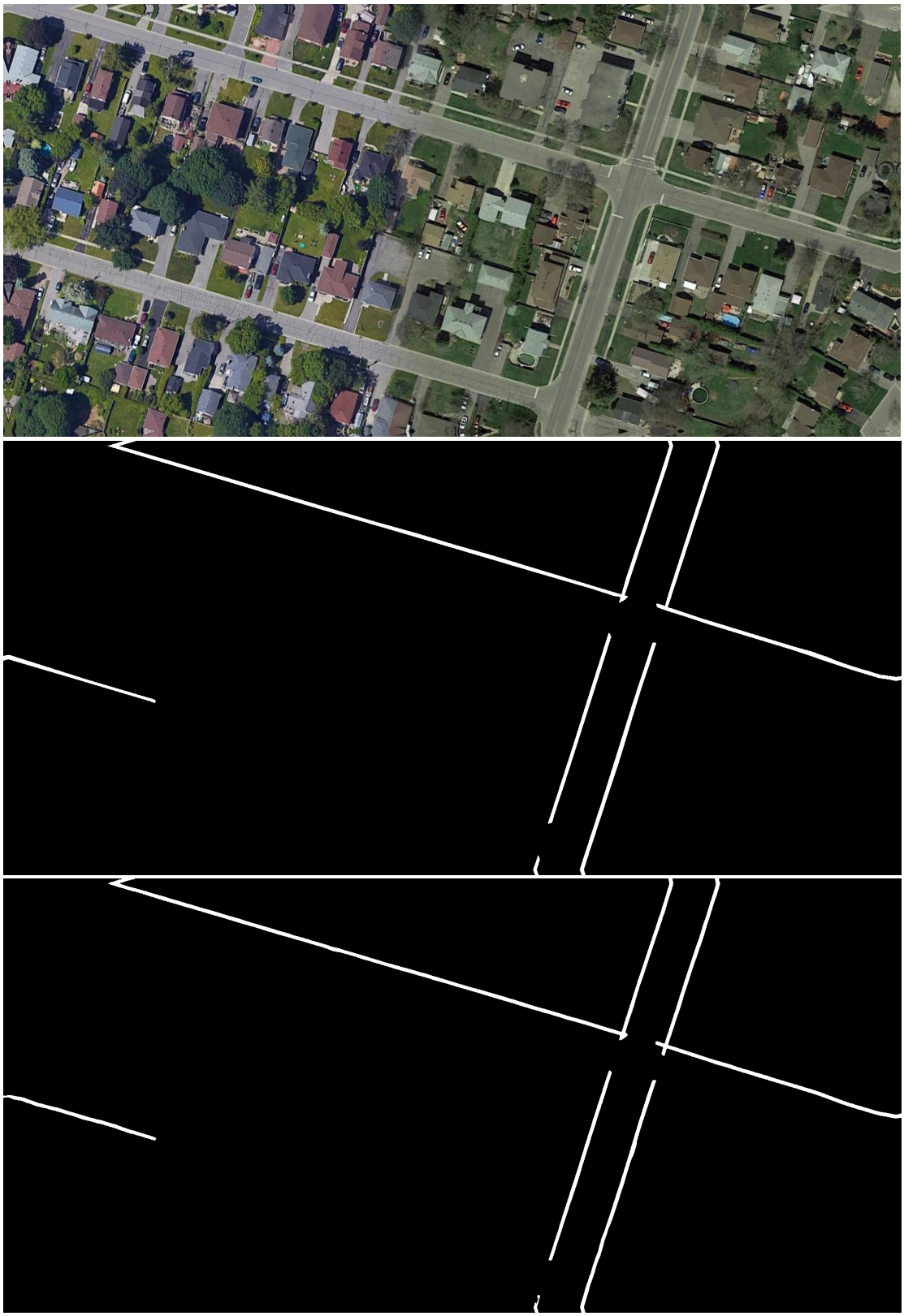

Figure 4.12 Model inference result #1

Figure 4.13 Model inference result #2

Figure 4.14 Model inference result #3

Figure 4.15 Model inference result #4

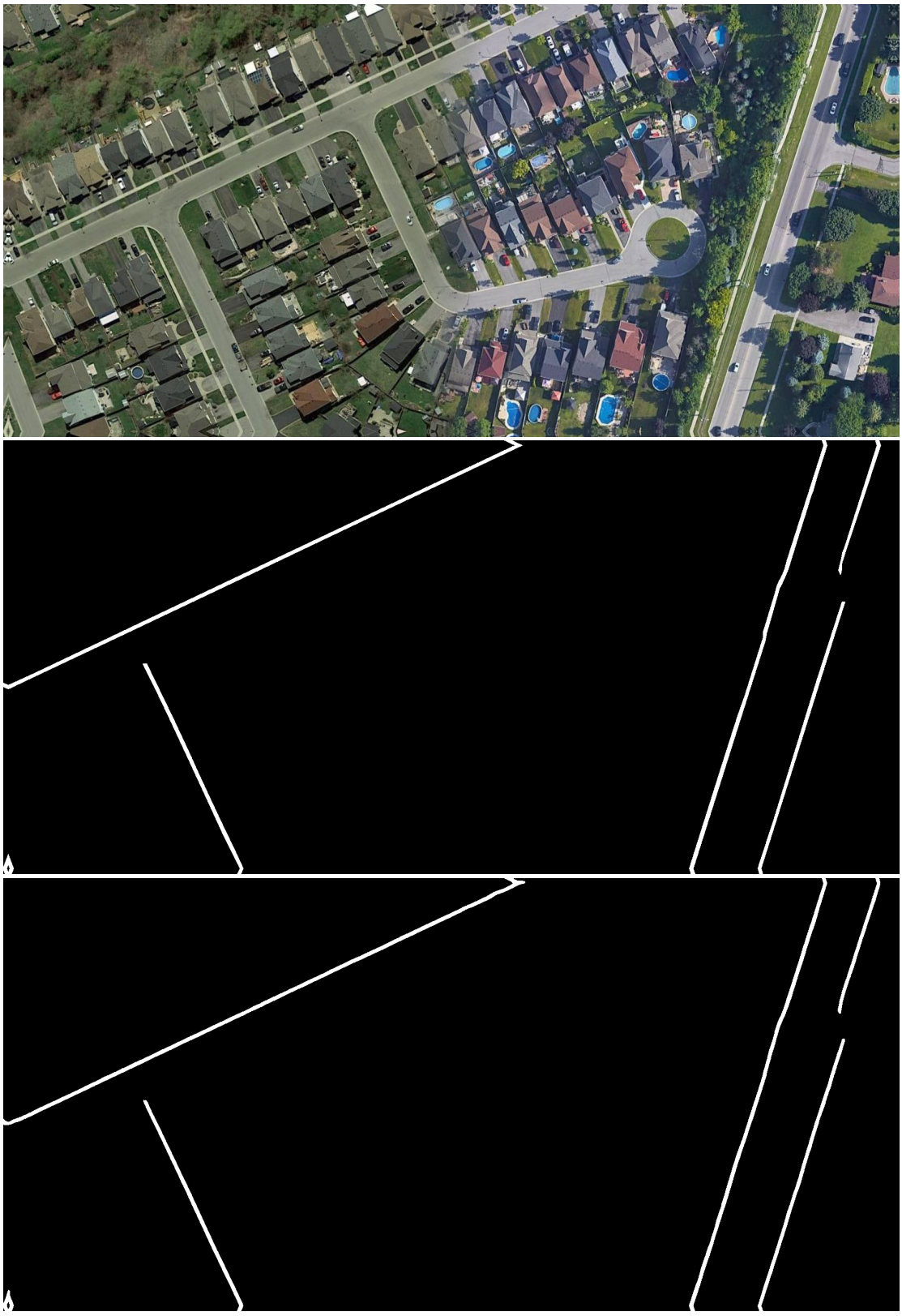

Figure 4.16 Model inference result #5

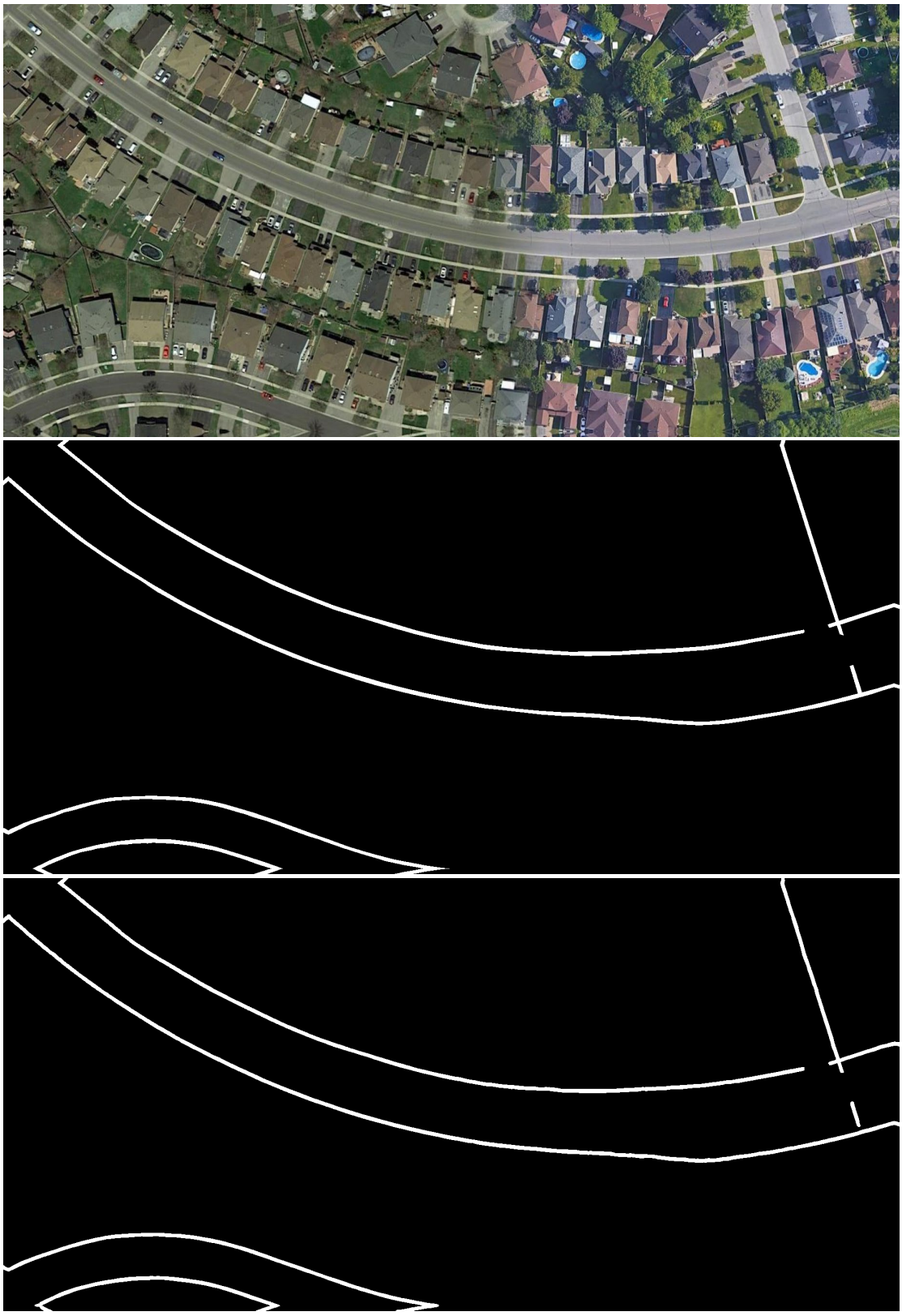

Figure 4.17 Model inference result #6

Figure 4.18 Model inference result #7

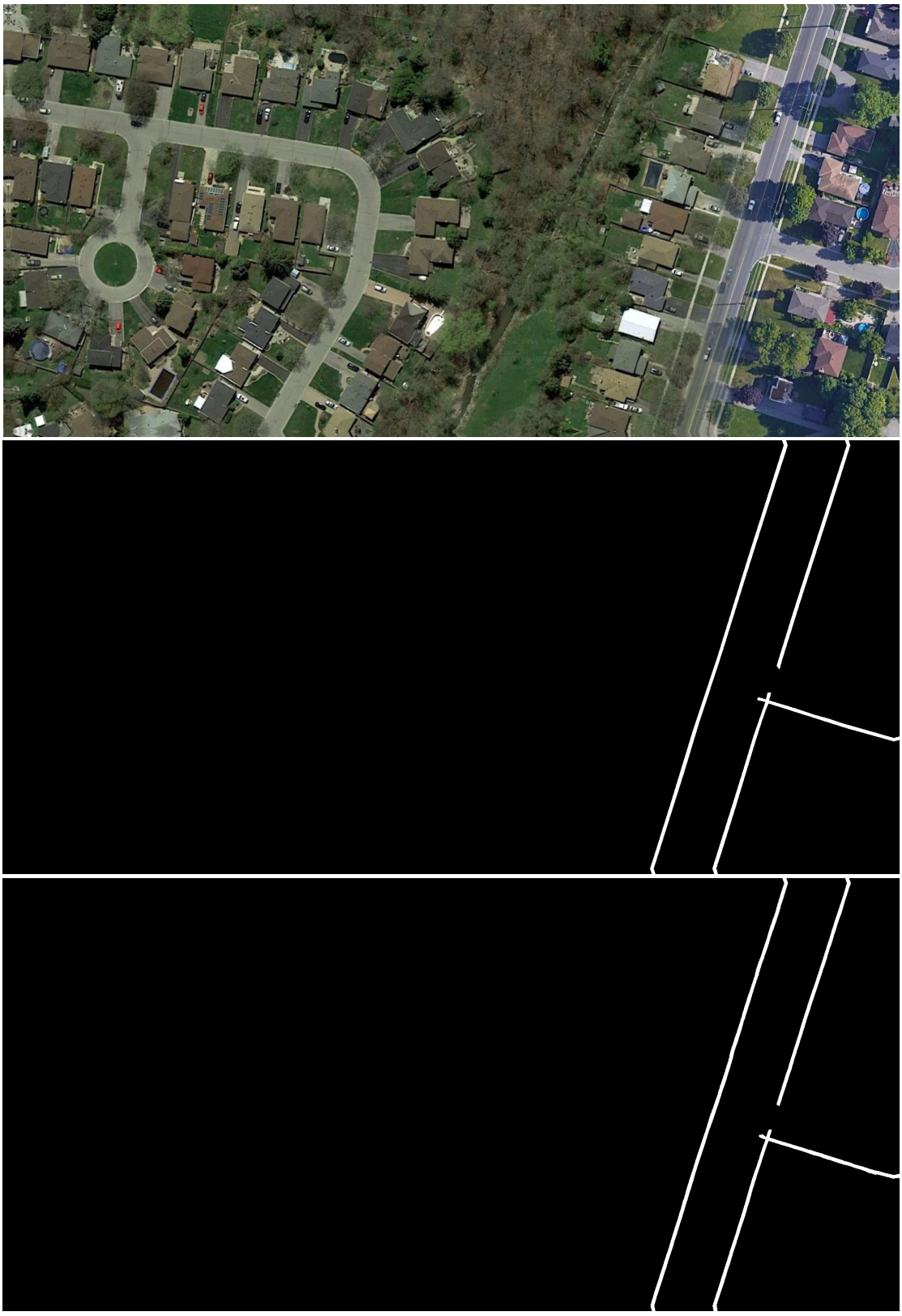

