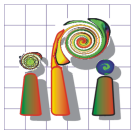



Master Thesis

Segmentation and Vectorisation of curbstones from high-resolution ortho-images for test sites in Bavaria, Germany

Gottfried Wilhelm Leibniz Universität
Institute for Photogrammetry and Geoinformation

Winter Semester 2023-24

Submitted by:
Mariya Jose
10043341

First Examiner:
apl. Prof. Dr. techn. Franz Rottensteiner
Second Examiner:
Mirjana Voelsen, M.Sc.

Supervisors:
Dr. Jiaojiao Tian , Dr. Stefan Auer

Statement

I declare that this thesis is the result of independent research conducted by me under the guidance of my supervisors. It does not contain the results of any other scientific research that has been published or written by any other individuals or groups, except for those already cited in the thesis. Furthermore, I state that this work in the same or a similar form has not been submitted to an examination authority.

Munich, 16.05.2024

Place, Date

Signature

Abstract

Generating road networks manually has always been an ineffective and labour-intensive task. Accurate representation of road networks is crucial for various applications, including urban planning, infrastructure management, navigation systems, and especially autonomous vehicle development. In the realm of autonomous driving, the accurate detection of curbstones holds particular significance, as they serve as critical boundaries for vehicle navigation and safety. However, current methods for online curbstone detection at the site are fraught with challenges, including real-time processing constraints and environmental variability. Fortunately, the availability of high-resolution aerial imagery presents an opportunity for offline curbstone detection, enabling more comprehensive and accurate mapping of road networks. In this thesis, we address the problem of curbstone detection as an iterative graph generation task, wherein curbstone edges are detected vertex by vertex from initial curbstone candidates identified through segmentation. Leveraging techniques from imitation learning, we take a high-resolution ortho-image as input and output a graph representing the detected curbstones. Our approach endeavours to enhance the accuracy and robustness of road edge detection through several enhancements. We introduce a loss function, termed Slope Penalty loss, aimed at refining the model training process by addressing the slight variations in gradients of the predicted vertices. Our experimental evaluations underscore the effectiveness of these enhancements, as demonstrated through comparisons with the already existing curbstone detection algorithms. The proposed approach is tested over the city area of Munich, Bavaria, Germany

Keywords: Segmentation, vectorisation, UNet, ResNet, Curbstone, Imitation learning

Contents

List of Figures

List of Tables

1 Introduction

The advancement of autonomous driving technologies has profoundly highlighted the critical role of automatic curbstone detection in ensuring safe and efficient navigation for vehicles, particularly in urban traffic environments where structured road curbs are common. Curbstone detection stands as a fundamental component in empowering autonomous vehicles to discern drivable areas on roads, thereby significantly enhancing overall road safety and mobility (Bastani et al. (2018)). Traditionally, curbstone detection has predominantly relied on employing sensors such as LIDARs and cameras mounted on vehicles (Xu, Sun, and M. Liu (2021)) to get the images required for the detection. However, the data acquired from these approaches confront substantial challenges, including variations in illumination conditions, occlusions from surrounding objects like cars, pedestrians, and vegetation (Bastani et al. (2018)), as well as the presence of shadows, and sparse point-cloud data, which can degrade detection accuracy, particularly in areas far from the ego-vehicle, all of which can potentially compromise the accuracy and reliability of detection. Furthermore, the hardware requisites for these data acquisitions inevitably contribute to the overall expense of the system, posing additional hurdles for widespread adoption and implementation (Xu, Sun, and M. Liu (2021)).

To effectively address these challenges and complement terrestrial mapping efforts, there has been a burgeoning interest in leveraging remote sensing data for mapping distinctive elements such as curbstones across expansive areas. By seamlessly integrating remote sensing information into maps utilised in automotive applications, such as Opendrive, there emerges the opportunity to increase the accuracy of the map data, thereby substantially enhancing the performance of autonomous driving systems (Xu, Sun, and M. Liu (2021)). Among the myriad avenues explored for curbstone detection, one particularly promising approach lies in the utilisation of high-resolution aerial images for curbstone detection.

Within the realm of curbstone detection from high-resolution ortho-images, two primary methodologies have emerged as focal points of investigation: segmentation followed by heuristic post-processing and end-to-end iterative graph generation. The former approach entails segmenting the curbstones from the ortho-images, succeeded by heuristic post-processing steps aimed at refining the results. While this method often yields commendable pixel-level outcomes, it grapples with challenges pertaining to poor topology correctness and the presence of noise in segmentation results (Bastani et al. (2018)), particularly evident in complex topological scenarios. In contrast, the iterative graph generation approach expresses the problem as an iterative process, wherein curbstones are predicted vertex by vertex from initial candidates identified through segmentation. This methodology presents the potential for enhanced topology correctness and reduced noise in the final results, thereby constituting an appealing avenue for curbstone detection in diverse and complex environments (Xu, Y. Liu, et al. (2022), Bastani et al. (2018), Xu,

Sun, and M. Liu (2021)). In summary, the integration of aerial imagery with advanced deep-learning techniques shows great promise for curbstone detection and road network extraction in urban environments.

1.1 Problem statement

The main objective of this thesis is to firstly, explore the performance of iCurb in segmenting curbstones from high-resolution ortho-images, and secondly, to refine and enhance the existing structure of iCurb. Leveraging the groundwork laid out by Xu, Sun, and M. Liu (2021), the aim is to adapt this framework to suit the specific dataset of Bavaria, Germany. Subsequently, the outcomes both visually and quantitatively are evaluated using the provided test datasets.

1.2 Contribution

In addressing the aforementioned challenges, this thesis makes a contribution aimed at enhancing the accuracy and robustness of road edge detection and vectorisation from high-resolution airborne orthophotos. This thesis introduces a loss function termed "Slope penalty loss" designed to improve the training of deep learning models by addressing the slight changes in gradients of the generated edges.

Chapter 2 will provide a comprehensive review of existing literature and research papers in the field of road edge detection and vectorisation, exploring various methodologies and techniques employed in previous studies. Chapter 3 will delve into the fundamental concepts and mathematical backgrounds underlying the methods utilised in this thesis. In Chapter 4, the proposed methodology for road edge segmentation and vectorisation, iCurb will be detailed, outlining the specific steps involved in its implementation. Chapter 5 will focus on the experimental setup, including details of the dataset used, training procedures, and evaluation metrics employed to assess the performance of the adopted approach. Visualisations of the results and their evaluation based on several evaluation metrics will be presented in Chapter 6, providing insights into the effectiveness of the methodology. Finally, Chapter 7 will draw conclusions based on the findings obtained, highlighting the contributions of the thesis and proposing avenues for future research in the domain of road edge detection and vectorisation from high-resolution ortho-images.

2 Related Work

The exploration of road network detection and extraction from aerial imagery has evolved over the decades, spanning from early heuristic-based approaches to modern deep-learning techniques. This section reviews relevant works in chronological order, highlighting key contributions and advancements in the field.

2.1 Segmentation based approaches

Segmentation-based approaches that have been used to detect linear structures like curbstones or roads mainly have two stages: predict the segmentation map and then process this segmentation map, to get vectorised outputs. Since both roads are curbstones are linear structures and are affected by complex topology, the detection methods used to detect roads can also be adapted to detect curbstones.

Mnih and Hinton 2010 are credited with pioneering the utilization of neural networks for road network detection in aerial images. Their approach involved initially partitioning the large aerial image into smaller patches. Within each patch, they predicted the road network, eventually consolidating the patches to form the final predicted road network segmentation map. Subsequent works in segmentation followed a comparable methodology, although employing more advanced segmentation networks like UNet (Ronneberger, Fischer, and Brox (2015)), FPN (Lin et al. (2016)), etc. Usually, these networks outputs map in raster format, so further post-processing is required to vectorise these maps, extract the centerline of the segmentation masks, filter out outlier detections and to correct discontinuities in the connections of the network.

To better extract the network, we can use several algorithms starting with Hough (1962) introduced the Hough Transform, an innovative mathematical method for recognising complex patterns especially lines in digital images. T. Zhang and Suen (1984) presented a fast parallel algorithm for thinning digital patterns into one pixel-wide representation. This method contributed to the field of digital image processing by enabling the efficient skeletonisation of binary images, which is crucial for extracting geometric features such as roads or curbstones from aerial imagery. Also Douglas and Peucker (1973) proposed algorithms for reducing the number of points required to represent a digitised line. This work addressed the challenge of efficiently representing line features in digital images, facilitating faster processing and especially storage of geometric data. So after obtaining the segmentation mask, we could skeletonise the mask, label the connected regions to get a dense list of coordinates and then use the Douglas Peucker algorithm to reduce the number of points in each instance. The above two methods (T. Zhang and Suen (1984), Douglas and Peucker (1973)) were combined and used to detect curbstones from orthoimages recently in the paper Tian et al. (2023).

As semantic segmentation solely operates on pixel-level predictions, it often overlooks topology information. Consequently, road network graphs derived from segmentation-based methods may exhibit inadequate topology accuracy. Furthermore, manually crafted or heuristic post-processing techniques struggle to rectify errors within the resulting road network graph. To create a baseline to compare different methods of curbstone detection algorithms, we implemented Hough (1962) and Tian et al. (2023), for their simplicity and more importantly efficiency.

2.2 End-to-end approaches

Different from segmentation-based approaches, end-to-end approaches for the generation of curbstone or road network detection directly output the vectorised results.

Moving forward to automatic approaches, Barzohar and Cooper (1996) introduced an automatic approach for finding main roads in aerial images using geometric-stochastic models and estimation techniques. This work represented a significant advancement in road detection algorithms, leveraging probabilistic models to accurately identify road networks from complex aerial scenes. Wegner, Montoya-Zegarra, and Schindler (2015) proposed a novel approach treating road networks as collections of minimum-cost paths, offering insights into the structural properties of road networks and their representation in remote sensing imagery. Continuing the trend of machine learning-based methods, Cheng et al. (2017) introduced an automatic road detection and centerline extraction framework using cascaded convolutional neural networks (CNNs). This approach demonstrated the power of end-to-end deep learning models in extracting road features with high accuracy and efficiency. Further advancing the state of the art, Mattyus, Luo, and Urtasun (2017) introduced DeepRoadMapper, a method for extracting road topology from aerial images using deep learning techniques. Their approach demonstrated superior performance in accurately mapping road networks from high-resolution aerial imagery.

Hinz and Baumgartner (2003) proposed an automatic approach that uses multi-view aerial imagery to extract urban road networks. By integrating information from different perspectives, their method upgrades the detection and construction of road networks, overcoming challenges such as occlusions and variations in perspective. This paper laid the foundation for utilising multiple viewpoints for robust road detection. Building upon this foundation, Hu et al. (2007) introduced a method for road network extraction and intersection detection from aerial images in their paper. Their approach combines feature extraction, tracking, and intersection detection techniques to accurately delineate road networks. By analysing the continuity and connectivity of road segments, the algorithm identifies intersections and captures the hierarchical structure of road networks. This work contributed to improving the accuracy and efficiency of road detection algorithms by incorporating high-order topological features such as intersections.

Recent advancements in deep learning have further propelled the field of graph-based end-to-end road detection methods. For instance, RoadTracer, developed by Bastani et al. (2018), represents a significant milestone in automated road extraction. Utilising deep learning and computer vision techniques, RoadTracer achieves state-of-the-art performance in road extraction tasks. By leveraging large-scale datasets and CNN architectures, RoadTracer demonstrates remarkable accuracy and robustness in delineating road networks from aerial imagery. It uses an iterative search process guided by a CNN-based decision function to predict the graph growth. Similarly, Vecroad, introduced by Tan et al. (2020), employs point-based iterative graph exploration for road graph extraction. By iteratively refining graph representations based on local features and connectivity constraints, Vecroad achieves robustness to occlusions and variations in road morphology. This approach showcases the effectiveness of iterative refinement strategies in enhancing road detection accuracy.

