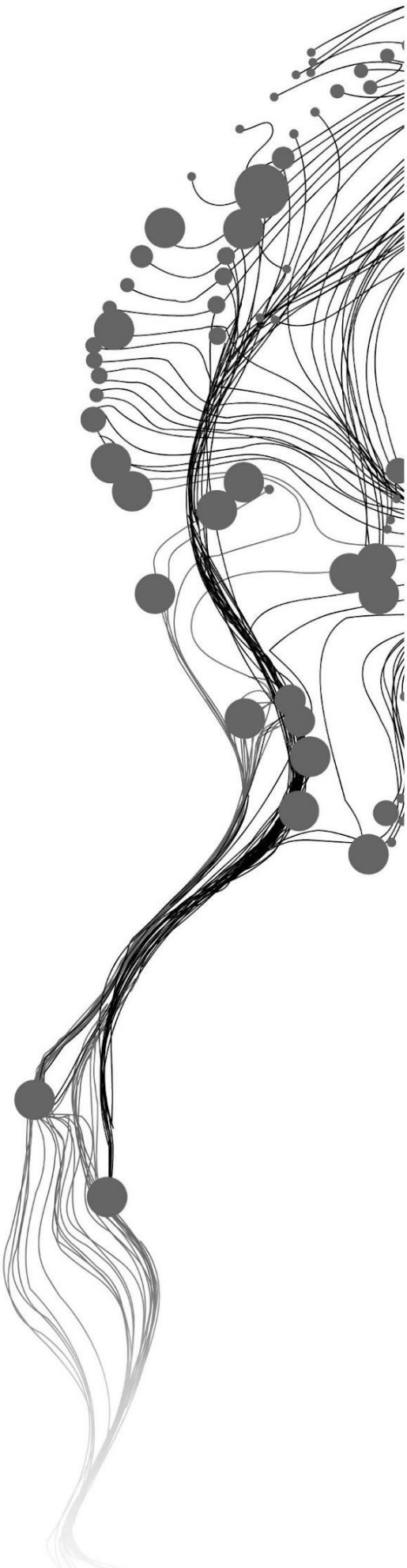
Deep learning on MLS Point Clouds: Feasibility of Improving Semantic Segmentation Result Using Part Segmentation

SARA YOUSEFIMASHHOOR
JULY, 2022

SUPERVISORS:
Dr. Ville Lehtola
Dr. ir. Sander Oude Elberink

ADVISORS:
Dr. Shayan Nikoohemat
Dr. Daan Bloembergen



Deep learning on MLS Point Clouds: Feasibility of Improving Semantic Segmentation Result Using Part Segmentation

SARA YOUSEFIMASHHOOR
Enschede, The Netherlands, July, 2022

Thesis submitted to the Faculty of Geo-Information Science and Earth Observation of the University of Twente in partial fulfillment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.

Specialization: Geoinformatics

SUPERVISORS:
Dr. Ville Lehtola
Dr. ir. Sander Oude Elberink

ADVISORS:
Dr. Shayan Nikoohemat
Dr. Daan Bloembergen

THESIS ASSESSMENT BOARD CHAIR:
Prof. Dr. Ir. George Vosselman

EXTERNAL ASSESSOR:
Dr. Florent Poux (University of Liege)

DISCLAIMER

This document describes work undertaken as part of a programme of study at the Faculty of Geo-Information Science and Earth Observation of the University of Twente. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the Faculty.

ABSTRACT

Smart cities are interested in using the latest sensing and processing technologies to increase their operational efficiency. Two cutting-edge technologies, mobile laser scanning (MLS) and deep learning (DL) can be utilized to serve this goal. On the one hand, as a rapid data acquisition method, mobile laser scanning (MLS) technology provides highly accurate geometry data in the form of dense point clouds. On the other hand, 3D deep learning (DL) algorithms operating on point clouds can provide the 3D understanding and answer many questions in academia and industry. Therefore, in this study, we work at the intersection of MLS and DL to investigate the quality of urban asset inventory.

The deep learning models are typically designed to work with an equally-distributed dataset, a characteristic that is naturally absent in an urban scene. Therefore, in most DL methods, underrepresented classes like urban furniture (such as poles, signs, benches, trashcans, etc.), with less than 1% share in the training and inference dataset, are considered noise and not segmented properly. To address this issue, a two-step pipeline is proposed and tested. At first, semantic segmentation provides a coarse prediction of the shape and location of the asset of interest. Then a more refined segmentation is achieved with part segmentation on the local neighborhood of the detected asset. The idea is inspired by a previous study showing that part segmentation can properly decompose isolated pole-like objects into their constituent parts(Yousefimashhoor, 2022). In this research, we are investigating whether a trained part segmentation model can be used to improve the semantic segmentation result in a local neighborhood.

To study the feasibility of the proposed pipeline, a pilot class of interest (pole), a state-of-the-art deep learning algorithm on point clouds (KPConv), and publicly available data for test and training (the NPM3D dataset) are chosen. The KPConv deep learning model is trained for two tasks, one for the scene semantic segmentation and the other one for part segmentation on poles' local neighborhoods. The trained models are used to infer labels from raw point clouds, and their output is compared to each other within the same spatial extents, once on tile level and once after inserting the tiles back to the original prediction point cloud, on the scene level. Also, some method adaptations are investigated to improve the results, such as enlarging the tiles or including intensity values. Finally, the two models are integrated into one pipeline to simulate the real-world scenario where the input of the part segmentation task is the output of the semantic segmentation result.

The results show that the on perfect tiles, the best adaptation of our proposed method, can achieve a maximum of 6.4% improvement in the IoU of pole class at the tile level and improve the result of semantic segmentation of poles by 1.6% in the whole scene. The qualitative inspection of the results reveals that the part segmentation model performance for each tile can be different and is dependent mainly on pole topological relation with other classes (isolated, intersecting, or attaching) and the point density of the tile. The simulation of a real-world scenario on our proposed pipeline shows that the neighborhood tile selection plays a crucial role in achieving the best result. With the imperfect tiles extracted based on semantic segmentation results, a minor improvement in recall (around 1%) and a significant decline in precision(around 5%) is achieved, which stems from the bias in training set toward the object of interest.The limitations of the proposed method, namely, the destructive role of semantic segmentation errors in escalating false predictions in the part segmentation, the limited contextual information due to the tile selection, and the model bias towards the pole class leading to a huge false positive rate confirm that the problem that we have investigated is not easy to solve. There is no clear-cut solution that could work in all scenarios. In this light, our work helps to clarify the specific approaches that can be viable and the pitfalls that should be avoided.

Keywords: Deep Learning, point cloud, urban asset management, pole-like objects, mobile laser scanning (MLS), semantics segmentation, part segmentation, class imbalance

ACKNOWLEDGMENTS

I would like to take the chance to express my sincere gratitude to those who assisted and accompanied me on this journey. First and foremost, my supervisors, Dr. Ville Lehtola and Dr. Sander Oude Elbernik, encouraged and supported me to pursue my dreams and reach beyond my expectations. Their insights, guidance, and patience were invaluable to my research process.

My research would have been impossible without the aid and support of my dedicated advisors. My special thanks to Dr. Daan Bloembergen and Dr. Shayan Nikoohemat, who helped me take my first steps in the 3D deep learning domain. I cannot put into words how grateful I am for their active involvement, availability, practical remarks, and support.

I am also thankful to the City of Amsterdam. It has been an honor to be part of the young and vibrant DataLab team, working with my bright friends and colleagues and learning from them.

And last but not least, I am grateful to my dearest partner, Mojtaba Mirakhorlo, whose warm heart, sparkling eyes, and cheerful smile were my drive throughout the way.

I dedicate this work to my beautiful and inclusive Amsterdam,
the place that I gladly call my second home.

TABLE OF CONTENTS

Abstract ..... **ii**

1. Introduction **1**

1.1. Background 1

1.2. Problem Statement 3

1.3. Research objectives and questions 4

1.4. Thesis Structure 4

2. Literature Review **5**

2.1. Semantic Segmentation 5

2.2. Part Segmentation 5

2.3. Deep learning algorithms on point clouds 6

2.3.1. Indirect Methods 6

2.3.2. Direct Methods 6

2.4. Kernel Point Convolution 8

2.5. Battling class imbalance 10

2.5.1. Data-level methods 10

2.5.2. Algorithm-level methods 10

2.5.3. Hybrid methods 10

3. Data **11**

3.1. The MLS dataset: NPM3D 11

4. Methodology **12**

4.1. The Algorithm: KPConv 14

4.2. Semantic Segmentation 15

4.3. Part Segmentation 16

4.4. Integrating the two steps 19

5. Results and Discussion **21**

5.1. Scene Semantic Segmentation on the NPM3D dataset 21

5.1.1. Pole Predictions in the scene 23

5.2. Part segmentation of pole tiles 25

5.3. Enlarging the tile size 30

5.4. Including an additional input feature 31

5.5. Integrating the two-step pipeline 34
6. Conclusion and Future Work 37
6.1. Contributions 37
6.2. Limitations 38
6.3. Answers to the research questions 39
3.3. Future Work 40
Appendix (1) : Confusion Matrices 44
Appendix (2) : IoU Comparison 49

LIST OF FIGURES

Figure 1- 3D scene analysis workflow (Weinmann et al, 2015) 2
Figure 2 - Part segmentation on isolated pole-like objects using KPConv algorithm 3
Figure 3 - Semantic Segmentation of a point cloud (adopted from NPM3D dataset) 5
Figure 4 - Part Segmentation of isolated objects (adopted from ShapeNetPart dataset) 5
Figure 5 - Indirect Deep Learning Methods on Point Clouds (Bello et al. 2020) 6
Figure 6 - PointNet ++ feature learning architecture (Qi et al., 2017b) 7
Figure 7 - Point convolution using kernels (Hua et al., 2018 and Guo et al., 2020) 7
Figure 8 - Extracting local correlation in PointCNN (Li et al., 2018) 8
Figure 9 - EdgeConv operation in DGCNN architecture (Wang et al., 2019) 8
Figure 10 - Continuous Convolution (Guo et al., 2020) 9
Figure 11 - Share of Classes in NPM3D Dataset 11
Figure 12 - An overall view to the methodology 12
Figure 13 - Intuitive understanding of the methodology 13
Figure 14 - KPConv Architecture 14
Figure 15 - Part Segmentation Data Processing Workflow 16
Figure 16 - Neighborhood extraction workflow 17
Figure 17 - Pole bounding box and tile extraction 18
Figure 18 - Proposed Pipeline for integrating Semantic Segmentation and Part Segmentation 20
Figure 19 - Problematic labeling choices in NPM3D dataset 22
Figure 20 - False Positive cases in segmenting poles from the whole scene 24
Figure 21 - Falsely False positive scenarios in pole segmentation in the scene semantic segmentation 24
Figure 22 - False negative cases in semantic segmentation for pole class 24
Figure 23 - Entirely missed poles in the semantic segmentation 25
Figure 24 - point density affecting semantic segmentation results 25
Figure 25 - Tiles and their difficulty levels 27
Figure 26 - Variety of possible hard scenarios (extracted from the ground truth) 28
Figure 27 - Edge problematic in tile prediction 28
Figure 28 - lack of contextual information in the tile edges 29
Figure 29 - Evaluation metrics change by enlarging the neighborhood tiles 30
Figure 30 - Difference of confusion matrices, focused on the class pole for tiles of different sizes 31
Figure 31- Effect of enlarging the tile size on a pole instance prediction 31
Figure 32 - Comparing the part segmentation results with and without itensity values 33
Figure 33- Comparing the part segmentation results with and without itensity values 33
Figure 34 - Comparison of Evaluation Metrics between the output of models trained with and without intensity value 33
Figure 35 - Pole prediction scenario in semantic segmentation 34
Figure 36 - Nested Bounding Boxes Problematic 35
Figure 37 - Incapability of the part segmentation to correct false positives 37

LIST OF TABLES

Table 1- Point and label count per point cloud in NPM3D dataset 11
Table 2 - Pole like objects type and distribution in NPM3D dataset 11
Table 3 - KPConv rigid and deformable performance on NPM3D dataset (adopted from Thomas et al., 2019) 15
Table 4 - Tabel 3 - Comparing the published and achieved results for the semantic segmentation task on NPM3D using KPConv algorithm 21
Table 5 - Share of classes in the NPM3D dataset 21
Table 6 - Confusion Matrix of Semantic Segmentation task on NPM3D dataset 23
Table 7 - Difference of Confusion Matrices : Part segmentation of 0.5m offset tiles 26
Table 8 - Evaluation Metrics for all the class in tiles : comparing semantic segmentation and part segmentation 26
Table 9 - Comparing the results of part segmentation with and without padding 29
Table 10 - Difference of Confusion Matrices : The effect of including intensity values 32
Table 11 - Evaluation metrics for the pole class after integration of two steps 36
Table 12 - Difference of Confusion Matrices : The effect of integrating two steps 36

1. INTRODUCTION

1.1. Background

Cities want to become smarter by using technological innovations, such as extracting useful information about the urban environment to help them in the decision-making process. One of the main concerns of contemporary urban management is the maintenance of the existing urban assets. An inventory describing the location, geometric information, condition, and type of the existing assets is crucial for the effective maintenance of the assets(Landa & Prochazka, 2014; Sairam et al., 2016). Formerly, the detection and evaluation of urban assets were done manually by specialist operators. However, due to the labor-intensive and time-consuming procedure of the field survey and the subjectivity of the results, automated methods have become more popular (Landa & Prochazka, 2014; Y. Li et al., 2020; Peraka & Biligiri, 2020; Sairam et al., 2016).

Automation has entered the asset detection process in both data acquisition and data processing phases. With the advancement of sensing technology, 2D and 3D mobile mapping systems have become more available and affordable over time(Guo et al., 2020). For automation of urban asset inventory, image analysis methods and machine learning algorithms have been utilized widely (Ma et al., 2018) because of the low computational effort required to work with structured 2D datasets. However, using RGB images has some limitations. The performance of the trained models is easily affected by the shadows or insufficient contrast (X. Chen et al., 2021). Moreover, inferring 3D coordinates from 2D images is prone to have more error than doing so from a LiDAR acquired point cloud.

Companies and governmental organizations are becoming increasingly interested in using mobile laser scanning (MLS) 3D point clouds for urban asset monitoring and autonomous driving applications instead or in combination with RGB images. The first competency of point clouds over other data types is its measurement quality. MLS point clouds contain more accurate shape, scale, and geometry information, higher point density, and large spatial coverage. Operational considerations such as independence from environmental and lighting conditions, acquisition safety, low acquisition time, cost-efficiency, and less traffic disruption are other plus points of using point cloud for urban asset monitoring applications (Guo et al., 2020; Karlsson, 2014; H. T. Li et al., 2021).

However, processing a 3D point cloud of road networks has its own challenges. The irregularity of the dataset, the high dimensionality (Guo et al., 2020), the permutation invariance, and the distance-dependent sparsity (He et al., 2020; Milioto et al., 2020) in the scene have prevented the traditional algorithms to be able to provide a semantic understanding of the urban scene. Traditional approaches[1] could only segment the point clouds without assigning any meaningful labels to the cluster (Grilli et al., 2017; Xie et al., 2020). The traditional approaches used handcrafted features (like surface normals, curvature, surface orientation, color information, point reflectance, etc,.) or prior knowledge to segment the point cloud (Grilli et al., 2017; Zhang et al., 2019) and usually required parameter fine-tuning which was not straightforward (Zhang et al., 2019).
Later, the early machine learning approaches successfully integrated segmentation and classification of points in a single pipeline by following the workflow illustrated in figure 1. In the first step, with the motivation of extracting

[1] Traditional approaches such as region growing, model fitting, edge-based segmentation, or machine learning approaches like k-mean clustering, mean shift and hierarchical clustering (Zhang et al., 2019)

3D structure, a neighborhood is defined around each point, and then mathematical solutions are used to extract features to characterize each neighborhood. Handcrafted geometric (like local dimensionality and height values) or radiometric features associated with neighborhoods were extracted in this step. In Some cases, the 2D projection of a neighborhood on a vertical and horizontal plane was also investigated to generate useful information(Weinmann et al., 2013). In the third step, The main concern was to choose the most informative feature for separating classes. Some features would be selected based on the properties of the class of interest, while some would be filtered because of their correlation with other features. Sometimes an algorithm (like Random Forest) was able to distinguish the features with high information gain and prioritize them, but in other cases, the best features were selected by trial and error (Weinmann et al., 2015). Finally, a classifier like SVM would be used to generate interpretable results per point either by considering only the feature vector of the particular point or by also considering the feature vector of its neighboring points due to their spatial correlation(Weinmann et al., 2015). The described workflow still requires handcrafting features and involves high human intervention and could not deal properly with the huge amount of 3D data captured from an outdoor scene (Peraka & Biligiri, 2020).

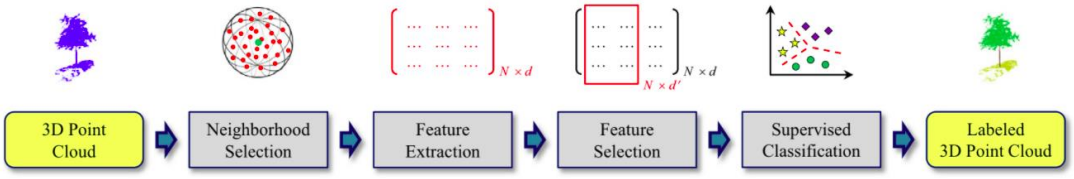


Figure 1- 3D scene analysis workflow (Weinmann et al, 2015)

Lately, The advancement in the computer vision and artificial intelligence domain and the ability of deep learning models to handle a huge amount of data and extract high-level features to distinguish classes have led deep learning methods to gain significant attention among scholars for the 3D scene understanding tasks(Liu et al., 2019; Shinde & Shah, 2018; Zhang et al., 2019). This research also uses deep learning techniques for the semantic segmentation task in both scene and cropped tile level.

Although deep learning (DL) methods show promising results, improving the performance of these models is still an open topic in this domain. One of the main classic yet underresearched (Chen et al., 2021) challenges even for the state-of-the-art algorithms trained with high-quality training data is the low performance of the DL models on the minority classes[2]. Due to the high frequency, prediction error for the majority classes[3] dominates the weights updating process (Bellinger et al., 2020). As a typical DL model assumes that the dataset has a balanced (or moderately imbalance) distribution over all the classes and since it is designed to achieve the maximum overall accuracy, the minority classes will be considered noise and ignored to benefit the majority class predictions(Y. Chen et al., 2021; Dong et al., 2019). This is precisely the problem we study in this thesis. Our main focus will be on improving the prediction performance of these minority classes in semantic segmentation tasks.

[2] By 'minority class' we mean those classes that are rare and underrepresented in the dataset. The share of representative points belonging to the minority class in the whole dataset is highly insignificant.
[3] By 'majority class' we mean the classes that is frequent and overrepresented in the dataset. The share of representing points belonging to the majority class has a huge gap with the rest of the classes.

1.2. Problem Statement

The translation of this problem in the context of urban asset management is as follows. Urban management bodies and all the parties involved in public space maintenance are interested in an updated inventory of the location, geometry, and condition of urban assets. Deep learning methods for semantics segmentation on the updated mobile laser scanning data can provide a fast baseline prediction. However, many categories of urban assets (such as poles, trashcans, benches, etc.) are minority classes in an urban outdoor scene, where most of the points belong to the building and ground class (Bloembergen & Eijgenstein, 2021). Therefore, the deep learning algorithms usually fail to segment these objects correctly in the point cloud. The urban assets' geometry is not always fully captured. Instead, they are being confused partly or entirely with other classes.

Since an asset inventory requires accurate details about the shape of the asset, this research aims to study the possibility of achieving a better semantic segmentation prediction for the urban assets using an additional part segmentation step. The main idea is to use the initial semantic segmentation predictions to spot the locations of the minority class of interest, crop a neighborhood around those spots, and test whether a trained part segmentation algorithm can assign more accurate labels to the points inside the crop. This experiment is motivated by previous work (Yousefimashhoor, 2022) on high-quality and manually labeled pole data belonging to the Gemeente Amsterdam, that shows the (KPConv) deep learning algorithm is capable of decomposing the pole-like objects to their constituent part(figure 2) with a mIoU of 84%. A similar idea is tested on an imperfect real-world data in this research, and it is assumed that since the class distribution in the crops is more balanced (or moderately imbalance in favor of our class of interest) compared to the whole scene, it will lead to less confusion between the minority and majority classes and improves the overall semantic segmentation results.

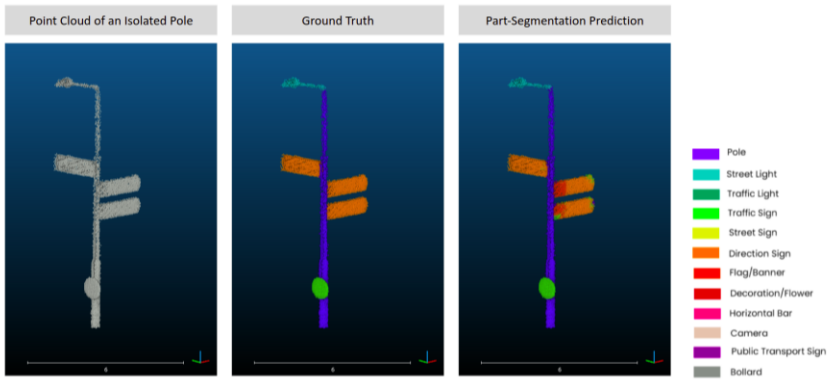


Figure 2 - Part segmentation on isolated pole-like objects using KPConv algorithm

Among all the minority classes, this research is focused on pole-like objects for four reasons.

1) They can be locally spotted because they are not extended objects with large neighborhoods, and it is assumed that the part segmentation task can handle its extent.
2) They have a variety of possible shapes that might be confusing for the model, and it is interesting to investigate how the proposed pipeline performs in each instance.
3) Some poles have intersections with the class natural (e.g., trees). Furthermore, there is a class similarity between the poles and some building parts or the tree trunk. It is interesting to investigate whether the proposed pipeline can deal with such hard cases.
4) Pole-like objects are important urban assets as they carry multiple attachments and are crucial to traffic flow guidance and safety matters.