Figure 4.19 Model inference result #8

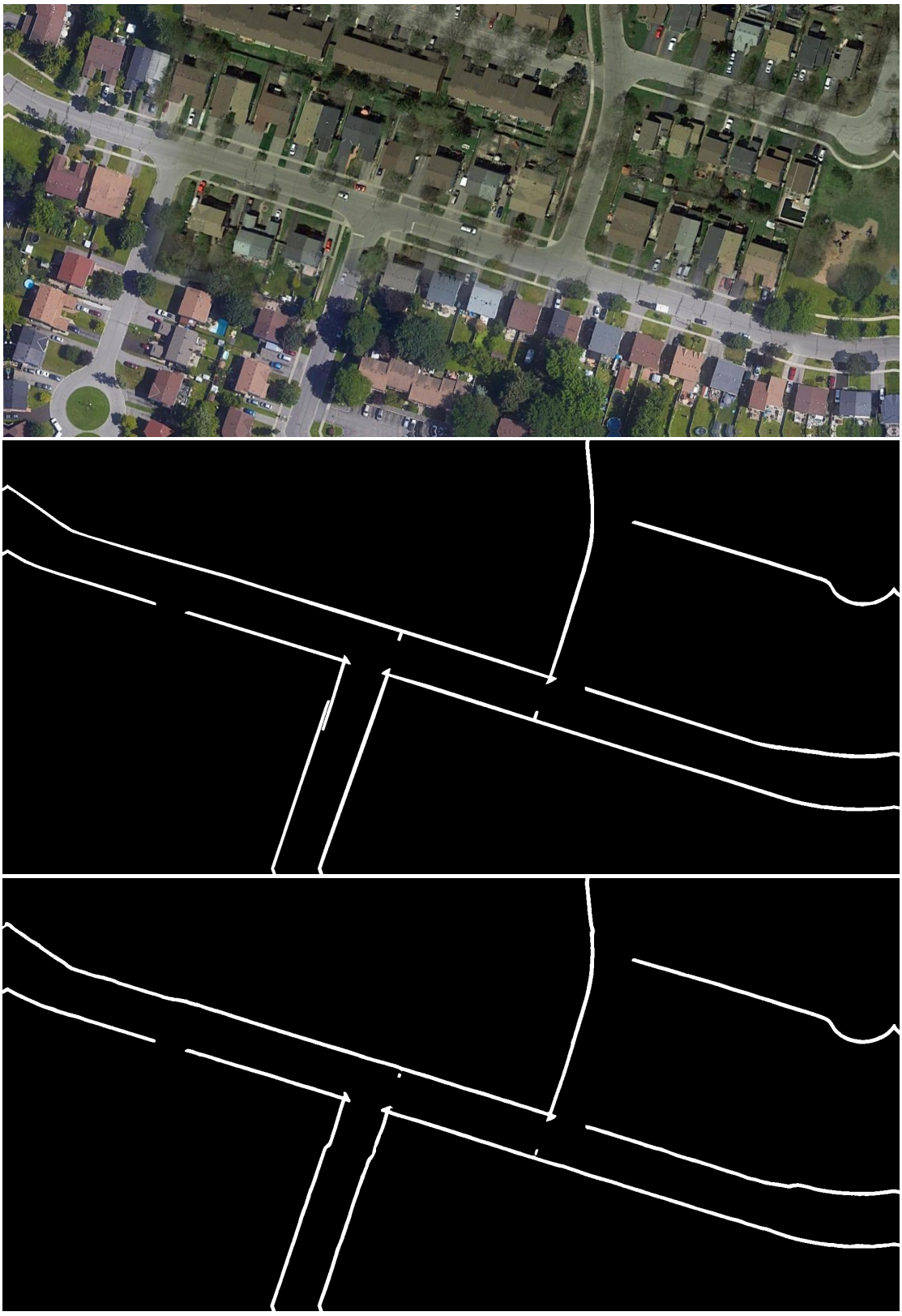

Figure 4.20 Model inference result #9

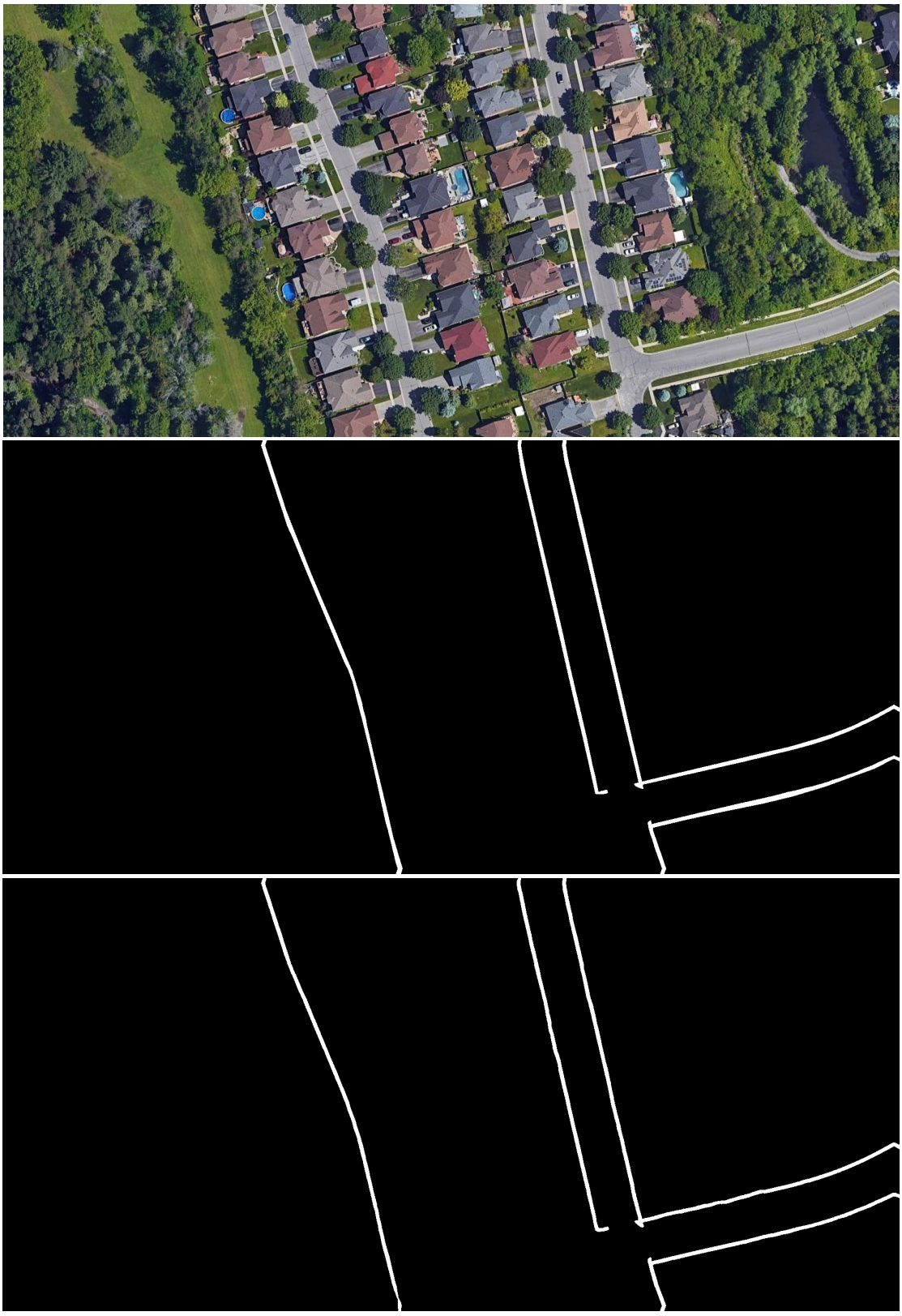

Figure 4.21 Model inference result #10

4.2.4 Summary (Part I)

The first part of this Chapter describes the process of the automatic sidewalk extraction using semantic segmentation in detail, from selecting and initializing the deep learning model to training and evaluating the model. The testing results shows the incredible performance of the trained segmentation model on unseen images. The best performing model, UNett++ with TrivialAugment, has a dice loss of 0.003955 and an IoU score of 0.9923, proving that the TrivialAugment helps enhance the model performance. The inferencing results from Figure 4.12 to Figure 4.21 also show the incredible generalization ability of the model on unseen images.