Moreover, Li, Wegner, and Lucchi (2019) work on topological map extraction from overhead images and Belli and Kipf's image-conditioned graph generation approach for road network extraction further demonstrate the potential of graph-based methods in road detection. By modelling road networks as topological graphs and leveraging graph neural networks for image-conditioned graph generation, these approaches offer context-aware road extraction capabilities, enabling accurate delineation of road topology from aerial imagery.Homayounfar et al. (2019)'s DAGMapper, introduces a learning-based approach for mapping road networks by discovering lane topology. By utilising deep learning techniques to infer lane configurations and road topology from aerial imagery, DAGMapper achieves scalability and generalisation to diverse environments. This work underscores the importance of incorporating learning-based approaches into graph-based road detection methodologies. Finally, iCurb is introduced in the paper by Xu, Sun, and M. Liu (2021), this approach uses imitation learning for iterative graph generation which predicts points one by one like visual navigation, from candidate points generated from a segmentation model such as FPN. The method proposed in Xu, Sun, and M. Liu (2021) showed promising outcomes, and since our dataset shares similarities, we aim to apply this approach.

In conclusion, end-to-end graph-based methods have significantly advanced the field of road detection from aerial imagery, offering robustness, accuracy, and scalability in delineating complex road networks. Through innovative algorithms, leveraging deep learning techniques, and incorporating iterative refinement strategies, these approaches continue to push the boundaries of road detection capabilities, paving the way for enhanced navigation systems, urban planning, and infrastructure development.

3 Theoretical background

In this chapter, the theoretical background needed to understand the segmentation and vectorisation of road edges will be discussed. The chapter is split mainly into 2 sections. In the first section, an overview of the Convolutional Neural Networks (CNN) and also about ResNet, FPN and UNet, the CNN architectures used in this thesis is presented. The next section will describe the iterative graph generation method using imitation learning.

3.1 Deep Neural Networks

3.1.1 CNN

CNN has emerged as a formidable tool in the realm of image processing and recognition. Inspired by the human visual cortex, CNNs mimic its hierarchical structure, with neurons in successive layers capturing increasingly complex features.

The architecture of CNNs mirrors the organisation of the visual cortex (Yamashita et al. (2018)), where neurons in localised regions respond to specific receptive fields, gradually covering larger areas through overlapping. This hierarchical arrangement enables neurons in initial layers to learn basic features like edges and points, while those in later layers understand high-level features such as objects or shapes. Originally proposed by Bengio and Lecun (1997), CNNs excel in handling data with grid-like topology, such as images.

CNN is a mathematical construct that is typically composed of three types of layers:

1. **Convolutional Layer**: At the core of CNNs lies the convolutional layer, where convolution, a mathematical operation, captures spatial context and a nonlinear activation function (Yamashita et al. (2018)). By employing multiple filters across the input data, CNNs extract local features crucial for learning. Each filter traverses the entire layer, producing a 2-dimensional map of activations termed a feature map (as seen in figure 1). This approach significantly reduces the number of learnable parameters compared to a fully connected layer, as filters are shared, thus efficiently representing both low-level and high-level features. Then these feature maps pass through smooth nonlinear functions like sigmoid or hyperbolic tangent (tanh) to model complex relationships within the data. However, the rectified linear unit (ReLU) has become the most prevalent choice for activation functions in modern neural networks. ReLU computes the function $f(x) = max(0, x)$, providing a simple yet effective way to introduce non-linearity into the network's computations.

2. **Pooling Layer**: While convolution alone may not suffice for complex tasks, its combination with other functions allows CNNs to model intricate relationships in data. Following the convolutional layer, pooling layers are often utilised to down-sample the feature maps, reducing their spatial dimensions while retaining essential

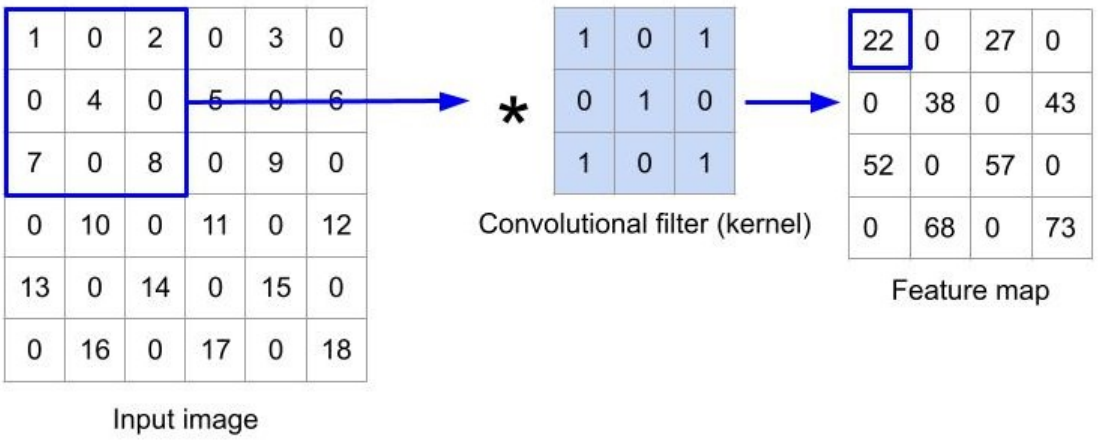


Figure 1: Example of convolution operation with a kernel size of 3×3, no padding, and a stride of 1. A kernel is applied across the input tensor, and an element-wise product between each element of the kernel and the input tensor is calculated at each location and summed to obtain the output value in the corresponding position of the output tensor, called a feature map

information. Max pooling, for example, selects the maximum value from each sub-region of the feature map, effectively highlighting the most significant features and Average pooling takes the average value of the same region (as shown in figure 2).

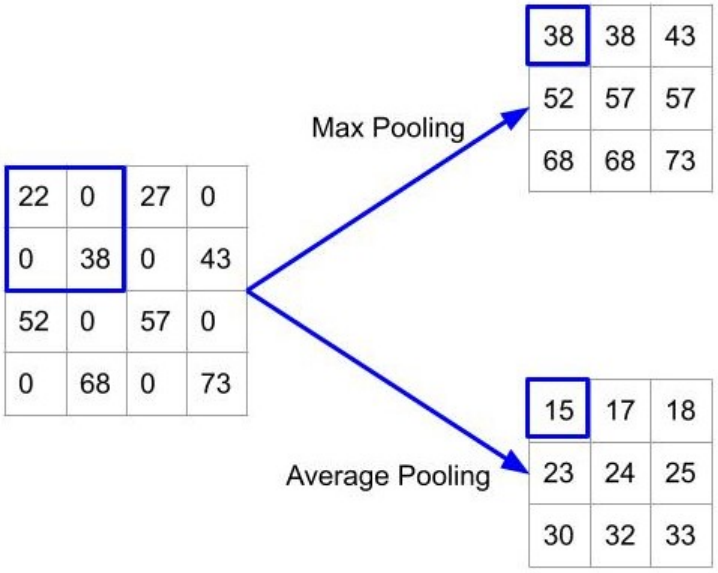


Figure 2: Example of max and average pooling with a kernel of size 2×2, no padding and a stride of 1.

3. **Fully connected layers**: The output feature maps from the final convolution or pooling layer are flattened into a one-dimensional array, and then connected to one or more fully connected layers, also referred to as dense layers. In these dense layers, every input is linked to every output through learnable weights. Once the features

extracted by the convolutional and pooled layers are generated, they are processed by a subset of fully connected layers to produce the final outputs of the network. Each fully connected layer is also followed by a nonlinear activation function, such as ReLU, to introduce nonlinearity into the network's computations (Yamashita et al. (2018)).

3.1.2 ResNet

The rise of deep networks brought forth a new challenge: increased difficulty in training. Deeper networks often encounter issues such as vanishing gradients and accuracy degradation upon convergence. In 2015, He et al. (2015) introduced the ResNet, short for Residual Neural Network architecture to address these challenges.

One of the key architectural elements of ResNet is the inclusion of shortcut connections, also known as skip connections. These connections serve as conduits for information flow, allowing gradients to propagate more effectively through the network. By bypassing one or more layers, skip connections mitigate the detrimental effects of vanishing gradients and alleviate the burden of learning complex mappings.

Notably, ResNet introduces the concept of residual blocks, which consist of stacked convolutional layers augmented by identity shortcut connections. This modular design enables the construction of deep networks while maintaining computational efficiency and ease of optimisation. Furthermore, ResNet advocates for the use of ReLU activation functions, which introduce non-linearity into the network and facilitate feature learning. The combination of residual blocks, skip connections, and ReLU activations empowers ResNet to achieve superior performance on various image classification tasks, surpassing previous state-of-the-art models.

The authors show that for any desired mapping denoted as $H(x)$, the non-linear layers stacked in ResNet fit another mapping $F(x) := H(x) - x$ and the original mapping is recast into $F(x) + x$. They argue that the optimisation of the residual mapping is much easier than the original mapping.

Feed-forward neural networks are used to realise the aforementioned residual formulation with the so-called "shortcut connections" (Ripley 1996). Shortcut connections refer to the skipping of one or more layers. Figure 3 shows one such network used in the ResNet architecture for residual learning using identity mapping and ReLu as the activation. The outputs of the identity mapping are added to the outputs of the stacked layers. Such a model can be trained in a conventional way using stochastic gradient descent with back-propagation.

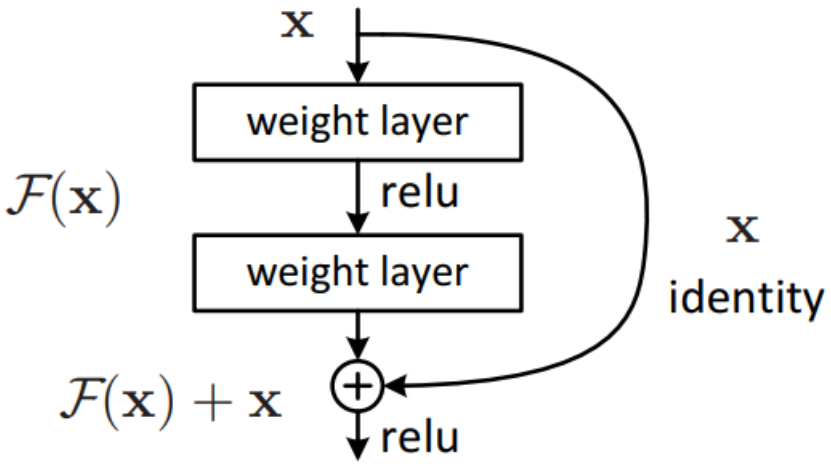


Figure 3: Residual learning: a building block (He et al. (2015))

The main idea of each of the ResNet blocks is to map $H(x) := F(x) + x$, which can be represented as a sum of a residual mapping $F(x)$ and an identity mapping x. The authors He et al. (2015), argue that stacking layers shouldn't degrade the network performance, because using the skip connection the error of the deeper model should not produce a training error higher than its shallower counterparts.

With the success of the architecture, ResNet has been widely used especially in research and for developing frameworks that use it as the backbone. It also serves as the agent for the imitation learning to vectorise road edges which will be further discussed in section 3.2.

3.1.3 FPN

Feature Pyramid Networks (FPNs) represent a pivotal advancement in the domain of computer vision, particularly in the realm of object detection. Traditionally, feature pyramids have been indispensable for recognising objects at different scales, forming the cornerstone of many recognition systems. However, recent strides in deep learning have seen a departure from pyramid representations due to their computational and memory-intensive nature. In response to this challenge, the FPN architecture leverages the inherent multi-scale hierarchy of deep convolutional networks to construct feature pyramids with minimal additional cost.