1.3. Research objectives and questions

The main objective of this research is to test whether it is feasible to improve the semantic segmentation results using the part segmentation task in the neighborhood around the pole-like objects in imperfect data. The research aims to design a pipeline to optimize the part segmentation task to achieve the best result for the pole class by doing the part segmentation. The sub-objectives of this research are as follows:

1. **Sub-objective (1):** To select and train a semantic segmentation model and investigate the results on the test MLS point cloud using a deep learning algorithm
   1.1. What are the challenges of the prediction in the scene semantic segmentation step? And why are they happening?
   1.2. How good is the prediction for the class pole? What challenges should be overcome in the part segmentation step?

2. **Sub-objective (2):** To train the part segmentation model with cropped tiles of the scene and investigate its performance.
   2.1. How does the part segmentation step influence the initial result? What parameters can affect the effectiveness of the part segmentation step?
   2.2. What are the limitations of the part segmentation on tiles?
   2.3. Is there any preprocessing or postprocessing step that could improve the outcome?

3. **Sub-objective (3):** To integrate the semantic segmentation and part segmentation into one pipeline
   3.1. What other complexities emerge when the two methods are integrated together?
   3.2. How will the results get affected by this?

Novelty

It is of great interest to investigate whether the acquired MLS point clouds could directly be used for a combination of scene segmentation and part segmentation. The novelty of this research lies in combining the semantic segmentation and part segmentation results of selected crops from the scene to improve the final segmentation result. Several initiatives (such as different sampling methods, data augmentation, or modifying loss calculation) have been tested to reduce the class imbalance effect on the final prediction and enhance the results for the minority classes. However, to the best of our knowledge, no one has tested tackling this issue by cropping the scene around the object of interest.

1.4. Thesis Structure

The remainder of this thesis is structured in five chapters. In the next chapter, the related literature is being reviewed. In the third chapter is the description of the NPM3D data. Chapter (4) is about the research methodology. It explains the motivation and method related to each experiment. The results are reported in the fifth chapter, and the analytical discussions around them are presented. In the final chapter, the conclusion, including the answer to the research questions, contributions and limitations of the research, and future work, are mentioned in summary.

2. LITERATURE REVIEW

2.1. Semantic Segmentation

Scene Semantic segmentation or scene labeling is one of the crucial steps in 3D scene understanding (Yu et al., 2015; Zhang et al., 2019). This process aims to group points based on a shared geometrical or radiometric characteristic, classify each group of points and assign meaningful labels to the points belonging to each group (Grilli et al., 2017; Zhang et al., 2019). Similar to what is illustrated in figure 3, the final product of the semantic segmentation task is a colored point cloud where a color is assigned to each class to distinguish objects in the scene.

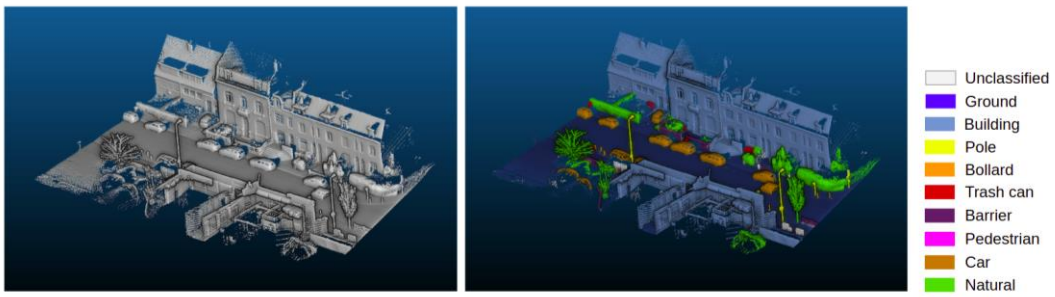


Figure 3 - Semantic Segmentation of a point cloud (adopted from NPM3D dataset)

2.2. Part Segmentation

Part segmentation also involves assigning semantics to each point, but the task is done on the object level. The ultimate goal is to decompose the object of interest into its constituent parts(Boulch, 2020). The main challenge lies in the variety of geometric shapes describing a single part and the unknown number of a particular part in an object instance (Guo et al., 2020). As shown in the figure below, the 'chair' class encompasses a variety of instances, each of which has a different number of parts. Some parts are absent in some tiles. For example, a chair might have or not have arms. Also, a single part, like the back of the seat, is in various shapes.

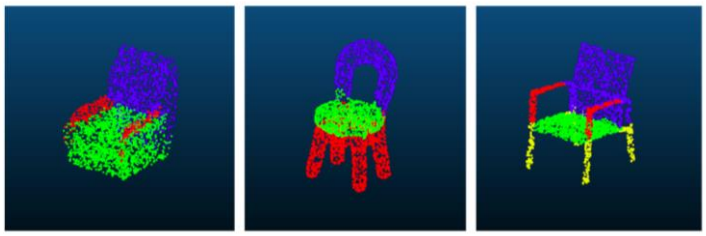


Figure 4 - Part Segmentation of isolated objects (adopted from ShapeNetPart dataset)

Deep learning (DL) algorithms have recently been widely used to perform the tasks above. Since both tasks are concerned with segmentation and point classification, a single DL algorithm can be used with minor modifications to handle both problems.

2.3. Deep learning algorithms on point clouds

Deep learning methods on point clouds have been categorized mainly based on the input format of the data ingested into the model(Guo et al., 2020; Xie et al., 2020), and can be generally divided into indirect and direct methods(Zhang et al., 2019). In the indirect methods, the irregular structure of the point cloud is transformed into a structured data format, while in the direct methods, the model can accept the points as the input and process them directly.

2.3.1. Indirect Methods

The unstructured data of point clouds, the high dimensionality (Guo et al., 2020), and the point density variance(He et al., 2020; Milioto et al., 2020) in the scene have led the scholars to begin with intermediate representations of the 3D dataset. The first attempts to overcome these challenges include running deep learning algorithms on projections of 3d datasets on 2D images (Multi-view based or projection-based methods) or voxelizing the data in a 3D grid (Voxel-based methods or Discretization-Based)(Guo et al., 2020; Qi et al., 2016). These methods aim to reduce irregularities of point clouds and make it easier to define a convolution operation.
These methods extract the aggregated features from the spatial neighborhood(Lu & Shi, 2020) The prediction results are being back-projected to the original point cloud in the final step. The early approaches' disadvantage is the inevitable elimination of information loss and details(Lazarek & Pryczek, 2018). Particularly, the projection-based method cannot deal with occlusion and point density variation in semantic segmentation tasks (Thomas et al., 2019). Also, the computational cost in voxelization methods grows exponentially by increasing the resolution (Guo et al., 2020).

Figure 5 - Indirect Deep Learning Methods on Point Clouds (Bello et al. 2020)

2.3.2. Direct Methods

The more recent approaches consume the points directly. The point-based methods can be categorized as follows:

1) **Multilayer Perceptron (MLP)-based / PointNet-based:** The early network proposed to label individual points (Qi et al., 2016)uses a shared MLP to label 3D data. The spatial encoding of individual points is given to the network as an input. The network will aggregate the input to create a global point cloud signature (max pooling) and predict based on it (Qi et al., 2017; Thomas et al., 2019). This method could deal with the inherent unstructured format of the raw point cloud (Chen et al., 2021). However, the points are labeled without considering their local geometry, spatial distribution, or interaction with their neighboring points, which leads to a noisy prediction (Guo et al., 2020).
   The following approaches are proposed to capture and include the spatial structure of neighboring points or homogeneous regions in the prediction pipeline(Chen et al., 2021).

2) **Approaches using Local Structure Modelling:** The initial step to capturing the local structure is to define the local neighborhood. The common pipeline for extracting neighborhoods starts from sampling. A number of points are chosen using random point sampling (RPS), farthest point sampling (FPS), or uniform sampling methods in the cloud(Bello et al., 2020). Then, in the grouping phase, methods such as

K-nearest neighboring points, ball query[4] (Qi et al., 2017), and K-d tree[5] (Klokov & Lempitsky, 2017)can be adopted to define neighborhood members. After generating the neighborhoods, there are two strategies to follow to map each group of points to a representative vector for their local structure:

a) **Neighborhood Feature Pooling**: In this strategy a local feature is extracted by summarizing the features of points in the local region using a max-pooling function while the correlation between the neighboring points is ignored (Bello et al., 2020). For instance, in PointNet++, overlapping local neighborhoods are defined, and the local features are extracted from each neighborhood. Then the neighborhoods are grouped with their adjacent neighborhoods if they have similar features(Qi et al., 2017). The generated regions are the input of the higher layers. The region's vectors are then fed to MLP as input, and the predictions are generated for each receptive field (region) and further interpolated to achieve per-point predictions (figure 6).

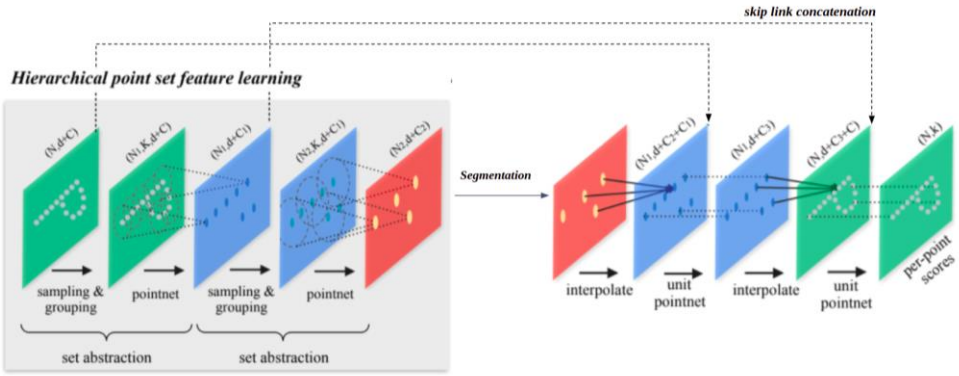


Figure 6 - PointNet ++ feature learning architecture (Qi et al., 2017b)

Also, in some convolutional networks(Hua et al., 2018), each point is convolved using a kernel without any sampling. The kernel is placed on each point and points located inside the kernel are considered neighboring points which are influencing the convolution result in the given location(figure 7).

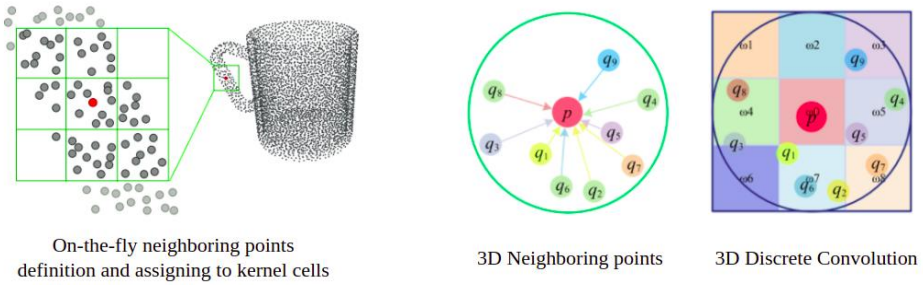


Figure 7 - Point convolution using kernels (Hua et al., 2018 and Guo et al., 2020)

b) **Extracting local correlation:** Exploring the relation between the points in a receptive field improves the discriminative capability of the model (Bello et al, 2020). To capture the spatial

[4] points located in a spherical neighborhood (within the distance x from the chosen points) are selected..
[5] Fracturing space into a binary tree structure using median (hyperplane) of the coordinate values (along the axis with the highest range) and repeat the same procedure for the generated splits (half-spaces).

layout/distribution of the local neighborhood, multiple strategies are used. For example, In PointCNN (Y. Li et al., 2018), for each neighborhood, the neighborhood coordinates are redefined in a local coordinate system using X-transformation where the origin is set to the representative point. As it is illustrated in figure 8, The relative coordinates of the points are included in the region's feature vector using an MLP(Y. Li et al., 2018).

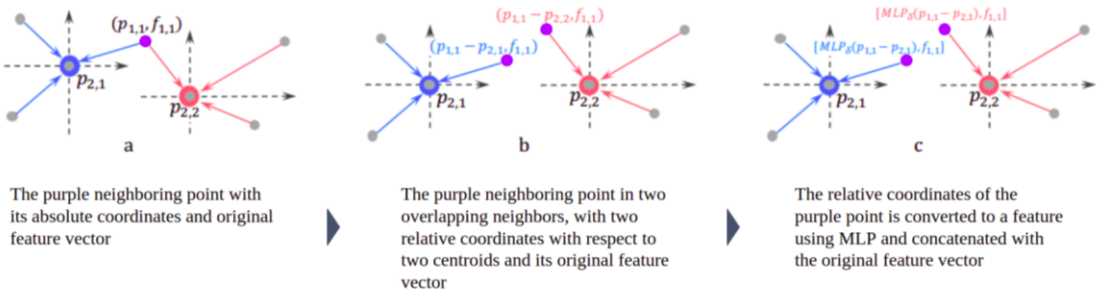


Figure 8 - Extracting local correlation in PointCNN (Li et al., 2018)

The graph-based methods are developed to capture the local geometric structure(Guo et al., 2020). DGCNN[6] (Wang et al., 2019) produces a graph structure in any given neighborhood. The EdgeConv layer generates edge features and describes the relation between the centroid and its neighboring points. The graph is updated in each layer. Since in this algorithm, the focus is on applying a filter to the edges instead of points, the algorithm is able to combine features on a local surface patch while it is unable to capture the deformation of patches(Thomas et al., 2019).

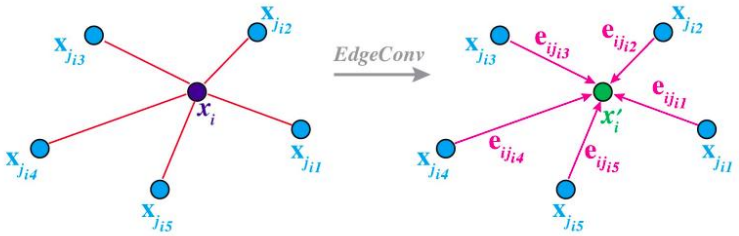


Figure 9 - EdgeConv operation in DGCNN architecture (Wang et al., 2019)

2.4. Kernel Point Convolution

This subsection will elaborate on the main concepts behind the algorithm's design. Kernel-point convolution, or KPConv(Thomas et al., 2019), is a deep learning algorithm operating on 3D point clouds. The algorithm's design is motivated by the concept of having a regular grid for convolution in 2D image processing. The kernel in this method is a spherical neighborhood with a predefined radius containing a specified number of points on its surface. The arrangement of points on the sphere surface is regular and predefined in the rigid form of KPConv, while in the deformable version of KPConv, the arrangement is learned to fit the local geometry of scene objects.

Each kernel point has a learned weight and an area of influence based on Euclidean space and expressed by a correlation function (Thomas et al., 2019). The main assumption is the point cloud's spatial localization property,

[6] Dynamic Graph Convolutional Neural Network

which allows a spatial kernel to influence a local neighborhood of each point and abstract the neighborhood features.

To create input for the next layer of the network, The kernel (sphere) center is located on each point in the point cloud. The surrounding points that fell into this neighborhood are considered neighboring points(figure 10a). For each of the neighboring points, a kernel function would be calculated considering its distance from all the kernel points and the learned weights of the kernel points (figure 10b). Each neighboring point influences more from the closest kernel points compared to the farther ones. The calculated kernel function for each neighbor will multiply by its feature matrix and generate a new feature with a lower dimension to describe the center point and its neighborhood as the input to the next layer in the network.

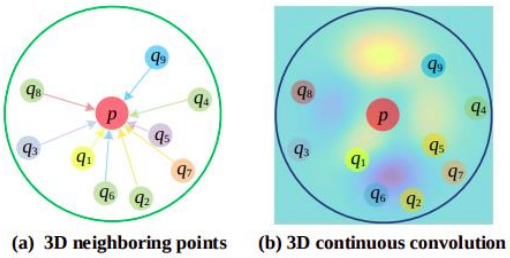


Figure 10 - Continuous Convolution (Guo et al., 2020)

A more detailed mathematical explanation of the algorithm is given below. Equation (1) explains the convolution process. For each point in the neighborhood (points located inside the sphere), a relative position to the central point is calculated and given to the kernel function (g). The feature of each point is multiplied by the result of the kernel function and summed up.

$$(\mathcal{F} * g)(x) = \sum_{x_i \in \mathcal{N}_x} g(x_i - x) f_i \qquad \textit{Equation (1)}$$

The kernel function(g) is defined per neighboring point and is calculated using equation (2). The learned weight for each kernel point is multiplied by the correlation function (h) between each kernel point (k) and the neighboring point and summed up. Equation (2) implies that all kernel points influence all neighboring points but with different intensities based on their spatial correlation.

$$g(y_i) = \sum_{k<K} h\left(y_i, \widetilde{x}_k\right) W_k \qquad \textit{Equation (2)}$$

The correlation function (h) is a function that is calculated per neighboring point (y_i) and per kernel point (x_k) using their relative position in Euclidean space as follows:

$$h\left(y_i, \widetilde{x}_k\right) = \max\left(0, 1 - \frac{\|y_i - \widetilde{x}_k\|}{\sigma}\right) \qquad \textit{Equation (3)}$$

Where σ hyperparameter is the influence distance and is chosen considering the point density (Thomas et al., 2019). The h function creates numbers close to one if the kernel point and the neighboring point are close to one another and generates values close to 0 if they are far away.

The KPConv algorithm can be used for semantic segmentation, classification, and part-segmentation tasks. The semantic segmentation task is used for segmenting and assigning meaningful labels to a whole indoor or outdoor scene. The classification task is used for isolated exclusive object classes that can be fully separated from one another (like a lamp vs. an airplane). The part segmentation task is for isolated individual objects with a known number of parts. The objective of this task is to divide the point cloud of the object into its constituent parts.

2.5. Battling class imbalance

In general, there are three methods to deal with an imbalanced dataset in the deep learning domain to reduce the model bias toward the majority classes and improve the prediction for the minority classes (Krawczyk, 2016) :

2.5.1. Data-level methods

The training data has a crucial role in the predictive performance of any deep learning algorithm. The poor performance of DL algorithms lies strongly in the distribution of classes in the training set. That is why in the data-based methods, the share of classes in the dataset is being modified with a number of initiatives(Johnson & Khoshgoftaar, 2019) The training set can be balanced by subsampling[7] the majority classes or oversampling[8] the minority classes, or both using various sampling methods(Lin & Nguyen, 2020). With increasing the number of records for the minority classes in oversampling strategy, the required computational effort will increase, and consequently, the training speed will be reduced(Johnson & Khoshgoftaar, 2019) Also, they can bring significant improvement for datasets with slight imbalance. Repeating the records in the oversampling strategy can easily lead to overfitting(Griffiths & Boehm, 2019). Undersampling can result in the loss of useful information. Both sampling methods will change the local point density and local structure of the point cloud (Boogaard, 2022).

Augmentation of selected patches mainly consisting of medium and minority classes is also tested as a solution to skewed data distribution (Griffiths & Boehm, 2019; Poliyapram et al., 2019). Furthermore, generating synthetic data is another way of increasing the frequency of the minority classes in the dataset; however, not only generating synthetic data can be time-consuming and labor-intensive, the class imbalance usually will remain in the simulated data (Synthcity: Griffith and Boehms, 2019). Also, in a study (Sander, 2020),class aggregation (like merging similar classes of trucks and cars) can be used for class imbalance correction (Sander, 2020).

2.5.2. Algorithm-level methods

This approach addresses the class imbalance during the training phase (Boogaard et al., 2022). Since the learning happens by computing the error (loss) between the model prediction and the ground truth, the scholars have adjusted the loss calculation methods in a way that the algorithm receives an additional award if it generates a correct prediction for the minority class (Lin & Nguyen, 2020). The weighted cross-entropy and focal loss are two common methods to implement cost-sensitive learning. In the weighted cross-entropy method, a higher weight is assigned to the minority classes in the calculation of the loss function, and the weights for the majority classes are decreased. In the focal loss, the same idea is pursued, but the weights are proportionate to the inverse of the class share in the dataset(Boogaard et al., 2022). The main drawback for the cost-sensitive processes in determining the weights belonging to each class in the loss function. It is costly and not straightforward for large datasets with multiple features and is highly problem-specific (Dong et al., 2019; Johnson & Khoshgoftaar, 2019). Also, the complexity of the loss function can reduce the training speed(Lin & Nguyen, 2020). Incremental transfer learning is another proposed method to prevent the majority classes from dominating the gradient. In this pipeline, classes are being represented to the machine step by step, starting from the least represented class. In each iteration, the next remaining minority class will be added to the scene, and the pipeline finishes by introducing the first majority class to the network.

2.5.3. Hybrid methods

The hybrid methods benefit from the combination of data-level and algorithm-level strategies.

[7] reducing or removing the number of records

[8] repeating or interpolating records

3. DATA

3.1. The MLS dataset: NPM3D

This project uses the Paris-Lille-3D (NPM3D) dataset as a publicly available benchmark dataset. The dataset is captured in two cities of, Paris and Lille, in 2018, with a Velodyne HDL-32e sensor. The total amount of points captured from 1940 meters is 143.1 million. The point cloud is registered and has been labeled point-wise manually. The point density of the point cloud is between 1000 to 2000 points per square meter. We set our baseline on the point cloud labeled with ten classes. The statistics about the classes can be found in table 1.

