Thesis Internship Report

Automatic Locating of Suspended Streetlights and Overhead Cables in Urban Point Clouds

by

Falke Boskaljon

First supervisor: Vincent François-Lavet
Second reader: Sandjay Bhulai
Daily supervisor I: Chris Eijgenstein (Gemeente Amsterdam)
Daily supervisor II: Daan Bloembergen (Gemeente Amsterdam)

Automatic Locating of Suspended Streetlights and Overhead Cables in Urban Point Clouds

Falke Boskaljon

Thesis Internship Report

Vrije Universiteit Amsterdam
Faculty of Science, Business Analytics
De Boelelaan 1081a, 1081 HV Amsterdam

Host organization
Gemeente Amsterdam
Amstel 1, 1011 PN Amsterdam

August 19, 2022

Preface

This thesis represents the final project for the degree Master of Science in Business Analytics at the Vrije Universiteit. The purpose of the thesis is to provide a scientific solution to the problem of a company using all learned skills during the course of the program. The research problem concerns the automatic detection of suspended streetlights and cables in mobile-laser scanned urban point cloud data. This project is the result of a six-month internship conducted at the Chief Technology Office of the municipality of Amsterdam.

I would like to thank my thesis supervisor Vincent-Francois Lavet from the Vrije Universiteit, for his time, support, and feedback during this project. I also would like to thank my second reader professor Sandjay Bhulai from the Vrije Universiteit, for his time and interest in this project. Moreover, I would like to sincerely thank Daan Bloembergen and Chris Eijgenstein at Gemeente Amsterdam, for their daily supervision. Without their support, valuable input, motivation, and relevant feedback I would not have obtained the results I have obtained. Finally, I would like to thank Iva Gornishka for her encouragement and endless assistance during my internship.

Amsterdam, July 2022.

Management Summary

One of a municipality's responsibilities is to regularly monitor suspended streetlights and cables for a safe and reliable environment. Mobile laser scanned 3D point clouds offer great potential for creating high-precision digital representations of assets in street scenes. However, manual object extraction from point clouds is a tedious and time-consuming task due to the large number of points they contain. In practice, it is infeasible to do this regularly for city-scale point clouds. The goal of this thesis is to develop a computationally efficient and reliable pipeline that automatically detects suspended streetlights in urban scene point clouds. This way, streetlights can be mapped and monitored by the municipality with minimal human effort.

In this thesis, we discuss the relevance of the problem to the municipality, review related work in object extraction from point clouds, present and describe our four-staged pipeline for the extraction of suspended streetlights and cables, and provide an extensive evaluation in terms of performance and applicability. This research provides four contributions to the field of point cloud processing: a robust cable extraction algorithm. Second, a cable type classifier, an algorithm that can detect cable attachments, and a labelled training dataset for supervised machine learning models.

We conclude that small assets like cables and streetlights can be extracted from urban point clouds with high precision and recall. Furthermore, efficient filtering steps can significantly reduce the data volume of point clouds and make processing of large-scale point clouds feasible. For better performance, we recommend acquiring a more accurate point cloud dataset for the monitoring of small assets. The used point cloud in this study had an average standard deviation of 10cm, which is rather high.

Table of Content

1 Introduction

The public spaces of cities contain a wide variety of assets. Due to the constant influence of weather and use, the condition of these objects changes and can affect the safety and livability of a city. One of a municipality's key responsibilities is to regularly monitor assets in public spaces. In particular, suspended objects such as streetlights are important due to their vulnerability and functional role. In urban environments, objects are often suspended when placement of a pole is impossible and the desired location is above ground level, i.e., the lighting of a narrow street or the suspension of tram cables for tram tracks integrated within roads. Since cables often stretch above roads and sidewalks, they carry many risks. Therefore, it is crucial to periodically inspect suspended cables and their attachments.

The traditional approach for municipalities to update their urban management system is on-site manual inspection. While sending a specialist out on the streets provides good quality assessments, it is rather slow and expensive, and for an entire city, it is practically impossible. Recently, advances in three-dimensional (3D) data acquisition made Mobile Laser Scanning (MLS) devices an excellent and efficient tool for capturing 3D information of complex objects and scenes (Shi et al., 2020). An example of an acquired MLS point cloud is shown in Figure 1.

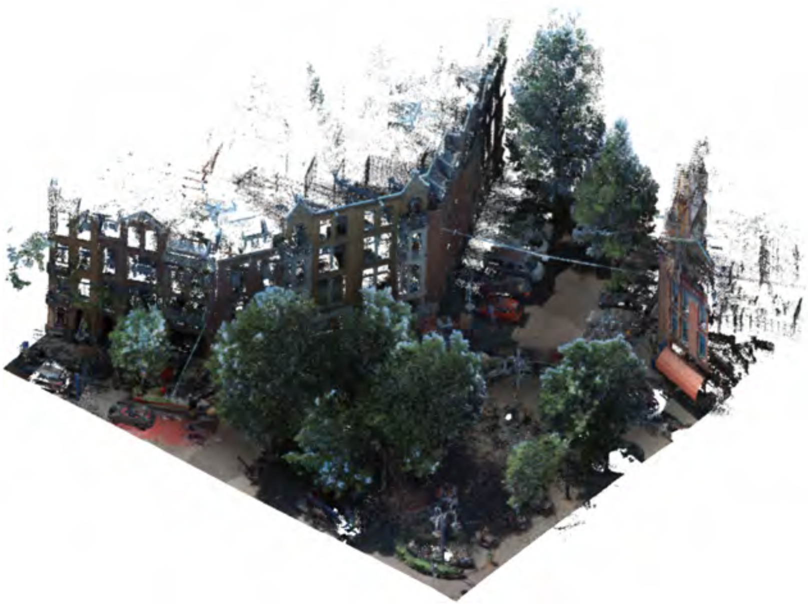


Figure 1: An example urban scene point cloud.

While raw point clouds provide the ability to inspect assets remotely and perform 3D digital measurements (i.e., the distance of a cable to the road), they are rather limited beyond a visual inspection. Research into the development of automatic processing pipelines for urban point clouds offers great potential for more efficient urban asset management (Bloembergen and Eijgenstein, 2021).

There are several challenges when developing automatic processing pipelines for urban point clouds. First of all, MLS point clouds are known for their immense data volume. Moreover, there are a lot of moving objects that create noise that have to be taken care of. Furthermore, available research on MLS point cloud processing is limited. The goal of this thesis is to develop a method that can automatically detect suspended streetlights in MLS urban point cloud data and that is also efficient to be applied to city-scale point clouds. We present a four-phased pipeline consisting of a search space reduction step, cable extraction step, cable classification step, and suspended streetlight detection step. The pipeline has been implemented using *Python* programming language and is publicly available[1].

In this study, we have made several contributions to the field of point cloud processing. Our main contributions are as follows:

1. We present a robust cable extraction algorithm for urban point clouds.
2. We propose a new method which classifies extracted cables into tram cables and other cables.
3. We propose a new method which can detect cable attachments and determine if it is a streetlight.
4. We present a new point cloud training data set with labelled suspending streetlight and cable points.

As can be seen in Figure 2, the contributions fit together the following way. First, we developed a robust cable extraction algorithm that detects cables in urban point clouds. This algorithm can extract different types of cables, including cables that carry attachments. On top of the cable extraction algorithm, the second contribution provides a classifier that distinguishes the extracted cables into tram cables and non-tram cables. The third contribution is an algorithm that detects cable attachments. In particular, it performs a search for streetlights along the classified non-tram. The top-right area of Figure 2 shows the asset management system of the municipality. The asset management system contains all assets in the city. The dashed line illustrates how the output of the third contribution is used to update the asset management system. Similarly, the bottom-left area of Figure 2 illustrates how contribution 4 uses the information of the other 3 contributions to create a labelled training dataset, which can be used for the training of deep learning models.

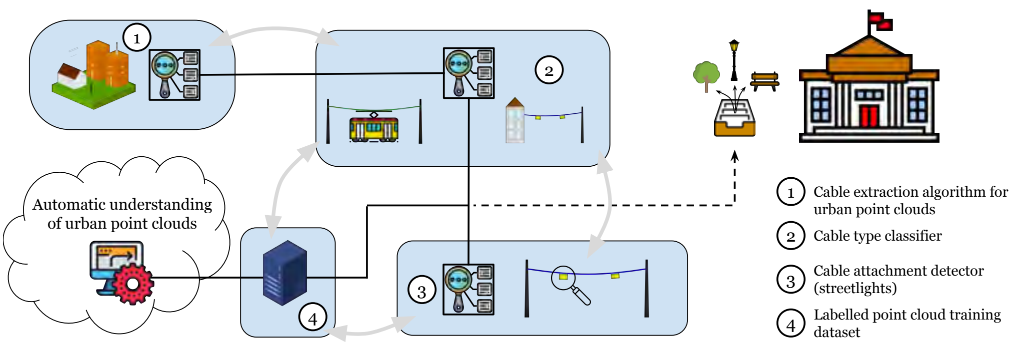


Figure 2: Illustration of how the contributions fit together.

The remainder of this thesis is structured as follows: Chapter 2 provides a brief company description, business context of the problem, scientific background of the problem, problem statement and relevance to the organisation. Chapter 3 reviews related work on the detection of suspended objects in point cloud data. Chapter 4 describes the data sources. Chapter 5 presents and describes the proposed method. Chapter 6 discusses the results of the experiments. Finally, Chapter 7 concludes this thesis and discusses future works.

[1] https://github.com/Amsterdam-Internships/UPC_Suspended_Streetlight_Extractor

2 Problem Description

In this chapter, we first briefly describe the host organisation, the business context and the scientific foundation. Next, the problem statement and research questions are formulated. Finally, the relevance of the problem to the host organisation is described.