The key innovation of FPN lies in its top-down architecture with lateral connections as seen in figure 4, which facilitates the generation of high-level semantic feature maps at all scales. By combining low-resolution, semantically strong features with high-resolution, semantically weak features, FPNs effectively bridge the semantic gap and provide rich semantics at all levels of the feature hierarchy (Lin et al. (2016)).

The motivation behind FPNs stems from the need to address the fundamental challenge of recognising objects across a wide range of scales. Traditional methods relied on featurised image pyramids, which were computationally intensive and impractical for real-world applications. Although deep convolutional networks (ConvNets) have emerged as a powerful alternative, they still require pyramid representations to achieve optimal accuracy (Lin et al. (2016)).

FPNs offer a compelling solution by exploiting the pyramidal shape of ConvNet feature hierarchies while maintaining strong semantics at all scales. Unlike previous architectures that focus on producing a single high-level feature map, FPNs operate as feature pyramids, enabling independent predictions to be made on each level. This approach ensures that object detections can be performed effectively across multiple scales, mirroring the functionality of featurised image pyramids.

Central to the architecture of FPNs is the bottom-up pathway, which involves the feed-forward computation of the backbone ConvNet to generate feature hierarchies at different scales. Each stage of the ConvNet produces output maps of varying resolutions, with the deepest layers containing the strongest features. The top-down pathway complements this process by hallucinating higher-resolution features through upsampling and merging them with the bottom-up features via lateral connections.

The efficacy of FPNs has been demonstrated through extensive evaluation of various detection and segmentation systems. Remarkably, FPNs achieve state-of-the-art results on the COCO detection benchmark without the need for additional enhancements. Furthermore, FPNs can run at practical speeds, making them a viable solution for real-time object detection applications(Lin et al. (2016)).

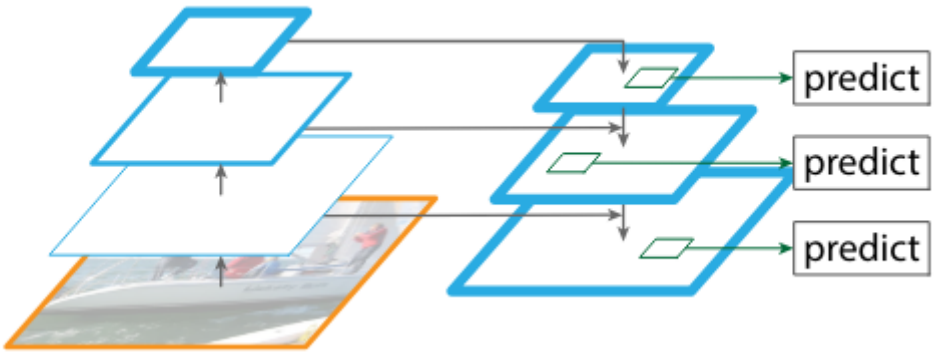


Figure 4: A feature pyramid, with predictions made independently at all levels.(Lin et al. (2016))

3.1.4 UNet

U-Net, a convolutional neural network (CNN) architecture, has gained significant popularity for semantic segmentation tasks, including medical image analysis, satellite image interpretation, and road scene understanding. Developed by Olaf Ronneberger, Philipp Fischer, and Thomas Brox at the University of Freiburg in 2015, U-Net exhibits a distinctive U-shaped architecture, hence its name.

At its core, U-Net comprises an encoder-decoder structure, with skip connections between corresponding encoder and decoder layers as seen in figure 5. This design enables precise localisation while capturing both high-level semantic information and fine-grained details. The encoder portion consists of successive convolutional and pooling layers that progressively downsample the input image, extracting hierarchical features. Meanwhile, the decoder section employs upsampling and convolutional layers to reconstruct the spatial resolution while integrating features from earlier layers via skip connections (Ronneberger, Fischer, and Brox (2015)).

The unique aspect of U-Net lies in its expansive contracting path followed by a symmetric expansive path. This architecture enables the network to learn feature representations at multiple scales, facilitating accurate segmentation even in the presence of complex spatial structures and varying object sizes.

U-Net has demonstrated remarkable performance across various segmentation tasks. In remote sensing applications, U-Net has been deployed for land cover classification, building footprint extraction, and road network detection. Moreover, in autonomous driving, U-Net plays a crucial role in road scene understanding, including lane detection, curbstone segmentation, and obstacle identification.

Despite its effectiveness, U-Net is not without limitations. One challenge is its tendency to produce segmentation artefacts, especially around object boundaries, due to the limited receptive field of the network. Additionally, U-Net may struggle with class imbalance in segmentation tasks, necessitating the use of appropriate loss functions and data augmentation strategies to mitigate this issue.

Overall, U-Net remains a versatile and powerful tool for semantic segmentation, offering a robust framework for addressing a wide range of image analysis tasks across different domains. Ongoing research efforts continue to refine and extend the capabilities of U-Net, making it an integral component in the arsenal of deep learning methods for image segmentation.

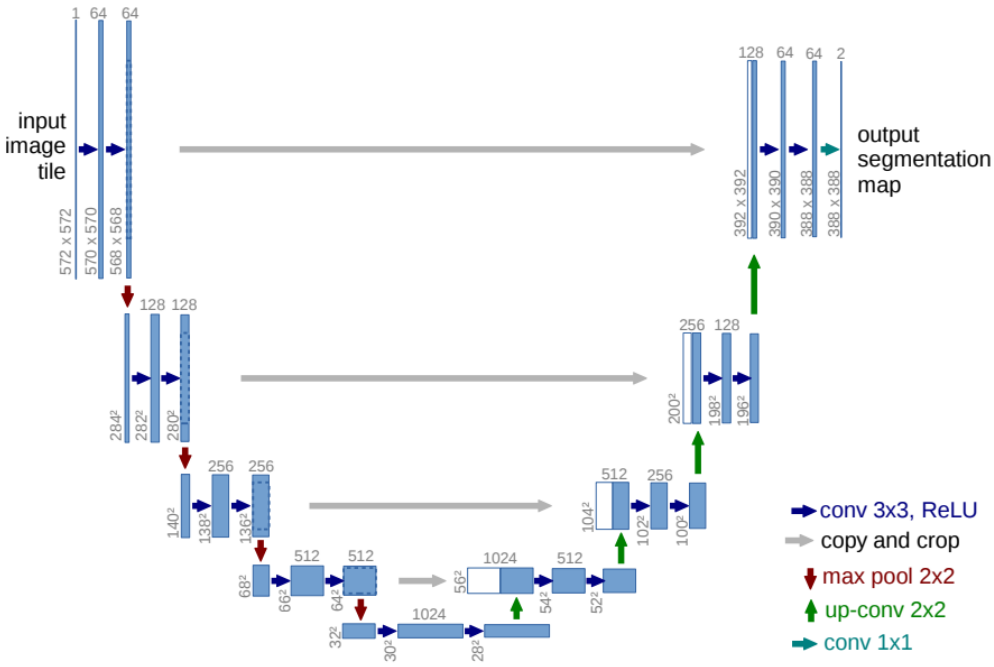


Figure 5: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. (Ronneberger, Fischer, and Brox (2015))

3.1.5 Training of Neural Networks

Training any neural network always involves adjusting the parameters, such as filters in convolutional layers and weights in fully connected layers, to minimise the difference between the predicted outputs and the true labels provided in the training dataset. This process is typically achieved through the backpropagation algorithm, which relies on a loss function and an optimization algorithm like gradient descent (Yamashita et al. (2018)).

The loss function, also known as a cost function, evaluates how well the network's predictions match the ground truth. The goal of training a model is to minimize the error between the ground truth and predicted values by minimizing the loss function. There are various types of loss functions that can be used based on the applications. In this thesis, predominantly two kinds of losses are used:

1. Cross entropy loss: used in classification tasks to measure the distance between class probabilities of ground truth and predicted. Usually, the sigmoid activation function should be used as the last layer before applying cross-entropy. In this thesis, the BCEWithLogitsLoss function available in the nn module of the torch package in Python is used, it combines the Sigmoid layer and binary cross entropy loss and it is calculated as follows:

$$L_{bce} = -\left[y \cdot \log \sigma(x) + (1-y) \cdot \log(1-\sigma(x))\right] \tag{1}$$

where L_{bce} is the binary cross-entropy loss, x is the predicted class probability, y is the class probability of ground truth and σ is the sigmoid function[1] (Pytorch Accessed on: 11.05.2024).

2. L1 and L2 losses: used in regression tasks to measure the difference between actual and predicted values. In L1 loss, the absolute difference between the values is measured and in L2 loss the squared differences between the values are measured. In this thesis, the SmoothL1Loss function available in the nn module of the torch package in Python is used, it is more robust and less sensitive to outliers and is calculated as:

$$L_{l_1} = \begin{cases} 0.5(x-y)^2/beta, & \text{if } |x-y| < beta \\ |x-y| - 0.5 * beta, & \text{otherwise} \end{cases} \tag{2}$$

where L_{l_1} is the Smooth L1 loss, x is the predicted value, y is the ground truth and beta specifies the threshold at which to change between L1 and L2 loss. The value must be non-negative and by default, it is 1 in Python implementations(Pytorch (Accessed on: 11.05.2024)).

Choosing the appropriate loss function is crucial and depends on the nature of the problem being solved (Yamashita et al. (2018)).

Gradient descent is a widely used optimization algorithm that iteratively updates the parameters of the network to minimise the loss function. The gradient of the loss function indicates the direction of the steepest increase, and the parameters are adjusted in the opposite direction of the gradient, with a step size determined by the learning rate hyperparameter. The update rule for a parameter is expressed as:

$$w := w - \alpha \frac{\partial L}{\partial w} \tag{3}$$

Here, w represents a parameter, α is the learning rate, and L denotes the loss function.

In practice, training is done using mini-batches of data due to memory constraints. This approach, known as mini-batch gradient descent or stochastic gradient descent (SGD), computes gradients using subsets of the training data and updates the parameters accordingly. The size of the mini-batch is another hyperparameter that needs to be specified before training begins (Yamashita et al. (2018)).

Several variants of gradient descent, such as Stochastic gradient descent (SGD)(Ruder (2016)) with momentum, Adam (Kingma and Ba (2017)), etc., have been developed to improve convergence and performance. These algorithms incorporate additional techniques to adjust the learning rate dynamically or adaptively during training. While these methods offer advantages, their specific details are beyond the scope of this discussion.

[1]Sigmoid function is one of the most commonly used activation functions; $\sigma(x) = \frac{1}{1+e^{-x}}$ (Chen et al. (2024))

In this thesis, Adam is used as the optimizer for the training process. While stochastic gradient descent maintains a single learning rate for all weight updates and the learning rate does not change during training (Ruder (2016)), the advantage of Adam is that it maintains a learning rate for each network weight (parameter) and is separately adapted as learning unfolds Kingma and Ba (2017).

3.2 Imitation learning

Mimicking human actions has been one of the prominent learning approaches which have proven its great role in the field of robotics and navigation. The method involves learning a policy or behaviour, through observing demonstrations made by experts rather than relying solely on simple trial-and-error exploration (Torabi, Warnell, and Stone (2018)). This approach holds particular significance in addressing complex tasks, as it necessitates minimal expert knowledge while offering applicability across a diverse range of applications (Hussein et al. (2017)).

Imitation learning utilises the concept of studying experts' demonstrations to guide the learning process. Instead of starting from scratch, the agent mimics the behaviour of the expert and, thus, achieves the task faster, acquiring the highest efficiency (Hussein et al. (2017), Torabi, Warnell, and Stone (2018)). The aforementioned is highly useful for cases when it's hard to set up reward functions or the exploration strategies are challenging.

As the basis for imitation learning, the concept of behavioural cloning is used whereby an agent learns a mapping from states to actions mimicking the expert on the state actions taken by it. This process is used mostly in supervised learning techniques, with its input data as state-action pairs drawn from expert demonstrations. It acts by reducing the difference between its forecast actions and the real actions of the teacher.