Besides the excellent results from above, segmentation-based sidewalk extraction method still faces the segment discontinuity issue due to occlusions. This issue and the solution to it will be further discussed in the second part of this Chapter.

4.3 Sidewalk Segmentation Refinement with Path-Planning Algorithm (Part II)

Segmentation refinement is a post-processing technique commonly used in enhancing the output of a primary segmentation algorithm to improve its accuracy, reliability, or completeness. Since the primary segmentation method may produce errors such as over-segmentation, under-segmentation, or misclassification of pixels, segmentation refinement has been frequently used to correct these errors and achieve more accurate and coherent segmentation results.

In this Chapter, a path-planning-based segmentation refinement method is introduced to fix the broken sidewalk segment issue and further enhance the completeness of the sidewalk extraction method.

4.3.1 The Discontinuity Issue in Sidewalk Segmentation

The discontinuity issue in sidewalk segmentation belongs to the under-segmentation category, which refers to the situation that the segmentation algorithm has failed to distinguish between different objects or structure that should ideally be in separate segments. In segmentation-based road network extraction from aerial images, the discontinuity issue often exists due to occlusion above the road, such as shadows, trees, and large buildings or constructions. This issue happens more frequently on sidewalks than on other road types which makes sidewalk extraction much harder. This occlusion issue causes the segmentation model sometimes labelling the covered sidewalk as the background class (black pixel). A graphical demonstration of such occlusion is shown in Figure 4.22.

Figure 4.22 Occlusion and segmentation discontinuity example

The aerial image used for prediction is at the top of the figure, and the sidewalk prediction is at the bottom. There are two occlusions above the sidewalk, which are located inside the red box on the aerial image. These parts of the sidewalk are fully covered by trees. When it was sent to the trained model for sidewalk segmentation prediction, the segmentation model failed to predict the occlusions as a sidewalk feature, as seen in the red box on the prediction image. Such broken segmentation is critical as it is no longer considered as a sidewalk feature by an autonomous driving system to give proper commands such as

moving forward. Thus, the segmentation refinement is required to fix this issue and produce a complete sidewalk extraction for the autonomous driving system.

4.3.2 Sidewalk Segmentation Refinement with $A*$ Algorithm

The $A*$ path-planning algorithm is used to fix the discontinued prediction issue. First, the endpoints (pixel location) of the broken part of the sidewalk are located. The sidewalk image is then processed into a graph where each pixel represents a navigable node, and the endpoints represent the start and end node respectively. Then, the $A*$ path planning algorithm is added to complete the missing sidewalk by finding the shortest path between two endpoints. $A*$ path-planning algorithm $f(n)$ leverages two functions when generating the shortest path between to nodes: a cost function $g(n)$ and a heuristic function $h(n)$, shown in Eq. 4.14.

$$f(n) = g(n) + h(n) \tag{4.14}$$

In Eq. 4.14, $g(n)$ represents the exact cost of the path from the starting point to any node $\boldsymbol{n}$, while $h(n)$ is an estimated cost of the path between node $\boldsymbol{n}$ and the end node. The Manhattan distance equation, Eq. 4.15, is used as the heuristic function.

$$d = |x_1 - x_2| + |y_1 - y_2| \tag{4.15}$$

Where (x_1, y_1) and (x_2, y_2) are the coordinate of two points, and d is the Manhattan distance. The heuristic function estimates the minimum cost of the path between node $\boldsymbol{n}$ and the end node without considering any obstacles, ensuring the path-planning algorithm finding the shortest path without overestimating the true cost. The algorithm computes the sum of the two costs and chooses the path with the lowest value of $f(n)$. Since the occlusions in the sidewalk dataset are all small shadows and trees, finding the shortest path

between the broken sidewalk endpoints becomes a suitable solution. After applying the *A**-based segmentation refinement to the segmentation prediction, the complete sidewalk extraction is shown in Figure 4.23: the sidewalk prediction (in white pixel) is overlayed on the aerial image, and the sidewalk completed by *A** is highlighted in red inside the red box. Zoomed-in views of both refinements are also shown in Figure 4.23(b) and Figure 4.23(c).

Figure 4.23 Sidewalk segmentation refinement with A* algorithm

4.4 Summary

The first part of this Chapter shows the strong generalization ability of segmentation models on sidewalk extraction. More importantly, the trained deep learning model is able to correctly label lots of sidewalk features that are covered by occlusions. This shows that

the deep learning model managed to learn those special features using the limited dataset. The second part of this Chapter introduces the segmentation refinement stage to the sidewalk extraction model. The refinement method fixes the broken sidewalk segmentation by finding the shortest path between two endpoints of the discontinuity using the A^* path-planning algorithm. This method is essentially ideal for completing the sidewalk segmentation since the occlusion covering the sidewalk is small trees and shadows.

Chapter 5. Route Planning and Optimization on Sidewalk Network

5.1 Introduction

The purpose of sidewalk extraction is to provide the sidewalk feature to the HD map and enrich its informative level, making autonomous driving on sidewalks available for small robots and vehicles. To keep autonomous robots operating within the sidewalk zone, it is necessary to have a safe and efficient route planned ahead. This Chapter describes the use of a cost-based route planning algorithm on the extracted sidewalk network at city-scale. An objective function considering travel distance and safety level of the generated path is also implemented to measure the safety and efficiency of the generated path.

5.2 Route Planning on the Extracted Sidewalk Network

5.2.1 City-scale Sidewalk Network Creation

Each sidewalk prediction from sidewalk extraction is only a small part of the aerial map. Thus, a city-scale aerial map with the sidewalk network is constructed using concatenation to allow autonomous robots to navigate through the city. This part was done by only overlaying the sidewalk pixels on the original aerial image and then concatenating the overlayed images together to construct the aerial map with the sidewalk network. In addition, crosswalks are also labelled for the completeness of the road network since they are inevitable when planning the route. Figure 5.1 shows the city-scaled aerial map with the sidewalk network, where the sidewalk and crosswalk network are highlighted in light blue. Figure 5.2 shows the city-scaled sidewalk network image that only contains the sidewalk (white pixel) and the crosswalk (gray pixel), which will be used for path planning.