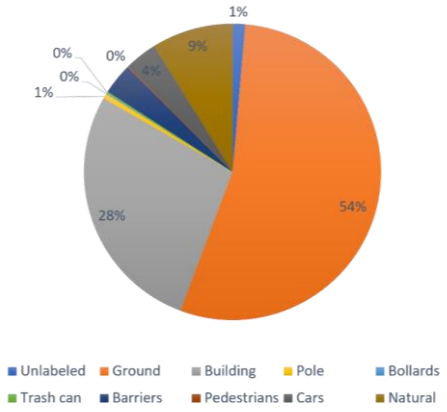


Figure 11 - Share of Classes in NPM3D Dataset

Table 1- Point and label count per point cloud in NPM3D dataset

	Uabeled	Ground	Building	Pole	Bollards	Trash can	Barriers	Pedestrians	Cars	Natural
Lille1_1	199627	17462868	7422179	167122	7735	67074	1210528	16290	1291186	2188821
Lille1_2	270626	17463872	9450594	185901	22355	85451	397685	7328	852111	1279065
Lille2	179841	12042099	7193259	109906	7298	115885	54818	11233	770451	917631
Paris	992204	18106481	8948787	193167	24327	28113	2389916	137281	1298547	6213720
Sum	1642298	65075320	33014819	656096	61715	296523	4052947	172132	4212295	10599237
Percentage	1%	54%	28%	1%	<<1%	<<1%	3%	<<1%	4%	9%

As it is illustrated in figure 11, there is a huge class imbalance between the classes in this dataset. More than half of the points represent the "Ground", and almost 30% of the point represent the "Building." The "Natural" class has the third-highest point count with less than 10% share, and the rest of the classes constitute less than 10% of the whole dataset in total. As it is indicated in the table below, the total number of points representing pole-like objects consists only 0.004% of the whole dataset. The point coordinates (x,y,z) and reflectance (intensity of unit8) are stored for each point in the cloud. In this dataset, pole-like objects are classified as static objects with representation as a point on a map (punctual). Poles with higher than 1-meter height are labeled as posts and lower than 1-meter height as bollards. The list of available pole-like objects can be seen in the table below.

Table 2 - Pole like objects type and distribution in NPM3D dataset

Class	**ID**	**Lille1**		**Lille2**		**Paris**		**Total**	
		#instances	#points	#instances	#points	#instances	#points	#instances	#points
Traffic light	302020500	14	25.82k	11	27.41k	16	80.76k	41	133.10
Traffic Sign	302020600	82	113.6k	39	69.89k	17	30.75k	138	214.3
Sign board	302020700	20	68.34k	12	43.14k	3	13.41k	35	124.9
Bollards	302020300	122	34.80k	64	7.982k	84	28.33k	270	71.10k
Posts	302020200	9	13.51k	0	0	0	0	9	13.51k
Total	-	247	256.1k	126	148.422k	120	153.25	493	556.91k

4. METHODOLOGY

The proposed two-step process (figure 12) begins with labeling the raw point cloud of the whole scene using trained KPConv model for the semantic segmentation task. This step provides information about the approximate location of the poles, where a neighborhood can be defined around the objects of interest. In the next phase, each tile is considered an instance of a similar object that can be decomposed to its constituent parts using the part segmentation. An intermediate step of extracting pole neighborhood is required to limit the input extent to be fed into part segmentation trained model because the part segmentation task is shown to be capable of handling isolated objects. Therefore, the pole points are clustered and cropped from the scene with their adjacent area into tiles. The tiles describing pole neighborhoods are then fed to a trained KPConv, without any provided labeles, for the part segmentation task to generate a secondary prediction per tile. Assessing the quality of the tile predictions is the main concern of this study. As the final step, the tile predictions are inserted back into the scene. It is assumed that replacing the initial prediction from the semantic segmentation with their corresponding tile predictions will improve the final semantic segmentation result. A more intuitive illustration of the pipeline is illustrated in figure 13.

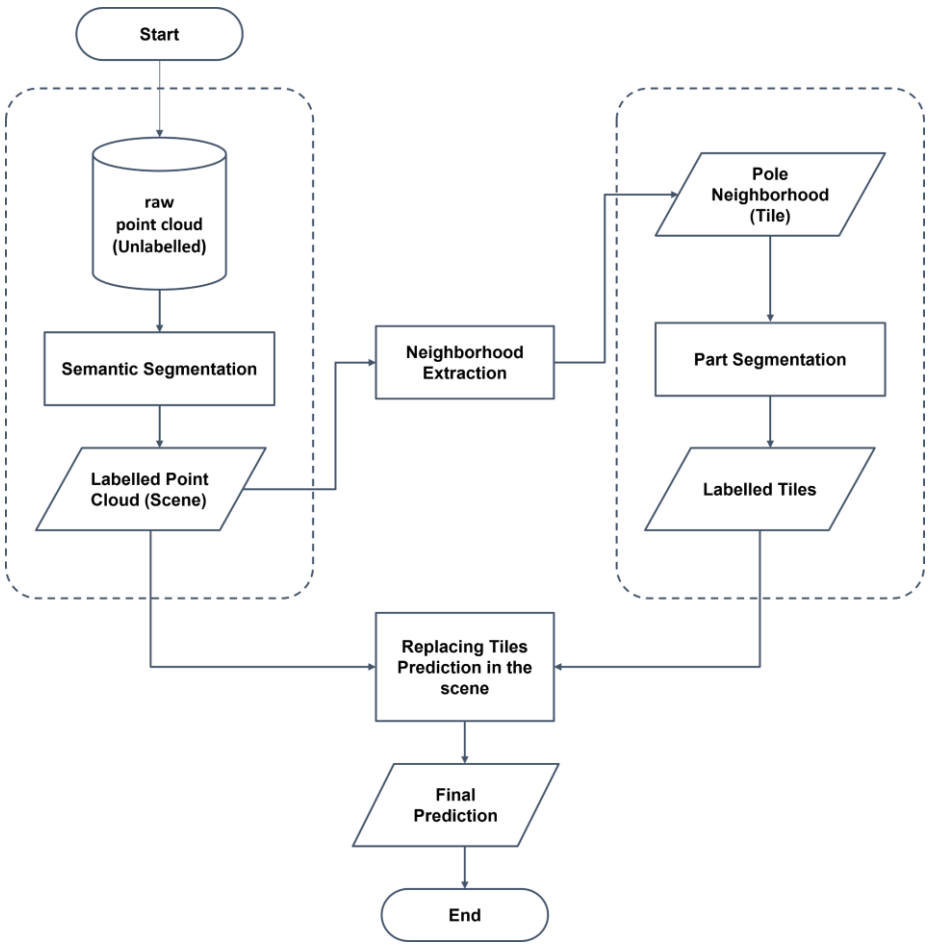


Figure 12 - An overall view to the methodology

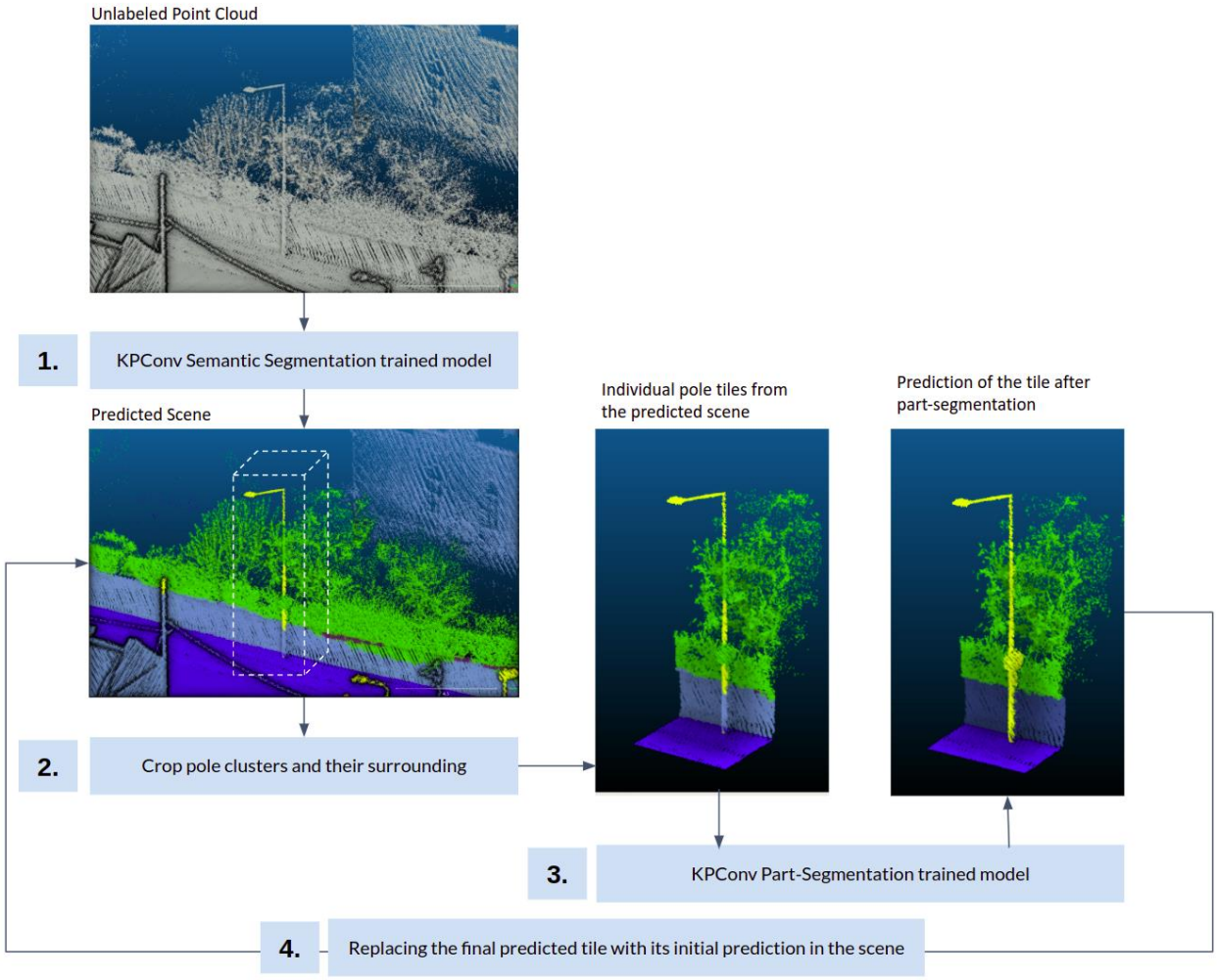


Figure 13 - Intuitive understanding of the methodology

For testing this idea, the KPConv state-of-the-art deep learning algorithm has been chosen to do both segmentation tasks, and the publicly available dataset of the NPM3D captured from urban scenes in France is used as the labeled set. The structure of this chapter matches the main step of the pipeline shown above and are as follows:

1- The Algorithm: KPConv
2- semantic segmentation
4- part segmentation
5- integration of two steps

4.1. The Algorithm: KPConv

The deep learning algorithm used for 3D semantic segmentation and part segmentation in this research is KPConv. The motivations behind choosing the KPConv algorithm are:

1) KPConv is the state-of-the-art semantic segmentation algorithm that has influenced many scholars in this domain. The KPConv paper was published in 2019 and has 566 citations. Many papers afterward propose modifications and optimizations for KPConv as they find this solution promising.
2) KPConv is one of the algorithms on the leaderboard of our chosen benchmark dataset (Paris-Lille-3D). At the beginning of this project, KPConv was ranked 4th on the NPM3D leaderboard. However, it was the first algorithm with publicly available, trusted, and well-supported code in the list.
3) The KPConv code is publicly available on GitHub. It has a well-documented code and descriptions. Also, the folder has 114 forks and 533 stars and the author is responsive and provides active support for the code issues.
4) This algorithm has been tested on several benchmark datasets and achieved acceptable results. Prior to this research, the algorithm was debugged and tested on a custom labeled dataset provided by Amsterdam municipality, and it generated promising results for the part segmentation of pole-like objects.

KPConv Architecture for Scene and Part Segmentation: The network architecture for both semantic segmentation task and part segmentation task is the same (KP-FCNN[9]) and is illustrated in the figure 14 . As it can be seen, a flat input containing the points and their corresponding feature(s) is fed to the algorithm. The grid subsampling strategy is used to create a harmonious point density in the point cloud, which makes the network flexible to deal with various point densities. An important hyperparameter is the 'first subsamping distance', which is the first size used for grid subsampling and it is set based on the point spacing in the point cloud. In each decoding step the number of points is reduced while the subsampling distance and consequently the receptive field area are increased (red balls). In other words, the kernel radius is also updating and enlarging at the same time proportionate to the subsampling distance(Thomas et al., 2019). In the encoding phase, the points are upsampled using the nearest upsampling layer, skip links, and MLP. The ResNet architecture is used to create skip links between encoding layers and the output of decoding layers. The output from the intermediate decoder layer is concatenated with the output of upsampling layers and processed with a shared MLP (unary or 1conv layer) to get the pointwise features(Thomas, 2020; Thomas et al., 2019).In the part segmentation model a single network is trained with multiple heads to distinguish parts of the objects. The point cloud is smaller and multiple instances can be processed simultaneously (Thomas et al., 2019). For the semantic segmentation task the point cloud is large and it cannot be processed at once. Therefore, a KDTree is built to process the point cloud.

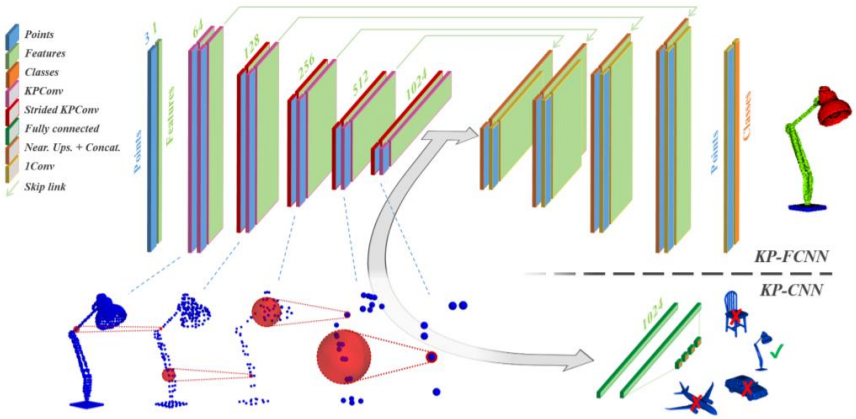


Figure 14 - KPConv Architecture

[9] Kernel Point Fully Convolutional Neural Network

As reported in the main paper, for the easier task of object classification rigid form of KPConv is used while for a more complex task like scene semantic segmentation or part segmentation or on large and diverse datasets, the deform version performs better(Thomas et al, 2019). As it can be seen in the table 3, the deformable kernels achieve higher mIoU on the NPM3D dataset by 3.5%. Therefore, in this project the same network architecture is used. The deformabale kernels are learned using two regularization losses: 1) with the fitting regularization loss, calculated after a local shift in the kernel point positions, the kernel points can capture the local geometry, and 2) by repulsive regularization loss, it is guaranteed that kernel points have acceptable distribution and are not concentrated in one point. (guarantee the best spatial coverage)

Table 3 - KPConv rigid and deformable performance on NPM3D dataset (adopted from Thomas et al., 2019)

methods	Semantic Segmentation on NPM3D dataset mIoU
KPConv rigid	72.3%
KPCon deformable	75.9%

4.2. Semantic Segmentation

Motivation: As the initial step, the KPConv algorithm for the scene semantic segmentation is trained and tested with its default parameters on the 10-class Paris-Lille-3D dataset. This experiment is to reproduce the KPConv published paper result and ensure the code runs smoothly. Furthermore, this initial result can be considered a baseline to determine the success or failure of the further experiments. Also, the predictions of semantic segmentation on the class of interest, pole, can provide an insight into the deficiencies that crop segmentation needs to resolve.

Data Preparation: The default data handling of the KPConv algorithm for this benchmark dataset includes considering the Lille1_2 as the validation set and the rest of the labeled clouds as the training set. The test set is a collection of three unlabelled point clouds. The output of the testing phase is the prediction point cloud which contains the probabilities of each point belonging to each class. The output is interesting for the visual interpretation of the results. However, no quantitative evaluation metric can be obtained from the testing phase because of the missing ground truth. Without knowing the correct prediction for each point, it is difficult to get a clear picture of the results and compare the outcome of different experiments.
Therefore, the main data preparation step for the semantic segmentation task is data partitioning to reach a finer split of the labeled set. Therefore, each point cloud is cut into two halves, generating eight clouds out of the default four. Then five of the clouds were used for the training phase, one for validation and two for testing. With the new split, 64% of labeled points are used for training, 13% for validation, and 23% for testing.

Processing: The hyper parameters setting in this experiment is preserved as the author has suggested and claimed to be optimized to create a more comparable result with the published results.

Evaluation: The same evaluation metrics (IoU and mIoU) cited in the paper is used to compare the achieved and the claimed results.

4.3. Part Segmentation

By 'Part Segmentation,' we mean using the KPConv code for the part segmentation task to generate predictions for the cropped tiles. Therefore, in this work, we use the terms' part segmentation' and 'crop segmentation' interchangeably.

Motivation: The main assumption is that the part segmentation model can consider each tile as an instance of an object and segment it into the given ten classes. The main motivation is to see how the predictions in the same tile will be modified after the part segmentation and check whether it is improving. The main rationale behind generating smaller tiles is to improve the model's performance for the minority class of interest (pole) by reducing the class imbalance. The better segmentation results on the minority class of interest, pole, would also improve the predictions for majority classes by reducing the confusion between the minority and the majority classes. Therefore, the initial assumption is that a finer prediction stage would improve the overall result of the semantic segmentation.

Data Preprocessing: The data preparation for the part-segmentation task is done in two consecutive phases illustrated in figure 15 The motivation behind the first phase is to extract the neighborhood of individual poles with which the part-segmentation model is going to be trained and tested. The second phase of data preparation is mainly focused on normalizing the data values per tile to make the learning process smoother for the algorithm and prepare the data to be readable for the part segmentation code.

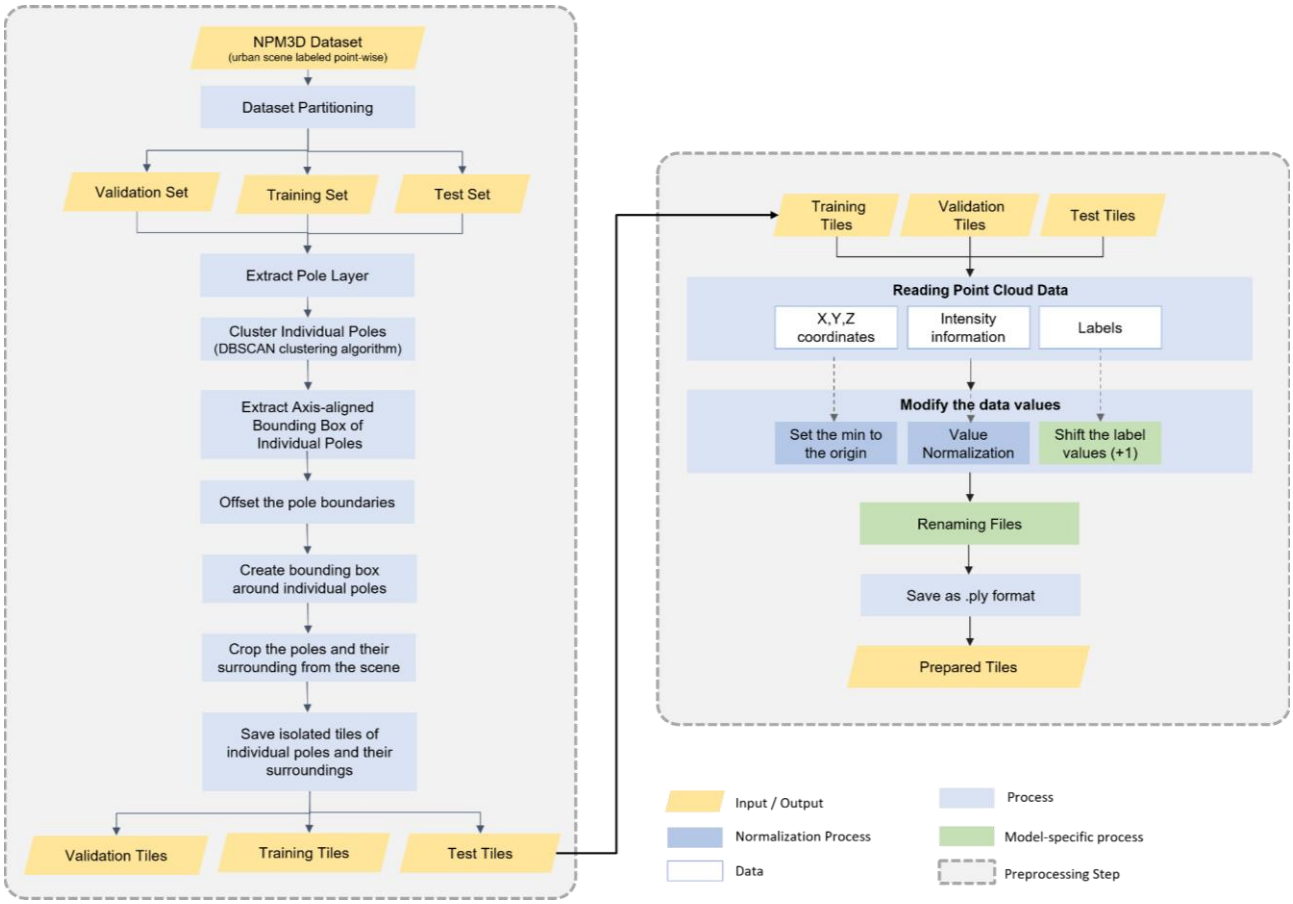


Figure 15 - Part Segmentation Data Processing Workflow

(A) Neighborhood extraction: In the first step, the same data partition[10] as the semantic segmentation phase is used. After partitioning the data, for each point cloud(figure 16-A), the pole points are extracted as a layer(figure 16-B). It should be mentioned that the pole layers were extracted from the ground truth labels of the scene. The reason behind this choice is threefold:

1) To simplify the problem for a smoother pipeline design and evaluation
2) To include all the pole instances and see how the proposed pipeline deals with different scenarios.
3) The tiles created from the ground truth benefit from perfect neighborhood definition. Thus, the best possible result that can be achieved with the part segmentation (the maximum potential of the method) can be reported.
4) This baseline facilitates spotting the complexities of integrating semantic segmentation and part segmentation separately in a later step.

After extracting the pole layer from each point cloud, a built-in functionality in the open3D library, DBSCAN clustering, is used to identify individual poles from the pole layer. This function groups the point based on the local point density. The required inputs to the function are the minimum number of points to form a cluster (set to 10) and the Density parameter (set to 0.5) that is used to cluster individual poles(figure 16-C) (open3d documentation).