Imitation learning is one of a kind concept that has some advantages and at the same time presents some challenges but for its effective use, these ought to be handled well. The first problem is to get over the distribution gap between the training data (expert's demonstration) and the testing data (real-world situation). It as a result may cause what is called covariate shift, the learned policy is not able to generalise to unknown states or situations (Attia and Dayan (2018)). Another problem is the issue of compounding errors whereby initially the small errors between the agent's actions and the expert's demonstrations are amplified as soon as time goes by until they become enlarged, leading to further divergence in the performance or suboptimal performance (Torabi, Warnell, and Stone (2018)).
As mentioned before the fundamental concept behind imitation learning is to replicate the behaviour of an expert control policy π^* through a learned policy $\hat{\pi}$. In this thesis, the

curbstone detection problem is addressed as an imitation learning problem. Let's imagine an agent A positioned at a vertex v_t within an ortho-image. Agent A is tasked with executing actions guided by $\hat{\pi}$ to progress to the next vertex v_{t+1} along curbstones, relying on visual cues or the environment around v_t. The expert demonstrations provided by π^* can be derived from ground-truth curbstone labels. This network endeavours to acquire a policy $\hat{\pi}$ that approximates π^*, directing agent A to traverse along curbstones from the starting vertex to the destination vertex. Consequently, agent A learns to construct graphs representing the curbstones vertex by vertex Xu, Sun, and M. Liu (2021).

4 Methodology

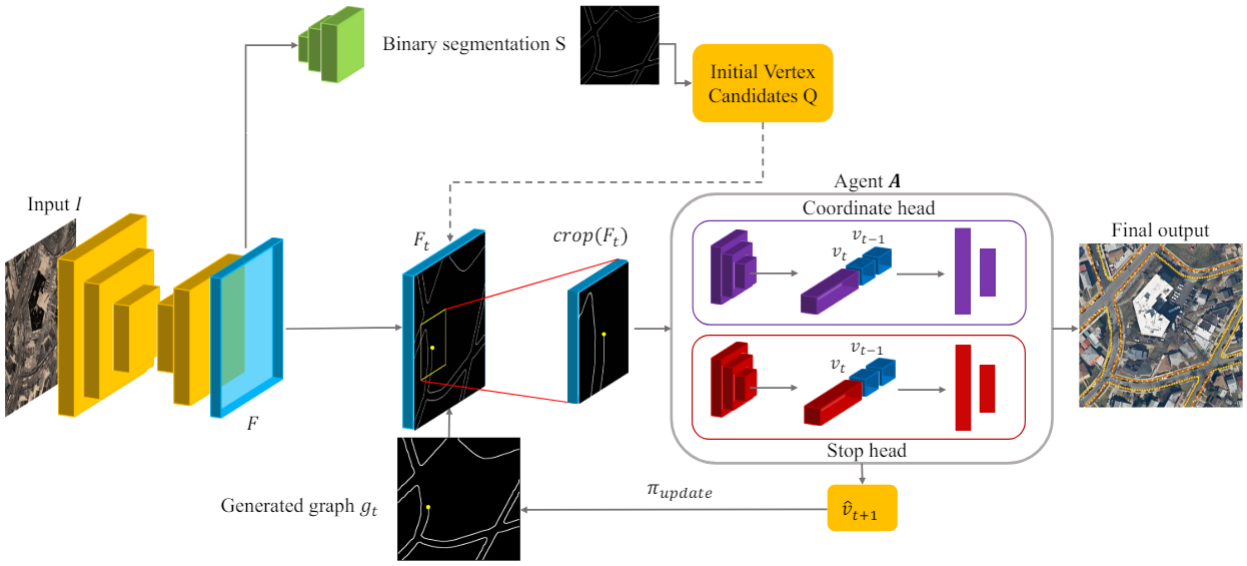


Figure 6: The overview diagram of the iCurb pipeline (Xu, Sun, and M. Liu (2021))

A schematic overview depicted in figure 6 illustrates the iCurb pipeline. An RGB orthoimage I, $C \times H \times W$ with C number of spectral bands and H and W as image height and width respectively, is given as an input and the desired output is a graph $G = V, E$ which represents the curbstones, where V is a vertex set of the generated vertices, $v_t = x_t, s_t$, where x_t is the 2D coordinate of v_t and s_t is a variable indicating whether the graph generation process should cease or continue. The neighbtheing vertices are connected to generate the edges E, during the graph generation.

iCurb encompasses two main problems: the extraction of initial vertex candidates and the subsequent generation of graphs from these points.

Initially, use a deep learning CNN framework like UNet as mentioned in section 3.1.4 to segment the provided input images to generate a binary segmentation mask, S and then extract potential candidates for initial vertices for each segment detected from S. This process yields a list of initial vertex candidates denoted as $Q = \{q_i\}_{i=1}^{N}$, where q_i is one such initial vertex with a total number of N potential vertex points. Subsequently, Q is only used at the beginning of each iteration to grow a graph instance as indicated in the dashed line in figure 6

Subsequently, utilising an imitation learning algorithm for the iterative graph generation approach, predict the curbstone graphs. Commencing from an initial vertex, the algorithm predicts subsequent vertices sequentially. Each vertex in the generated vertices list V comprises 2D coordinates along with a variable indicating whether the graph gen-

eration process should cease or continue, $\{v_t = x_t, s_t\}$. When the value of s_t equals 1, the algorithm halts further point generation for that instance and proceeds to initiate a new graph generation process using an unused initial vertex from the list of initial vertex candidates $Q = \{q_i\}_{i=1}^{N}$.

The iCurb pipeline consists of three steps:

1. Extract the feature map F using the UNet (3.1.4) from the input I as mentioned above. The initial vertex candidates $Q = \{q_i\}_{i=1}^{N}$ are extracted by processing the binary segmentation map S. To save historical information, set all the pixels covered by the predicted vertices or edges until time t as 1 in g_t (initially g_t will be initialised as a matrix containing zeros, like a completely black image).

2. Concatenate F and g_t into a new multi-channel feature map F_t. Here, F_t can be regarded as the environment in imitation learning. Assuming that the current vertex is v_t denoted with the yellow node in figure 6, we crop a $d \times d$ square block $crop(F_t)$ with the centre at v_t to represent the local features of v_t. With $crop(F_t)$ given as input, the agent A predicts the next vertex $v_{t+1} = \{x_{t+1}, s_{t+1}\}$:

$$v_{t+1} = \underset{v_{t+1} \in \text{crop}(F_t)}{\arg\max}\ \pi[v_{t+1}|\text{crop}(F_t), v_t, v_{t-1}] \tag{4}$$

   where π is the policy of agent A, $crop(F_t)$ is the current environment, v_t is the current vertex and v_{t-1} is the previous vertex, x_{t+1} and s_{t+1} are predicted by the coordinate head and stop head, respectively which we will see in detail in the further sections.

3. g_t is updated given the policy π_{update}. During testing, π_{update} directly adds the predicted v_{t+1} into g_t. However, during training, the graph is grown with more complex strategies like DAgger (Ross, Gordon, and Bagnell (2011)) to generate training samples to teach the agent. The detailed training strategy is described in section 4.4. After all the initial vertex candidates in Q being processed, the agent A is trained on D and the policy π is updated based on the equation:

$$\hat{\pi_{update}} = \arg\min_{\pi} \sum^{t} |\pi[\text{crop}(F_t), v_t, v_{t-1}] - v^*_{t+1}| \tag{5}$$

   where v^*_{t+1} is the ground truth, which will be selected based on dynamic labelling discussed in further sections.

In the following sections, we will explain the above steps in more detail.

4.1 Feature extraction backbone

In the implementation, the high-resolution RGB ortho-images I, with dimensions $C \times H \times W$, are fed into the segmentation model to generate a corresponding feature map

F, with dimensions $C_R \times nH \times nW$, where C_R is the number of dimensions in the feature map generated by the segmentation model before any non-linear operations and n is a factor by which the size of the feature map reduces when compared to the input image.

In the work, we experiment with FPN and UNet for the feature extraction backbone. An important advantage of UNet is that the resulting feature map matches the size of the input image, eliminating the need for upsampling and it has 63 dimensions.

In contrast, when utilising FPN as the backbone, the resulting feature vector was of size $8 \times H/2 \times W/2$, necessitating upsampling to generate a feature map of the original image size. This step is crucial for graph growth and further processing.

It is noteworthy that for future development if a different network is opted as the backbone, the feature map generated at the end should match the size of the input image. This consistency is vital for maintaining the integrity and accuracy of subsequent processing steps.

4.2 Segmentation and initial vertex candidates

In the pipeline, the segmentation head plays a crucial role in extracting the binary segmentation mask from the high-resolution ortho-images. This process begins with the feature map generated in the preceding section 4.1. The feature map undergoes further processing through a convolutional layer layers, followed by a softmax layer, resulting in the generation of a curbstone probability map. This probability map provides insight into the likelihood of each pixel belonging to a curbstone or not.

Subsequently, utilising the curbstone probability map, apply a threshold to filter out pixels with low probability (less than 0.75 probability), effectively generating a binary mask representing the segmentation of curbstones. This binary mask undergoes further refinement through skeletonisation, which reduces the binary mask to one pixel-wide representation (T. Y. Zhang and Suen (1984)), then using the measure function available in skimage package in Python we label connected regions and then eliminate short and small segments to enhance the accuracy of the segmentation and ignore outliers.

Finally, proceeds to select the initial vertices of the connected regions (which is the first element in the list of coordinates), which is then added to the list of initial vertex candidates denoted as Q. This process ensures that the pipeline is equipped with reliable initial vertex candidates, laying the foundation for subsequent graph generation and analysis.

4.3 Agent network

Once both the feature map matching the size of the input image, and the list of potential initial vertex candidates are generated, we move on to the imitation learning segment of the pipeline.

The agent within the iCurb pipeline receives inputs comprising a cropped portion of the feature map concatenated with the graph generated to provide historical context $crop(F_t)$ centred at the current vertex v_t. In this implementation, the size of the crop ($d \times d$) was opted to be 63×63, the value should be in correspondence with the size of the input images used. Having the centre, v_t and size, d we select or crop the necessary region by finding the coordinates of the window in the real image. Initially, this crop is centred at the first initial vertex candidate and subsequently shifts to the point predicted by the agent, and so forth. Additionally, the input includes information about the two preceding vertices.

The iCurb agent is equipped with two prediction heads: the coordinate head and the stop head. The coordinate head focuses on predicting the next 2D coordinates (x_{t+1}) for graph expansion, while the stop head determines whether the graph expansion should continue or stop (s_{t+1}). They have the same network structure i.e., ResNet architecture except for the number of output channels.

- **Coordinate head**: To extract the local spatial details around the current vertex (v_t), the cropped feature map concatenated with the historical graph information, g_t is fed to the coordinate head. It then predicts a relative displacement (Δx_t) of the predicted vertex within the scope of the crop. The updated vertex coordinate (x_{t+1}) is computed as the sum of the current vertex coordinate (x_t) and the displacement (Δx_t). During training, the displacement (Δx_t) is scaled from a range of $[0, d]$ to $[1, 1]$ for numerical stability. To enhance historical information and encode absolute coordinates, the feature vector is concatenated with two past vertices (v_t and v_{t-1}).
- **Stop head**: On the other hand, the stop head also extracts the local spatial details around the current vertex (v_t), we feed the cropped feature map concatenated with the historical graph information to the coordinate head. It then predicts the stop action based on certain conditions. These conditions include reaching the end vertex of a curb instance, generating incorrect graphs, or spending excessive time on growing a curb instance, which could lead to local infinite loops. When a stop action is triggered, iCurb initiates the growth of another curb instance if there are remaining vertices in the candidate list (Q). Otherwise, iCurb outputs the final graph for the image. Since stop actions are less frequent than normal actions, the training data may be unbalanced and thus may result in the graph growing beyond the necessary limits.