Figure 5.1 City-scale aerial map with sidewalk network

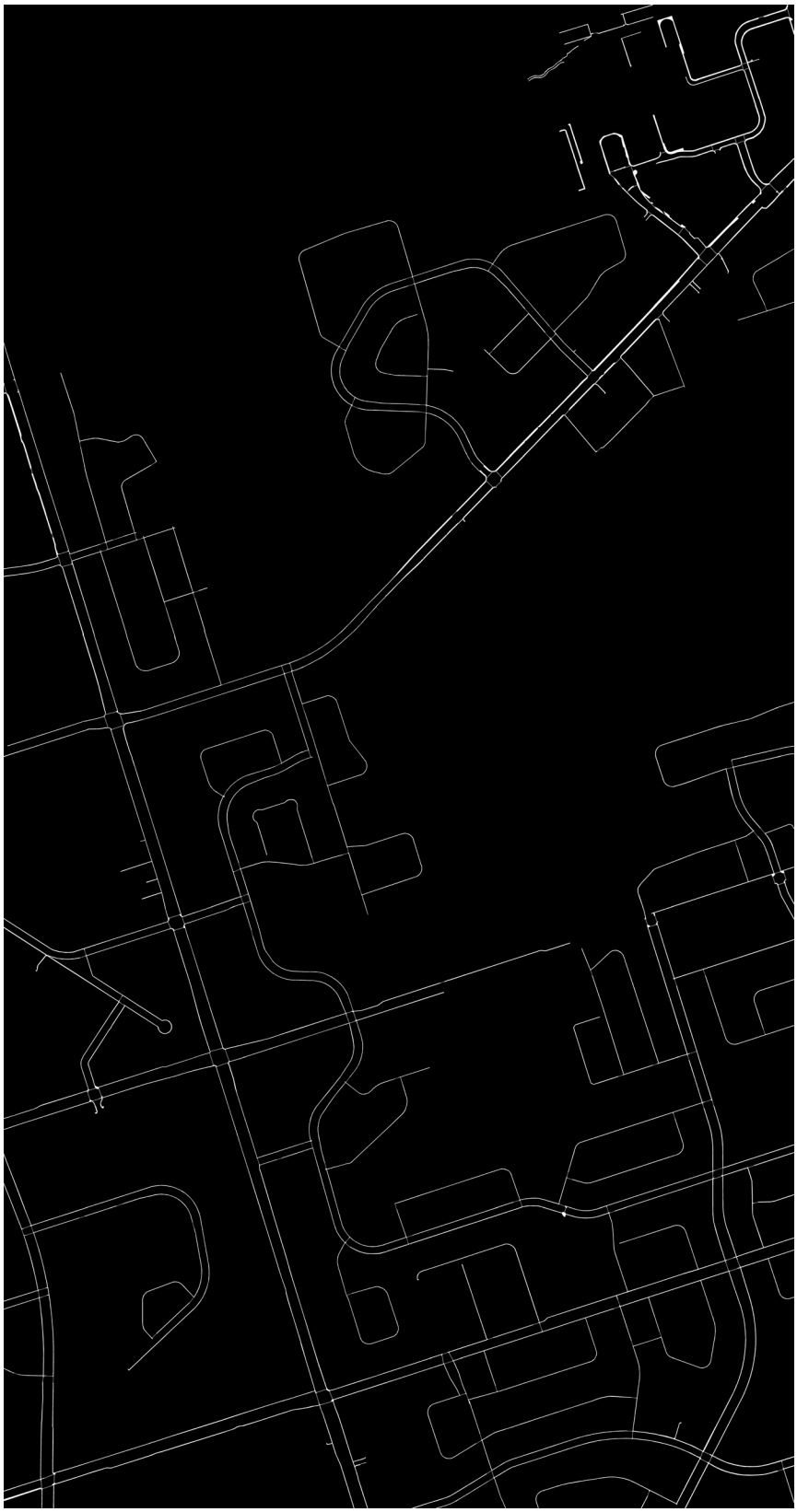

Figure 5.2 Sidewalk and crosswalk network

In Figure 5.2, the crosswalk is purposely labelled in gray pixels to differentiate between sidewalks. In addition, Figure 5.2 uses the GeoTIFF format to store the GPS coordinate information for each pixel on the image. The route planning algorithm can use the GPS value for routing instead of using the pixel which does not provide any localization information. The stored coordinate system also provides the distance between each pixel, allowing it to calculate the actual distance of the route.

5.2.2 Cost-based *A** Routing Algorithm on the Sidewalk (Crosswalk) Network

Route planning on the sidewalk network is a multi-stage process. Firstly, the sidewalk GeoTIFF image is opened using the *rasterio* Python library to store the GPS coordinate system in the memory. The image is then converted into a cost array that consists of pixel values of 0 (black), 255 (white), and 128 (gray). In the cost array, all black pixels are initiated as non-navigable since they represent the background. White and gray pixels are both initiated as navigable but with different costs since they represent sidewalks and crosswalks, respectively. The *skeletonize* function from the *skimage.morphology* library is used to skeletonize the cost array. The skeletonization aims to shrinks down the pixel width of the sidewalk and the crosswalk. This step increases the efficiency of the route planning algorithm when going through all the navigable, and it also prevents it from generating a route that has a zig-zag shape. Secondly, a node graph is created from the skeletonized cost array using the *networkx* library. The node graph is generated by iterating through the navigable pixels and adding edges between the navigable neighboring pixels. For each edge, the cost is set as the average of the costs of the two nodes it connects. The idea here is to build a weighted graph where the weight of each node represents the cost to traverse that pixel, and the weight of each edge is an average of the costs of the nodes it

connects. By giving different costs to sidewalk and crosswalk nodes, various routes can be generated between two endpoints. Lastly, two variants of $A*$ path-planning algorithm are applied on the weighted node graph using the GPS coordinate of the start and end nodes. The first variant does not contain any heuristic function and the second one uses the Euclidean Distance, Eq. 5.1, as the heuristic function.

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2} \tag{5.1}$$

Where $d(p,q)$ is the Euclidean Distance, and p and q are two points represented by (p_1, p_2) and (q_1, q_2).

5.2.3 Objective Function for Optimal Route Planning

An Objective function considering total travel distance and route safety level is proposed to evaluate the "quality" of a generated route, as defined in Eq. 5.2.

$$Q = w_1 \cdot D + w_2 \cdot (1 - S) \tag{5.2}$$

Where Q is the quality of a particular segment in the route, w_1 and w_2 are weights such that $w_1 + w_2 = 1$, D is the Euclidean distance for a given segment, and S is the safety score for a given segment/node, which is a value between 0 and 1, with 1 indicating maximum safety (e.g., a sidewalk) and 0 indicating minimum safety (e.g., a crosswalk). Since crosswalks have more uncertainties than sidewalks, a safer route means taking as many sidewalk nodes as possible to reduce the number of uncertainties from the crosswalk. The idea of this objective function comes from the Nash Equilibrium that the function tries to minimize the total travel distance of the route while maximizing the total safety score. The term $(1 - S)$ ensures that a higher safety score will result in a lower quality, as the goal is to minimize the objective function. w_1 and w_2 are weights that control the importance of distance and safety, respectively. If the route safety is extremely important,

(e.g., traveling in downtown where traffic is generally much busier and taking crosswalks is not ideal), the safety weight can be assigned much greater than the distance weight so that the selected optimal route will use as many sidewalks as possible. If the route distance has higher priority, (e.g., ego-robot may not have enough energy for long distance deliveries), the distance weight can be assigned much greater than the safety weight to achieve a short-distance route. If tasks happen in residential areas where traffic is less busier and crosswalks are generally safe to take, a route planning with equal weights for distance and safety, where safety and distance have the same importance, can be used. Thus, by fine-tuning the weights w_1 and w_2, the objective function can be adapted to various route planning applications.

5.3 Route Planning Results and Discussion

The route planning algorithm is applied between two locations, the starting location is on the sidewalk at Founders Dr and Conlin Rd, and the ending location is on the sidewalk of Clearbrook Dr and Blackwood Blvd intersection. The coordinates, longitude and latitude, of the starting and ending location under the EPSG:3857-WGS 84 coordinate system are (-8783043.932, 5457265.278) and (-8778264.198, 5456317.822), respectively. The cost for sidewalk is fixed at 0.5, while different costs for crosswalk, including 0.5, 1.0, 5.0, 10.0, 20.0, 30.0, 40.0, and 50.0, are used for different runs. The highest cost for crosswalk stops at 50.0 due to the reason that the route planning algorithm starts generating the same results for cost equal to 50.0 and above. This indicates that, when the cost for crosswalk is much higher than it for sidewalk, there are some inevitable crosswalks that the route planning algorithm must take to get to the destination. Three scenarios are considered when initializing the objective function weights: 1. Safety has the highest priority ($w_1 =$

$0.1, w_2 = 0.9$); 2. Mileage has the highest priority ($w_1 = 0.9, w_2 = 0.1$); 3. Both safety and mileage have an equal amount of priority ($w_1 = 0.5, w_2 = 0.5$). The A*-based route planning algorithms with and without the heuristic function are run with different combinations of the objective function weights and the crosswalk costs. The results are shown in the tables below (E.D represents Euclidean Distance and lowest values are bolded).