2.1 Host organisation

The City of Amsterdam (Gemeente Amsterdam) is the municipal organisation of Amsterdam and Weesp. The municipality is the administrative hub of the city, offering a wide variety of public services. Its responsibilities include mobility, housing, social security, and urban asset management. The organisational structure of the City of Amsterdam consists of four clusters, the services support teams, and seven district organisations responsible for executive tasks.

The Chief Technology Office (CTO) is a department part of the Community Services Cluster. The department's main task is to make innovation happen in the city. CTO works in close collaboration with all other departments in the municipality. They conduct research and develop new solutions with the help of data and technology intended to serve residents. Themes include e-health, circular economy, smart mobility, sharing economy, cooperation with start-ups and innovative procurement. The CTO department has a dedicated Artificial Intelligence (AI) team consisting of data scientists, academics and other professionals who work on urban AI solutions for Amsterdam.

Research and innovation conducted at CTO follow the ideologies of the municipality. All research at CTO is published to their open digital platform[1]. The platform is part of the sustainable knowledge infrastructure of municipalities, universities and other stakeholders in the Amsterdam metropolitan area to show transparency, share knowledge, and facilitate collaboration.

2.2 Business context

The streets in Amsterdam contain a wide variety of assets, such as benches, bicycle racks, trash bins, traffic signs, and streetlights. The condition of these objects decreases over time through use and constantly being exposed to weather influences. One of a municipality's key responsibilities is to regularly monitor assets in public spaces for the city's development and to maintain a safe and reliable environment. On-site manual inspection of assets is costly, time-consuming, and prone to human errors as it depends heavily on the administrator's knowledge and skill (Suárez-González, 2020). A city administrator has to travel back and forth to the location, perform a visual inspection, take notes and manually enter the report data. In some cases, a digital inspection of the public space could produce similar results to a physical inspection.

MLS point clouds can provide a detailed 3D representation of the current state of objects in public space relatively cheap and efficiently (Yen et al., 2011). This digital 3D copy allows city administrators to inspect and monitor streets remotely without having to be present on location. However, manually identifying objects in the point clouds is still a tedious task. An automatic understanding

[1] https://www.openresearch.amsterdam

of urban scene point cloud data offers great potential to municipalities. The development of automatic processing pipelines of urban scene point clouds allows cities to periodically monitor public space more efficiently (Bloembergen and Eijgenstein, 2021).

2.3 Scientific foundation

While urban scene point clouds can be very useful for monitoring assets in public space, the processing is challenging. MLS urban scene point clouds are characterized by immense data volumes due to their high level of detail. Moreover, street scene point clouds contain various noise caused by moving objects such as flying birds and cars that pass by the scanner, making accurate extraction difficult. The introduction of MLS point clouds created the need for efficient and robust processing techniques. Previous studies have successfully investigated point cloud data processing for surveying and mapping purposes (Shan and Toth, 2018; Vosselman and Maas, 2010; Yu et al., 2015). Some studies focused on identifying specific types of objects in point clouds, like ground or buildings. Other studies attempted to semantically segment the whole point cloud.

2.4 Problem statement

Manual object extraction from point clouds is a tedious and time-consuming task due to the large number of points they contain. In practice, it is infeasible to do this regularly for city-scale point clouds. Currently, there are no automated methods that can detect suspending streetlights in point clouds. To address this problem, the goal of this thesis is to develop a computationally efficient and reliable pipeline that automatically detects suspended streetlights in urban scene point clouds. This way, streetlights can be mapped with minimal human effort for the entire city.

In this thesis, we optimise the performance of the pipeline in terms of efficiency and validity. The validity provides a detailed understanding of the quality of the pipeline's output. To measure the quality, the detected points are compared to the ground truth. We will compare the output of the pipeline in both a point-wise and object-wise manner. The point-wise comparison can be used to assess the quality of the output for the usage as a training dataset for semantic segmentation models. The object-wise comparison can be used to assess the quality of the pipeline for the municipality, as their main interest is locating the streetlights.

Next to validity, we measure the efficiency of the pipeline for scalability purposes. In order for the pipeline to be scalable, the computational load should not explode. For instance, if the pipeline can process 1 000 points/sec. and a 2 500 m^2 point cloud contains on average 6 million points, it will take roughly 2 hours to process. The computational efficiency will be measured in terms of processing time needed to evaluate 1 million points using a consumer-market laptop. More specifics on the evaluation of the pipeline can be found in Section 6.2.

We formulate the research question as follows:

RQ *To what extent can suspended streetlights and overhead cables be automatically extracted from urban point clouds?*

We validate the main research question in terms of efficiency and quality. In order to answer the main research question, we introduce three sub-questions. Each sub-question provides information to answer the main research question. The first sub-question addresses the problem of the immense data volume of urban scene point clouds.

SQ 1 *How can we efficiently reduce the search space using prior knowledge?*

Urban scene point clouds are massive in size. Only a small proportion of points in the point cloud are suspending streetlights and cables. The majority of points are redundant and belong to other objects. Analysing each individual point in the point cloud will take an excessive amount of time and is very inefficient. For most objects, it is possible to roughly estimate where they appear in the urban scenes. We can employ this knowledge to reduce the search space with minimal computational costs and thus limit the processing of redundant points in the proceeding steps. We analyse the effectiveness of reducing the search space reduction in terms of accuracy and the number of points removed.

The second sub-question focuses on detecting individual cables in the reduced point cloud. Finding cables is essential in the process of detecting and suspending streetlights because they are suspended by a cable. In other words, the cables can be used for a targeted search for suspending streetlights. After most of the redundant points have been removed, we attempt to detect cable points and identify individual cables.

SQ 2 *How to detect cables in the reduced urban point cloud?*

We propose a point-based cable detection algorithm and introduce various input parameters. We optimise some of the input parameters using a sensitivity analysis Chapter 6. Some studies proposed to use deep learning to classify cable points in point clouds (Qi et al., 2017; Thomas et al., 2019; Yan et al., 2022). We investigate the transferability of the proposed RandLA-Net (Hu et al., 2021) semantic segmentation model that was trained on the Toronto 3D dataset (Tan et al., 2020).

In the last sub-question, we address the problem of detecting if a given cable has an attachment or not and if this attachment is a streetlight. Each of the identified cables is analysed for streetlight attachments.

SQ 3 How to detect suspending streetlights given a cable?

Not all cables in urban scenes will have streetlights attached to them. Instead of using all identified cables, can we ignore certain types of cables? How do we identify these types of cables? Cable attachments in urban scenes are more difficult to extract due to the close proximity of other objects to the cable. What cable properties can we utilise to pinpoint where attachments are located? How can we separate attachments from other objects? Finally, what properties can be used to determine if an attachment is a streetlight or not?

2.5 Relevance of the problem to the organisation

The research into automated detection of suspended streetlights in urban point clouds is relevant to the municipality in two ways: firstly, the public space department can use the pipeline to update the streetlight database; and secondly, the AI team can extend the (semi-)automated urban point cloud annotation pipeline[2] with the proposed pipeline.

It is the municipality's responsibility to maintain objects in the public spaces. The municipality has an online form available where citizens can report problems related to public space[3]. In case of malfunctioning streetlights, citizens are asked to select the corresponding streetlight on a map. However, the map is not complete, some of the suspending streetlights are missing others might have incorrect coordinates. Manually inspecting the complete jurisdictional area for suspending streetlights is impractical. A pipeline that detects suspending streetlights in urban point clouds can help the public space department update the streetlight map.

At the same time, the detection pipeline is relevant to the AI team. The AI team currently works on a large semantic segmentation model for urban point cloud data. The training dataset is labelled using a (semi-)automatic annotation pipeline (more on this can be read in Bloembergen and Eijgenstein (2021)). The current annotation pipeline for creating the training dataset is limited to six classes: ground, building, car, tree, lamp post, and traffic sign. Adding an additional class to the training dataset could help the model distinguish classes better and improve prediction scores.

[2] Github repository: https://github.com/amsterdam-ai-team/urban_pointcloud_processing
[3] https://meldingen.amsterdam.nl/incident/beschrijf

3 Related Work

In this chapter, we will briefly review related works in the research area of this thesis and discuss their main limitations. First, a general review of point cloud processing is given (Section 3.1). Next, existing methods for cable extraction are grouped and reviewed (Section 3.2). Finally, a summary of the most relevant studies for this thesis is given.

3.1 Point cloud processing

Scholars have been studying the extraction and modelling of objects from 3D point clouds since the introduction of laser scanning devices. Many studies have focused on the processing of LiDAR point cloud data for mapping and surveying purposes (Shan and Toth, 2018; Vosselman and Maas, 2010). Yi et al. (2017); Yastikli and Cetin (2021); Wu et al. (2017) identify building facades from MLS point clouds. Wang and Glenn (2009); Maguya et al. (2014) extract digital terrain models from airborne LiDAR. Xu et al. (2007); Safaie et al. (2021); Yu et al. (2012) extract trees from point clouds. Arastounia (2017) recognises rail tracks and power cables from terrestrial and airborne LiDAR point clouds. Yang et al. (2013) and Pu et al. (2011) extract roads from LiDAR point clouds.

MLS point clouds are known for their immense data volumes and high variance in point density (Yadav et al., 2015). Some researchers have proposed to use voxelisation to reduce the number of points (Cabo et al., 2018; Wu et al., 2013; Shi et al., 2018; Jung et al., 2020). The original scanned points are binned into equally spaced 3D-grid cells called 'voxels'. The center coordinates of the non-empty voxels are used as new points. One of the key advantages of voxelisation is that it down-samples the point cloud without losing information is sparse regions. In contrast, reducing the volume of a point cloud using random sub-sampling can result in the loss of sparse regions. Besides computational gains, some researchers utilised the concept of voxelisation to detect specific characteristics of objects. Munir et al. (2019) extracted pole-like objects from urban point clouds using a voxel-based algorithm. Poles were extracted by analysing vertical continuity in the voxelised point cloud. To cope with the large size of the MLS street scene point clouds in this thesis, voxelisation is used to down-sample the high-density regions, like trees, while preserving the sparse cable points.