4.4 Training strategy

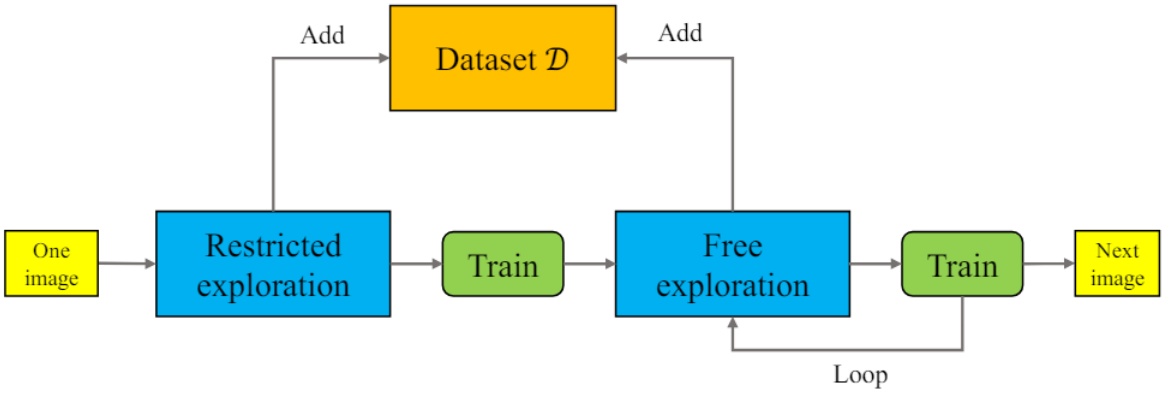


Figure 7: The diagram of the training strategy Xu, Sun, and M. Liu (2021)).

We use a DAgger-based training strategy (Ross, Gordon, and Bagnell (2011)) to teach agent A of imitation learning to mimic expert policy which will be explained in further sections. This algorithm has two different kinds of exploration methods to generate training samples for imitation learning: it has a common framework but a different update policy. The pipeline is visualised in figure 7. Given an ortho-image I as input, iCurb first runs the restricted exploration method. Then it adds the generated samples into the aggregated dataset D and we train iCurb on D. After this, we run N rounds of free exploration. In each iteration, we aggregate the data and train iCurb on D. To trade off effectiveness and efficiency, we set N as 1 so that we have a balance between restricted and free exploration. Finally, iCurb finishes processing the current ortho-image and turns to work on the next one.

To make sure that the agent moves forward until the stop action is triggered, we use the following constraints on the ground truth label $v*_t$ used to train the agent:

1. $v*_{t+1}$ should be after $v*_t$ i.e., $v*_{t+1}$ is more closed to the end vertex along the curb instance.

2. $v*_{t+1}$ should be at a distance from $v*_t$ to prevent the agent from being trapped in a local infinite loop. So in the implementation, we first obtain points of the ground-truth road curb that are after $v*_t$ and have enough distance to $v*_t$, and then find $v*_{t+1}$ among them.

Such a process is visualised in figure 8.

Restricted exploration: In the restricted exploration method, we have some restrictions on the agent A:

1. A is restricted to be within 15 pixels of the ground-truth road curb graphs.

2. A stops when and only when it is close to the end vertex.

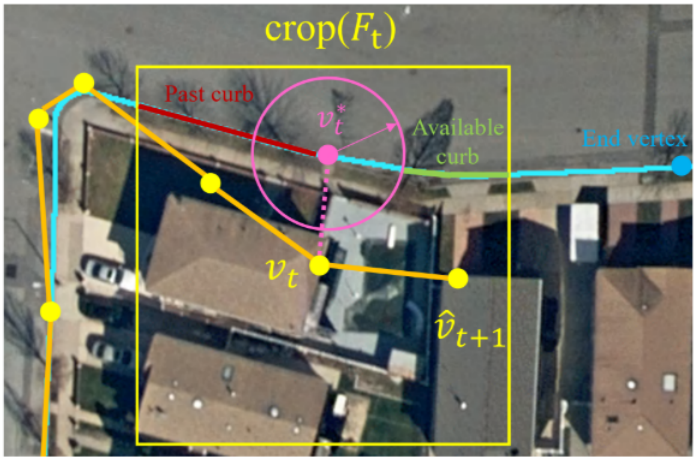


Figure 8: The visualisation of the principles to generate labels for agent training. The ground-truth road curbs are shown by cyan lines. The label used to train the agent is shown by the pink node. The generated vertices are denoted by yellow nodes, and the edges are solid orange line segments. The yellow rectangle is the environment (i.e., $crop(F_t)$). Suppose now we have v_t and $v*_t$. In this example, the agent moves from left to right towards the blue end vertex. Within $crop(Ft)$, the red lines represent explored road curb pixels (i.e., past road curb). Then, we draw a pink circle centring at $v*_t$, whose radius is 15 pixels in the experiments. $v*_{t+1}$ should be outside of this circle. In this example, $v*_{t+1}$ must be located on the green road curb pixels (i.e., available road curb). In this way, we can guarantee that $v*_{t+1}$ is after $v*_t$ and away from it far enough at the same time (Xu, Sun, and M. Liu (2021)).

The above restrictions can make sure that A travels along road curbs and generates relatively high-quality training samples. The process is visualised in Fig. 3 for better demonstration.

Correspondingly, the above restrictions are realised by the following principles:

1. When the distance between predicted vertex v_{t+1} and its corresponding ground truth $v*_{t+1}$ is smaller than 15 pixels, we directly addv_{t+1} into the graph and continue. But if the distance is larger than 15 pixels, we will make a correction and add $v*_{t+1}$ into the graph instead to prevent A from being far from the ground-truth road curb.

2. If $v*_t$ is closed to the end vertex within a threshold distance, A should stop.

Using the restricted exploration technique can indeed enhance the accuracy of training samples near ground-truth road curbs, leading to faster convergence and improved overall results. However, it's important to note that this method can create a disparity between training and testing. If left unaddressed, this could result in unexpected actions by the agent during testing, especially in unexplored scenarios. Therefore, relying solely on restricted exploration may not align with the objectives.

As the training progresses, the effectiveness of the restricted exploration approach diminishes. This can be likened to the scenario where a student grows more proficient over

time, thereby reducing the necessity for constant guidance from the teacher.

Figure 9: The demonstration of the restricted exploration method. The ground-truth road curbs are shown by cyan lines. The generated vertices are denoted by yellow nodes, and the edges are solid orange line segments. The dashed orange lines represent possible edges that could be added in step $t+1$. Assume A is now at the vertex v_t whose ground-truth label is $v*_t$. Based on $crop(F_t)$ represented by the yellow rectangle, agent A makes the prediction of the next vertex $\hat{v}_{t+1}$. Then we find the nearest point of the ground-truth road curbs to $\hat{v}_{t+1}$ as the label $v*_{t+1}$. There are some restrictions on $v*_{t+1}$, e.g., it must be after $v*_t$ and the distance between them should be larger than a threshold. If the distance between $v*_{t+1}$ and $\hat{v}_{t+1}$ is lower than 15 pixels, $\hat{v}_{t+1}$ is directly added into the graph ($\hat{v}_{t+1}$ and dashed edge 1 are picked); but if the distance is larger than 15 pixels, $v*t+1$ is used instead to update the graph ($v*_{t+1}$ and dashed edge 2 is picked) (Xu, Sun, and M. Liu (2021)).

Free exploration: Several kinds of uncommon states have a great impact on the results:

1. Normal stop action: Normal stop action means that A stops when it is very close to the end vertex.

2. Stop action when A is too far from the current road curb: A may reach someplace where it loses sight of the current curb instance. In some cases, it finds another curb instance and follows the wrong instance, which may happen when two curb instances are parallel and close to each other.

3. Stop action when A is trapped locally: A may be trapped in a local infinite loop, and it may or may not be able to get out, so it should be stopped after such an infinite loop is detected. Once such a state is not handled properly, the processing time would greatly increase.

4. Stop action when A goes backwards (i.e., A turns back and moves towards the initial vertex instead of the end vertex along a road curb).

These states rarely happen during testing, but if the stop head fails to trigger the stop action when they occur, the obtained graph will have serious errors. For example, if A is not stopped from leaving the ground-truth road curbs, a long and incorrect curb instance

will be generated. Thus, there should be enough samples for the agent to learn how to handle these cases during the training period. Some examples are visualised to show the above states that occurred during the free exploration in figure 10.

Graph-based baselines in the comparison experiments cannot effectively handle these states due to their naive training strategy. Thus, they tend to generate more noisy outputs. The free exploration can generate enough samples to cover these uncommon states so that iCurb could achieve much better performance.

(a) Example of state (2). Agent A fails to trigger a stop action or make corrections when it generates incorrect vertices (in the red rectangle). Then an obvious incorrect curb instance is obtained.

(b) Example of state (2) and state (1). Agent A fails to stop when it is close to the end vertex, turns in the wrong direction, and generates many meaningless vertices.

(c) Example of state (4). The agent A correctly generates the graph at first (1). However, when it gets close to the end vertex, it suddenly turns and goes back (2). Then it generates incorrect vertices along the curb in the wrong direction towards the initial vertex (3).

Figure 10: Visualisation of the generated graph once uncommon states is not properly handled. For all images, cyan lines are ground-truth curbs. Blue points denote the end vertices and green points denote the initial vertices. Orange line segments are the edges of the trajectories. Yellow points are normally added vertices, and red points represent vertices that stop action should have been triggered. Red points with pink fill mean that a local infinite loop is detected (A may or may not escape from the loop).

4.5 Slope Penalty loss

To further improve the training process for the curbstone detection model, we introduced an additional loss component known as the Slope Penalty Loss. This novel loss mechanism is designed to penalise deviations in the slopes of predicted curbstone coordinates compared to their ground truth counterparts.

If $(x_p, y_p), (x_t, y_t)$ are the predicted and true vertices respectively. The implementation of the additional slope penalty loss can be summarised as:

1. Calculate the angles of rotation for both the predicted curbstone coordinates and the corresponding ground truth coordinates relative to the positive X-axis.

$$\theta_p = atan2(y_p, x_p)$$
$$\theta_t = atan2(y_t, x_t)$$

2. The difference between the predicted and ground truth angles is computed to quantify the degree of deviation.

$$\Delta\theta = |\theta_p - \theta_t|$$

3. To ensure that the angle differences are properly handled, angles greater than 180 degrees are replaced with their complementary angles. This step is crucial for maintaining consistency and accuracy in the loss calculation process.

$$\Delta\theta = \begin{cases} \pi - \Delta\theta & if \Delta\theta > \pi \\ \Delta\theta & otherwise \end{cases}$$

4. The final step involves finding Smooth L1 loss between the angle differences and a zero tensor. The Smooth L1 loss is chosen due to its robustness in handling outliers and its ability to provide a balance between accuracy and smoothness in the loss landscape.

$$L_{slope} = SmoothL1Loss(\Delta\theta, 0)$$

Integrating the Slope Penalty Loss into the overall training process is achieved by combining it with the existing coordinate loss which is regression loss calculated using smooth L1 loss between the predicted and true vertices using a weighted sum. The weighting factor, w determines the influence of the slope penalty on the overall loss calculation. By adjusting this coefficient, we can effectively control the impact of the slope deviation on the training dynamics.