Table 5.1 Route planning results ($w_1 = 0.1, w_2 = 0.9$)

Crosswalk Cost	Heuristic Function	Total Nodes	Total Distance (km)	Total Quality (Q)
0.5	E.D	37298	9.88	**12737.8**
1.0	E.D	37278	9.87	14395.4
5.0	E.D	37266	9.87	16656.8
10.0	E.D	38524	10.20	22443.6
5.0	None	33096	8.77	22827.8
0.5	None	33262	8.81	23731.8
1.0	None	**33064**	**8.75**	23989.2
10.0	None	34284	9.08	24654.8
20.0	E.D	36554	9.68	25388.5
20.0	None	36086	9.56	25954.6
40.0	E.D	51076	13.52	37030.5
30.0	None	51076	13.52	37178.1
40.0	None	51076	13.52	37178.1
30.0	E.D	52128	13.80	37752.2
50.0	E.D	52130	13.80	37765.9
50.0	None	52130	13.80	37913.5

Table 5.2 Route planning results ($w_1 = 0.9, w_2 = 0.1$)

Crosswalk Cost	Heuristic Function	Total Nodes	Total Distance (km)	Total Quality (Q)
5.0	None	33096	8.76	**31954.2**
1.0	None	**33064**	**8.75**	32054.8
0.5	None	33262	8.81	32202.2
10.0	None	34284	9.08	33213.2
0.5	E.D	37298	9.88	34568.2
1.0	E.D	37278	9.87	34734.6
20.0	None	36086	9.56	34959.4
5.0	E.D	37266	9.87	34975.2
20.0	E.D	36554	9.68	35312.5
10.0	E.D	38524	10.20	36736.4
40.0	E.D	51076	13.52	49514.5
30.0	None	51076	13.52	49530.9
40.0	None	51076	13.52	49530.9
30.0	E.D	52128	13.52	50529.8
50.0	E.D	52130	13.80	50549.5
50.0	None	52130	13.80	50549.5

Table 5.3 Route planning results ($w_1 = 0.5, w_2 = 0.5$)

Crosswalk Cost	Heuristic Function	Total Nodes	Total Distance (km)	Total Quality (Q)
0.5	E.D	37298	9.88	**23653.0**
1.0	E.D	37278	9.87	24565.0
5.0	E.D	37266	9.87	25816.0
5.0	None	33096	8.76	27391.0
0.5	None	33262	8.81	27967.0
1.0	None	**33064**	**8.75**	28022.0
10.0	None	34284	9.08	28934.0
10.0	E.D	38524	10.20	29590.0
20.0	E.D	36554	9.68	30350.5
20.0	None	36086	9.56	30457.0
40.0	E.D	51076	13.52	43272.5
30.0	None	51076	13.52	43354.5
40.0	None	51076	13.52	43354.5
30.0	E.D	52128	13.80	44141.0
50.0	E.D	52130	13.80	44149.5
50.0	None	52130	13.80	44231.5

Table 5.1 shows the route planning results when safety has the highest priority. The minimum total quality value is 12737.8 when the route planning algorithm is applied with Euclidean distance heuristic function and the cost for crosswalk is 0.5. The total distance of the generated route is approximately 9.88 kilometers. Table 5.2 shows the route planning results when mileage has the highest priority. The minimum total quality value is 31954.2 when the route planning algorithm is applied without a heuristic function and the cost for crosswalk is 1.0. Table 5.3 shows the route planning results when safety and mileage share an equal amount of priority. The minimum total quality value is 23653 when the route planning algorithm is applied with Euclidean distance heuristic function and the cost of crosswalk is 0.5. Thus, when safety has the highest priority or safety and distance share an equal amount of priority, the optimal route is generated when A^*-based route planning algorithm is applied with Euclidean distance as heuristic function and the cost for crosswalk is at 0.5. When distance has the highest priority, the optimal route is generated when A^*-based route planning algorithm is applied without a heuristic function and the cost for crosswalk is 1.0. The graphical representation of two optimal routes are shown in Figure 5.3 and Figure 5.4, where the generated route is labelled in red. Some major differences between the first and the second route can be observed from the figure. Starting from the Ontario Tech University campus, route #1 takes the south bound of the sidewalk then turns onto the west bound sidewalk of the Simcoe St. It switches to the east bound sidewalk of the Simcoe St after passing the Commencement Dr and continues until it reaches the Glovers Rd. The route then turns onto the north bound sidewalk of the Glovers Rd and continues until it reaches the Sarasota St. The route starts heading south on the Sarasota St then turns onto the north bound sidewalk of the Ormond. After following the

Ormond Dr heading northeast, it turns onto the south bound sidewalk of the Blythwood Square to reach the Wilson Rd. After heading south on the east bound sidewalk of Wilson Rd, the route turns onto the Greenvailey Trail. The route makes its last turn from the Greenvailey Trail onto the Blackwood Blvd north bound sidewalk and continues heading east until it reaches the ending location. On the other hand, optimal route #2 continues heading south on the west bound sidewalk of the Simcoe St after departures from the school campus. At the Glovers Rd, it switches to the east bound sidewalk of the Simcoe St and continues heading south until it reaches the Taunton Rd, instead of taking the Glovers Rd. It then turns onto the north bound sidewalk of the Taunton Rd and heads east until the Wilson Rd. The route then heads north on the west bound sidewalk of the Wilson Rd and turns onto the south bound sidewalk of the Blackwood Blvd. After switching to the north bound sidewalk of the Blackwood Blvd, it continues heading east on the sidewalk until reaches the ending location. The major differences between two optimal routes are the total travel distances and the route complexity. Since for optimal route #1, safety has more or equal weights compared to mileage, the route planning algorithm tries to use as many sidewalk nodes as possible, resulting in more routes generated in the residential area, as shown in Figure 5.3. This leads to more detours in the route in order to avoid taking crosswalks, resulting in a more complex route. Optimal route #2, on the other hand, has mileage as the top priority, resulting in more routes generated in the busy traffic area and using as many crosswalk nodes as possible as "shortcuts" to achieve low travel distance. Thus, optimal route #2 has a higher Q value than optimal route #1.

By observing the table, it can be noticed that as the cost for crosswalk increases from 10.0 to 50.0, the total quality value dramatically increases regardless of the weight of

distance and safety. This indicates that distance has a stronger impact on the Q value when the crosswalk cost is much higher than the sidewalk cost. The relationship is reasonable since the higher the crosswalk cost is, the more distance the autonomous robot will travel, leading to higher Q. Higher travel distance also leads to more uncertainties on the road. When the cost for crosswalk is between 0.5 and 10.0, a low travel distance does not guarantee an optimal route. In the table above, the lowest travel distance is achieved, approximately 8.75 kilometers, when A^* is used without heuristic function and the crosswalk cost is 1.0. However, the Q value of this route is not at minimum.

Figure 5.3 Optimal Route #1

Figure 5.4 Optimal Route #2

5.4 Path Following Simulation on the Generated Route

To mimic the real-life scenario of an autonomous robot travelling between two locations on the generated route, a path following simulation is created using ROS (Robot Operating System) and Gazebo simulator. The route from Figure 5.4 is selected for the path following simulation. Since the route is generated by connecting all selected nodes together, pure pursuit is used in the simulation as the waypoint following algorithm.