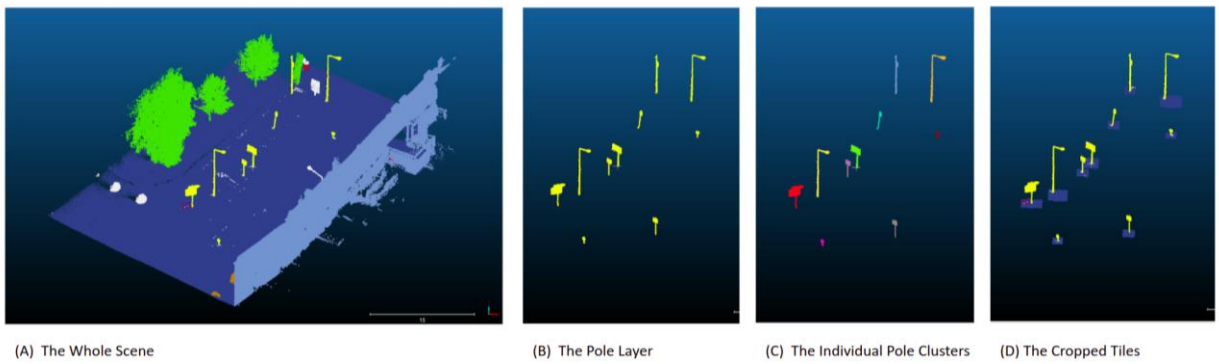


Figure 16 - Neighborhood extraction workflow

After extracting individual poles, their axis-aligned bounding box on the XY plane is extracted using their min(X, Y) and max(X, Y) coordinate values(figure 17). It should be mentioned that the experiment is repeated with an oriented bounding box (or minimum bounding box) around the poles and yields poorer results. The reason might be that more information about the background classes is captured using an axis-aligned neighborhood definition that helps the model distinguish them better.
The XY bounding box of each pole is enlarged by adding a constant value (offset parameter) to provide richer contextual information(figure 16- D).

[10] Explained in section 4.2.

It is important to set the offset parameter to a number so that the created tile captures the geometrical essence of existing objects. For instance, the tile should not be small to the extent that the part of the cropped building becomes similar to a pole-like object. Also, it should not be so large that the part-segmentation model cannot handle the task properly. The offset parameter in this project was set to 0.5m, 1m, and 1.5m and fine-tuned during the experiments.

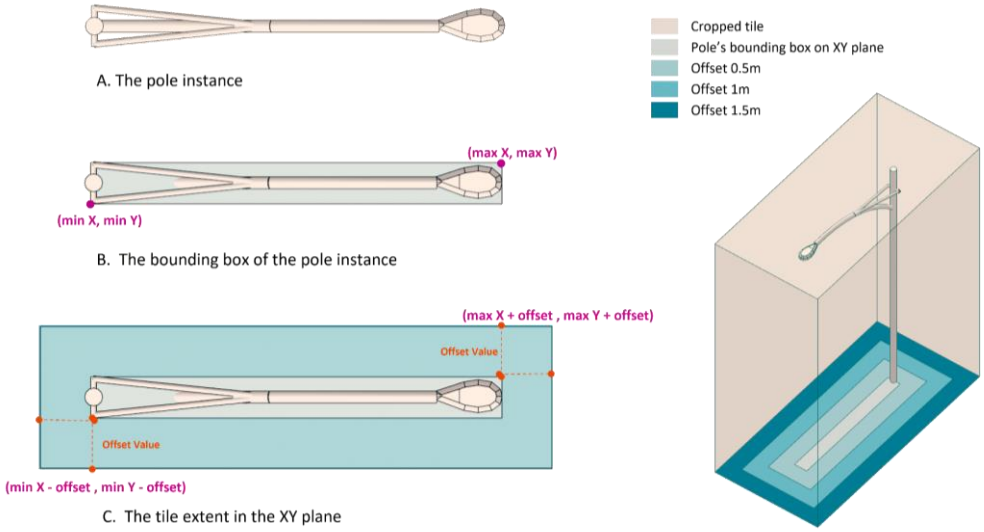


Figure 17 - Pole bounding box and tile extraction

(B) Data Normalization and Preparation: After generating the pole neighborhood tiles, the second phase of data preparation begins. In this step, the point features (coordinate values, intensity values, and labels) are normalized as it is illustrated in figure 15 . The normalization applied for coordinate and intensity values is done to make the learning process easier for the model. For instance, absolute values of x,y, and z coordinates might confuse the model as they are a variety of big numbers, while the relative positions to the lowest point can be easier to learn as it is smaller and is almost within a similar range for different tiles.
It is important to mention that in the semantic segmentation task, the normalization of the feature values is done throughout the default pipeline. The motivation behind highlighting these steps in the part-segmentation task is that the preparation steps are modified to fit the purpose of the project and make the data understandable for the model.

In the default pipeline, the coordinate values are normalized by placing the object's center of mass to the origin (0,0,0). However, in this study, the lowest point in each tile is set to origin so that the model can learn the upright linear objects more easily. By default, the part segmentation task only uses the coordinate values for training and inference. However, since the intensity values are included in this study as additional features to the algorithm input in one experiment, the intensity values are normalized and stored in the prepared data. A similar normalization method as reflectance normalization done in the semantic segmentation task is used to make the results comparable. In this normalization method, all values larger than 50 in an 8bit reflectance data are mapped to 50, and the smaller values remain intact. Also, some code-specific modifications like shifting the label values by +1 and renaming the files are done instead of modifying the code to accept 0-based labeling and random file names.

Processing: In this experiment, the tiles are cropped from the ground truth with the same process explained in section 4.3. The KPConv algorithm is trained for the part segmentation task and then tested with these cropped

tiles. The pole neighborhoods are defined with 0.5m offset from the poles' axis-aligned boundaries. Due to the technical limitations(out of memory issues), the batch_size is set to 10, and the first subsampling distance is set to 0.05. The rest of the parameters are set to the default optimized by the author.

Evaluation: Aside from the visual interpretation of the results and the common evaluation metrics (Recall[11], Precision[12], F1[13] , and IoU[14]), a method is used to simplify the results efficiently. To compare the results to the original predictions of the semantic segmentation, the same tiles are cropped from the prediction point cloud generated with the semantic segmentation task. The confusion matrix is generated for each tile, once for the cropped semantic segmentation predictions and once for the part segmentation predictions. Two matrices are accumulated over all the tiles to reflect the overall performance of each task. Finally, as we are interested in the improvements that the crop segmentation phase has made, the accumulated confusion matrix of the semantic segmentation task is subtracted elementwise from the accumulated part segmentation confusion matrix to generate the 'Difference of Confusion Matrices'. In this type of matrix, it is ideal to get positive values only on the diagonal and negative values on the rest of the cells because, in an ideal scenario, the number of true positive points increases, and the number of false positives and false negatives reduce.

4.4. Integrating the two steps

Motivation: After evaluating the capability of the part segmentation task in generating predictions for tiles cropped around the poles in the ground truth, a real scenario is tested where the inference tiles given to the part segmentation trained model are extracted from the semantic segmentation results. The main motivation for this experiment is to assess the applicability of the proposed pipeline in a real-world scenario and highlight its possible complexities. Also, with this experiment, the pipeline design is finalized to be adapted in a real-world application, shown in figure 18.

Method: A similar data preparation process as section 4.3. is pursued here with some minor changes as follows:

1) The pole layer is extracted from the imperfect and noisy results of the semantic segmentation task.
2) Only the test (or inference) point clouds are prepared because the previously trained network[15] for the part segmentation will be used for inference.
3) The offset value is increased to 1m with two motivations: (a) to guarantee that the entire pole is captured in the tile even if only a small part of the pole is segmented initially, and (b) to give the possibility of cropping the tile padding where the lack of contextual information worsens the performance. Therefore, the model is being tested on larger tiles compared to tiles with which it is trained.
4) An additional process of omitting multiple bounding boxes generated around the same pole due to the unstable semantic segmentation predictions is required(discussed in section 5.6 – Complexity(1)).
5) Padding removal is also added to the pipeline to remove the poor predictions around the edges due to the lack of contextual information (discussed in section 5.2 – Edge Problematic)
6) Inserting back to the scene: As the part segmentation model is trained with specific attention to the poles (discussed in section 5.2 – Background Classes Problematic), only labels belonging to the pole class are altered when inserting the tiles back to the scene. To this end, all the points labeled as pole in each tile

[11] Recall = TP /(TP + FN)
[12] Precision = TP / (TP + FP)
[13] F1 = 2TP/ (2TP + FP + FN)
[14] IoU = TP/(TP + FN + FP)

1) [15] The trained network is trained with tiles of 0.5m offset from the pole axis-aligned bounding boxes.

after the part segmentation will be kept as poles. The labels for points that were initially labeled as pole in the scene segmentation will be updated with their corresponding part segmentation prediction.

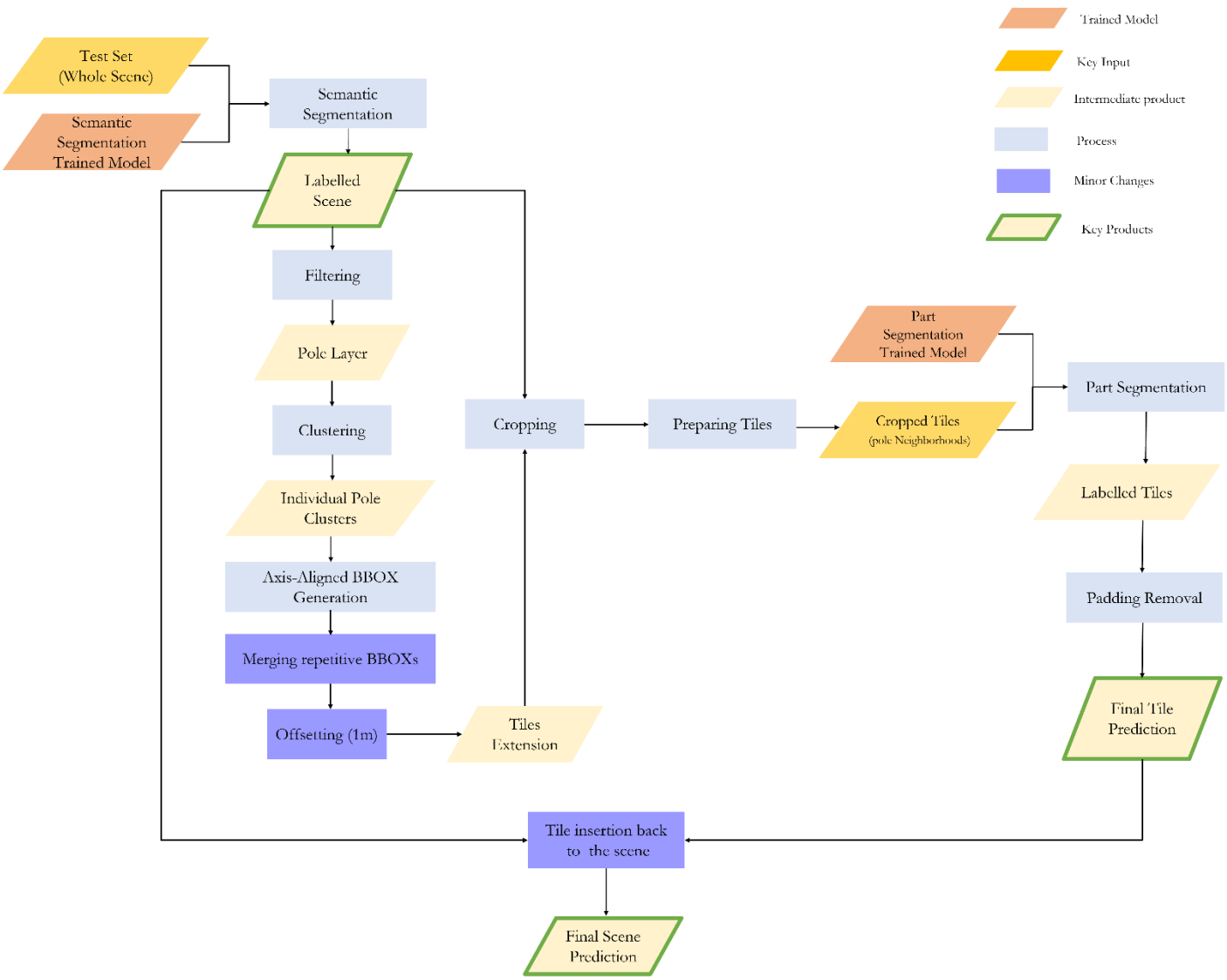


Figure 18 - Proposed Pipeline for integrating Semantic Segmentation and Part Segmentation

5. RESULTS AND DISCUSSION

5.1. Scene Semantic Segmentation on the NPM3D dataset

In this section, the scene semantic segmentation task is done on the NPM3D dataset using KPConv default code and parameter setting. As it can be seen in table 4, the achieved intersection of union (IoU) values for most of the classes is less than the published result. The reason is that the training set is smaller in this experiment due to the partitioning of the publicly available labeled [16] point clouds. Consequently, the trained models are tested on different point clouds, another reason for minor discrepancies between the claimed and achieved results. However, the results are close enough (on average, there is a 6% difference between the claimed and achieved IoUs) to claim that the code is debugged, runs smoothly, and the model can generate meaningful predictions.

Table 4 - Tabel 3 - Comparing the published and achieved results for the semantic segmentation task on NPM3D using KPConv algorithm

KPConv	mIoU	Ground	Building	Pole	Bollard	Trash can	Barrier	Pedestrian	Car	Natural
Claimed	82.0%	99.5%	94.0%	71.3%	83.1%	78.7%	47.7%	78.2%	94.4%	91.4%
Achieved	79.1%	97.8%	93.6%	65.3%	70.5%	70.1%	42.6%	85.5%	96.3%	90.2%

The class-imbalance problematic

The quantitative inspection of the results reported in table 5 shows more than 20% difference between the IoU of majority classes like ground and building compared to the minority classes like poles, bollards, or trashcans. The KPConv algorithm, like most deep learning models, is assumed that the dataset has an equal distribution among all the classes. Therefore, by maximizing its overall predictive performance, the classes with a higher number of representative points will be segmented with better results. In contrast, the limited number of points representing minority classes will lead the algorithm to ignore the corresponding false predictions and produce poorer results for these classes.

Table 5 - Share of classes in the NPM3D dataset

Class	Unclassified	Ground	Building	Pole	Bollard	Trash can	Barrier	Pedestrian	Car	Natural
Number of points	1642298	65075320	33014819	656096	61715	296523	4052947	172132	4212295	10599237
Share in the dataset	1%	54%	28%	1%	<<1%	<<1%	3%	<<1%	4%	9%

As can be seen in table 5, the class imbalance is an inherent characteristic of the data used in this research. The motivation behind the part segmentation step is to tackle this issue by cropping the scene around the object of interest to learn from and predict on a more class-balanced tile set.

[16] The test set for NPM3D dataset is publicly available but it is unlabeled. The absence of annotation, prevents us to generate quantitative results out of the predictions. Therefore, a part of training set is separated and used during the test phase. See section 4.2.

The class similarity problematic

Another spotted issue is the inter-class confusion which stems from the class similarity in the training set. In the table 5 the class' barrier' with 3% of share in the whole dataset has the lowest mIoU compared to other minority classes with less than 1% share. In this case, the class imbalance cannot explain the poor result, but the class similarity justifies it.

Some classes are similar regarding their appearances or location. For instance, bollards and pole-like objects are both cylindrical objects located along the sidewalks but have different heights. The other example is the barrier class, which represents the fences, short walls, and transparent facades located in the continuation of the buildings. Buildings and barriers are both planar objects defining the edge of the scene (figure 19 - C, D, and E).

Furthermore, for some classes like 'natural', different instances of the class, might have different geometries. For instance (figure 19 - A and B), green components in an urban scene can be in the shape of a tree, a green wall, a cubical bush, a garden, grass, or a cylindrical or hanging component. The variation in geometrical shape results in a huge confusion between class natural and other classes.

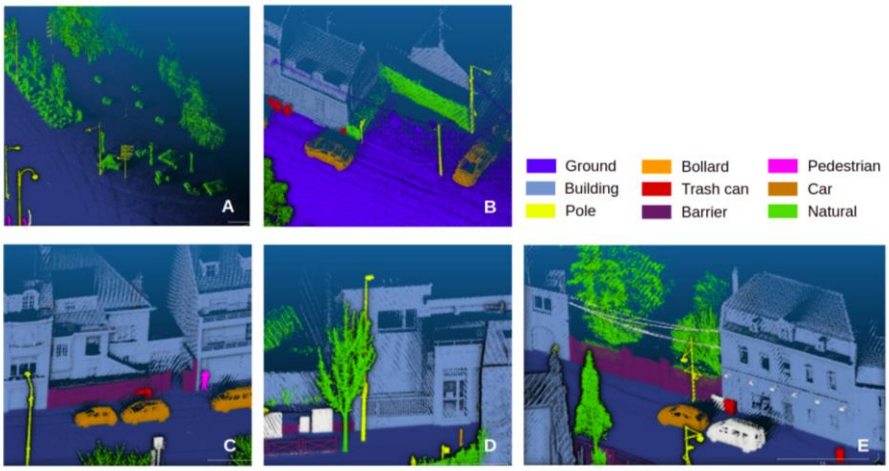


Figure 19 - Problematic labeling choices in NPM3D dataset

Table 6 - Confusion Matrix of Semantic Segmentation task on NPM3D dataset

False Negative →

Prediction

Ground truth ↓ Class	Ground	Building	Pole	Bollard	Trash can	Barrier	Pedestrian	Car	Natural	SUM
Ground	37439936	84645	879	315	2486	20304	315	3556	174468	37726904
Building	148462	41062476	14903	319	7080	10870	1004	2546	144597	41392257
Pole	2198	99109	401180	1454	68	8576	290	351	36436	549662
Bollard	3321	167	5877	37727	79	139	475	191	172	48148
Trash can	21742	42851	3315	2891	364087	9692	1159	3931	56408	506076
Barrier	31233	1801110	10303	208	549	2114429	3307	16142	829109	4806390
Pedestrian	1061	505	2190	0	26	8	163345	817	7179	175131
Car	18102	19860	1683	53	47	38598	8487	3930854	33840	4051524
Natural	321213	430043	25544	100	2698	71701	957	2349	19708412	20563017
SUM	37987268	43540766	465874	43067	377120	2274317	179339	3960737	20990621	109819109

↓ False Positive

Correct Predictions
Significant Confusion - False Negative
Significant Confusion - False Positive

The inspection of the confusion matrix (Table 6) shows that in almost all the classes, a huge amount of points are labeled as 'natural' due to the geometrical variety of the natural components that confuses the model. The next confusing class is 'building'. The geometrical and topological similarity between the barrier and building class and the proximity of poles, trash cans, and natural classes with building class has led to huge confusion. Also, the intersection of the regions belonging to pole and natural classes and natural and barrier classes and their closeness justifies the confusion.

The annotation of classes' building', 'barrier', and 'natural' in the NPM3D dataset is confusing for the model. Many green walls and walls around the properties, as upright planar components, are labeled as natural and barrier, respectively. The geometrical similarity between these classes confuses the model. Therefore, additional features like the number of returns or color information should be included to distinguish these classes better. Since the NPM3D dataset only has coordinate and noisy reflectance values, there are not enough informative features to distinguish them properly in the whole scene.

5.1.1. Pole Predictions in the scene

False Positives: As shown in figure 20, after the semantic segmentation, parts of other classes are confused with the class pole. For instance, the trunk of a tree(A), a part of a bollard(D), a cylindrical piece of a barrier (B), and even the trace of a pedestrian (C). All these confusing objects have cylindrical and vertical geometry, which makes it understandable if confused with poles. A previous study (Yousefimashhoor, 2022) shows that objects like bollards, poles, or vegetation attached to poles can be distinguished using a part segmentation model trained to decompose pole attachments. Therefore, merging classes like poles and bollards into the same class and dividing them using an additional PLO part segmentation task might improve the results. On the one hand, it will slightly increase the pole class size in the dataset. On the other hand, it will distinguish poles and bollards with higher accurate boundaries.

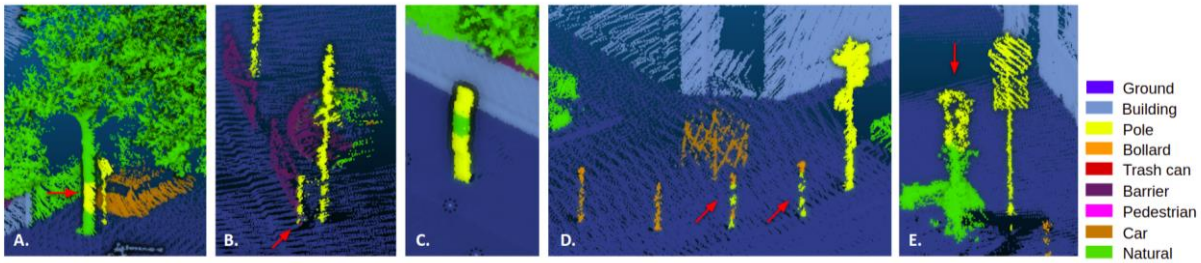


Figure 20 - False Positive cases in segmenting poles from the whole scene

Falsely False Positive: In some rare cases (figure 21) the semantic segmentation can segment a part of the pole attached to the buildings. However, these points are labeled as unclassified. They will not be calculated in the confusion matrix nor reflected in the quantitative results as an improvement. In these cases, poor labeling will distort the results slightly. Since these poles provide lighting, they are also interesting for asset management purposes. It is informative to know that the semantic segmentation can detect the pole's light bulb even with no prior similar instance in the training set. Since such cases are progressing to the second step of part segmentation, there is a chance of improvement for them.

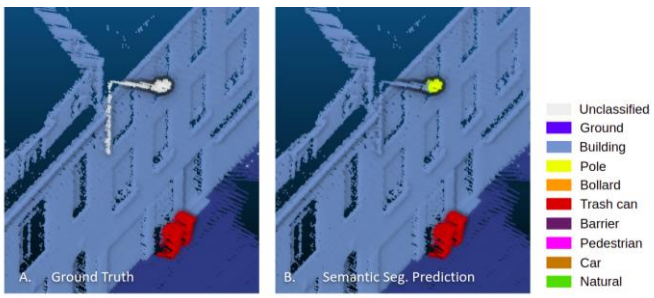


Figure 21 - Falsely False positive scenarios in pole segmentation in the scene semantic segmentation

False Negatives: As illustrated in figure 22, parts of the poles are sometimes missing due to their intersection with other classes like tree branches (C), or when it is located close to other classes like buildings and barriers (A and B).