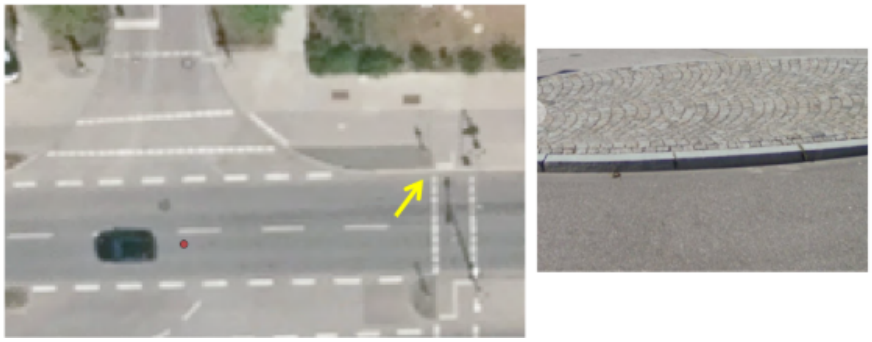
$$L = L_{l_1} + w \cdot L_{slope} \tag{6}$$

5 Experimental Setup

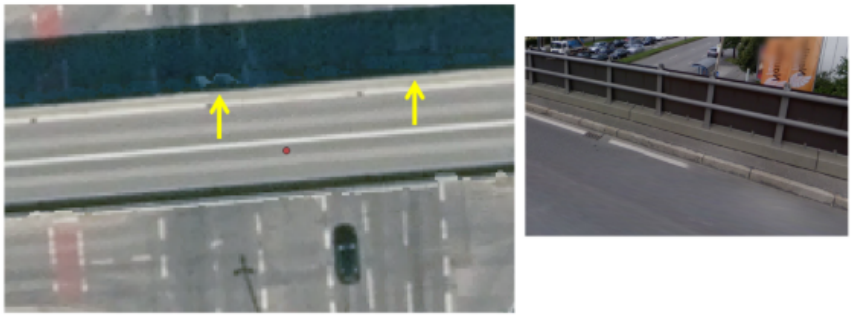
5.1 Dataset

A novel dataset was used for training, validation and testing of the model used in this thesis. The data was provided by the Agency for Digitisation, High-Speed Internet and Surveying (LDBV), Bavarian Germany. This dataset comprises true orthophotos (TrueDOP-20) with 20cm ground resolution with three channels (RGB = red, green, blue) capturing Munich, Germany. The training data comprised 49, 3500×3500 sized ortho-images taken from different regions like Englischer Garten, Ismaning, Obersending, Ostpark and Pasing in Bavaria, which were split into 512 × 512 sized patches (each image is split into 144 patches with 50% overlap), 6048 RGB patches for training and 150 RGB patches out of 1008 available for validation of the model. 12 ortho-images from regions like Lindau and Milbertshofen of sizes 3500 × 3500 pixels are used directly (without splitting) to test the trained model.

For each ortho-image, a curbstone vector ground truth data is provided by a national surveying agency in a vector format as 2D Line Strings with attributes. To facilitate the training and testing, this data is converted into a rasterised form for segmentation and also a dense annotation sequence for the iterative graph generation using imitation learning.



(a) Curbstone next to a pedestrian way



(b) Curbstone next to the fence of a bridge

Figure 11: Areas with curbstones (marked with the yellow markers) (Tian et al. (2023))

Figure 12: Areas without curbstone annotations (marked with the yellow lines), curbstones occluded by buildings and tree branches (Tian et al. (2023))

Due to the aerial perspective of the camera sensors, inherent challenges arise, including occlusions caused by buildings, tree branches, and parked vehicles. In regions where such occlusions occur as seen in figure 12, the ground truth data are considered outliers and are not annotated [2].

5.2 Training process

A robust pipeline is adopted by leveraging the algorithm detailed in Section 4 alongside the dataset outlined in Section 5.1 to detect and generate graphs of road edges from ortho-images.

The training process consists of two phases. Initially, the focus is on pre-training the segmentation model. Subsequently, train the iCurb pipeline, utilising the pre-trained weights obtained from the segmentation model.

During the training phase of the segmentation model, feed batches of images sized 512×512 into the model, each batch comprising 8 images. The model undergoes training for 30 epochs, employing the Adam optimisation algorithm to update the weights. Initialised the learning rate at 0.0001, which decreases by a factor of 0.5 using a learning rate scheduler when the learning stagnates for more than 3 epochs i.e, in the validation process if the F1-score doesn't increase for more than 3 epochs the learning rate is reduced and the training is continued. The loss function employed for the segmentation model is the Binary Cross Entropy Loss with Sigmoid Function (BCEWithLogitsLoss) (Pytorch Ac-

[2] the outlier ground truths are removed using the curbstone map refinement method described in Tian et al. (2023)

cessed on: 11.05.2024). To prevent overfitting and enhance generalisation, validate the model after every epoch and save the model with the highest F1 score.

For the training of the imitation learning agent in the iCurb pipeline, begin by generating training samples, following the procedure outlined in Section 4.4, where each image is processed individually. These image patches, sized 63×63, are then input into the agent in batches of 128 images for training. The training of the agent involves both restricted and free exploration for sample generation, conducted sequentially. This sequential exploration process contributes to the time-consuming nature of model training. The agent model is trained for 3 epochs, utilising the Adam optimisation algorithm to update the weights. Similar to the segmentation model, start with an initial learning rate of 0.0001, which is halved using a learning rate scheduler upon learning stagnation observed for more than 3 validations. To prevent overfitting and enhance generalisation, validate the model after every 500 images and save the model with the highest F1 score.

The agent network comprises two heads, each serving distinct purposes. The loss of the coordinate head is calculated using a combination of two different losses, as discussed in Section 4.5. The first loss involves the regression loss of the predicted point, calculated using the smooth L1 loss function (Pytorch (Accessed on: 11.05.2024)), while the second loss, Slope penalty loss quantifies the angle between the ground truth and predicted points. For this experiment we consider the weight of slope penalty loss to be 0.01, i.e., $w = 0.01$ in equation 6. The loss of the stop head is measured using the Binary Cross Entropy Loss with Sigmoid Function (BCEWithLogitsLoss).

All training procedures were conducted using an NVIDIA GeForce GTX TITAN X GPU.

5.3 Evaluation metrics

Metrics are used to measure the performance of the iCurb pipeline and the evaluation of the results is done using pixel-wise metrics like precision, recall and F1-score.

Let's assume two sets of pixels: P_{GT} represents the pixels covered by the ground truth graphs, while P_{pre} represents the pixels covered by the predicted graphs.

Precision is the capability of a model to identify only relevant points(Padilla et al. (2021)). So in this case, precision quantifies the accuracy of the predicted graphs by measuring the proportion of pixels in the predicted graph, P_{pre} that align with the ground truth within a specified threshold τ. This is computed by dividing the number of pixels in P_{pre} that fall within τ of the ground-truth graph by the total number of pixels in P_{pre} (Xu, Sun, and M. Liu (2021)).

$$Precision = \frac{|\{p|d(p, P_{GT}) < \tau, \forall p \epsilon P_{pre}|}{|P_{pre}|} \tag{7}$$

Recall is the capability of a model to find all the relevant points(Padilla et al. (2021)). It is determined by calculating the ratio of pixels in P_{GT} that fall within τ of the predicted graph to the total number of pixels in P_{GT} Xu, Sun, and M. Liu (2021).

$$Recall = \frac{|\{p|d(p, P_{pre}) < \tau, \forall p \epsilon P_{pre}|}{|P_{GT}|} \tag{8}$$

here the function $d(a, B)$ computes the euclidean distance from a point a to the nearest point in a set B. Here,$\{.\}$ denotes a set, and $|.|$ represents the number of elements in the set. In this thesis, experimental results with different threshold values τ set to 2.0, 5.0, and 10.0 pixels respectively are presented. Curbstone, the object to be detected is a thin linear element along the road, so experiments are conducted with small threshold values corresponding to 40, 100 and 200cm respectively based on the ground resolution.

A larger τ yields superior results because it provides greater flexibility in prediction tolerance. Essentially, increasing τ allows for more leniency in assessing the proximity of predicted points to the ground truth.

Precision and recall serve as complementary metrics. Precision focuses on the model's ability to minimise false positives, so a high precision score indicates that the predicted graphs closely match the ground truth graph, with minimal erroneous inclusions. On the other hand, recall emphasises the model's capability to minimise false negatives. so a high recall score suggests that the model effectively captures the entirety of the ground-truth graphs, minimising omissions. So to always try to strike a balance between precision and recall, thus calculating the metrics **F1-score** as:

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

6 Results and Discussion

This section presents and evaluates the results that have been obtained. In the first part of the experiment, the semantic segmentation results of FPN and UNet variants that were introduced in section 3 are evaluated, because the accuracy of the segmentation results further impacts the curbstone vectors detected. In the next set of experiments, the vectorised results of these segmented results using heuristic processes (Hough transform, Douglas Peucker) and iterative graph generation (iCurb and improved iCurb) are evaluated.

6.1 Segmentation results

Upon evaluating the segmentation results generated by the FPN and UNet models, UNet outperformed FPN in terms of precision, recall and F1 score. Examining the evaluation metrics presented in Table 1, a significant difference in recall values between UNet and FPN is noted. Specifically, the recall value increased from 0.427 for FPN to 0.622 for UNet, which is a 20% increase in recall, indicating a notable improvement in the model's ability to correctly identify relevant pixels. Even the precision value increased from 0.615 to 0.656 for UNet, a 4% increase in the precision value, the increase in recall and precision is translated to a 15% improvement in the F1-score. Consequently, for subsequent comparisons, UNet as the segmentation model is opted.

Segmentation	**Precision**	**Recall**	**F1-score**
FPN	0.615	0.427	0.487
UNet	0.656	0.622	0.628

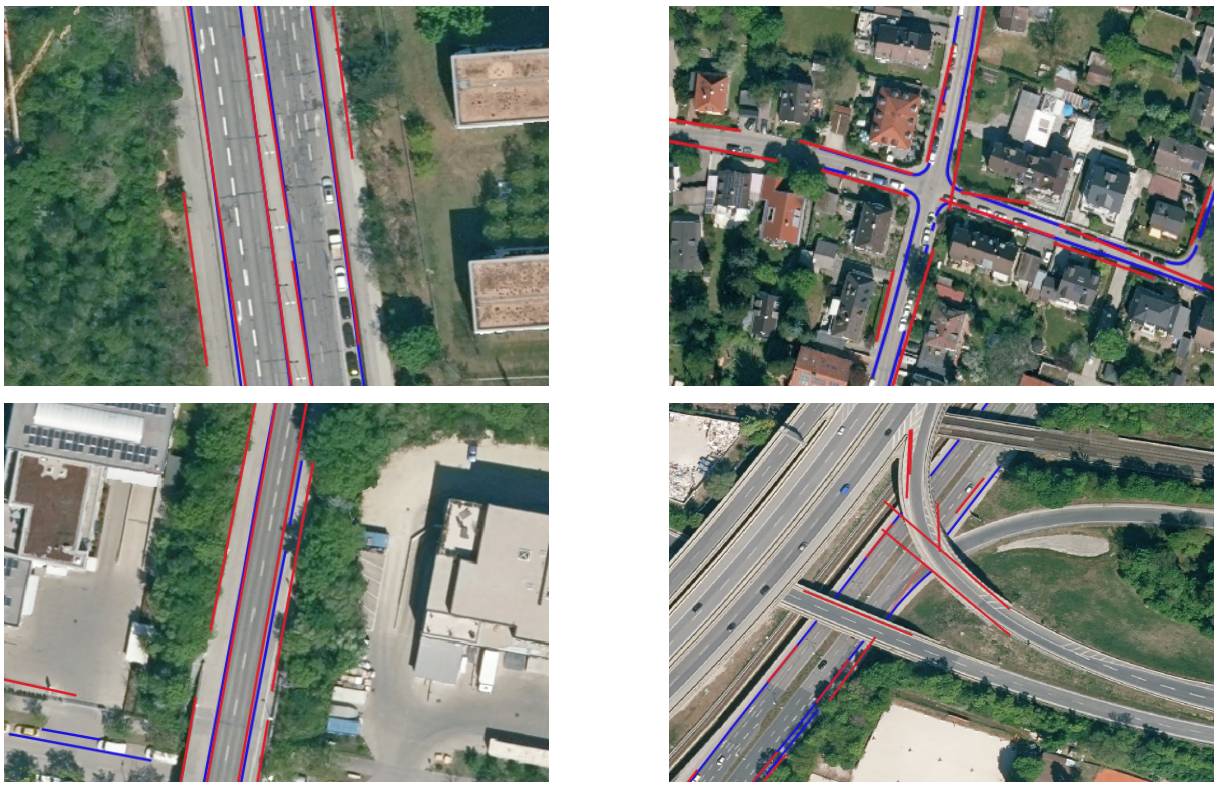
Table 1: Segmentation evaluation results

6.2 Vectorisation with traditional methods

Upon obtaining the segmentation results, the next task was to convert the rasterised binary mask into a vectorised format. For a naive comparison of the results produced, created a baseline by doing a series of heuristic processing like Hough transform, and Douglas Peucker on the segmentation results obtained.