3.2 Cable extraction

To the best of our knowledge, no research has been published on the detection of suspended streetlights in urban point clouds (or cable attachments in general). However, research has been published on the automated extraction of suspended cables from point clouds. Most studies focused on the extraction of cables from powerline and railroad corridors. Melzer and Briese (2004) were the first to address the problem of powerline detection in laser-scanned point clouds. They used a two-dimensional Hough Transform (HT) algorithm to detect straight lines in the top-down projection of the point cloud. They assumed cables are straight lines looking down from the sky. Troublesome non-line regions with high point density in the top-down projection were removed using a culling mechanism. Next, the cables were modelled using the RANdom Sample Consensus (RANSAC) algorithm proposed by Fischler and Bolles (1981). In general, the existing methods used for the detection, extraction, and modelling of cables can be divided into two main groups: supervised (Kim et al., 2010; Guo et al., 2015; Wang et al., 2017) and unsupervised (Fang and Lafarge, 2019; Guan et al., 2016; Ma et al., 2019).

Supervised methods

The supervised approach uses a classifier to segment cables from other objects in the point cloud. Wang et al. (2017) proposed to use a Support Vector Machines (SVM) for classification, 26 different features were extracted from the local neighbourhoods. Kim et al. (2010) used a Random Forest (RF) classifier to classify objects in the power line corridor into vegetation, wires, pylons, and buildings. They achieved an accuracy of 97.17% on the wire class. The height feature was the most important feature during their classification and most of their errors appeared near ground. However, supervised methods are time-consuming because they require large amounts of manually labelled training data to obtain a good prediction score (Lehtomäki et al., 2019). Moreover, the performance of supervised models is easily influenced by the composition of the training samples and scene complexity (Che et al., 2019; Li et al., 2019; Ma et al., 2019). The aforementioned supervised extraction methods focused on high-voltage transmission lines in similar scenes. Power lines operate at high heights and have a lot of clearance. In this study, the point cloud scenes are more complex because cables stretch at lower height levels and have less clearance to other objects such as buildings, trees, and poles. Furthermore, the street scenes of Amsterdam are very diverse; thus, the amount of data that must be manually labelled is immense.

Unsupervised Methods

Most researchers prefer unsupervised methods to extract cables. The unsupervised methods usually focus on the geometrical properties of cables and have no need for large amounts of training data. Munir et al. (2019) categorised the unsupervised methods into grid-based and point-based methods.

Grid-based Methods

Grid-based methods process the input data by first projecting the 3D point cloud onto a 2D grid space. Next, efficient and powerful image processing techniques can be used to process the point cloud and detect cables. Although processing the data is now significantly faster, information is lost due to a missing dimension. The majority of researchers used Hough Transformation (HT) (Tajima and Masuda, 2020; Shokri et al., 2019; Melzer and Briese, 2004; Guan et al., 2016; Yadav and Chousalkar, 2017) and RANSAC (Tajima and Masuda, 2020; Shi et al., 2018; Melzer and Briese, 2004; Lehtomäki et al., 2019; Sánchez-Rodríguez et al., 2019) to extract straight lines from 2D gridded data, due to their efficiency (Guan et al., 2016). However, HT and RANSAC do not work well for complex areas with vertically overlapping objects and high-density regions, i.e., vegetation and buildings (Melzer and Briese, 2004). Hence, a pre-processing step is often necessary to remove troublesome non-line regions. Another limitation of grid-based methods is the fact that these methods fail to extract vertically arranged cables. In response to the limitation of grid-based methods, Awrangjeb (2019) extracted cables that vertically overlap other objects by converting the point cloud into binary masks for different height levels, and then detecting straight lines in the masks. Even though their proposed algorithm can separate vertically overlapping cables, it will still have difficulties when multiple cables intersect at the same heights level.

Point-based Methods

Point-based methods process every single 3D point in the point cloud to determine its class. Although point-based methods don't lose information like grid-based approaches, the processing time is highly sensitive to the point cloud's volume. Hence, point-based methods often employ a pre-processing step to remove redundant points and speed up computations, i.e., ground filtering (Shokri et al., 2021; Shi et al., 2020; Shokri et al., 2019; Guan et al., 2016). In addition, point clouds are often organised into data structures such as kd-trees and oct-trees to speed up the time-consuming neighbourhood search operations (Hackel et al., 2016). Most approaches utilise the concept of principal component analysis (PCA) to derive a point's local geometric features (Shi et al., 2020; Husain and Vaishya, 2019; Lehtomäki et al., 2019; Zhang et al., 2016a; Cheng et al., 2014). The local geometric features are often used to cluster points with similar properties into individual objects, i.e., cables can be assumed to be a set of linear points and trees to be a set of volumetric points.

For this study, we prefer a point-based approach for the extraction of suspending cables. Grid-based approaches will have trouble detecting cables due to the presence of vertically overlapping cables and other objects. Furthermore, objects like bicycle racks and fences can produce straight lines in top-down projections and have to be taken care of. The main limitation of point-based methods is the computational load. However, an efficient filtering step can reduce the search space significantly and allow point-based analysis to be performed in a reasonable time (Jung et al., 2020).

3.3 Summary of most relevant studies

The following reviews the most recent and relevant studies on the detection, extraction and modelling of cables from 3D point cloud data.

Arastounia (2017) recognised railroad assets such as rail tracks and cables from airborne laser-scanned point clouds using a computationally efficient fully data-driven method. The author first recognised rail track pairs in the low-height point group using eigendecomposition. Next, cable seed points were identified and grown by employing the smooth curvature gradient property of rail tracks. Although all objects of interest were identified with an average precision and accuracy greater than 95%, the configuration was limited to non-intersecting straight rail tracks and cables. In our approach, we will focus on cables that can intersect other cables. We will use a similar growing technique which employs the last bit of cable to determine the growing direction. Some studies utilised the assumption that cables must be attached between two consecutive poles. Shokri et al. (2021) proposed a three-step algorithm for automatically extracting distribution power lines from both rural and urban MLS point clouds. In the pre-processing step, they eliminated 90% of unneeded points by using the noise removal algorithm proposed by Rusu et al. (2008) in combination with the removal of "low-height" points. Next, utility poles were identified using horizontal and vertical density information in search areas containing lines. The search areas were detected using the HT algorithm on a top-down projection that was processed using a closing morphology and canny edge filter in respective order. Finally, a 3D box between adjacent poles was voxelized and cable points were fit to a polynomial. The three-step algorithm managed to extract cables with average correctness and completeness of 100% and 95.6%. The assumption that cables must be attached between two consecutive poles limits the applicability of the proposed methods. In our method, cables can have any type of connection. Shi et al. (2020) developed a technique to extract power lines from MLS point clouds by first identifying candidate power line points based on the analysis of the linear geometrical feature. Points with a linear feature were labelled as candidate power line point and clustered based on Euclidean distance. The over-segmented cables were re-clustered based on similar fitted catenary models to the cable segments. This assumption holds for most cables but it is not the case for cables that carry attachments, such as streetlights. The catenary assumption limits the applicability of the algorithm to cables without attachments, such as power lines. Instead of fitting a catenary model, we adopt a condition-based approach for re-clustering that suffices for cables with attachments. Few studies have focused on extracting road crossing cables in urban areas. In this regard, Tajima and Masuda (2020) presented a technique to specifically reconstruct cables crossing roads. Since the direction of the scanning pattern of MLS is almost parallel to the road-crossing cables, points on the cable become sparse and cables crossing roads are more difficult to detect. They proposed a reconstruction technique using both point clouds and camera images. In our study, camera images were not available.

In summary, point clouds processing techniques can be used for the mapping and surveying of objects. In order to cope with the immense data volume, voxelisation can be used to improve efficiency. Although there have been many successful studies on the automated extraction and modelling of cables from point clouds, there are still some shortcomings in these works. The majority of studies focus on structured cables in power line and railroad corridors and employ characteristics such as connectivity between poles. These methods will fail to extract utility cables in urban environments that are less structured, intersect, and stretch between building facades. Another shortcoming is that methods use a catenary model or polynomial to reconstruct and merge cables. This assumption holds for cables without attachments, like powerlines, but does not suffice for cables that carry streetlights. To address these shortcomings, this thesis aims to develop an automated extraction algorithm for cables in the urban scene that can carry attachments. In addition, we propose an algorithm that can detect streetlights attached to cables. To the best of our knowledge, we are the first to propose a complete pipeline for the automated extraction of suspended streetlights and their attachments from point clouds.

4 Data

In this chapter, we describe the data sources used in this research for the detection of suspended streetlights and cables.

4.1 3D point cloud

The primary source of data in this thesis is the MLS point cloud of the jurisdictional district of the City of Amsterdam[1]. A point cloud is an unordered set of geo-referenced points (x,y,z) with radiometric information such as intensity, colour, and returned number of pulses. The Amsterdam point cloud dataset was captured by a Velodyne HDL-32 LiDAR sensor[2] mounted on a car. An illustration of the setup is visualised in Figure 3.

Figure 3: Setup used to capture the point cloud.

The point density of the Amsterdam point cloud ranges between 1.000 - 2.500 points/m^2. The driving speed of the vehicle and the distance of objects to the sensor influence the point density in the point cloud. The relative precision of the points is 2cm with an average standard deviation of 10cm. The acquired point cloud contains 240+ billion points and approximately 66.75 km^2 of urban scenes. The point cloud's size after post-processing is about 1.5TB of data. For processing reasons, the data collection team divided the point cloud into smaller segments called "tiles" of 50 x 50m. On average, a tile has about 9 million points, which is more suitable for processing on laptops. The information on the point cloud is summarised in Table 1.