Pure pursuit is a geometric approach to compute the steering commands for an autonomous robot to follow a path. It operates by calculating the curvature that will take the robot from its current position to a lookahead position on the path. In the algorithm, a lookahead distance L_d is firstly defined, which is the distance ahead of the robot at which it targets a point on the path. This point is called the lookahead point (p_{la}). The curvature (κ) required to reach the lookahead point is then calculated using the following equation:

$$\kappa = \frac{2y}{L_d^2} \tag{5.3}$$

Where y is the lateral offset between the robot's current position and the lookahead point, which is measured perpendicular to the robot's current heading. The steering angle δ required to follow the curvature is then determined using the bicycle vehicle model, defined as:

$$\delta = \arctan\left(\frac{L \cdot \kappa}{1 + \kappa^2 \cdot L_r^2}\right) \tag{5.4}$$

Where L is the distance between the front and rear axles of the robot and L_r is the distance from the rear axle to the center of mass of the robot. Since for most path following

applications where the speed is constant and κ is relatively small, the term $\kappa^2 \cdot L_r^2$ becomes negligible. Eq. 5.4 simplifies to:

$$\delta = \arctan(L \cdot \kappa) \tag{5.5}$$

The waypoints used in the simulation are from the node positions from the generated path after they are converted to the Gazebo coordinate. The robot used for the simulation is the Husky UGV (Unmanned Ground Vehicle) model. The simulation environment is initialized with the aerial map, Figure 5.4, as the base texture. The Husky robot is placed on the map texture at the starting location. A Python script is implemented to start the simulation, read the waypoint file, and send the steering command to the robot using the pure pursuit algorithm. The simulation continues running until the last waypoint, the ending location, has been reached. A short clip of the simulation has been made available at https://youtu.be/X5sh19oVV2w. The process of the simulation is also demonstrated in the algorithm below:

Algorithm 4 Pure Pursuit Waypoints Following

Input: waypoints from generated path
Lookahead distance
Wheelbase length of the robot

Output: steering angle

While robot has **not** reached the final waypoint:
Determine the current position of the robot
From the current position, **find** the next lookahead point based on the lookahead distance
Compute the angle between the robot's current heading and the lookahead point
Determine the curvature required to turn from the current heading to the angle calculated
Calculate the steering angle using the curvature and wheelbase length
Apply the steering angle to the robot
Update the robot's position based on the movement
End While

5.5 Summary

This chapter describes the cost-based *A** route planning algorithm for routing on sidewalk and crosswalk, and an objective function for choosing an optimal route for different scenarios. The cost-based *A** route planning algorithm is applied multiple times with various costs for crosswalk, and with and without the Euclidean distance as the heuristic function. The objective function uses the minimax methodology to minimize the total travel distance while maximizing the route safety level. Three conditions including highest priority for safety, equal priority for both distance and safety, and highest priority for distance are considered when using the objective function to obtain the optimal route. However, the proposed objective function is not limited to the above three scenarios. By initializing the different weights for distance and safety, optimal routes for different scenarios or applications can be determined by finding the route with the lowest quality value. The chapter then finishes with describing a simulation of autonomous path following application using one of the optimal routes with ROS and Gazebo simulator. The simulation aims to mimic as well as prove the feasibility of the real-life scenario of robots autonomously following the generated routes on the sidewalk.

Chapter 6. Conclusion

6.1 Conclusion

On the way to achieving fully autonomous driving, HD maps have become a key component of every aspect of an autonomous driving system, including mapping, localization, navigation, perception, mission planning, and motion control. An informative HD map provides an autonomous driving system with the ability to plan ahead and avoid accidents. In recent years, automatic road network extraction for HD maps has become a mature technique thanks to the fast development of powerful GPUs and incredible deep learning algorithms. Automatic sidewalk extraction, as a missing part of the HD map, has already shown its necessity with the growing diversification of modern urban transportation options, such as autonomous delivery robots, E-bikes, and E-scooters. This lack of research makes methods for automatic sidewalk extraction in demand.

Semantic segmentation algorithms have been known for their precision and efficiency in recognize and separate different objects at pixel level. The automatic sidewalk extraction on aerial image method proposed in the thesis also utilizes segmentation-based methods. To create a deep learning model that segments sidewalks from the background on an aerial image, the deep learning model must first be trained on a sidewalk specific dataset. Thus, a sidewalk dataset containing high-resolution aerial images with precise sidewalk annotations is created. The dataset covers sidewalk features in different terrains, such as school campus, residential area, traffic crossings, and motorways, aiming to have the deep learning model learn all types of sidewalks. Data augmentation technique is also used in dataset preparation to increase the size of the dataset. When training the sidewalk extraction model, multiple SOTA semantic segmentation models, including U-Net [25], LinkNet

[12], FPN [21], MA-Net [33], UNet++ [34], PSPNet [35], PAN [36], and DeepLabV3+ [37], are trained, tested, and evaluated using the aerial image dataset. TrivialAugment [32] and transfer learning techniques are used during training to enhance the model's final performance. A callback function is also implemented so that the program constantly updates and keeps the best result. By comparing the results of evaluation metrics in Table 4.1, the UNet++ with TrivialAugment model is observed to have the best sidewalk extraction performance. The inference results from Figure 4.12 to Figure 4.21 also show the incredible performance of the model, especially when sidewalks are covered by occlusions.

Segmentation refinement technique is applied on the discontinued sidewalk segmentation predictions that are caused by occlusion, such as trees, shadows, and buildings. The A^*-based segmentation refinement method is proposed to fix the disconnected sidewalk segmentation issue by finding the shortest path between two endpoints. This method is an ideal sidewalk segmentation refinement method since, comparing to other road network extraction, sidewalks have low network complexity and occlusion on the sidewalk are also relatively small. The refined sidewalk extraction images are then concatenated together to construct a city-scale sidewalk network that can be used for path planning and other topology applications.

Route planning algorithm is applied on the city-scale sidewalk network to generate routes for sidewalk and crosswalk-only autonomous driving through the city. The cost-based A^* route planning algorithm is proposed to generate different routes based on the cost of crosswalk. The higher the crosswalk cost is, the fewer crosswalk nodes will be taken. An objective function adopting the mini-max technique to determine the optimal

route when travel distance and safety are both considered. By varying the weights for travel distance and safety, the optimal routes for different applications or preferences can be determined by finding the minimum Q value from the objective function. The proposed objective function can be applied to not only route planning on sidewalks but also to other route planning applications.

To show the usage of route planning on sidewalk and crosswalk and its feasibility for navigating through the city, a path following simulation is conducted. Pure pursuit is used as the path tracking algorithm which utilizes the lookahead distance to compute the steering angle by calculating the curvature between the current robot position and the lookahead point. The simulation shows that path planning on sidewalk and crosswalk can be used for applications like delivery robot that are only allowed to operate on the sidewalk and crosswalk. It also proves the feasibility of traveling through the city using solely sidewalks and crosswalks.

In conclusion, the proposed sidewalk extraction on aerial image method with A^* path planning algorithm-based segmentation refinement technique offers an automatic way to precisely extraction sidewalks for HD maps, filling the gap in sidewalk extraction research. The prepared aerial image dataset can be used for further sidewalk extraction research or semantic segmentation algorithm research. For route planning on sidewalks and crosswalks, the cost-based A^* route planning algorithm is able to generate different routes based on the safety requirement. The proposed objective also provides a way to determine an optimal route for different route planning applications and preferences.

6.2 Recommendations and Future Work

The work discussed in this thesis can be expanded upon by the following additions and implementations.