Initially, explored segmentation-based approaches, leveraging the understanding that most road edges exhibit linear characteristics. Employing the Hough transform, attempted to detect line elements within the segmentation mask. While this approach yielded satisfactory results for linear segments of curbstones as in figures 13a, its performance faltered in more complex scenarios involving curves and junctions. From figures 13b top, even though the linear detections seem okay, the curves and the complex structure of the junction are not captured by the Hough transform. And in 13b bottom, when there are over

bridges in the scene, wrong linear detections are predicted.

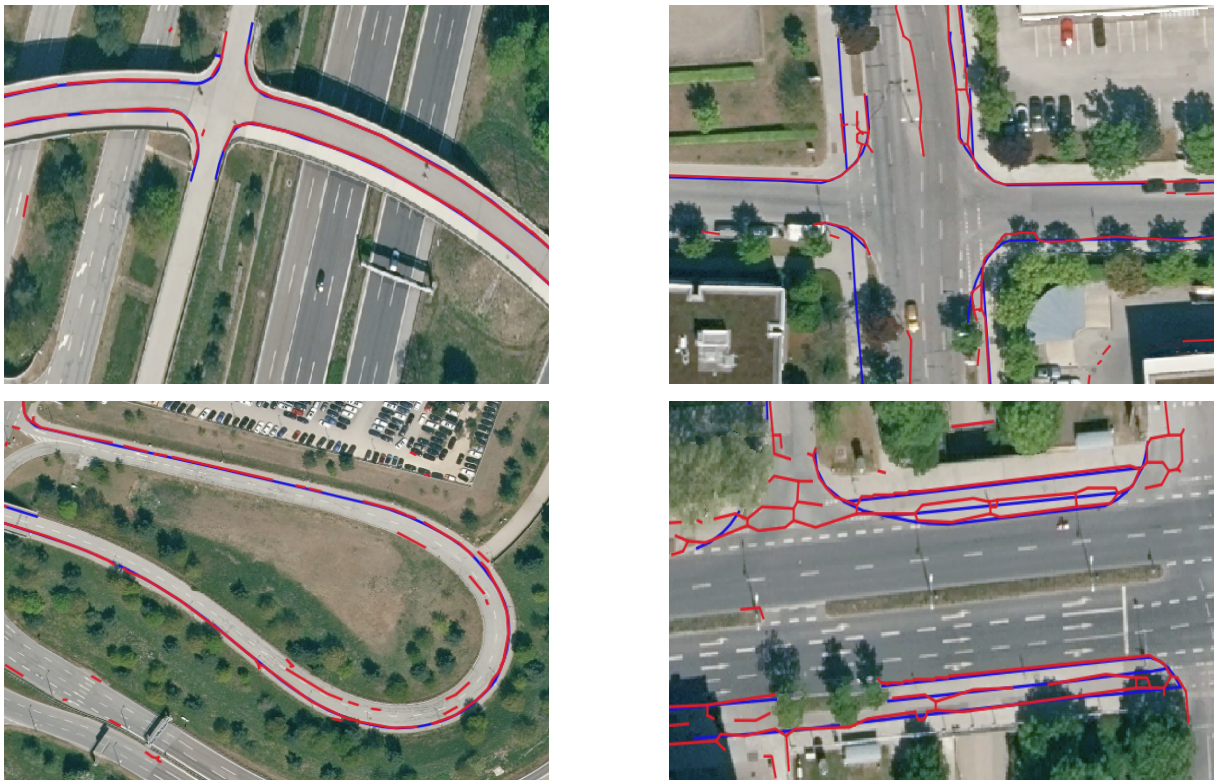


(a) (b)

Figure 13: Examples of Hough transform results (Blue lines indicate the ground truth, red lines indicate the predictions)

Subsequently, experimented with the Douglas-Peucker algorithm to vectorise the segmentation results. Although Douglas-Peucker showed improvement over the Hough transform in detecting curves and junctions as seen in figures 14a, it failed to accurately constrain the shape of the vectors produced. This led to the detection of numerous spurious vectors that lacked practical utility in real-life applications. In figures 14b, the egregious errors in understanding the shape of the segments detected can be attributed to the skeletonising algorithm to reduce the binary mask to one pixel-wide representation (T. Y. Zhang and Suen (1984)). Furthermore, despite employing the Douglas Peucker algorithm, these errors remain unmitigated.

One of the primary limitations of the Hough transform is its focus on detecting linear elements exclusively. However, Douglas Peucker addresses this by vectorising all segments, resulting in a significantly higher recall compared to the Hough transform—nearly 42% better. Nevertheless, as previously mentioned, the precision of the detected vectors varies, with some being highly inaccurate. Consequently, the final F1-score of the Douglas Peucker algorithm does not differ significantly from that of the Hough transform.



(a) (b)

Figure 14: Examples of Douglas Peucker results (Blue lines indicate the ground truth, red lines indicate the predictions)

6.3 Vectorisation with iterative graph generation

Recognising the limitations of these methods, graph-based approaches are explored. Inspired by the iterative graph generation framework proposed in the paper by iCurb, decided to tackle vectorisation as an iterative graph generation problem. Implementing the iCurb pipeline for curbstone vectorisation yielded significantly improved results compared to previous techniques.

From examining the results presented in figures 15, a considerable enhancement in the generated vectors is evident. Despite some areas where the vectors lack smoothness in linear regions, the overall outcome is notably improved and visually more appealing. Although there is a decline of 24% in recall compared to the previous Douglas Peucker method, the detected vectors exhibit a 10% increase in precision over those produced by Douglas Peucker. Consequently, there is a 12% improvement in the F1 scores.

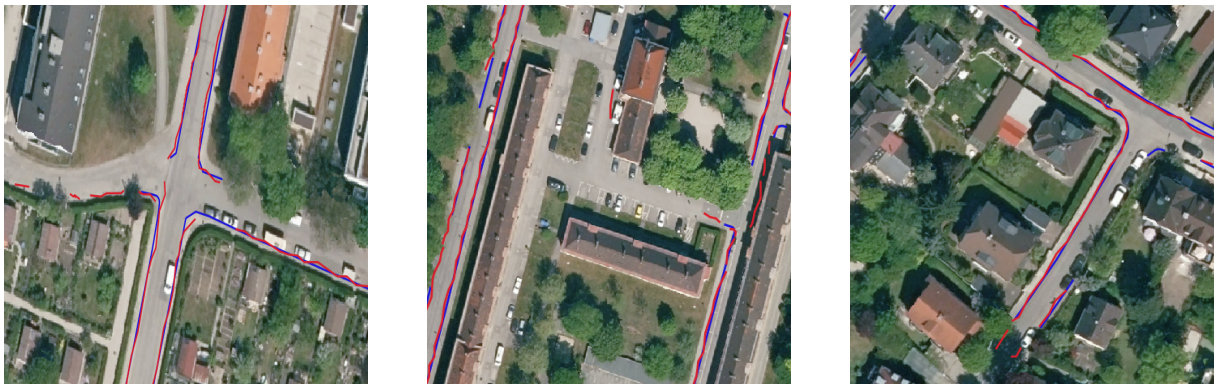


Figure 15: Examples of iCurb algorithm results (Blue lines indicate the ground truth, red lines indicate the predictions)

To further enhance the quality of the vectorised outputs, experimented with incorporating an additional slope loss as mentioned in section 4.5 into the training process of the iCurb pipeline. The addition of the slope loss proved to be beneficial, as it contributed to the increased smoothness of the generated graphs. This resulted in more continuous and visually appealing representations of curbstone boundaries as shown in figures 15.

A remarkable 30% surge in the precision of the detected vectors was reported. Despite still encountering a lower recall rate compared to Douglas Peucker, the F1 score saw a substantial 25% increase from the iCurb algorithm without the slope loss during training.

Methods	**Precision**			**Recall**			**F1-score**		
	1.0	2.0	5.0	1.0	2.0	5.0	1.0	2.0	5.0
Hough transform	0.071	0.092	0.127	0.249	0.322	0.438	0.108	0.140	0.193
Douglas Peucker	0.093	0.106	0.112	**0.711**	**0.806**	**0.856**	0.163	0.185	0.196
iCurb	0.148	0.191	0.220	0.421	0.536	0.616	0.212	0.272	0.313
iCurb(+ slope loss)	**0.391**	**0.506**	**0.530**	0.468	0.605	0.636	**0.417**	**0.539**	**0.565**

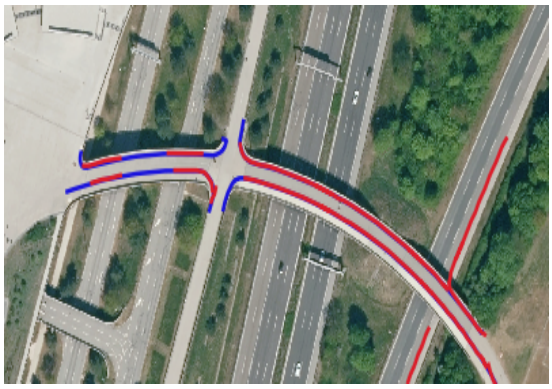
Table 2: Quantitative results for the comparison between segmentation-based and graph-based approaches. The best results are highlighted in bold format. 1.0, 2.0 and 5.0 are the different thresholds in pixels used to calculate the different metrics as mentioned in section 5.3.

(a) Ground truth (b) Hough transform (c) Douglas Peucker (d) iCurb (e) iCurb (+ slope penalty loss)

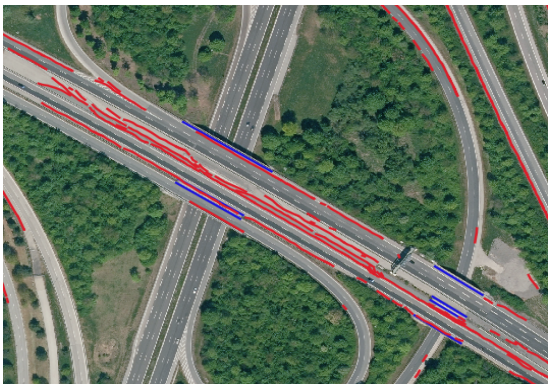
Figure 16: Qualitative demonstrations (Blue lines indicate the ground truth, red lines indicate the predictions)

6.4 Limitations

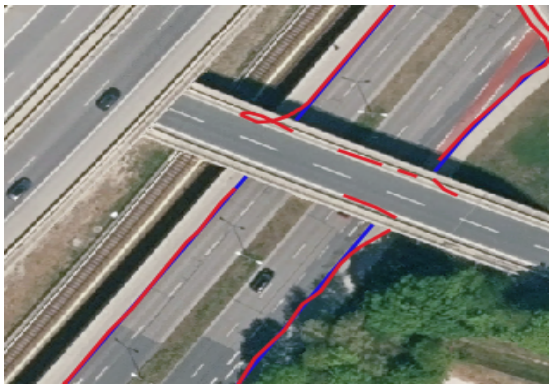
Although strides have been made in improving the results, there remain several scenarios where the algorithm falls short. In figures 17a and 17c, for instance, errors in graph growth when dealing with overbridges are observed. Here, the algorithm struggles as it perceives the graph instance growing along the overbridge and the one growing under it as a single instance. Figure 17b and 17d present another challenge, wherein the high number of false positive detections in the segmentation results significantly impacts the final outcomes, with numerous detections occurring where none exist. Additionally, Figures 17e and 17f depict instances of parallel curbstone instances. While one instance showcases promising results possibly due to better segmentation in that area, the other exhibits poor outcomes.