Table 1: Point cloud dataset specifications

Specification	**Values**
Laser sensor	Velodyne HDL-32 LiDAR
Point features	(x,y,z), intensity, number of returns, RGB
Total points (billion)	240+
Covering area (km^2)	66.75
Dataset size (TB)	1.5
Availability	not public (license)

[1]Point cloud data provided by Cyclomedia: https://www.cyclomedia.com/
[2]https://velodynelidar.com/products/hdl-32e/

4.2 BGT data

The BGT[3] (*Basisregistraties Grootschalige Topografie*) is a register of all objects that are in contact with the ground in the Netherlands. The dataset contains geographical information of most objects in terms of (x,y) coordinates. We use the BGT dataset to extract the location of tram tracks in Amsterdam (Section 5.3). Tram track segments are stored as a set of (x,y) coordinates forming a line.

4.3 AHN elevation data

The AHN4[4] (*Actueel Hoogtebestand Nederland*) dataset provides an accurate elevation model of the Netherlands obtained by aerial laser scanning. The elevation model is a labelled point cloud with a precision of 0.1m. We use the ground class to estimate the height of scanned points relative to ground level. A pre-processing step interpolates the gaps in the AHN4 using the nearest neighbour's value. An example of the elevation data is shown in Figure 4.

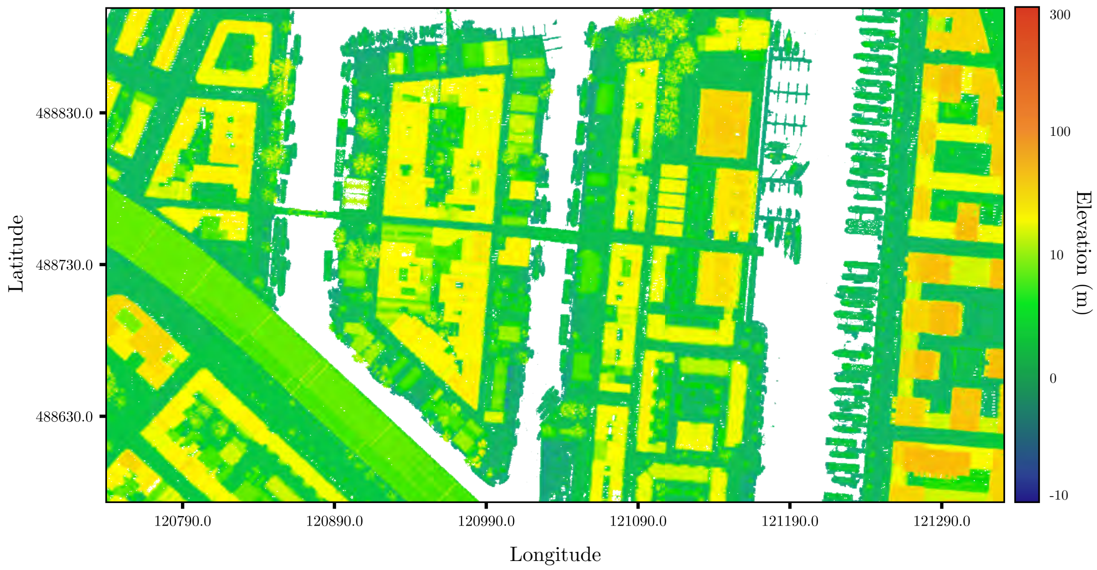


Figure 4: AHN elevation data of an example neighbourhood in Amsterdam

[3]https://www.pdok.nl/introductie/-/article/basisregistratie-grootschalige-topografie-bgt-
[4]https://www.pdok.nl/introductie/-/article/actueel-hoogtebestand-nederland-ahn4-

4.4 BAG building surface data

The BAG[5] (*Basisregistraties Adressen en Gebouwen*) dataset contains rich information about all buildings in the Netherlands. The dataset is updated frequently and stores attributes of every building, including condition, surface outline, geometry, (x,y) coordinate, date of construction and purpose. In this thesis, we use the BAG building outline polygons as ground truth to filter out buildings from the point cloud, see section subsection 5.1.1. The dataset was acquired by scraping the 3D BAG API[6] for buildings within the area covered by the point cloud. Figure 5 shows an example of the BAG building outline polygons for a specific area in Amsterdam.

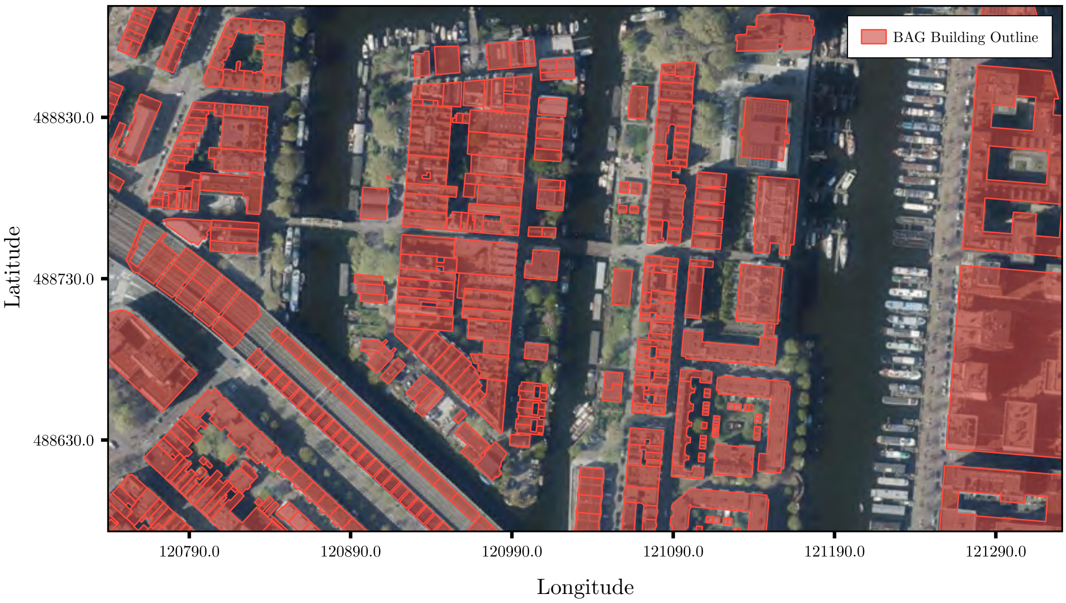


Figure 5: BAG buildings outlines projected on satellite image of an example neighbourhood in Amsterdam

[5]https://www.pdok.nl/introductie/-/article/basisregistratie-adressen-en-gebouwen-ba-1

[6]Provided by 3D geoinformation research group, TU Delft: https://data.3dbag.nl/api/BAG3D_v2/wfs

5 Methods

In this chapter, we describe our proposed method for the detection of suspended streetlights and cables in urban point clouds. The literature review indicated that there is still a need for a method that can efficiently detect suspended streetlights and cables in urban scene point clouds. The proposed method automatically detects suspended streetlights and cables in urban scenes using a four-stage pipeline. In the first stage, the search space of the point cloud is reduced using constraints based on prior knowledge about the city infrastructure to speed up the processing (Section 5.1). In the second stage, cable points are detected in the reduced point cloud using a point-based analysis in combination with clustering and refinement steps (Section 5.2). In the third stage, individual cables are classified into tram cables and non-tram cables (Section 5.3). In the fourth and final stage of the pipeline, the detected non-tram cables are analysed for streetlight attachments using a connected component algorithm in combination with bounding box analysis (Section 5.4). An overview of the steps and stages of the proposed pipeline can be seen in Figure 6. The inputs of the pipeline are an unprocessed point cloud tile and corresponding AHN, BAG, and BGT data. The pipeline has two outputs: suspended streetlights and horizontal cables of at least 2 meters long. Vertical cables are considered outside the scope of this research.