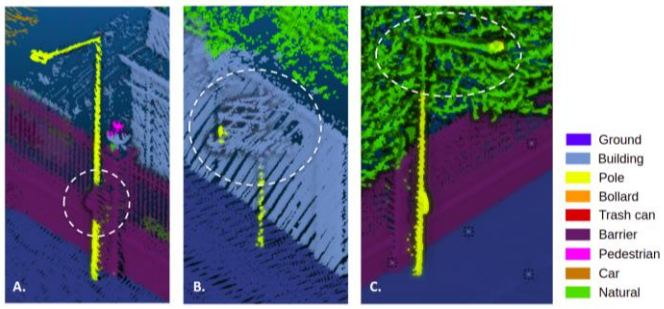


Figure 22 - False negative cases in semantic segmentation for pole class

Entirely Missed poles: There are five poles in the test dataset where not a single pole point is detected on these poles. Two of these poles are illustrated in figure 23. As it can be seen, they are located very close to the building that the model has confused with the building signs and attachments, which are very common in the training set. These cases will consequently be eliminated and would not go further into the second phase of prediction with the part-segmentation trained model. Thus, their corresponding results cannot be improved with our proposed pipeline.

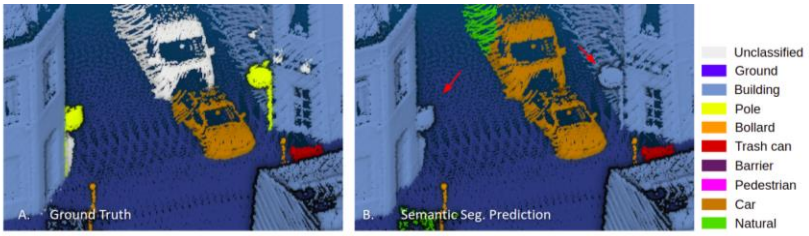


Figure 23 - Entirely missed poles in the semantic segmentation

Point Density or Edge problematic: Another important observation is the false predictions in the scene edges where the point cloud has become sparse. The predictions become unstable in these locations. As shown in figure 24, as we get closer to the edge of the scene, more points belonging to the pole class are confused with the building class.

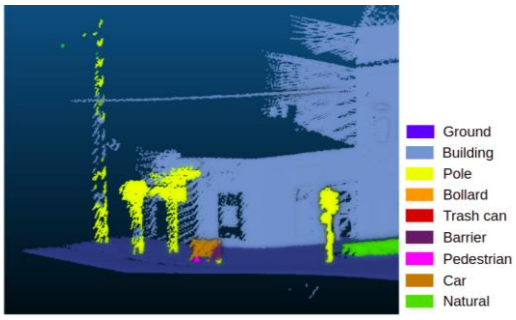


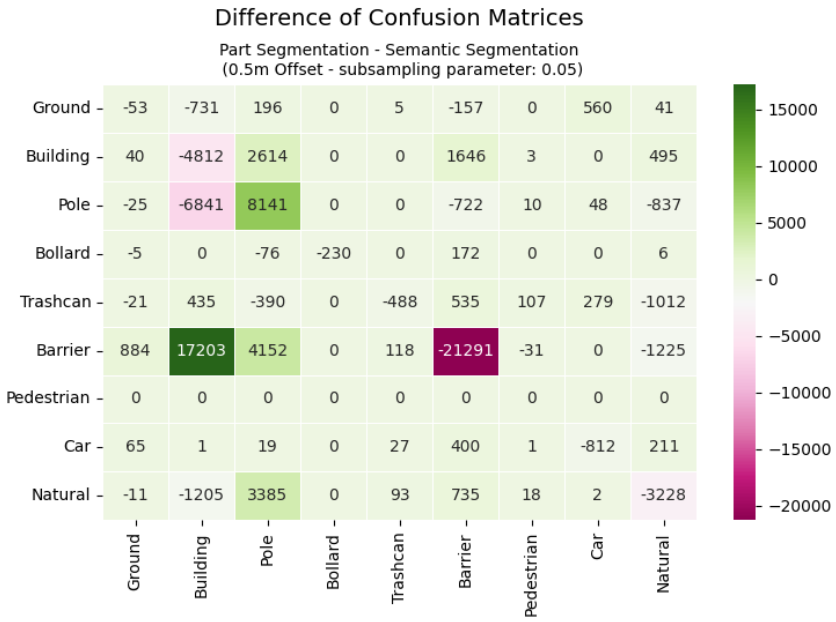
Figure 24 - point density affecting semantic segmentation results

5.2. Part segmentation of pole tiles

In this section, the extracted tiles of pole neighborhood are used to train the part segmentation model to generate predictions on the test tiles. The confusion matrices for semantic segmentation and part segmentation per tile are generated and compared. The difference of accumulated confusion matrices[17] (table 7) shows that for the class pole, the number of true positives after the crop segmentation increased by more than 8000 points. It also shows that around 6800 points belonging to the class pole, which was previously labeled as buildings in the semantic segmentation step, are now correctly labeled as poles. However, more than 4800 points belonging to the class 'building' are labeled as poles, which were previously labeled correctly. Another interesting observation is that the negative numbers along the 'pole' row highlight that the number of points belonging to the class pole that was initially confused with other classes is reduced in all of the classes. In contrast, the positive numbers along the column 'pole' show that the part segmentation has worsened the prediction of all other classes by labeling them as poles (high false positives rate). For the detailed confusion matrices of semantic segmentation and part segmentation of tiles, see Appendix (1).

[17] See the evaluation method for part segmentation (section 4.3.)

Table 7 - Difference of Confusion Matrices : Part segmentation of 0.5m offset tiles



A summary of the result is presented in table 8 . As can be seen, due to the decrease of false negatives and increase of true positives, the recall is increased for the pole class after the part segmentation phase by around 6.5% and has reached 92%. However, this improvement is at the cost of precision. The precision is reduced by more than 7% compared to the semantic segmentation result because of the increase in false positives rate. The IoU and F1 scores for the pole class in both predictions are comparable. The reason is that the decline in false negatives and increase in false positives have neutralized one another in calculating these metrics.

Table 8 - Evaluation Metrics for all the class in tiles : comparing semantic segmentation and part segmentation

Evaluation Metric	Semantic Classes																	
	Ground		Building		Pole		Bollard		Trashcan		Barrier		Pedestrian		Car		Natural	
Task	Sem.	Part.	Sem.	Part.	Sem.	Part.	Sem.	Part.	Sem.	Part.	Sem.	Part.	Sem.	Part.	Sem.	Part.	Sem.	Part.
Precision	98.7%	98.4%	75.6%	69.3%	97.5%	90.3%	100%	0%	100%	73.7%	94.6%	75.3%	0%	0%	99.8%	10.4%	83.7%	86.2%
Recall	99.2%	99.3%	99.8%	94.6%	85.5%	92.0%	54.5%	0%	30.9%	21.1%	63.7%	25.5%	0%	0%	96.7%	1.2%	91.9%	86.5%
F1	99.0%	98.8%	86.0%	80.0%	91.1%	91.1%	70.6%	0%	47.2%	32.8%	76.1%	38.1%	0%	0%	98.2%	11.1%	87.6%	86.4%
IoU	98.0%	97.6%	75.4%	66.7%	83.7%	83.7%	54.6%	0%	30.9%	19.6%	61.4%	23.5%	0%	0%	96.5%	5.9%	77.9%	76.1%

Semantic Segmentation (baseline) | Decline | Improvement | No Significant Change

- Sem = Semantic Segmentation
- Part = Part Segmentation

False Positives problematic

The most apparent observation is the dramatic increase in the false-positive rate for the class pole, which arises from the methodology. The design of the training dataset has created a bias toward pole class. Since a pole object exists in every tile of the training set, and other classes are rare, the model assumes a pole in every tile and assigns

as many points as possible to this class. Cropping (equal to reducing the contextual information) and changing the local geometry of classes also worsen the results.

Difficulty level problematic

The performance of the part segmentation trained model is not the same in every tile, and it seems to be affected by the tile difficulty level. In some easy tiles belonging to an isolated pole with enough distance from other objects, the part segmentation generates nearly perfect results similar to the semantic segmentation.

In more complicated scenarios, where a pole is located close to a bush, a fence, or a building, the semantic segmentation cannot segment the entire pole properly. However, as visual results show, the part segmentation model can improve the segmentation. As can be seen in the figure 25 the middle part of the pole with its attached traffic sign is initially predicted as 'barrier', while the part-segmentation model can recognize the whole pole correctly.

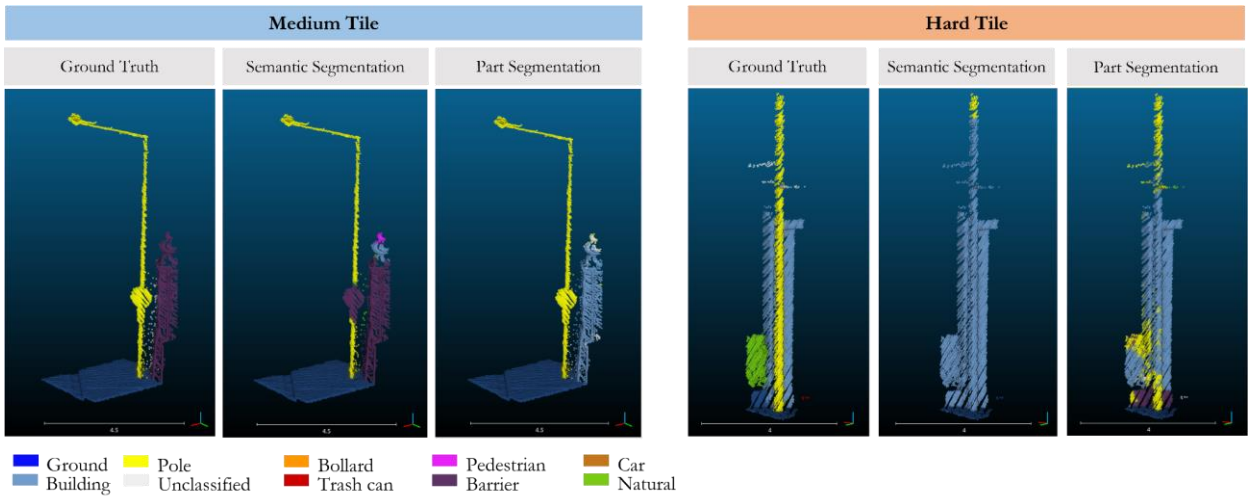


Figure 25 - Tiles and their difficulty levels

There are also some hard scenarios where poles are intersected with a tree or are attached to a building. The trained semantic segmentation model predicts most of the pole points as part of the adjacent or intersecting class. The part-segmentation model can segment more points belonging to the pole correctly. Meanwhile, it increases the number of false positives and pole confusion with other classes, particularly in the edges. The reason might be the limited number of instances (27 hard instances out of 167 training examples[18] equal to 16% of the dataset) of a similar scenario in the training set, which seems to be insufficient for the model to learn to distinguish them.

In some cases, the classes are intertwined at a level that even a human operator cannot distinguish easily. A wide variety of possible problematic scenarios are illustrated in figure 26 , a pole can be attached to a building, it can be surrounded by multiple classes like trash cans, natural and barrier, a pedestrian might lean against it or it can be located inside a bush or a tree crown. The various combinations of hard instances make it almost impossible to provide well-presented training data for such scenarios.

[18] Counted manually by exploring the training set

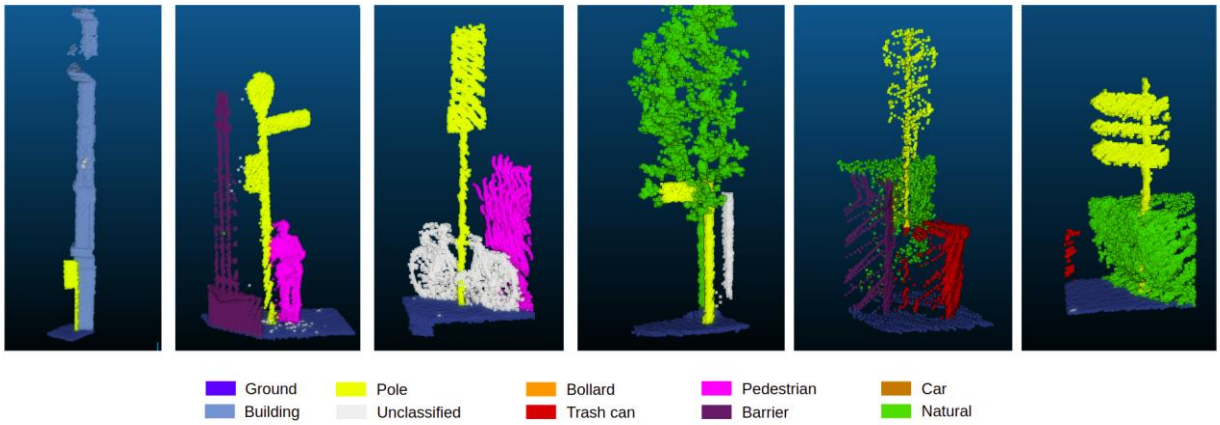


Figure 26 - Variety of possible hard scenarios (extracted from the ground truth)

Edge Problematic

The visual investigation of the tiles reveals that the quality of part segmentation predictions reduces near the tile edges. This observation can be justified by how the convolution works in the algorithm. The area captured in the spherical kernel near the tile edges is artificial due to the cropping. Therefore, when centered around edge points, a part of the kernel receives empty space while those locations contain points and information in the original scene. The artificial emptiness captured by the kernel points seems to result in poor predictions around the tile edges.

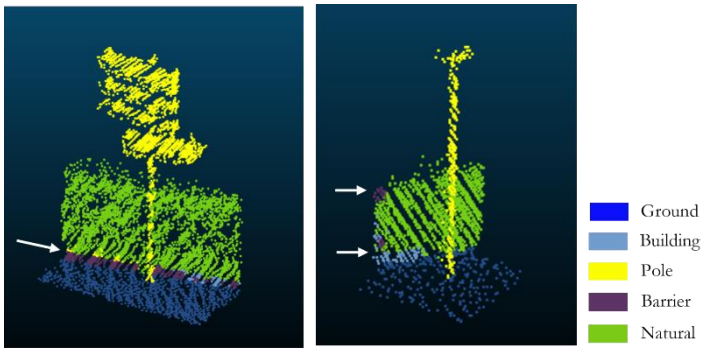


Figure 27 - Edge problematic in tile prediction

The kernel radius in the first layer is set to 0.1m by default and doubled after each layer. The architecture of KPConv for the segmentation tasks has five layers, meaning the kernel radii sequence will be [0.1, 0.2, 0.4, 0.8][19] , and the largest value would be 0.8m. Therefore, a crop with the offset of 0.8m from the edges of the tile will give the points that their complete neighbor has captured by the kernels, and thus their corresponding predictions can be trusted. Table 9 shows the IoU result calculated over all the tiles in two scenarios of the full extent and cropped 0.8m from the edges. As can be seen, the IoU of the pole class is 6.4% higher than the semantic segmentation result in the same extent of tile in the trusted zone. While in the entire extent of the tile, the IoU for the pole class generated by the part segmentation does not show any improvement compared to its corresponding semantic segmentation results. See Appeendix (2) to observe how padding removal in the ideal scenario will improve the semantic segmentation result.

[19] Since after the fifth layer the decoding phase begins, the kernel size does not enlarge.

Table 9 - Comparing the results of part segmentation with and without padding

Point Cloud	Task	IoU per Class								
		Ground	Building	Pole	Bollard	Trashcan	Barrier	Pedestrian	Car	Natural
Tiles (Full extent)	semantic segmentation	98.0%	75.4%	83.7%	54.6%	30.9%	61.4%	0%	96.5%	77.9%
	part segmentation	97.6%	66.7%	83.7%	0%	19.6%	23.5%	0%	5.9%	76.1%
Cropped tiles (0.8m offset from the edge)	semantic segmentation	99.6%	0%	85.9%	0%	0%	43.9%	0%	0%	83.5%
	part segmentation	99.2%	0%	92.3%	0%	0%	13.4%	0%	0%	80.3%

Semantic Segmentation (baseline) — No improvement
Part Segmentation — Improvement

Background Classes Problematic

Another important observation in table 9 is that after the tile segmentation, the IoU for all the classes except pole reduces. This decline stems from the pipeline's focus on the pole class: in this methodology, only the pole class is captured fully in each tile with enough contextual information due to the offset from its bounding box. While other classes might be completely absent or have a limited number of instances in the training dataset. Also, some large-scale or extended objects, such as cars and barriers, will always capture partially in the tiles. The lack of contextual information and the remaining parts with random or floating shape makes it even hard for the human brain to identify the classes correctly.

An example is illustrated in figure 28. The floating points belonging to the class 'building' are similar to the tree branches. Or in another case, as barriers and green walls are both vertical planar objects, distinguishing them is not easy. Particularly, when the barrier class gets sparse, it becomes similar to a bush. On that account, it is understandable that the performance of the part segmentation model for classes other than pole is not as good as the semantic segmentation model that captures the complete geometry of all objects. Thus, changing labels of classes other than 'pole' based on part segmnetation output is highly risky due to the lack of contextual information.

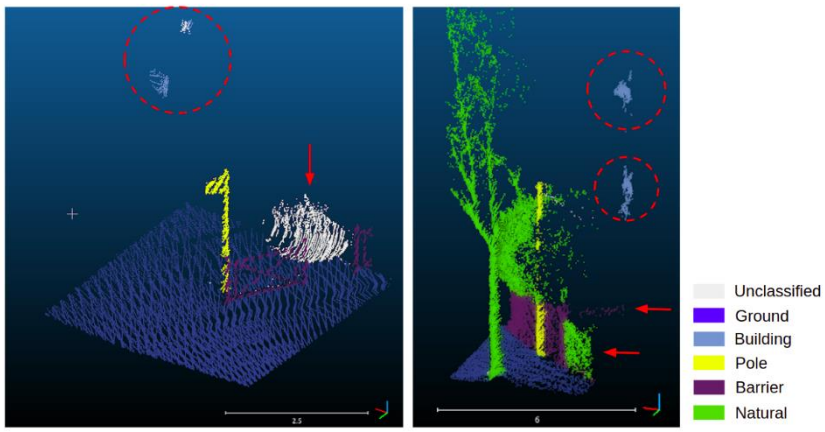


Figure 28 - lack of contextual information in the tile edges

5.3. Enlarging the tile size

With the motivation of fine-tuning the "offset parameter" or to find the tile size range that the part segmentation task can handle, the same method described in section 3.3.2 is repeated with two different offset sizes of 1m and 1.5m around the pole boundaries. The part segmentation model is trained separately in each scenario with the corresponding tile size.

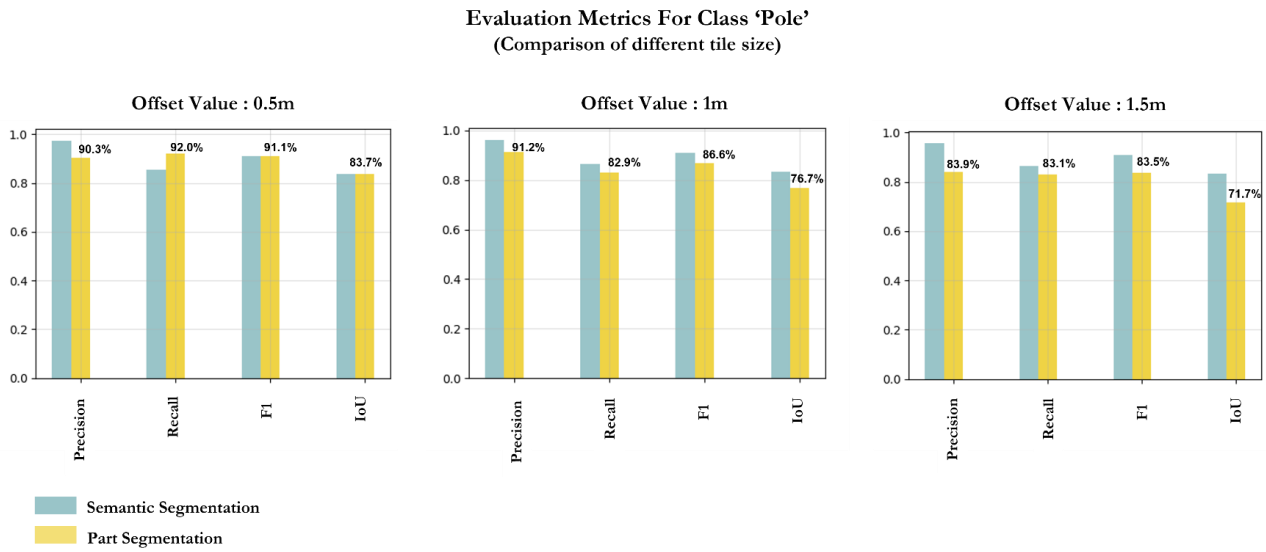


Figure 29 - Evaluation metrics change by enlarging the neighborhood tiles

The comparison of evaluation metrics among different tile sizes shows that the most promising result belongs to the smallest tiles of 0.5m offset. As seen in figure 29, the evaluation metrics of recall, F1, and IoU for the pole class are better with 0.5m offset compared to larger tiles with an average of 8.5%, 6%, and 9.5%, respectively, which are considerable differences.
However, the precision is slightly higher with the tiles of 1m offset. The reason is that enlarging the tiles can capture more contextual information, and the model can learn from more points representing the background classes. Also, there is a better chance of battling changes in the overall geometry of an object. For instance, buildings or barriers will be presented with higher similarity to their original geometry as vertical planar objects in a wider area. More importantly, the share of the class pole is reduced as the tiles get larger. The share of the pole in the training set is 28.5%, with a 0.5m offset. This number will is reduced to 15.3%, and 9.3% in 1m and 1.5m offset tiles, respectively. The decline in the proportion of pole points results in a less biased network towards the pole class. Therefore, a lower rate of false positives and better precision are achieved with a 1m offset.