(a)

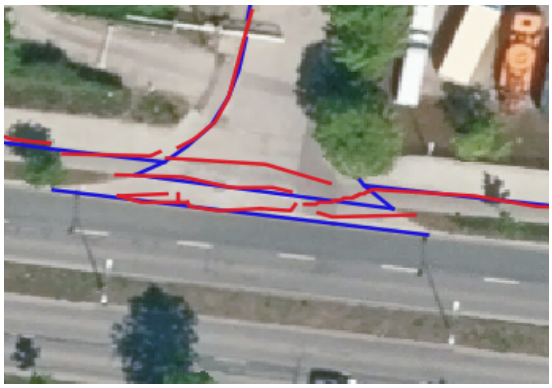


(b)

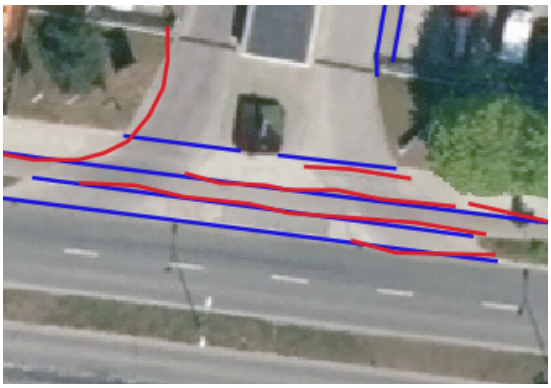


(c)

(d)



(e)



(f)

Figure 17: Examples where limitations of the iCurb algorithm can be seen. Blue lines indicate the ground truth, and red lines indicate the iCurb (+Slope Loss) predictions.

7 Conclusion and Outlook

In this thesis, an approach for detecting and generating graphs of curbstones from ortho-images using the iCurb pipeline was adopted. Leveraging a novel dataset provided by LDBV, comprising ortho-images of Munich, Germany, the model is trained and evaluated, showcasing significant advancements in curbstone detection and graph generation.

The experimental setup for training involved two main phases: pre-training the segmentation model and training the iCurb pipeline for graph generation. Utilised deep learning techniques such as the UNet architecture for segmentation and imitation learning for graph generation. Through extensive experimentation and evaluation, demonstrated the effectiveness of the approach in accurately identifying road edges from ortho-imagery.

Comparative results with segmentation-based approaches revealed the superiority of the iCurb pipeline, especially when incorporating slope penalty loss during training. The incorporation of the Slope Penalty Loss serves as a valuable addition to the training regimen, contributing to the model's ability to learn and adapt to variations in curbstone slopes. The iCurb algorithm showed substantial improvements in precision and F1-score compared to traditional methods, indicating its potential for real-world applications in urban planning, navigation systems, and autonomous driving.

However, despite the promising results, several limitations and challenges remain. The algorithm struggles in certain scenarios, such as graph growth around overbridges, highly complex intersections, high false positive detections in segmentation results, and handling parallel curbstone instances. Addressing these limitations will be crucial for enhancing the robustness and reliability of the iCurb pipeline.

Looking ahead, there are several avenues for future research and development to further improve the performance and applicability of the iCurb pipeline:

1. **Advanced Segmentation Models**: Exploring more advanced segmentation architectures beyond UNet, such as Dense UNets (Cai et al. (2020)) or transformers (Z. Liu et al. (2021)) or combinations, could lead to better segmentation results, especially in complex urban environments with occlusions, shadows and not always symmetric.

2. **Graph Refinement Techniques**: Developing post-processing techniques to refine the generated graphs, such as graph pruning (Murray, Konidaris, and Sorin (2020)) or smoothing algorithms, could help mitigate errors and improve the overall quality of the outputs.

3. **Integration with LiDAR and Radar Data**: Integrating complementary sensor modalities, such as LiDAR and radar data, could provide additional depth and con-

textual information, enhancing the accuracy and robustness of curbstone detection and graph generation.

4. **Dataset Expansion and Diversity**: Continuously expanding and diversifying the dataset with images from different cities, seasons, and weather conditions would help improve the model's robustness and generalisation capabilities.

By addressing these challenges and exploring these opportunities, the iCurb pipeline has the potential to significantly improve the way road edges are detected and represented from ortho-imagery. This could lead to advancements in urban planning and navigation systems, ultimately contributing to safer and more efficient urban environments.

8 References

[1] Alexandre Attia, Sharone Dayan. Global Overview of Imitation Learning. In: ((2018)).

[2] M. Barzohar, D. B. Cooper. Automatic finding of main roads in aerial images by using geometric-stochastic models and estimation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18.7 ((1996)), pp. 707–721.

[3] Favyen Bastani et al. *RoadTracer: Automatic Extraction of Road Networks from Aerial Images*. (2018). arXiv: 1802.03680 [cs.CV].

[4] Y. Bengio, Yann Lecun. Convolutional Networks for Images, Speech, and Time-Series. In: (Nov. (1997)).

[5] S Cai et al. Dense-UNet: a novel multiphoton in vivo cellular image segmentation model based on a convolutional neural network. In: *Quantitative Imaging in Medicine and Surgery* 10.6 ((2020)), pp. 1275–1285. DOI: 10.21037/qims-19-1090.

[6] Yunji Chen et al. "Chapter 2 - Fundamentals of neural networks". In: *AI Computing Systems*. Ed. by Yunji Chen et al. Morgan Kaufmann, (2024), pp. 17–51. ISBN: 978-0-323-95399-3. DOI: https://doi.org/10.1016/B978-0-32-395399-3.00008-1. URL: https://www.sciencedirect.com/science/article/pii/B9780323953993000081.

[7] G. Cheng et al. Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. In: *IEEE Transactions on Geoscience and Remote Sensing* 55.6 ((2017)), pp. 3322–3337.

[8] D. H. Douglas, T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 10.2 ((1973)), pp. 112–122.

[9] Kaiming He et al. Deep Residual Learning for Image Recognition. In: *CoRR* abs/1512.03385 ((2015)). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[10] S. Hinz, A. Baumgartner. Automatic extraction of urban road networks from multi-view aerial imagery. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 58.1 ((2003)), pp. 83–98.

[11] N. Homayounfar et al. "DAGMapper: Learning to map by discovering lane topology". In: *Proc. IEEE Int. Conf. Comput. Vis.* (2019), pp. 2911–2920.

[12] Paul V.C. Hough. "Method and Means for Recognizing Complex Patterns". U.S. Patent 3,069,654. Dec. (1962).

[13] J. Hu et al. Road network extraction and intersection detection from aerial images by tracking road footprints. In: *IEEE Transactions on Geoscience and Remote Sensing* 45.12 ((2007)), pp. 4144–4157.

[14] A. Hussein et al. Imitation Learning. In: ((2017)).

[15] Diederik P. Kingma, Jimmy Ba. *Adam: A Method for Stochastic Optimization*. (2017). arXiv: 1412.6980 [cs.LG].

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

[17] Tsung-Yi Lin et al. Feature Pyramid Networks for Object Detection. In: *CoRR* abs/1612.03144 ((2016)). arXiv: 1612.03144. URL: http://arxiv.org/abs/1612.03144.

[18] Ze Liu et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In: *CoRR* abs/2103.14030 ((2021)). arXiv: 2103.14030. URL: https://arxiv.org/abs/2103.14030.

[19] G. Mattyus, W. Luo, R. Urtasun. "DeepRoadMapper: Extracting road topology from aerial images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. (2017), pp. 3438–3446.

[20] Volodymyr Mnih, Geoffrey E. Hinton. "Learning to Detect Roads in High-Resolution Aerial Images". In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 210–223. ISBN: 978-3-642-15567-3.

[21] Sean Murray, George D. Konidaris, Daniel J. Sorin. Roadmap Subsampling for Changing Environments. In: ((2020)).

[22] Rafael Padilla et al. A Comparative Analysis of Object Detection Metrics with a Companion Open-Source Toolkit. In: *Electronics* 10.3 ((2021)). ISSN: 2079-9292. DOI: 10.3390/electronics10030279. URL: https://www.mdpi.com/2079-9292/10/3/279.

[23] Pytorch. *BCEWithLogitsLoss*. Accessed on: 11.05.2024. URL: https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html.

[24] Pytorch. *SmoothL1Loss*. (Accessed on: 11.05.2024). URL: https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html.

[25] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[26] Olaf Ronneberger, Philipp Fischer, Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Ed. by Nassir Navab et al. Cham: Springer International Publishing, (2015), pp. 234–241. ISBN: 978-3-319-24574-4.

[27] Stéphane Ross, Geoffrey J Gordon, J Andrew Bagnell. "A reduction of imitation learning and structured prediction to no-regret online learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. (2011), pp. 627–635.

[28] Sebastian Ruder. An overview of gradient descent optimization algorithms. In: *arXiv* ((2016)). Accessed 23 Jan 2018. URL: https://arxiv.org/pdf/1609.04747.pdf.

[29] Y.-Q. Tan et al. "Vecroad: Point-based iterative graph exploration for road graphs extraction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. (2020), pp. 8910–8918.

[30] Jiaojiao Tian et al. "Fusion of Stereo Aerial Images and Official Surveying Data for Mapping Curbstones Using AI". In: *2023 8th International Conference on Signal and Image Processing (ICSIP)*. (2023), pp. 176–180. DOI: 10.1109/ICSIP57908.2023.10271056.

[31] Fereshteh Torabi, Garrett Warnell, Peter Stone. Behavioral Cloning from Observation. In: *arXiv preprint arXiv:1805.01954* ((2018)).

[32] J. D. Wegner, J. A. Montoya-Zegarra, K. Schindler. Road networks as collections of minimum cost paths. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 108 ((2015)), pp. 128–137.

[33] Zhenhua Xu, Yuxuan Liu, et al. RNGDet: Road Network Graph Detection by Transformer in Aerial Images. In: (Feb. (2022)).

[34] Zhenhua Xu, Yuxiang Sun, Ming Liu. iCurb: Imitation Learning-Based Detection of Road Curbs Using Aerial Images for Autonomous Driving. In: *IEEE Robotics and Automation Letters* 6.2 (Apr. (2021)), pp. 1097–1104. ISSN: 2377-3774. DOI: 10.1109/lra.2021.3056344. URL: http://dx.doi.org/10.1109/LRA.2021.3056344.

[35] Riki Yamashita et al. Convolutional neural networks: an overview and application in radiology. In: *Insights Imaging* 9 ((2018)), pp. 611–629. DOI: 10.1007/s13244-018-0639-9. URL: https://doi.org/10.1007/s13244-018-0639-9.

[36] T. Zhang, C. Y. Suen. A fast parallel algorithm for thinning digital patterns. In: *Communications of the ACM* 27.3 ((1984)), pp. 236–239.

[37] T. Y. Zhang, C. Y. Suen. A fast parallel algorithm for thinning digital patterns. In: *Communications of the ACM* 27.3 (Mar. (1984)).