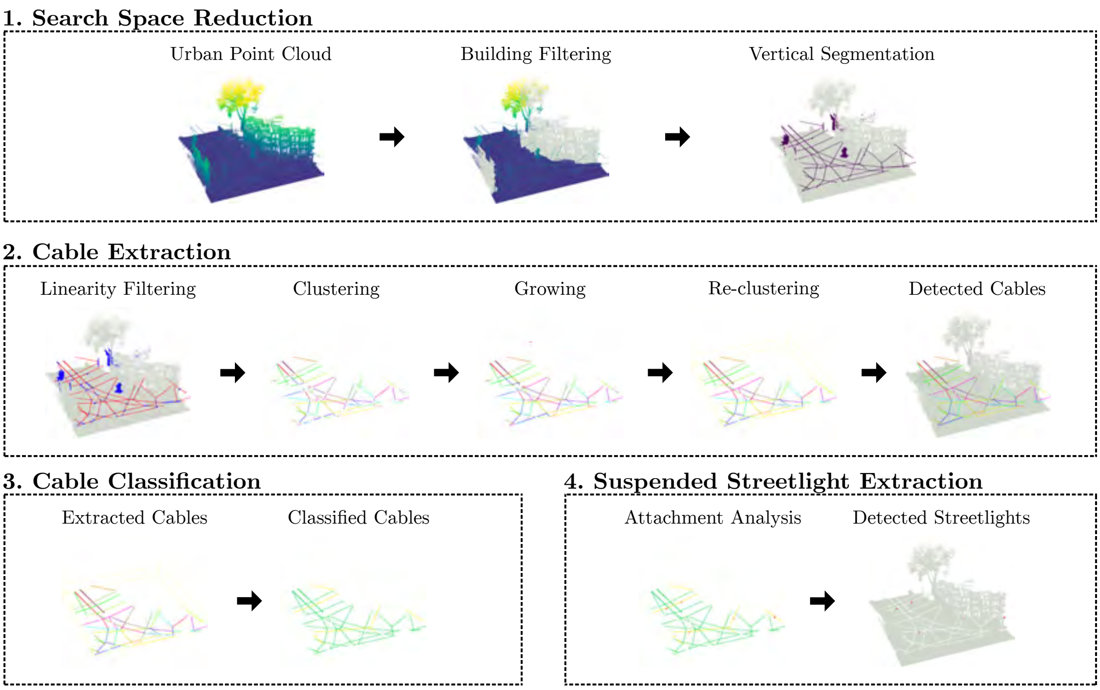


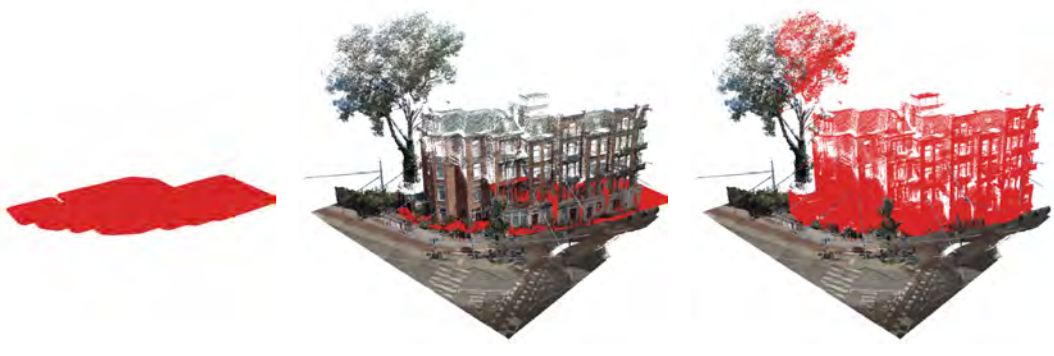
Figure 6: Workflow of the proposed pipeline for automated detection of suspended streetlight and cable from MLS urban scene point clouds.

5.1 Search space reduction

Due to the immense data volume of MLS urban scene point clouds, processing can take a significant amount of time. In urban scene point clouds, the majority of points belong to the ground surface and building facades. Directly processing the full point cloud is inefficient because of the large number of redundant points that will be evaluated using costly neighbourhood searches (Hackel et al., 2016). To overcome this limitation, the point cloud's spatial and computational complexities have to be reduced before performing sophisticated point-based techniques (Yu et al., 2014). Cserép et al. (2018) mentioned that linear runtime complexity algorithms can serve as adequate filters to reduce the point cloud with a low algorithmic cost. In order to improve the efficiency of the pipeline, two filtering algorithms that go through the point cloud in linear time are proposed: (1) building filter; and (2) vertical segmentation filter. The performance of the filtering steps in terms of computational reduction and time are analysed in Chapter 6.

5.1.1 Building filter

Assuming that cables do not stretch through urban structures, we will remove all points in the point cloud that belong to buildings. In order to do this, a data fusion-based approach inspired by Bloembergen and Eijgenstein (2021) is used. Data fusion combines different data sources into a unified data set which is more valuable and reliable than a single data source. The integration step is often followed by a reduction or replacement step. The BAG dataset offers geo-referenced ground truth building outlines which are frequently updated. The proposed filtering step combines the point cloud with the BAG building outlines to determine which of the points belong to an urban structure and removes them. The building outlines are stored as a set of (x,y) coordinates forming a polygon. Points that fall inside the polygons are marked as urban structure points. Bloembergen and Eijgenstein (2021) initiated to inflate the polygons to take into account the margin of error both data sets have. Inflating the building outline polygon helps to increase the number of marked building points. On the other hand, inflating the polygons too much can result in cable points being incorrectly marked. A typical example is cables connected to housing facades. The more the footprint is increased, the more cable points can be incorrectly labelled. The inflation parameter I is set to 0.5 m. We take care of potential false negatives in the cable growing step (Section 5.2). An illustration of the data fusion and building point filtering is shown in Figure 7.



(a) Building outline **(b)** Building outline + point cloud **(c)** Marked building points

Figure 7: Example illustration of data fusion between BAG and the point cloud to filter out building points.

5.1.2 Vertical segmentation filter

In general, ground points account for the largest proportion of points in point clouds (Yu et al., 2014). Since our goal is detecting suspended assets, we employ a height segmentation filter to retain only points located above ground level between a minimum (H_{min}) and maximum (H_{max}) height thresholds. The points are categorised into the following three groups based on their z-coordinate: "low-height" points (e.g. ground, benches, and cars), "medium-height" points (e.g. cables and tree foliage) and high-height points (e.g. treetops). An example of the vertical segmentation of a point cloud scene is shown in Figure 8. The red marked points indicate the medium-height class and consist mainly of cables, tree foliage and poles.

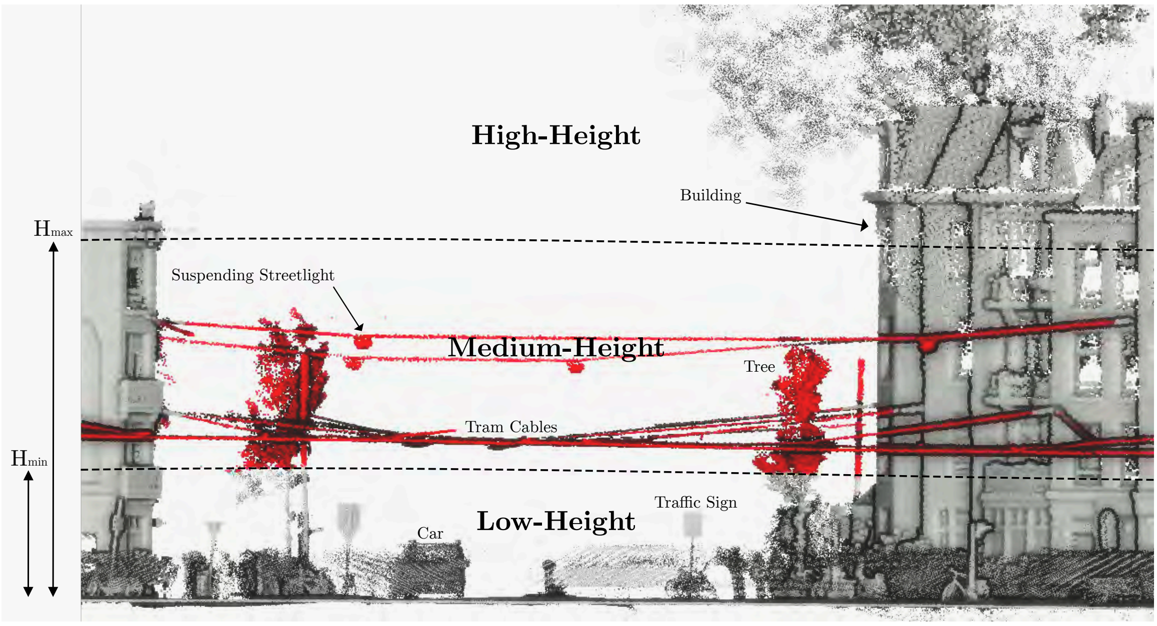


Figure 8: Illustration of the vertical segmentation filter on an example point cloud tile. The points belonging to the medium-height class are coloured red.

The H_{min} parameter defines the separation height between the "low-height" and "medium-height" groups. In other words, H_{min} is the minimal vertical clearance between the road and overhead wires. In most regions, the vertical clearance is specified by law (Tajima and Masuda, 2020). For the study area of this thesis, the minimal operating height of overhead cables is 4.5 m. To account for the margin of error in the point cloud, we set H_{min} to 4 m. The H_{max} parameter, separating the "medium-height" and "high-height" points, is set to an increased value of 15m based on an empirical analysis of the training tiles and information provided by the municipality (cables for suspended streetlights operate between 6-12 m).

It should be noted that the parameters H_{min} and H_{max} are relative to ground level. However, the z-coordinate of a point in a point cloud is with respect to NAP[1]. Simply evaluating if $H_{min} < z \leq H_{max}$ will not suffice. To obtain the height value relative to the ground surface for each point, we vertically shift each point with its corresponding ground elevation. The elevation of the ground is often represented by a Digital Surface Model (DSM)[2]. A DSM can be obtained in various ways. A common method is to fit horizontal planes to the point cloud (Zermas et al., 2017). Shi et al. (2020) and Zhang et al. (2016b) used a cloth simulation to identify ground points. The cloth simulation generates surfaces by covering a stiff fabric over the inverted point cloud. A drawback is that these methods require a lot of computational power. In this study, we fuse the publicly available AHN3 elevation data with the point cloud and use it as a reference surface. The grid cells of the reference surface contain the ground elevation relative to NAP. The (x,y)-coordinates of a point are used to

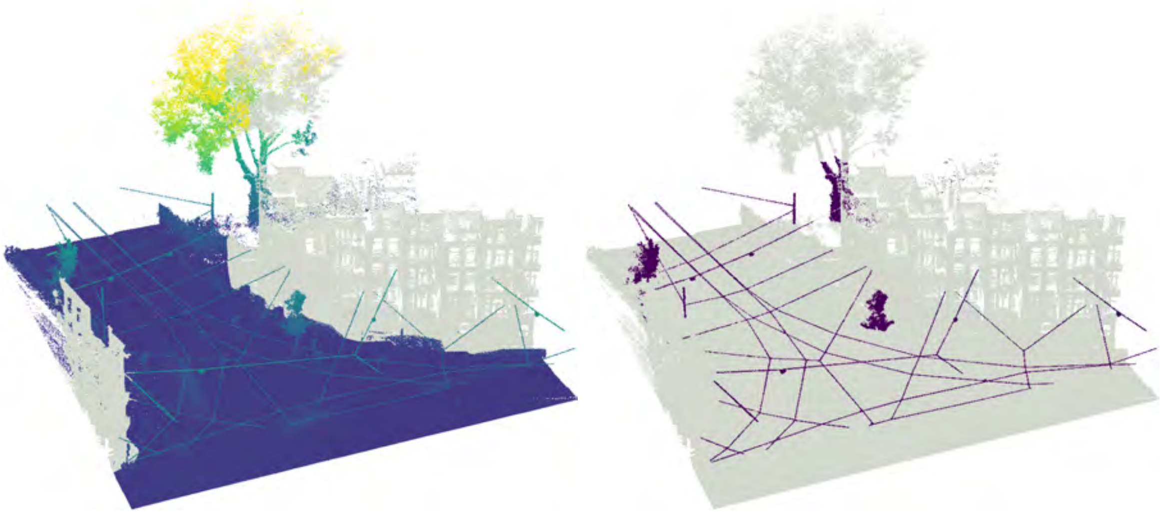
[1] Normaal Amsterdams Peil (NAP) is a vertical reference system.
[2] A DSM is a 3D topographical representation of the elevation of terrain.

determine the grid cell and extract the corresponding ground surface value z_g. The height of a point relative to ground surface is given by $z_h = z - z_g$. The filtering rules are summarised in Table 2.