The investigation of the difference of confusion matrices for pole class (figure 30) shows that the number of true positives with 1m and 1.5m offset is reduced drastically after the part segmentation.The number of true positives for the pole class is increased by around 8000 points with 0.5m offset, while it decreased by about 4800 points with each of the larger tiles. Since the share of the pole class has decreased in the enlarged tiles, the model was not able to learn to segment it properly, and the number of true positives has reduced significantly in the bigger tiles. The highlight for the largest tile size (1.5m offset) is the decline of false negatives for the pole class that was caused by the barrier and natural classes. Except that, the confusion between pole and all other classes is increased. Especially, the number of false positives is increasing dramatically with the contribution of almost all of the classes.

Difference Matrix (0.5m offset) - Focused on the Pole Class
(Part Segmentation - Semantic Segmentation)

Class	Ground	Building	Pole	Bollard	Trashcan	Barrier	Pedestrian	Car	Natural
Ground			196						
Building			2614						
Pole	-25	-6841	8141	0	0	-722	10	48	-837
Bollard			-76						
Trash can			-390						
Barrier			4152						
Pedestrian			0						
Car			19						
Natural			3385						

Difference Matrix (1m offset) - Focused on the Pole Class
(Part Segmentation - Semantic Segmentation)

Class	Ground	Building	Pole	Bollard	Trashcan	Barrier	Pedestrian	Car	Natural
Ground			30						
Building			5906						
Pole	-46	6199	-4760	11	121	-1150	0	0	-515
Bollard			409						
Trash can			-1384						
Barrier			-483						
Pedestrian			78						
Car			6						
Natural			1938						

Difference Matrix (1.5m offset) - Focused on the Pole Class
(Part Segmentation - Semantic Segmentation)

Class	Ground	Building	Pole	Bollard	Trashcan	Barrier	Pedestrian	Car	Natural
Ground			68						
Building			7323						
Pole	-3	7530	-4806	0	205	-1388	148	0	-1776
Bollard			533						
Trash can			619						
Barrier			1311						
Pedestrian			565						
Car			0						
Natural			6745						

Figure 30 - Difference of confusion matrices, focused on the class pole for tiles of different sizes

The visual interpretation (figure 31) of the results also shows the enormous false positive rate in the smallest tile. At the same time, it shows the fewer true positive rate for the larger tile of 1m offset. Again, the increase in false-positive rate can be seen for tiles of 1.5m offset. It should be mentioned that enlarging the tiles did not consistently affect all the pole instances. Therefore, the quantitative results give a better overall insight.

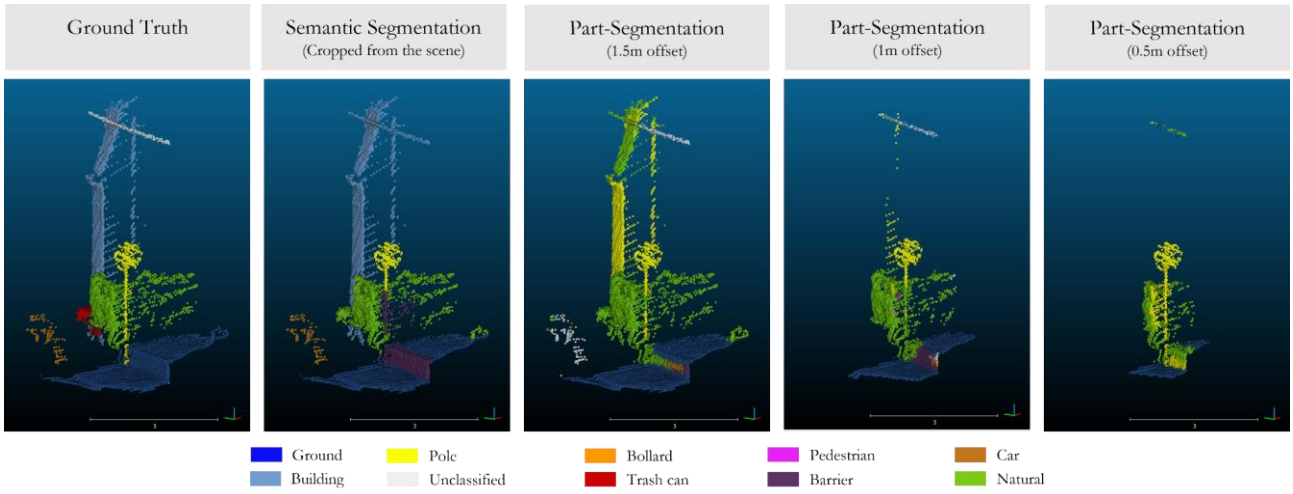


Figure 31- Effect of enlarging the tile size on a pole instance prediction

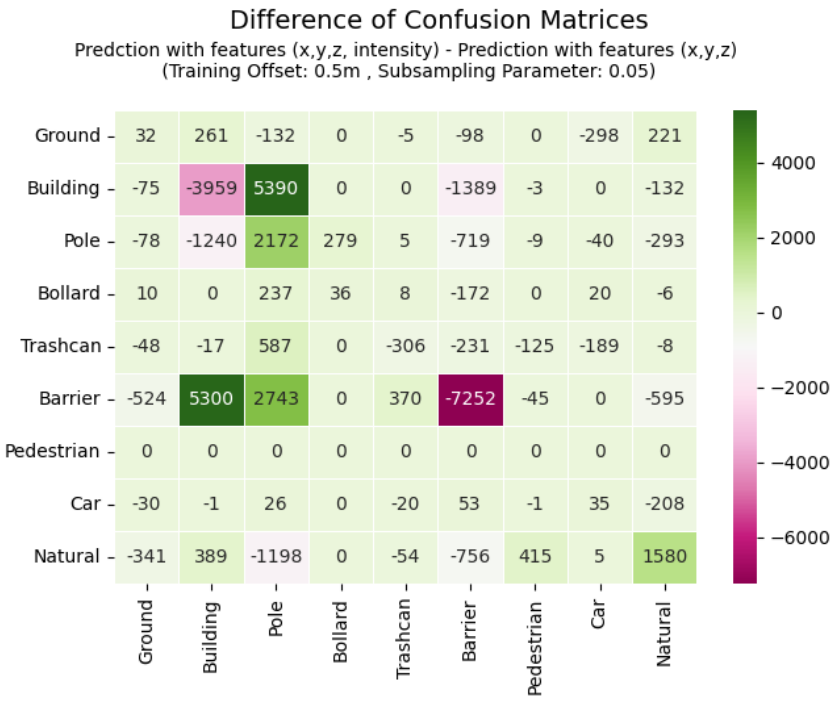
5.4. Including an additional input feature

'input features', as it is called in the KPConv configuration, refer to different attributes of points available in the dataset. Aside from the coordinate values, reflectance, color values, and the number of returns can be stored as point features in a given point cloud. These attributes can be included in the input tensor of a model and be learned and used for inference.

The only additional feature in the NPM3D dataset is the intensity value. With the motivation to use the full potential of the existing features in the dataset and see whether any additional feature can improve the performance of the result or not, the reflectance data is used in addition to coordinate values to train the part segmentation model. A previous study on the part segmentation of pole-like objects (Yousefimashhoor, 2022) shows that due to the usage of highly-reflective material or distinct colors in the attachments of the poles, including additional features like RGB and intensity values, has helped the network to distinguish the attachments better. Therefore, it is expected that the additional reflectance feature would help in segmenting the pole attachments better.

The same normalization method used by the author for the semantic segmentation task on the NPM3D dataset is used for this experiment to make the results comparable. In this method, all the reflectance values(8bit) higher than 50 are mapped to 50, and all the intensity values less than 50 remain intact.
As seen in the confusion matrices below, including intensity values has increased the number of false positives further. In all classes except the 'ground' and 'natural,' the number of points falsely labeled as pole has increased. The number of true positives is increased by 2172, and the number of pole points labeled as ground, buildings, barrier, and natural decreases. However, the true positives for classes like building and barrier have decreased significantly by 3959 and 7252 points, respectively. Also, the number of barrier points labeled as building has increased by 5300.

Table 10 - Difference of Confusion Matrices : The effect of including intensity values



As shown in figure 32 and 33, On the one hand, the signs of the poles are highly reflective. On the other hand, the incident angle and surface orientation have led the building or barrier's parts to return high-intensity laser beams. The intensity values are highly correlated with the incident angle, which means the relative position of the scanner and the surface orientation would affect the reflectance of each point(Gross et al., 2008). Therefore, to extract the reflectance values that are only influenced by the surface material, the trajectory of the scanner is needed. Since the trajectory data is not provided, the KPConv author has used a normalization method that removes the difference in intensity features between the classes on a global scale. However, in the local neighborhood, the intensity values are distinctive enough to influence the model performance. The normalized intensity of the tiles is illustrated to emphasize that the intensity values are noisy and are confusing for the model.

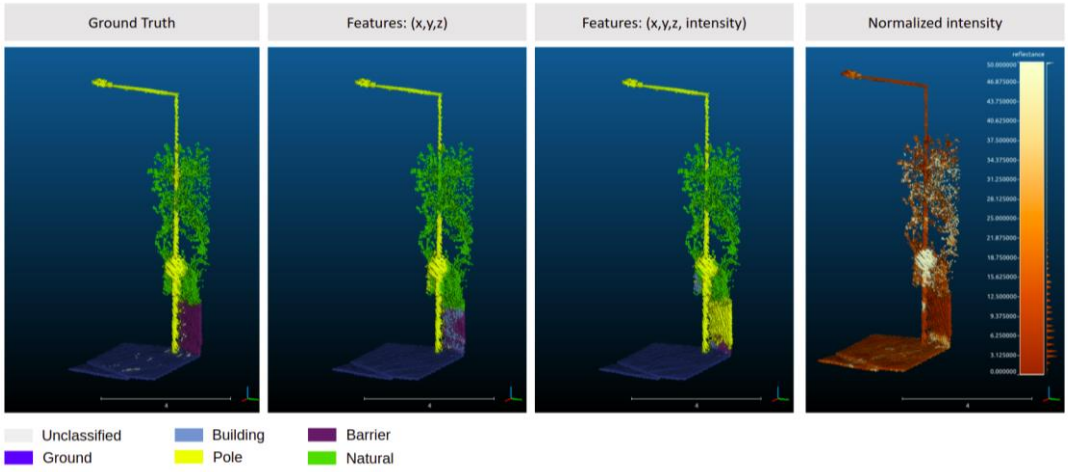


Figure 32 - Comparing the part segmentation results with and without itensity values

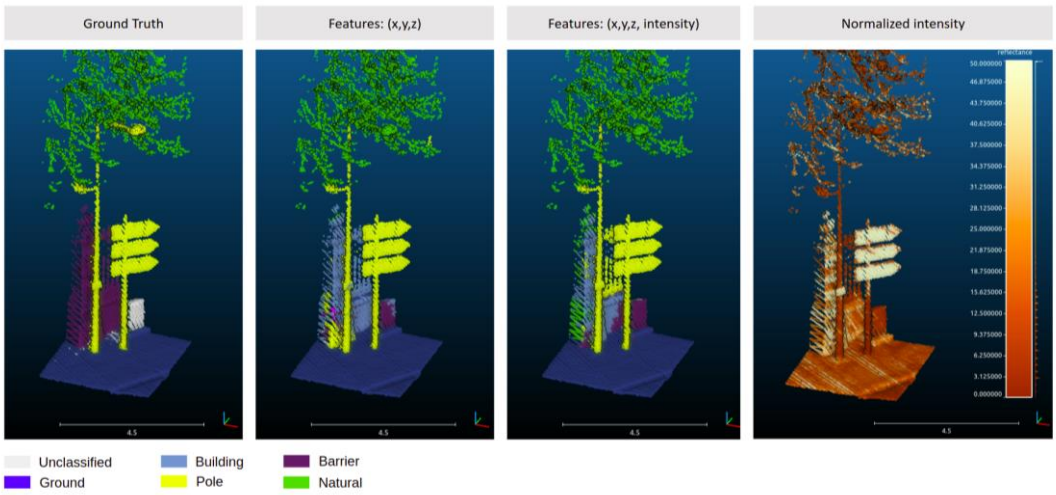


Figure 33- Comparing the part segmentation results with and without itensity values

The comparison of the evaluation metrics (figure 34) also shows that including the reflectance feature in the pipeline increases the recall compared to using only coordinate values to train and predict. However, the precision, F1, and IoU will decrease, which stems from the increase in the false positives and the increased confusion between other classes.

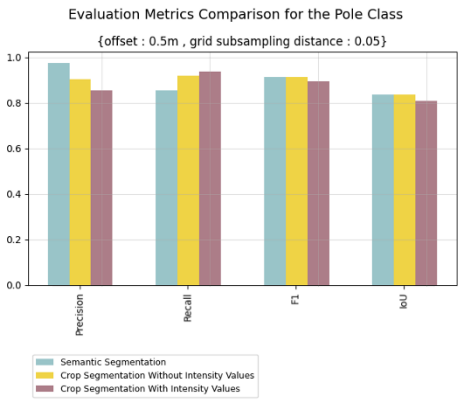


Figure 34 - Comparison of Evaluation Metrics between the output of models trained with and without intensity value

5.5. Integrating the two-step pipeline

In all the previous experiments with the part segmentation, the pole tiles' extents were extracted based on the ground truth. The main reason behind that choice was to trace the part segmentation performance on all possible pole neighborhood scenarios and reach the maximum potential of the proposed pipeline. However, in a real-world scenario, where pole neighborhood tiles should be extracted from the initial semantic segmentation prediction, the pipeline's complexities will increase and affect its performance. As can be seen in the figure 35 , the predictions for the class pole resulting from the semantic segmentation can be categorized as follows:

Entirely true positives: The whole pole is perfectly segmented (entirely green poles in figure 35 These poles do not need refinement in the further step, but due to the absence of ground truth in real-world data, they cannot be recognized to be removed from the pipeline. Therefore, the pipeline goes through a repetitive computation for these poles and generates the same result for the second time. A similar result achieved in the experiments with the part segmentation on the cropped tiles will be achieved on these poles.

Entirely false negatives: The whole pole is missed (entirely white poles) in the semantic segmentation. As discussed earlier, these poles are eliminated and would not proceed to the part segmentation step. No further improvement can be expected for these poles, which is one of the limitations of the designed pipeline. It can be said that since the input to the part segmentation step is based on an initial prediction of the semantic segmentation, the error will be propagated from throughout the pipeline.

Entirely false positives: The poles in warm colors (yellow to brown) in figure 35, are labeled as poles in the semantic segmentation step but belong to another class. These poles cannot be identified or eliminated in real-world scenarios where no ground truth is available. By proceeding to the part segmentation step, as the model is trained to find a pole inside the tile, they will be assigned to the pole class and preserved.

Partly missed, partly detected: The partially white and partially green poles are not entirely segmented with the semantic segmentation task. Because of the correctly predicted labels, the unsegmented points in the neighborhood will be passed to the next step, where the proposed pipeline could effectively improve the predictions. However, these instances are rare in the NPM3D dataset. Therefore, the improvement might not be significant numerically.

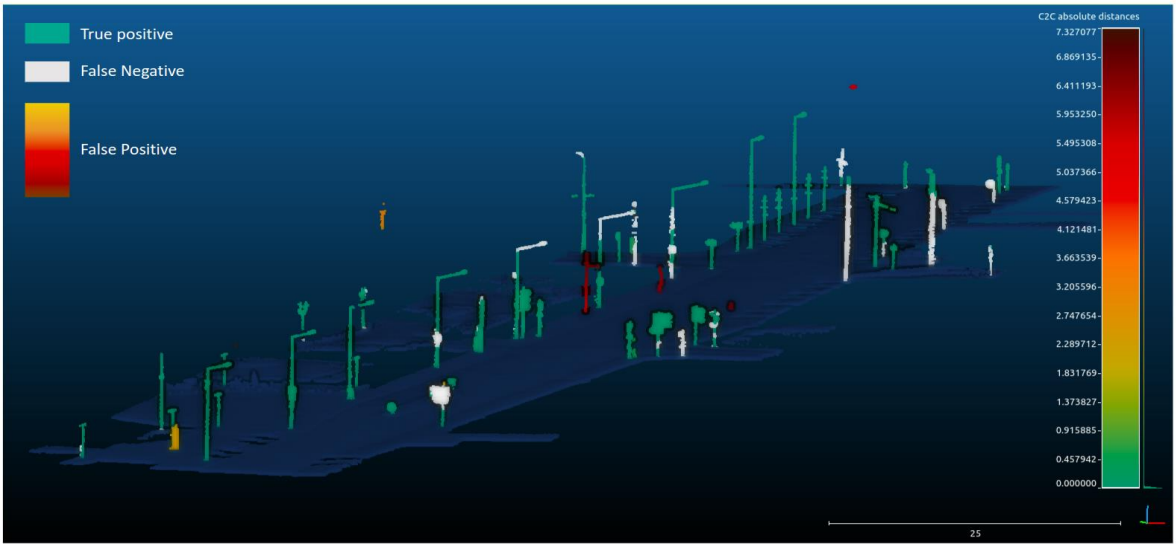


Figure 35 - Pole prediction scenario in semantic segmentation

Complexity (1) - Nested Bounding Boxes

When using semantic segmentation results to find the pole neighborhoods, the first complexity that emerges is a single pole broken into multiple clusters (figure 36) This results in creating multiple tiles for an individual pole in the same area. The repetitive tiles add some deficiencies to the pipeline: (a) they lead to extra unnecessary computations, which increase the time and memory for completing the task. (b) multiple prediction tiles will be generated for points located at the intersection of the tiles, which might be contradictory. In that case, extracting the final vote from multiple predictions is required, which adds to the complexity of the pipeline. (c) repeating the same tile can distort the quantitative result because any improvement or degradation will be included multiple times in the confusion matrix. To resolve this issue, the bounding boxes that are completely within another bounding box are eliminated as shown in figure 36 . In our test point clouds, after eliminating the small bounding boxes, the number of tiles reduced from 65 to 58 and from 42 to 34.

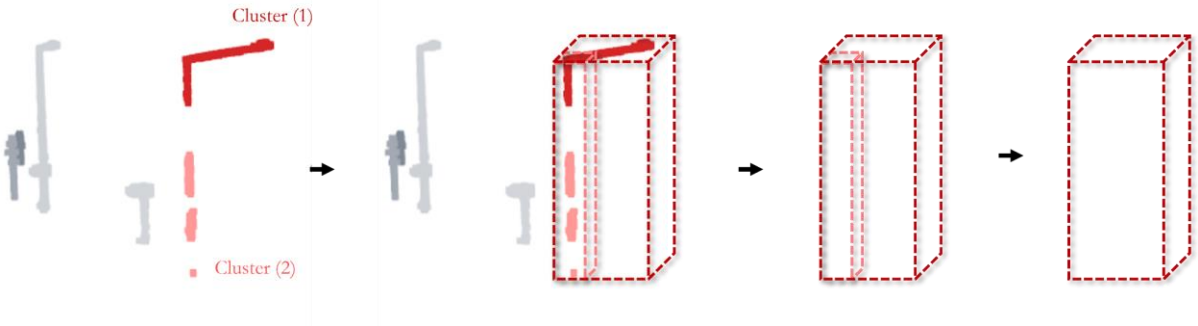


Figure 36 - Nested Bounding Boxes Problematic

Complexity (2) - offset value

In the previous experiments in which the pole neighborhoods were extracted based on the ground truth, the offset value was chosen so that the part segmentation model could handle it to segment the poles properly. However, with the imperfect semantic segmentation results, as only some parts of the pole might be segmented, it cannot be guaranteed that the whole pole is captured within the bounding box. Therefore, the offset value to enlarge the bounding box of pole clusters gains a crucial role in passing the entire pole to the next step. Fine-tuning the offset value is dependent on the dataset and the quality of the semantic segmentation result. For this dataset and algorithm, a 1m offset works properly. However, in some cases, an unsegmented part of the pole is close to tile edges, and there is a high chance of being eliminated during the padding cropping stage. Achieving the best result by including the entire tiles extent confirms that the poor performance is caused by eliminating pole parts in the padding removal phase.

Following the pipeline described in section 4.4. the final result for the pole class is as shown in table 11 . The results are reported in two scales: at the tile level and at the scene level after inserting the tiles back into the scene. As can be seen, the precision, F1 score, and IoU metrics are degraded after the part segmentation in the tile level, but the recall is increased by 3%. Similarly, the recall is increased but only 0.6% at the scene level. The F1 and IoU did not change significantly for the pole class, but the precision is the most affected evaluation metric in both levels. Precision is decreased by 1.2% at the scene level, but at the tile level, it is declined by 5.3%.

Table 11 - Evaluation metrics for the pole class after integration of two steps

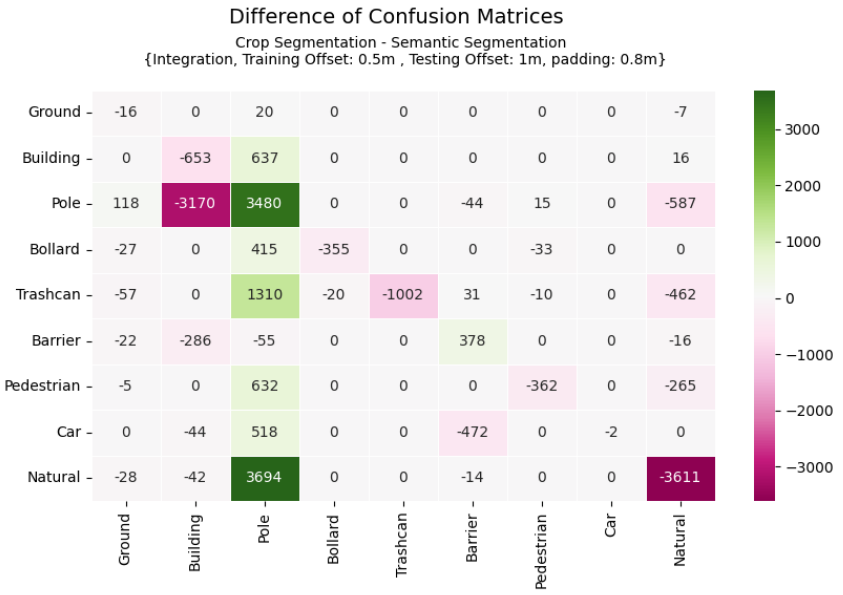
Evaluation Metric	Pole Class (Scene Level)		Pole Class (tile level)	
Task	**Sem.**	**Part.**	**Sem.**	**Part.**
Precision	86.1%	84.9%	95.2%	89.9%
Recall	73.0%	73.6%	90.5%	93.5%
F1	79%	78.9%	92.8%	91.7%
IoU	65.3%	65.2%	86.6%	84.7%

Semantic Segmentation (baseline) | Improvement | Decline | No Significant Change

- Sem = Semantic Segmentation
- Part = Part Segmentation

In table 12 , the bias of training set toward the pole class has affected the predictions. As can be seen from the differences of confusion matrices, the green cells along the pole column highlight the huge confusion and false-positive rate in the prediction. The points belonging to the natural class and trashcans are labeled as poles after the part segmentation.