1. Compared to other semantic segmentation datasets, the aerial image dataset is a small dataset for segmentation research. The dataset also only contains two classes, sidewalk and background, which limits it to binary classification application. Thus, future work should focus on expanding the size and the category of the dataset for more semantic segmentation research. In addition, sidewalks also have different shapes, sizes, and colors depending on where they are located. Thus, classifying sidewalk types by annotating each type in a specific label should also be conducted in future work.
2. The sidewalk dataset is labelled on the Google Satellite Image, which is not always up to date. This could lead to mismatch between the extracted sidewalk network on HD map and the actual sidewalk network. Thus, finding a reliable satellite image resource or developing a method to keep the sidewalk dataset up to date should be considered for future work.
3. Besides segmentation-based sidewalk extraction method proposed in this thesis, other types of methods including graph-generation and iterative graph growing methods are also suitable for sidewalk extraction. Thus, more sidewalk extraction algorithms should be implemented to compare with the work in this thesis and further improve the extraction results.
4. For route planning on sidewalk applications, A^* with and without heuristic function is used in this work. Future research should explore more route planning algorithms

to plan routes on sidewalk and crosswalk network. Additionally, route planning application should consider more aspects, such as considering the travel directions when on different bounds of the sidewalk, considering stop signs to reduce the stop and start frequencies, and considering taking alternative routes other than sidewalks and crosswalks.

Reference

[1] Z. Bao, S. Hossain, H. Lang, and X. Lin, “A review of high-definition map creation methods for autonomous driving,” *Eng Appl Artif Intell*, vol. 122, p. 106125, Jun. 2023, doi: 10.1016/J.ENGAPPAI.2023.106125.

[2] R. Liu, J. Wang, and B. Zhang, “High definition map for automated driving: overview and analysis,” *Journal of Navigation*, vol. 73, no. 2, pp. 324–341, Mar. 2020, doi: 10.1017/S0373463319000638.

[3] Z. Xu, Y. Sun, and M. Liu, “Topo-Boundary: A benchmark dataset on topological road-boundary detection using aerial images for autonomous driving,” *IEEE Robot Autom Lett*, vol. 6, no. 4, pp. 7248–7255, Jul. 2021, doi: 10.1109/LRA.2021.3097512.

[4] M. Cordts *et al.*, “The Cityscapes Dataset for Semantic Urban Scene Understanding,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016. doi: 10.1109/CVPR.2016.350.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[6] T. Y. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns,” *Commun ACM*, vol. 27, no. 3, pp. 236–239, Mar. 1984, doi: 10.1145/357994.358023.

[7] G. Mattyus, W. Luo, and R. Urtasun, “DeepRoadMapper: Extracting road topology from aerial images,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3458–3466.

[8] A. van Etten, D. Lindenbaum, and T. M. Bacastow, “SpaceNet: A remote sensing dataset and challenge series,” *arXiv: 1807.01232*, Jul. 03, 2018.

[9] I. Demir *et al.*, “DeepGlobe 2018: A challenge to parse the earth through satellite images,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 172–181.

[10] A. Mosinska, P. Marquez-Neila, M. Kozinski, and P. Fua, “Beyond the pixel-wise loss for topology-aware delineation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3136–3145.

[11] G. Máttyus and R. Urtasun, “Matching adversarial networks,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8024–8032.

[12] A. Chaurasia and E. Culurciello, “LinkNet: Exploiting encoder representations for efficient semantic segmentation,” in *2017 IEEE Visual Communications and Image Processing, VCIP 2017*, 2018, pp. 1–4.

[13] F. Bastani *et al.*, “RoadTracer: Automatic extraction of road networks from aerial images,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4720–4728.

[14] H. Ghandorh, W. Boulila, S. Masood, A. Koubaa, F. Ahmed, and J. Ahmad, “Semantic segmentation and edge detection—approach to road detection in very high resolution satellite images,” *Remote Sens (Basel)*, vol. 14, no. 3, Jan. 2022, doi: 10.3390/rs14030613.

[15] G. Cheng, Y. Wang, S. Xu, H. Wang, S. Xiang, and C. Pan, “Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 6, pp. 3322–3337, Mar. 2017, doi: 10.1109/TGRS.2017.2669341.

[16] N. Xue, S. Bai, F. Wang, G. S. Xia, T. Wu, and L. Zhang, “Learning attraction field representation for robust line segment detection,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1595–1603.

[17] S. He *et al.*, “Sat2Graph: Road graph extraction through graph-tensor encoding,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020, pp. 51–67.

[18] N. Girard, D. Smirnov, J. Solomon, and Y. Tarabalka, “Polygonal building extraction by frame field learning,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5887–5896.

[19] A. Vaswani *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, 2017, pp. 5999–6009.

[20] Z. Xu *et al.*, “CsBoundary: City-scale road-boundary detection in aerial images for high-definition maps,” *IEEE Robot Autom Lett*, vol. 7, no. 2, pp. 5063–5070, Apr. 2022, doi: 10.1109/LRA.2022.3154052.

[21] T. Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” in *Proceedings 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017, pp. 936–944.

[22] M. Schreiber, C. Knöppel, and U. Franke, “LaneLoc: Lane marking based localization using highly accurate maps,” in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2013, pp. 449–454.

[23] W. Jang, J. An, S. Lee, M. Cho, M. Sun, and E. Kim, “Road lane semantic segmentation for high definition map,” in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2018, pp. 1001–1006.

[24] L. Ma, Y. Li, J. Li, J. M. Junior, W. N. Goncalves, and M. 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*, Feb. 2021, doi: 10.1109/TITS.2021.3055366.

[25] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015. doi: 10.1007/978-3-319-24574-4_28.

[26] I. Goodfellow *et al.*, “Generative adversarial networks,” *Commun ACM*, vol. 63, no. 11, pp. 139–144, Nov. 2020, doi: 10.1145/3422622.

[27] Y. Li, L. Xiang, C. Zhang, and H. Wu, “Fusing taxi trajectories and rs images to build road map via dcnn,” *IEEE Access*, vol. 7, pp. 161487–161498, Nov. 2019, doi: 10.1109/ACCESS.2019.2951730.

[28] D. Yu, H. Xiong, Q. Xu, J. Wang, and K. Li, “Multi-stage residual fusion network for LIDAR-camera road detection,” in *IEEE Intelligent Vehicles Symposium, Proceedings*, 2019, pp. 2323–2328.

[29] J. D. Wegner, J. A. Montoya-Zegarra, and K. Schindler, “A higher-order CRF model for road network extraction,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2013, pp. 1698–1705.

[30] A. Batra, S. Singh, G. Pang, S. Basu, C. V. Jawahar, and M. Paluri, “Improved road connectivity by joint learning of orientation and segmentation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10377–10385.

[31] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.

[32] S. G. Müller and F. Hutter, “TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2021. doi: 10.1109/ICCV48922.2021.00081.

[33] T. Fan, G. Wang, Y. Li, and H. Wang, “Ma-net: A multi-scale attention network for liver and tumor segmentation,” *IEEE Access*, vol. 8, 2020, doi: 10.1109/ACCESS.2020.3025372.

[34] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, “Unet++: A nested u-net architecture for medical image segmentation,” in *Lecture Notes in Computer*

Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2018. doi: 10.1007/978-3-030-00889-5_1.

[35] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017. doi: 10.1109/CVPR.2017.660.

[36] H. Li, P. Xiong, J. An, and L. Wang, "Pyramid attention network for semantic segmentation," in *British Machine Vision Conference 2018, BMVC 2018*, 2019.

[37] L. C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018. doi: 10.1007/978-3-030-01234-2_49.

[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[39] X. Lei *et al.*, "STDC-MA network for semantic segmentation," *IET Image Process*, vol. 16, no. 14, 2022, doi: 10.1049/ipr2.12591.

[40] Y. Cai and Y. Wang, "MA-Unet: an improved version of Unet based on multi-scale and attention mechanism for medical image segmentation," 2022. doi: 10.1117/12.2628519.

[41] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Rethinking Atrous Convolution for Semantic Image Segmentation Liang-Chieh," *IEEE Trans Pattern Anal Mach Intell*, vol. 40, no. 4, 2018.

[42] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int J Comput Vis*, vol. 115, no. 3, 2015, doi: 10.1007/s11263-015-0816-y.

[43] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[44] Pavel Iakubovskii, "Segmentation Models Pytorch," GitHub. Accessed: Jun. 18, 2023. [Online]. Available: https://github.com/qubvel/segmentation_models.pytorch