Table 2: Filtering rules of the vertical segmentation filter, where z_h is the height of a point with respect to ground level.

Group	Condition	Description
Low-height	$z_h < H_{min}$	Ground, low-vegetation, cars,
Medium-height	$H_{min} \leq z_h \leq H_{max}$	Cables, tree foliage, pole-like objects
High-height	$z_h > H_{max}$	Tree tops

Figure 9 shows an example point cloud after the vertical segmentation filter. The filter can eliminate a large proportion of unneeded points without introducing false negatives. The remaining objects in the filtered point cloud are mainly cables, tree foliage, and parts of poles.



(a) before **(b)** after

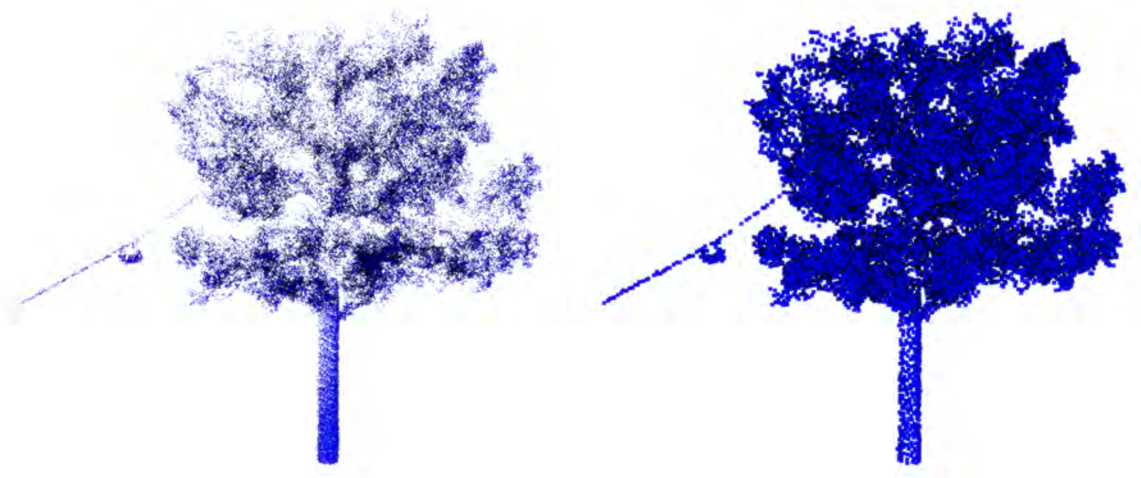
Figure 9: Example of an urban point cloud before (a) and after (b) applying the vertical segmentation filter. The removed points are coloured grey.

5.2 Cable extraction

The previous stage reduced the original point cloud by filtering out the majority of the redundant points (Section 5.1). In this stage, individual cables are identified and extracted from the reduced point cloud. The extracted cables will be used in the next stages to perform a targeted search for cable attachments. The cable extraction stage can be divided into five steps: (1) voxelising the reduced point cloud; (2) identifying candidate cable points; (2) clustering candidate cable points into cable segments; (3) growing cable segments; and (4) re-clustering over-segmented cables.

5.2.1 Voxelisation

Cables in MLS urban point clouds are sparse and exhibit strong variations in point density (Tajima and Masuda, 2020). The irregularity in point density makes it difficult to detect cables. To overcome this problem, points are binned into equally spaced 3D-grid cells called 'voxels'. The cubic voxel's size is defined by s. In this thesis, s is set to 9 cm based on the results of some preliminary experiments and similar values used in related works. We use the center coordinates of the non-empty voxels as new points in the pipeline. The voxelisation improves the uniformity of point density in cables. In addition, high-density regions in the reduced point cloud such as tree foliage are down-sampled and improve extraction efficiency due to the fewer points that have to be analysed. See Figure 10 for an illustration of the voxelisation of a cable together with a tree.



(a) Original **(b)** Voxelised

Figure 10: Illustration of voxelising original scanned points (a) into voxels (b).

5.2.2 Candidate cable point selection

To identify candidate cable points, we utilise local geometrical features of a point's neighbourhood. The geometrical features of a point's neighbourhood can express valuable information about a point. A key property of a cable point is that its neighbourhood point set has a linear structure. A tree point, on the other hand, is most likely to have a volumetric local point set, expressing the property of trees growing in all directions.

To obtain accurate local geometrical features of a point, it is key to choose an appropriate method for selecting the neighbourhood of points (Shi et al., 2020). If the neighbourhood is chosen poorly, points from other objects or cables could be included in the neighbourhood point set. In that case, the geometrical features of the point set would not suffice to identify cable points accurately. The neighbourhood of a point can be determined in two different ways: a k-nearest neighbour search or geometrical radius search. The k-nearest neighbour search finds the k nearest neighbours to a query point, whereas the radius search finds all neighbouring points within a certain distance to the query point.

The point density of the point cloud plays an important role in choosing the right search method. If the point density is uniform, radius search is the appropriate method since the search radius directly reflects a geometrical scale in object space (Hackel et al., 2016). In contrast, the k-nearest neighbour approach is not bounded by a maximum distance and can therefore be seen as a density adaptive search radius.

In this study, we utilise a radius search to determine a point's neighbourhood. As can be seen in Figure 11, the endings of the road-crossing cables are extremely sparse and miss multiple points. In sparse regions, the k-nearest neighbours approach will likely enclose points from other objects due to the sparsity of cables and the close proximity to objects in the street scene. Moreover, the voxelisation step improved uniformity in point density allowing the radius to be an accurate geometric scale. The parameter of the neighbourhood radius must be chosen carefully. A large r can capture different properties compared to a small r. For instance, a too large search radius can identify the geometrical property of two parallel cables as a planar. The radius parameter r is set using a sensitivity analysis which analyses the impact of the parameter value on the performance of the pipeline. More on this can be read in Section 6.3.

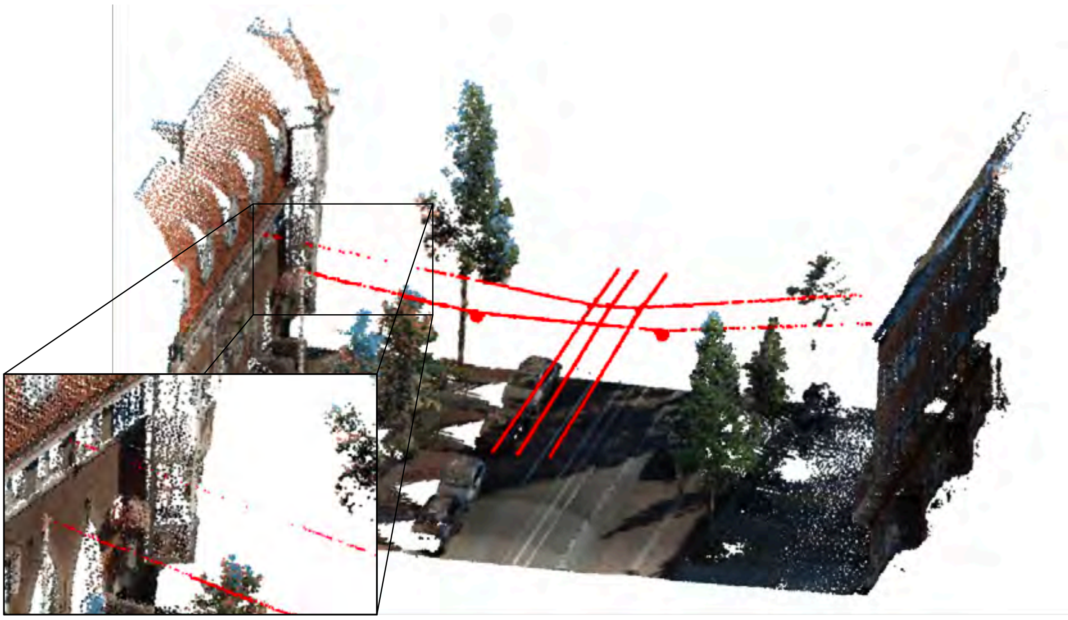


Figure 11: Example of sparsity in road-crossing cable endings. Cables are shown in red.

The computational complexity of a neighbourhood search using brute force is $\mathcal{O}(n)$, where n is the total number of points (Hackel et al., 2016). In literature, various search algorithms have been proposed that are significantly faster. In this thesis, we use a kD-tree as described in Maneewongvatana and Mount (1999) to structure the point cloud and speed up the search with $\mathcal{O}(\log n)$[3].