Table 12 - Difference of Confusion Matrices : The effect of integrating two steps



the visual inspection of the results shows that the part segmentation model cannot resolve the small noises of the semantic segmentation prediction. As shown in figure 37, a tiny cluster of falsely labelled poles is turned to a large cluster of false positives after the part segmentation. It can be said that the part segmentation successfully expands the seed prediction offered by the semantic segmentation, but it cannot help eliminate the false seeds.

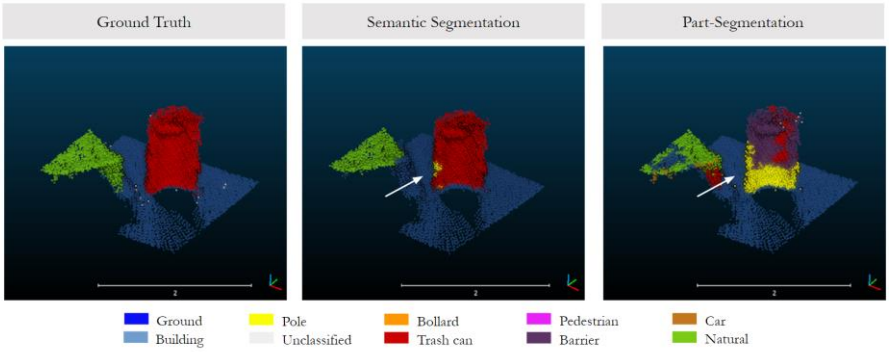


Figure 37 - Incapability of the part segmentation to correct false positives

6. CONCLUSION AND FUTURE WORK

Intending to achieve a better segmentation result for urban assets in a scene, we designed and tested a two-step pipeline by combining semantic segmentation and part segmentation in this research. In the proposed method, first, a deep learning model (KPConv) is trained for the semantic segmentation task using a publicly available mobile laser scanning dataset, NPM3D. Then, the trained model is tested on the test point cloud, generating an initial prediction. With the motivation of enhancing the results for the chosen asset of interest(pole), the neighborhoods around the poles are cropped into tiles and fed into the same deep learning model trained for the part segmentation task with similar tiles cropped from the training scene.

The results show that part segmentation can improve the partly missed poles and segment their entire shape with a more accurate boundary than semantic segmentation. However, the instances were rare in the NPM3D dataset, and their improvement did not reflect significantly in the semantic segmentation results. The key limitations of the proposed pipeline are the error propagation, the model bias towards the class of interest, the inconsistency of the performance in tiles of different difficulty, and parameter fine-tuning. Also, the proposed pipeline's effectiveness depends on the object geometry, and it cannot be used for extended ones. Overall, the results show that the pipeline's accomplishment is highly dependent on the tile definition. If all the poles are captured with enough contextual information, a 6.4% improvement in IoU can be achieved at the tile level and 1.6% at the scene level for the pole class. However, the part segmentation could not improve the semantic segmentation results in the real-world scenario. The instability, discontinuity, incompleteness, and noise in the semantic segmentation prediction result in inefficient tile selection, which significantly affects the part segmentation results.

6.1. Contributions

The main contribution of this work is to design a novel and simple pipeline that combines two 3D understanding tasks (part segmentation and semantic segmentation) to yield more refined results for the semantic segmentation task. The cases where the part segmentation could improve the initial prediction from the semantic segmentation

can be considered an important contribution applicable selectively for extracting more accurate results for urban asset management. This study also reveals the complexities of the problem at hand, discusses the edge cases, and investigates the possibilities of making the pipeline work.

6.2. Limitations

Error Propagation: In this pipeline, the final prediction is based on an initial prediction. Although the part segmentation aims to make the semantic segmentation predictions more accurate, some errors will be propagated. For instance, if no pole part is detected in the semantic segmentation phase, then the pole instance will be eliminated, and the part segmentation task will not have the opportunity to find the points belonging to the pole class. Or, in another scenario, where two clusters of points belonging to a single pole instance are generated, each of them will be captured in a different tile, and with the elimination of edge predictions, the entire pole can never be captured.
Lack of Contextual Information: Cropping discards the contextual information and changes the geometry of the classes. Tile selection results in the appearance of floating points clusters in the tile representing an incomplete object. In many cases, it is impossible even for the human brain to identify them.

Dependence on Object Geometry: The usability of the method for a particular class is dependent on the class geometry. The pipeline cannot be used for extended objects like buildings, barriers, guardrails, or sidewalks. The part segmentation trained model, with enough training examples, is able to segment a part of these extended objects, but it cannot capture the whole object at once because the part segmentation does not work properly for large tiles.

Repetitive calculations: The proposed pipeline suffers repetitive calculations, particularly when it has to segment easy poles twice, once in the whole scene and once from its neighborhood, generating the same output. Also, the padding removal after the part segmentation can be considered another waste of time and memory since the prediction is generated but is not used and removed finally.

Inconsistent Performance: The inconsistency in the performance of part segmentation in tiles of different difficulty makes it impossible to generalize the effectiveness of the part segmentation step for any given tile. Some tiles have better predictions with some adaptations of the method, while others do not work properly with that experiment setting. Many influential factors can affect the predictions and make them unreliable.

Parameter fine-tuning: The pipeline requires some parameter fine-tuning that can be highly data-dependent and not straightforward, particularly when ground truth labels are unavailable. For example, if the point cloud is captured from the historical and touristic part of the city with many poles and guidance signs close to each other, the parameters should limit the clusters so that single poles can be extracted. While in the modern parts of the city or highways where the pole-like objects are located far from each other, the parameters can be loosened. All in all, the range and extent of these parameters should be investigated based on the project context and available data.

6.3. Answers to the research questions

1.1. What are the challenges of the prediction in the scene semantic segmentation step? And why are they happening?

The first challenge is the intrinsic **class imbalance** of the urban scene leading to a biased prediction in favor of well-represented classes (such as ground and buildings) and relatively poorer IoU (smaller by 20% and more) in the prediction cloud for minority classes like poles, bollards, trashcan, and barrier compared to majority classes. The second challenge is the **data quality** in terms of captured points, point features, and labeling. The publicly available dataset (NPM3D) is not well-processed; the data suffers from noisy points, especially where two classes are located close to one another, and the floating noise points connect the two classes. Furthermore, the existing anisotropic patterns have led to point density variance within a local neighborhood. Moreover, the reflectance values are not normalized to compensate for the incident angle effect; therefore, they are highly noisy and do not represent the surface material properly. Also, **poor labeling choices** have led to substantial inter-class similarities. As the geometric characteristics of some instances from a particular class (like green walls) are close to the general geometry of another class (like building), and in the absence of high-quality, distinctive features like intensity and color information, the model confuses these classes easily.

1.2. How good is the scene semantic segmentation prediction for the class pole? What challenges should be overcome in the part segmentation step?

The poles are segmented with the intersection of union **(IoU) of 65.3%**, which is relatively low compared to majority classes like building and ground with more than 90% IoU. The visual interpretation of the results shows that some poles are **completely missed,** particularly those connected or close to the buildings. The reason is that the model confuses the pole with facade extensions (such as store signs, flower boxes, attached lamps, and columns). Some other poles are **partly missed** where pole intersects with trees or are located close to barriers or bushes. Some parts of the bollards are also **confused** with poles due to their **similar geometry** (vertical upright cylinders). Another observation is that at the edges of the point cloud, where there **is less point density**, the semantic segmentation prediction for poles becomes unstable, and the model confuses the pole points with other classes like building and natural.

However, the part segmentation step is only effective for partly missed poles since the model biased towards the pole class intensifies the false positive rate and cannot refine the segmentation of geometrically similar objects. Also, In other problematic scenarios where the pole is totally missed, the part segmentation does not get the chance to improve the results as the cases are being eliminated.

2.1. How does the part segmentation step influence the initial result? What parameters can affect the effectiveness of the part segmentation step?

In the ideal scenario, where all the poles are adequately captured within the tiles, the part segmentation can improve the IoU by 6.4% and reach 92.3% for the class pole. However, the integration results show that in the real-world scenario, the part segmentation cannot improve the semantic segmentation results. The recall improves by 3%, but at the same time, the precision, IoU, and F1 score will worsen. Based on the experiments in this work, the tile size and its capability to capture the whole pole, the data quality, the point density, the share of pole points in the training set, and the tile difficulty level influence the part segmentation result.

2.2. What are the limitations of the proposed pipeline?

It is discussed in section 6.2.

2.3. Can any additional step in the pipeline (preprocessing or postprocessing) improve the part segmentation outcome?

In this project, two preprocessing initiatives were Each initiative has its own limitations. The reflectance records were too noisy and required corrections related to the incidence angle. Therefore it could not improve the results of this project. However., maybe a high-quality intensity data could enhance the results of the part segmentation. The quantitative results on enlarged tiles show that larger tiles cannot be handled by part segmentation as they encompass more classes, and the class imbalance is reproduced and benefits classes like building and ground. The postprocessing step of padding removal (eliminating the uncertain edge predictions) enhanced the pole IoU on the tile level by more than 8.5%.

3.1. What other complexities emerge when the two methods are integrated together?

The main bottleneck in the integration of two pipelines is the tile selection. In some cases, only a focused point cluster (e.g., the light bulb) is segmented in the semantic segmentation step. In these cases, setting an offset parameter that could capture the whole pole in all cases but still be small enough to be properly handled by the part segmentation task is a challenge. Also, the discontinuity in the poles' prediction results in the generation of multiple bounding boxes around a single pole. However, many of these cases can be handled by eliminating the small bounding boxes located inside the bigger ones. However, some edge cases exist where multiple pole-like objects are located nearby. In these cases, their bounding boxes overlap considerably, but none cover the other. These cases will remain in the dataset and be calculated multiple times in a similar location.

3.2. How will the results get affected by this?

The results are significantly affected. The minor improvements that were achieved in the ideal scenario have turned into degradation in the real-world scenario. When part segmenting a perfect neighborhood containing a pole-like object, the model can segment that pole with acceptable performance. But with the imperfect tiles encompassing any random class, the model, greedily looking for the pole points, will assign pole labels to the points belonging to other classes and decrease the precision significantly. Therefore, The IoU and F1 got slightly worse (between 1% to 2%) compared to the semantic segmentation result. But the precision was reduced by more than 5% on the tile level after the part segmentation.

3.3. Future Work

For further research in this direction, we recommend designing a pipeline to determine the difficulty level of a 3D tile. Finding a method (even with the help of deep learning) to determine whether a tile can be improved with an additional part segmentation step or not can help optimize our proposed pipeline. Also, it is an interesting line of research for the industry as it has applications in reducing labeling efforts. By identifying hard instances that cannot be improved further with a deep learning algorithm, the human operator is guided to focus on labeling the hard instances and leave out the rest for labeling using smart methods.

Another possible direction is designing a synthetic dataset that could train the part segmentation model with edge cases. Since the variety of hard cases exist, and their perturbation is high, maybe a designed artificial dataset can be added to the training set to make the model familiar with the complex scenarios.

LIST OF REFERENCES

Bellinger, C., Corizzo, R., & Japkowicz, N. (2020). *ReMix: Calibrated Resampling for Class Imbalance in Deep learning*. http://arxiv.org/abs/2012.02312

Bello, S. A., Yu, S., Wang, C., Adam, J. M., & Li, J. (2020). Review: Deep learning on 3D point clouds. In *Remote Sensing* (Vol. 12, Issue 11). MDPI AG. https://doi.org/10.3390/rs12111729

Bloembergen, D., & Eijgenstein, C. (2021). *Automatic labelling of urban point clouds using data fusion*. http://arxiv.org/abs/2108.13757

Boogaard, F. P., van Henten, E. J., & Kootstra, G. (2022). Improved Point-Cloud Segmentation for Plant Phenotyping Through Class-Dependent Sampling of Training Data to Battle Class Imbalance. *Frontiers in Plant Science*, *13*. https://doi.org/10.3389/fpls.2022.838190

Boulch, A. (2020). ConvPoint: Continuous convolutions for point cloud processing. *Computers and Graphics (Pergamon)*, *88*, 24–34. https://doi.org/10.1016/j.cag.2020.02.005

Chen, X., Li, J., Huang, S., Cui, H., Liu, P., & Sun, Q. (2021). An automatic concrete crack-detection method fusing point clouds and images based on improved otsu's algorithm. *Sensors*, *21*(5), 1–19. https://doi.org/10.3390/s21051581

Chen, Y., Xiong, Y., Zhang, B., Zhou, J., & Zhang, Q. (2021). 3D point cloud semantic segmentation toward large-scale unstructured agricultural scene classification. *Computers and Electronics in Agriculture*, *190*. https://doi.org/10.1016/j.compag.2021.106445

Dong, Q., Gong, S., & Zhu, X. (2019). Imbalanced Deep Learning by Minority Class Incremental Rectification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(6), 1367–1381. https://doi.org/10.1109/TPAMI.2018.2832629

Griffiths, D., & Boehm, J. (2019). Weighted point cloud augmentation for neural network training data class-imbalance. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(2/W13), 981–987. https://doi.org/10.5194/isprs-archives-XLII-2-W13-981-2019

Grilli, E., Menna, F., & Remondino, F. (2017). A review of point clouds segmentation and classification algorithms. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, *42*(2W3), 339–344. https://doi.org/10.5194/isprs-archives-XLII-2-W3-339-2017

Gross, H., Jutzi, B., & Thoennessen, U. (2008). *INTENSITY NORMALIZATION BY INCIDENCE ANGLE AND RANGE OF FULL-WAVEFORM LIDAR DATA*.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2020). Deep Learning for 3D Point Clouds: A Survey. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* (Vol. 43, Issue 12, pp. 4338–4364). IEEE Computer Society. https://doi.org/10.1109/TPAMI.2020.3005434

He, H., Khoshelham, K., & Fraser, C. (2020). A multiclass TrAdaBoost transfer learning algorithm for the classification of mobile lidar data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *166*, 118–127. https://doi.org/10.1016/j.isprsjprs.2020.05.010

Hua, B.-S., Tran, M.-K., & Yeung, S.-K. (2018). *Pointwise Convolutional Neural Networks*. http://arxiv.org/abs/1712.05245

Johnson, J. M., & Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, *6*(1). https://doi.org/10.1186/s40537-019-0192-5

Karlsson, R. (2014). Implementation of Advanced Monitoring Techniques in Road Asset Management-Results from the TRIMM project. In *Transport Research Arena*.

Klokov, R., & Lempitsky, V. (2017). *Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models*. http://arxiv.org/abs/1704.01222

Krawczyk, B. (2016). Learning from imbalanced data: open challenges and future directions. In *Progress in Artificial Intelligence* (Vol. 5, Issue 4, pp. 221–232). Springer Verlag. https://doi.org/10.1007/s13748-016-0094-0

Landa, J., & Prochazka, D. (2014). Automatic Road Inventory Using LiDAR. *Procedia Economics and Finance*, *12*, 363–370. https://doi.org/10.1016/s2212-5671(14)00356-6

Lazarek, J., & Pryczek, M. (2018). A Review on Point Cloud Semantic Segmentation Methods. *JOURNAL OF APPLIED COMPUTER SCIENCE*, *26*(2), 99–105. https://doi.org/10.34658/jacs.2018.26.2.99-106

Li, H. T., Todd, Z., Bielski, N., & Carroll, F. (2021). 3D lidar point-cloud projection operator and transfer machine learning for effective road surface features detection and segmentation. *Visual Computer*. https://doi.org/10.1007/s00371-021-02103-8

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., & Chen, B. (2018). *PointCNN: Convolution On $\mathcal{X}$-Transformed Points*. http://arxiv.org/abs/1801.07791

Li, Y., Ma, L., Zhong, Z., Liu, F., Chapman, M. A., Cao, D., & Li, J. (2020). Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. *IEEE Transactions on Neural Networks and Learning Systems*, 1–21. https://doi.org/10.1109/tnnls.2020.3015992

Lin, H. I., & Nguyen, M. C. (2020). Boosting minority class prediction on imbalanced point cloud data. *Applied Sciences (Switzerland)*, *10*(3). https://doi.org/10.3390/app10030973

Liu, W., Sun, J., Li, W., Hu, T., & Wang, P. (2019). Deep learning on point clouds and its application: A survey. In *Sensors (Switzerland)* (Vol. 19, Issue 19). MDPI AG. https://doi.org/10.3390/s19194188

Lu, H., & Shi, H. (2020). *Deep Learning for 3D Point Cloud Understanding: A Survey*. http://arxiv.org/abs/2009.08920

Ma, L., Li, Y., Li, J., Wang, C., Wang, R., & Chapman, M. A. (2018). Mobile laser scanned point-clouds for road object detection and extraction: A review. In *Remote Sensing* (Vol. 10, Issue 10). MDPI AG. https://doi.org/10.3390/rs10101531

Milioto, A., Behley, J., McCool, C., & Stachniss, C. (2020). LiDAR panoptic segmentation for autonomous driving. *IEEE International Conference on Intelligent Robots and Systems*, 8505–8512. https://doi.org/10.1109/IROS45743.2020.9340837

Peraka, N. S. P., & Biligiri, K. P. (2020). Pavement asset management systems and technologies: A review. In *Automation in Construction* (Vol. 119). Elsevier B.V. https://doi.org/10.1016/j.autcon.2020.103336

Poliyapram, V., Wang, W., & Nakamura, R. (2019). A point-wise LiDAR and image multimodal fusion network (PMNet) for aerial point cloud 3D semantic segmentation. *Remote Sensing*, *11*(24). https://doi.org/10.3390/rs11242961

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2016). *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. http://arxiv.org/abs/1612.00593

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. http://arxiv.org/abs/1706.02413

Sairam, N., Nagarajan, S., & Ornitz, S. (2016). Development of Mobile Mapping System for 3D Road Asset Inventory. *Sensors (Switzerland)*, *16*(3). https://doi.org/10.3390/s16030367

Sander, R. (2020). *Sparse Data Fusion and Class Imbalance Correction Techniques for Efficient Multi-Class Point Cloud Semantic Segmentation*. https://doi.org/10.13140/RG.2.2.12077.03042

Shinde, P., & Shah, S. (2018). *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*.

Thomas, H. (2020). *Rotation-Invariant Point Convolution With Multiple Equivariant Alignments*. http://arxiv.org/abs/2012.04048

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., & Guibas, L. J. (2019). *KPConv: Flexible and Deformable Convolution for Point Clouds*. http://arxiv.org/abs/1904.08889

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., & Solomon, J. M. (2019). Dynamic graph Cnn for learning on point clouds. *ACM Transactions on Graphics*, *38*(5). https://doi.org/10.1145/3326362

Weinmann, M., Jutzi, B., Hinz, S., & Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, *105*, 286–304. https://doi.org/10.1016/j.isprsjprs.2015.01.016

Weinmann, M., Jutzi, B., & Mallet, C. (2013). Feature relevance assessment for the semantic interpretation of 3D point cloud data. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *2*(5W2), 313–318. https://doi.org/10.5194/isprsannals-II-5-W2-313-2013

Xie, Y., Tian, J., & Zhu, X. X. (2020). Linking Points with Labels in 3D: A Review of Point Cloud Semantic Segmentation. In *IEEE Geoscience and Remote Sensing Magazine* (Vol. 8, Issue 4, pp. 38–59). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/MGRS.2019.2937630

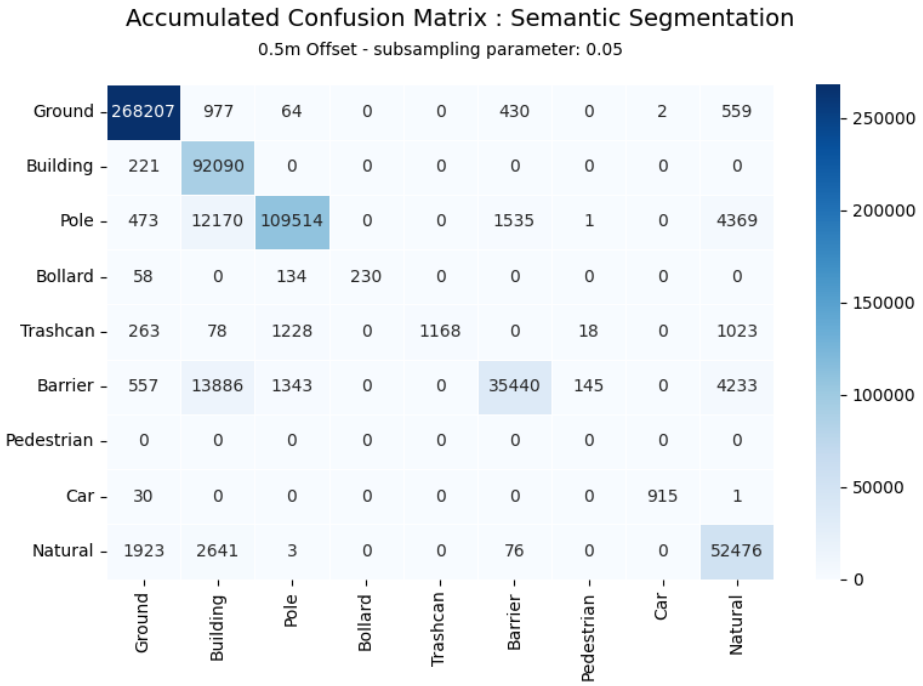
Yousefimashhoor, S. (2022). *Pole-like objects part-segmentation*.