The local geometrical linearity feature L of a point is calculated using the concept of principal component analysis (PCA) on a point's neighbourhood. We perform PCA as follows: Given a point p and its neighbourhood point set P, we calculate the covariance matrix $\mathbf{C}_{3\times3}$ of P as in Equation 5.1. Next, $\mathbf{C}_{3\times3}$ is decomposed into eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$ and corresponding eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$ as shown in Equation 5.2, where λ is the matrix of eigenvalues, $\mathbf{V}$ is the matrix of eigenvectors, and $\mathbf{I}$ an identity matrix.

$$\mathbf{C} = \begin{bmatrix} \mathrm{var}(X)^2 & \mathrm{cov}(X,Y) & \mathrm{cov}(X,Z) \\ \mathrm{cov}(X,Y) & \mathrm{var}(Y)^2 & \mathrm{cov}(Y,Z) \\ \mathrm{cov}(X,Z) & \mathrm{cov}(Y,Z) & \mathrm{var}(Z)^2 \end{bmatrix} \tag{5.1}$$

$$\begin{aligned} \mathbf{CV} &= \lambda\mathbf{V} \\ (\mathbf{C} - \lambda\mathbf{I})\,\mathbf{V} &= 0 \\ \det(\mathbf{C} - \lambda\mathbf{I}) &= 0 \end{aligned} \tag{5.2}$$

The eigenvalues denote the magnitude of variance in the direction of the corresponding eigenvector. Equation 5.3 is used to calculate the linearity L of neighbourhood point set P. L can take values between 0 and 1, where 1 corresponds to a perfect linear point set and 0 to a non-linear point set. We introduce L_t as the threshold value to determine if a point is considered linear or not.

$$L = (\lambda_1 - \lambda_2)/\lambda_1 > L_t \tag{5.3}$$

Besides a cable point being linear, its direction should be roughly parallel to the ground surface (horizontal). The main direction of point p is given by the first principal component $\mathbf{v}_1$ of P (Guan et al., 2016; Qin et al., 2017). We introduce a as the slope between the cable direction and the

[3]The kD-tree algorithm is available in the python library SciPy

horizontal XY-plane. The computation of a is shown in Equation 5.4. We introduce A_t as the threshold value for the maximum slope a cable can make.

$$a = \arccos \frac{\vec{\mathbf{v}}_1 \cdot \vec{\mathbf{z}}}{|\vec{\mathbf{v}}_1||\vec{\mathbf{z}}|} < A_t, \quad \vec{\mathbf{z}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.4}$$

The proposed candidate cable point step has three input parameters: neighbourhood radius (r), linearity threshold (L_t), and slope threshold (A_t). We set A_t to 10°, this will filter out all vertical linear points such as poles, but still takes into account the sagging characteristic of cables. The other two parameters are determined through a sensitivity analysis in Chapter 6.

Figure 12 shows an example of the reduced point cloud with candidate cable points marked red and others blue. It can be seen that most of the cable points have a red colour and thus have a linear feature. Besides cables, some small branches in the tree show linearity as well. Furthermore, we can see that not all cable points have been identified. Especially parts close to other objects were left undetected. For example, the intersection between two cables and the part where an attachment is connected to the cable. The next steps will take care of these problems.

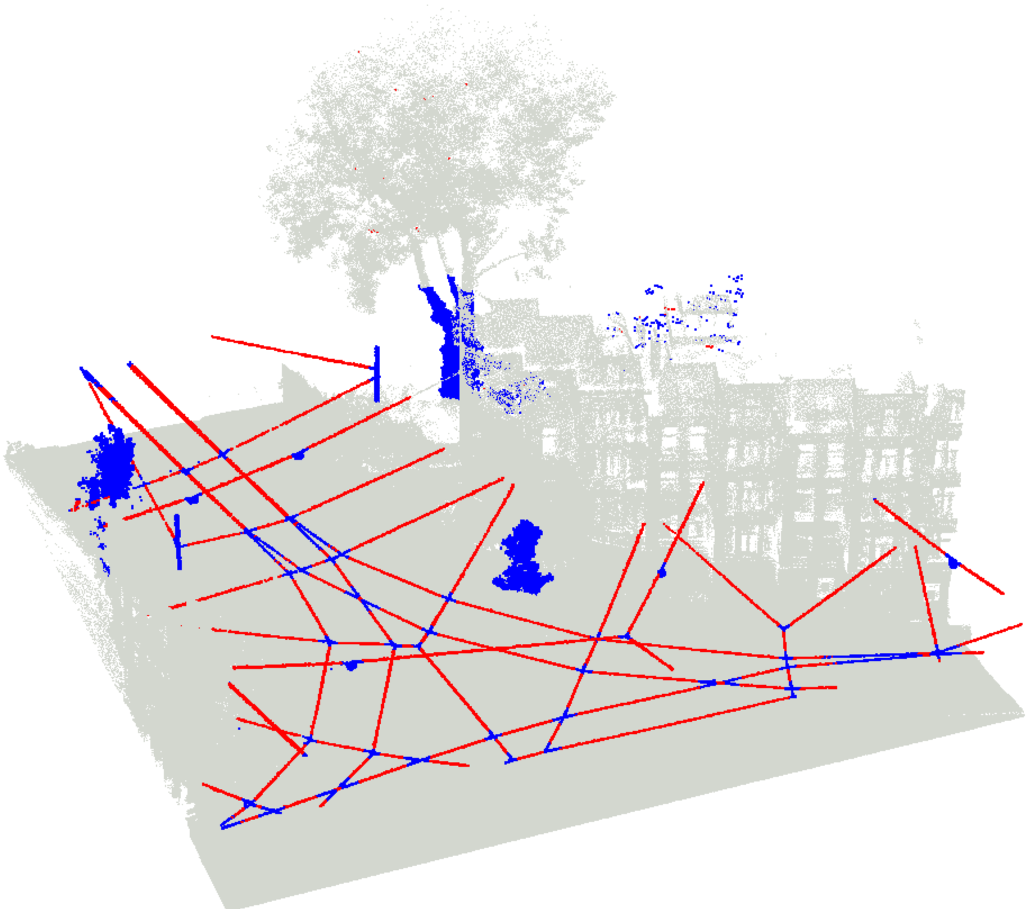


Figure 12: Example of identified candidate cable points (red) in a reduced urban point cloud tile. Non candidate points are coloured blue. The grey points were removed in previous filtering steps.

5.2.3 Candidate point clustering

The second step in the process of extracting cables is organising the identified cable points into individual groups of cable segments. The points are clustered based on Euclidean distance (Equation 5.5). The parameter d defines the minimum distance between clusters. If the Euclidean distance between two points is less than d, the points are put in the same cluster. The parameter d is set equal to the r (neighbourhood radius). A value larger than r could result in under-segmentation of the cables, where parallel cables are clustered into the same group.

$$\text{euclidean}\,(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{5.5}$$

The clustering of candidate cable points into individual cable segments offers the possibility to reconstruct cables and retrieve missed points in previous steps. subsection 5.2.4 will propose a technique to grow the cable segments and subsection 5.2.5 will propose a technique to fill gaps in over-segmented cables. Another advantage of clustering cable points into cable segments is that non-cable points, with a linear horizontal feature, can be filtered out, i.e., tree branches or pole extensions. Cable segments usually span multiple meters, whereas tree branches or pole extensions are limited (also due to gravity).

The clustered candidate cable points are shown in Figure 13, some cables are over-segmented due to missed cable points in the previous step or sparsity in the cable. Besides the larger cable clusters, there are some small non-cable clusters. These small clusters mainly originate from points that have a linear feature but belong to a different object (i.e., a tree branch). To cope with the small non-cable clusters, a minimal cluster length will be introduced at the end of the cable extraction stage.

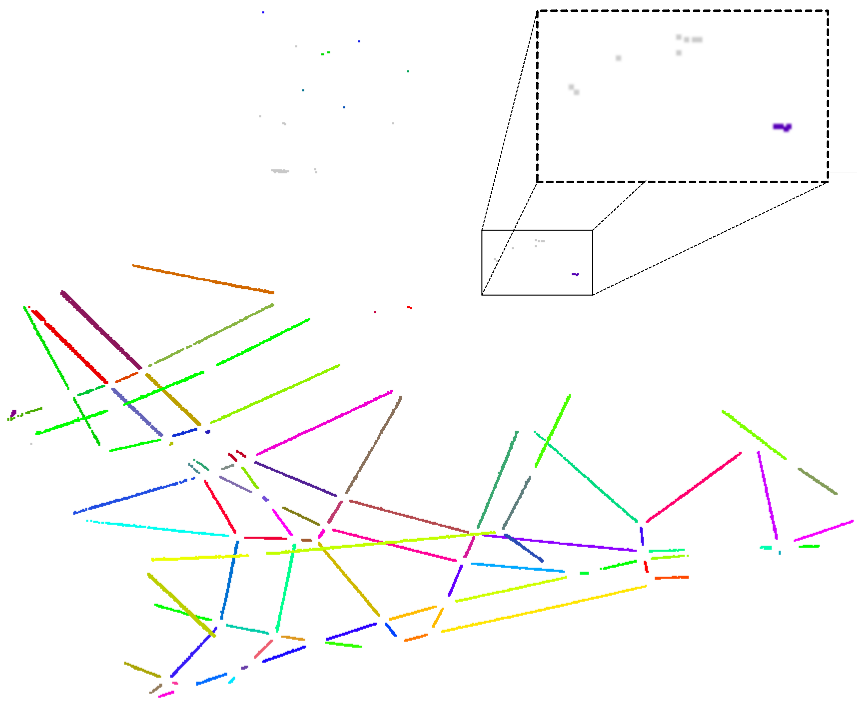


Figure 13: Candidate cable points clustered into segments.

5.2.4 Cluster growing

The clustering of candidate cable points is followed by growing of each cable segment in both directions. The goal of growing is to retrieve cable points that were missed in the previous steps. In most cases, these missed cable points have points from other objects nearby that influence the geometrical linearity feature. To retrieve some of the missed points close to the already identified cable cluster, we perform the following procedure for each cable cluster.

First, the endpoint of a cluster is identified using the main direction of the cable; let us call the endpoint P_{end} whose directional vector is defined by $\mathbf{v}_1$. Note, $\mathbf{v}_1$ is the already previously calculated first principal component of the local neighbourhood of cable point P_{end} (subsection 5.2.2). We use $\mathbf{v}_1$ to define the growing direction of the cable. A schematic illustration of the cluster growing is shown in Figure 14.