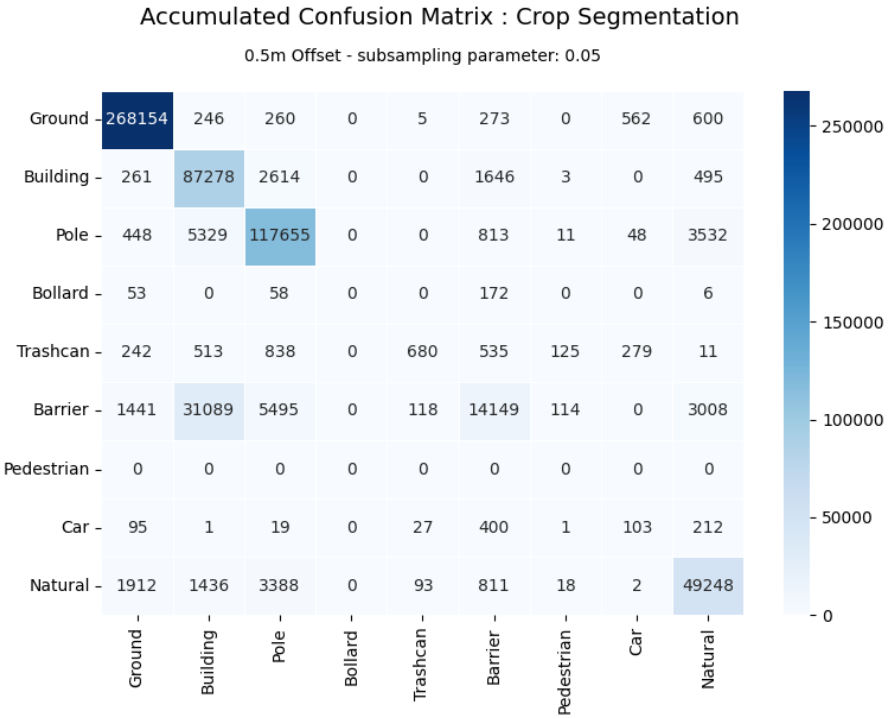
Yu, Y., Li, J., Guan, H., & Wang, C. (2015). Automated Extraction of Urban Road Facilities Using Mobile Laser Scanning Data. *IEEE Transactions on Intelligent Transportation Systems*, *16*(4), 2167–2181. https://doi.org/10.1109/TITS.2015.2399492

Zhang, J., Zhao, X., Chen, Z., & Lu, Z. (2019). A Review of Deep Learning-Based Semantic Segmentation for Point Cloud. In *IEEE Access* (Vol. 7, pp. 179118–179133). Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/ACCESS.2019.2958671

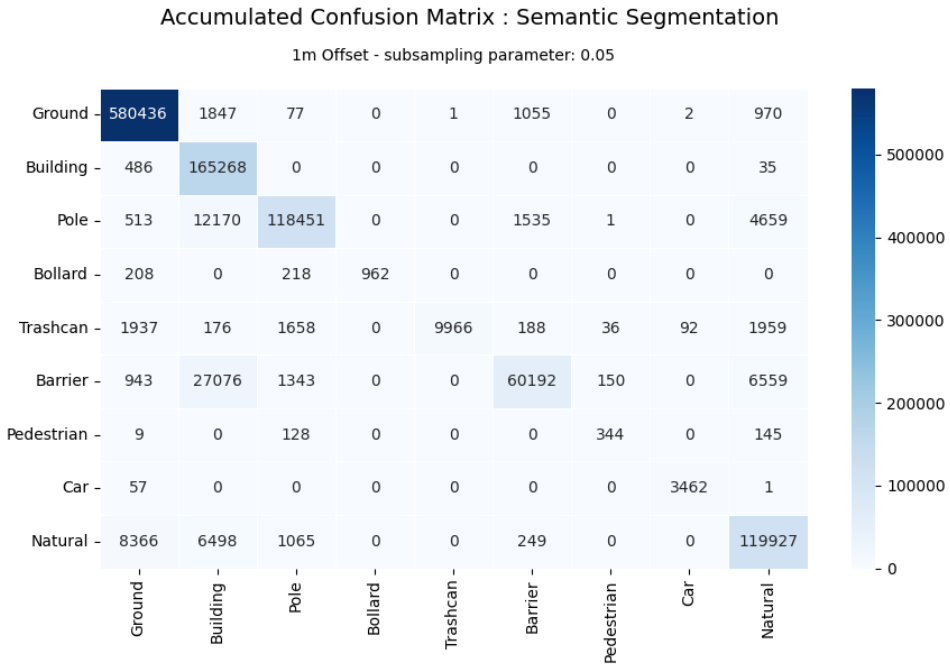
APPENDIX (1) : CONFUSION MATRICES

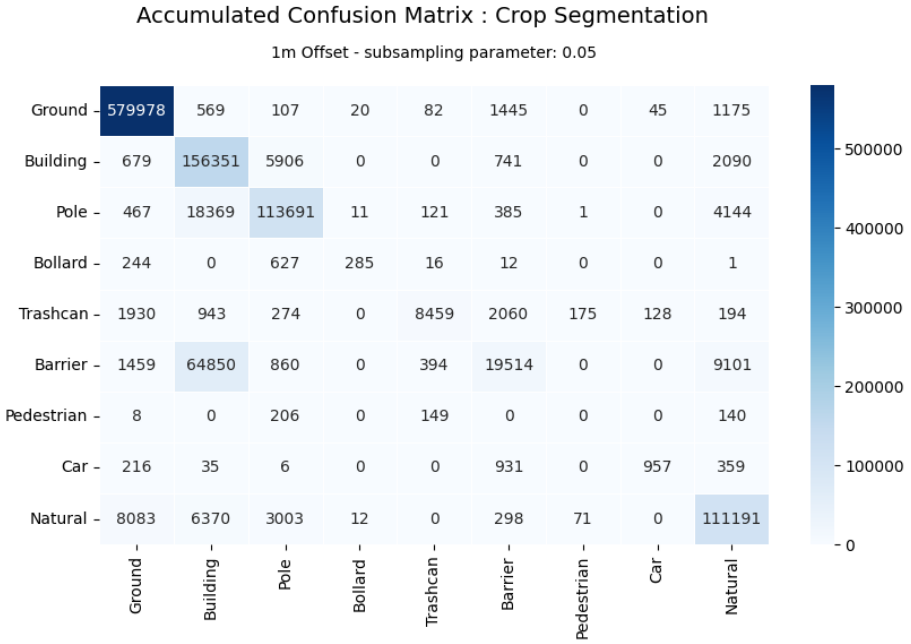
Semantic Segmentation and Part Segmentation on Tiles of 0.5m offset



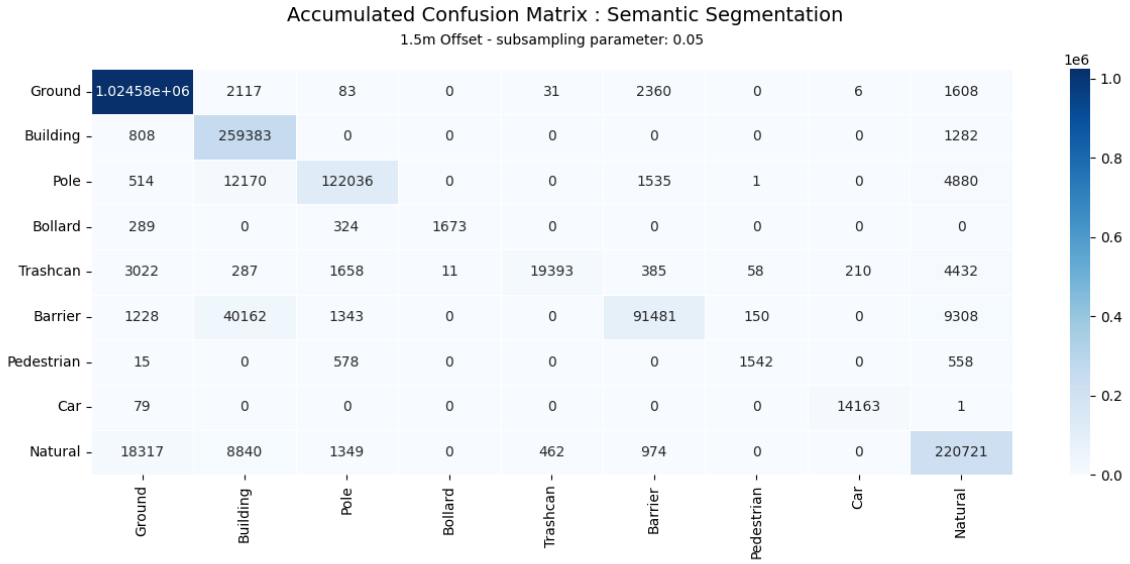


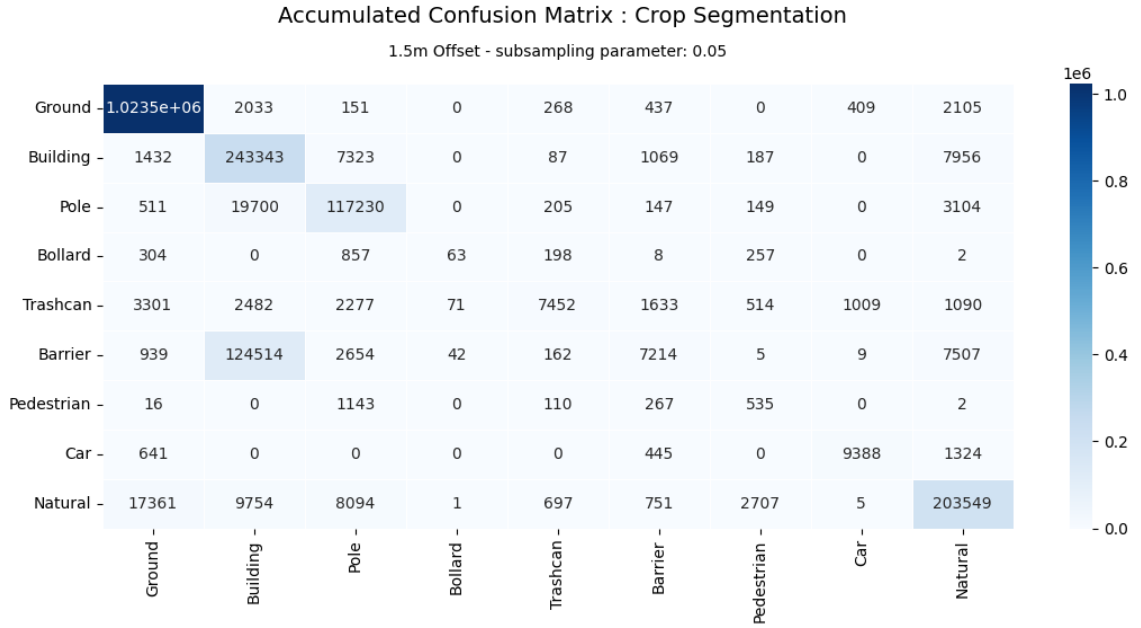
Semantic Segmentation and Part Segmentation on Tiles of 1m offset



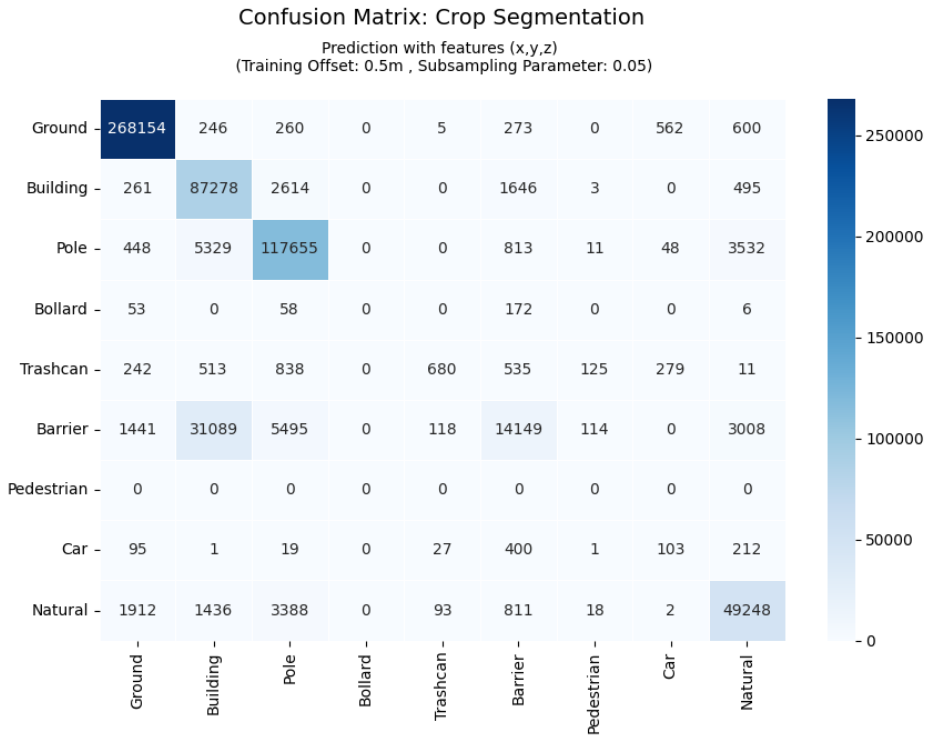


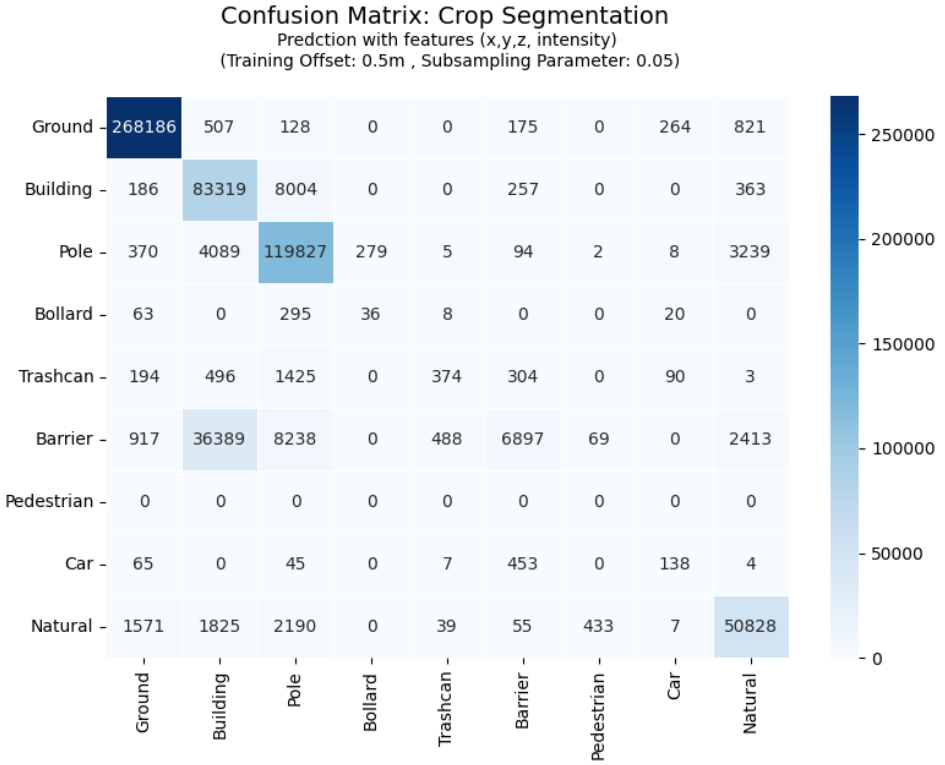
Semantic Segmentation and Part Segmentation on Tiles of 1.5m offset



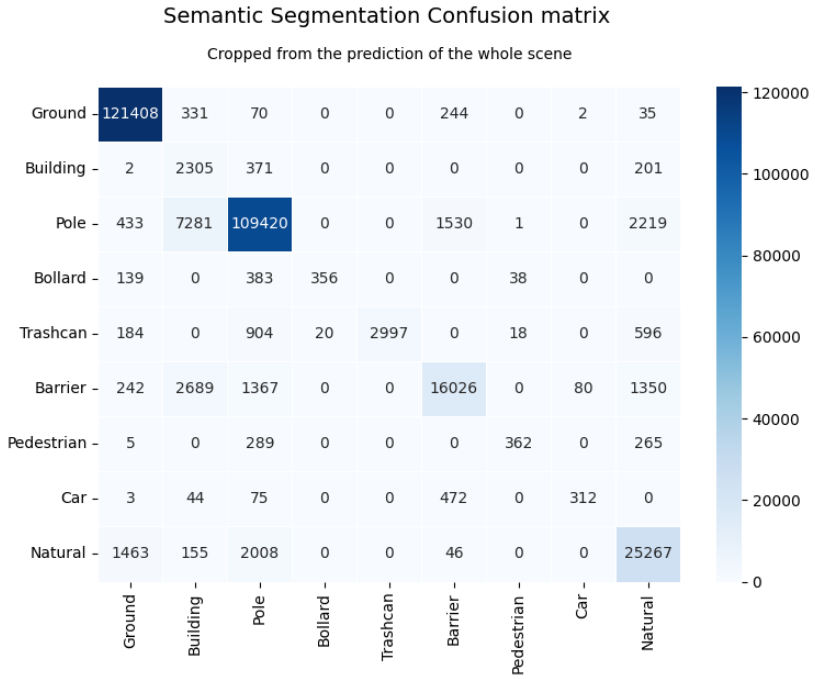


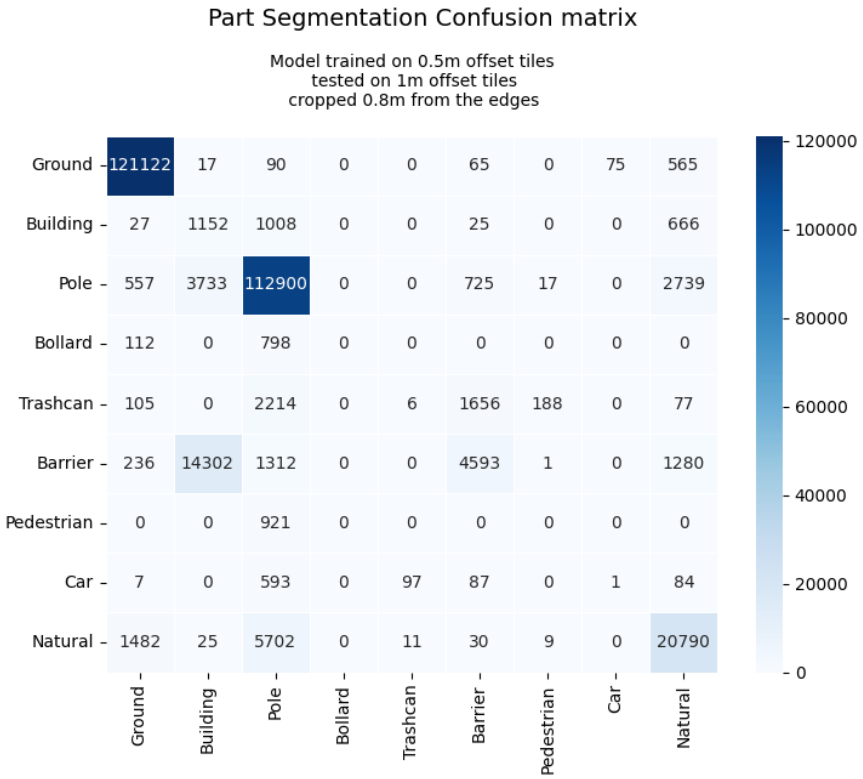
Semantic Segmentation and Part Segmentation (With and Without Intensity Values)





Integration of Semantic Segmentation and Part Segmentation
Trained on tiles of 0.5m offset
Tested on tiles of 1m offset
After padding removal 0.8m





APPENDIX (2) : IOU COMPARISON

In this table, the results of part segmentation in the tile level is compared to the results with padding removal (cropping after the part segmentation), and finally, the ideal result is inserted back into the scene, and the semantic segmentation result after the modification is reported. Note that the semantic segmentation (baseline result) is calculated within the effective tile extent in each scenario, which is why the numbers differ.

Neighborhood Offset	Point Cloud	Cropping size after part segmentation (from the edges)	Task	IoU per Class								
				Ground	Building	Pole	Bollard	Trashcan	Barrier	Pedestrian	Car	Natural
Training : 0.5m Testing: 0.5m	Tiles	0 m (Full Extent)	semantic segmentation	98.0%	75.4%	83.7%	54.6%	30.9%	61.4%	0%	96.5%	77.9%
		0 m (Full Extent)	part segmentation	97.6%	66.7%	83.7%	0%	19.6%	23.5%	0%	5.9%	76.1%
Training: 0.5m Testing: 0.5m	Cropped tiles	0.8 m	semantic segmentation	99.6%	0%	85.9%	0%	0%	43.9%	0%	0%	83.5%
		0.8 m	part segmentation	99.2%	0%	92.3%	0%	0%	13.4%	0%	0%	80.3%
Training: 0.5m Testing: 0.5m	Cropped tiles	0.5 m	semantic segmentation	99.0%	32.7%	84.5%	0%	4.1%	62.3%	0%	100%	72.4%
		0.5 m	part segmentation	98.8%	29.8%	90.7%	0%	12%	16.2%	0%	0%	77.8%
Training: 0.5m Testing: 1m	Cropped tiles	0.8 m	semantic segmentation	98.8%	68.1%	84.2%	0%	12.1%	63.7%	0%	97.6%	78.3%
		0.8 m	part segmentation	98.2%	49.7%	85.7%	0%	3.9%	12.2%	0%	14.3%	75.3%
Training: 0.5m Testing: 0.5m	Whole Scene	-	semantic segmentation	97.8%	93.6%	65.3%	70.5%	70.1%	42.6%	85.5%	96.3%	90.2%
		0.8 m	after replacing the predicted cropped tiles in the scene	97.8%	93.6%	65.7%	70.5%	70.1%	42.6%	85.5%	96.3%	90.3%
		0.5 m	after replacing the predicted cropped tiles in the scene	97.8%	93.6%	66.7%	70.5%	70.1%	42.6%	85.5%	96.3%	90.3%
Training 0.5m Testing: 1m	Whole Scene	0.8 m	after replacing the predicted cropped tiles in the scene	97.8%	93.6%	66.9%	70.5%	70.1%	42.6%	85.5%	96.3%	90.3%

Semantic Segmentation (baseline)
Part Segmentation
Combining Semantic Segmentation and Part Segmentation

Scene level: no improvement — Maximum improvement

Tile level: no improvement — Maximum improvement