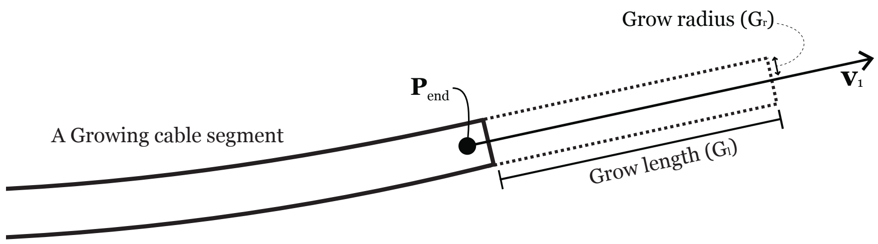


Figure 14: A schematic illustration of growing a cable cluster with endpoint P_{end} and it's growing direction vector $\mathbf{v}_1$.

Next, all unclassified points that are within a certain neighbourhood of endpoint P_{end} and fulfil the following two conditions are added to the cluster:

1. The distance of the candidate point to endpoint P_{end} is within G_l meter.
2. The candidate point is within G_r meter distance from growing direction vector $\mathbf{v}_1$.

The distance from a candidate growing point P_i to the growing vector $\mathbf{v}_1$ is known as the scalar rejection of P_j from $\mathbf{v}_1$. The formula for the scalar rejection of P_j from $\mathbf{v}_1$ is given in Equation 5.6, where θ is the angle between $\mathbf{v}_1$ and the vector from P_{end} to P_i. Figure 15 visualise the computation of the scalar rejection of P_j from $\mathbf{v}_1$.

$$\text{scalar rejection} = \|P_i - P_{end}\| \sin\theta \tag{5.6}$$

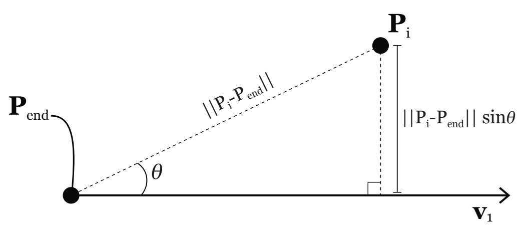


Figure 15: Illustration of the computation of the distance of point P_{end} to vector $\mathbf{v}_1$

The above-described growing algorithm is simultaneously performed on both sides of the cable clusters. In order to avoid growing the cable too far, we set G_l equal to the neighbourhood radius parameter r. The maximum distance to the growing vector (G_r) is set equal to 0.1 m (the standard deviation of MLS laser).

5.2.5 Segment merging

In this step, the over-segmented candidate cable points are merged into individual cables. In Chapter 3, we highlighted that most works utilise a catenary model to validate if cable segments belong to the same cable. Although the catenary model takes into account the sagging of a suspended cable, it does not work well when one or more objects are attached to the cable, the weight significantly influences the sagging making a good model fit impossible. See Figure 16 for an illustration of the influence of an attachment on the cable's sagging. To overcome this problem, we propose an algorithm that merges cable segments based on three conditions.

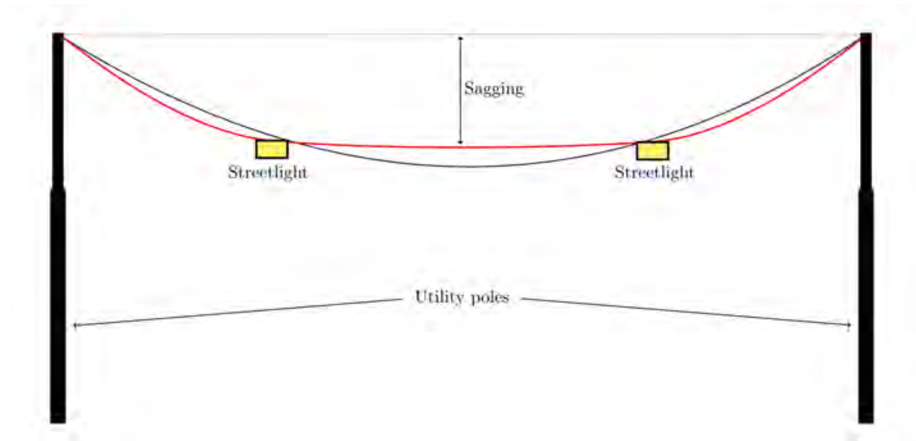


Figure 16: An illustration of an attachment's weight influencing the sag of a cable.

The merging is executed as follows. First, all cable segments are sorted in descending order based on the number of points they contain $C = \{C_1, C_2, .., C_m\}$, where C_1 is the largest cluster and C_m the smallest cluster. Next, we sequentially pair each cluster with all remaining clusters $C_{rem} = \{C_{j=i+1:m}\}$. The pairing follows ascending order based on the distance between the clusters, where the most nearby clusters are validated first. We determine if two clusters belong to the same cable by testing three conditions:

1. The difference between horizontal directions of the cable segments should be less than 5°.
2. The distance between the clusters (merge distance) should be less than 5 meters.
3. The endpoint of the cluster should be in line with the directional axis of the other cluster.

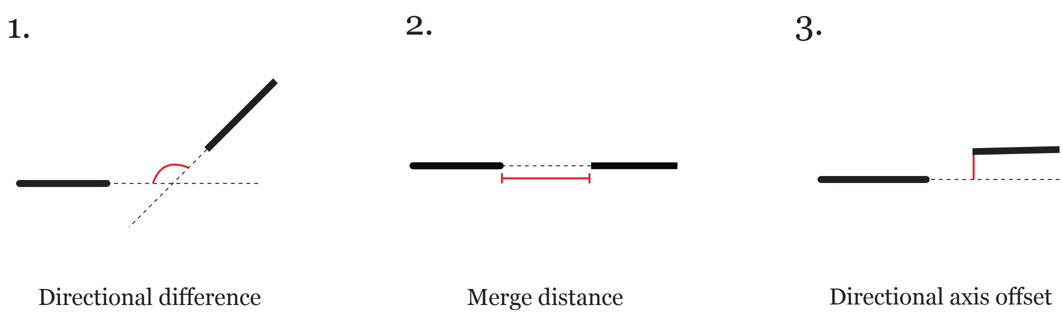


Figure 17: Schematic illustration of the three merging conditions for two cable clusters.

As illustrated in Figure 17, the first condition checks if the horizontal directions of the two clusters are similar, the difference should not exceed 5°. The horizontal direction of a cluster is calculated using two endpoints of the cluster. The second condition limits the Euclidean distance between two clusters to be at most 5 meters. This prevents the merging of clusters that are on the same line but are too far separated to belong to the same cables. The third condition tests if the endpoint of a cluster is in line with the directional axis of the other cluster. Note, that we only use the last 1.5 m of a cable to compute the directional axis to have a more accurate direction.

If a cluster pair satisfies all three conditions, the cluster j is added to cluster i ($C_i \leftarrow C_i \cup C_j$), and C_j is removed from C. Subsequently, the missing part between the clusters is reconstructed using a non-linear curve fit on the last 1.5 meters of both cable clusters. All points within 0.1 m (standard deviation of the MLS scanner) from the fitted curve are added to cluster C_i as well. Finally, cable clusters with horizontal length less than 2 meters are discarded. This value is chosen based on the minimum one-lane road width in Amsterdam. An example of the merging of over-segmented cables is shown in Figure 18.

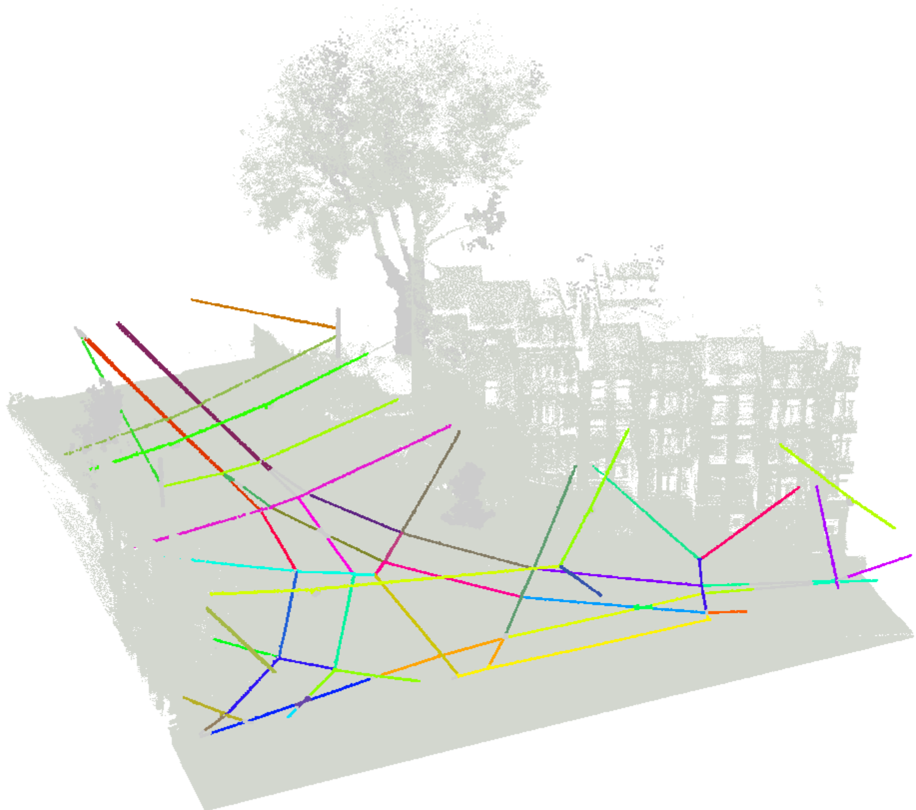


Figure 18: An illustration of the merged cable clusters. Each individual cable is shown in a different colour.

5.3 Cable classification

In this step, the identified cable clusters are classified into tram cables and non-tram cables using information from the BGT dataset. We classify the cables into tram cables and non-tram cables because the tram cables cannot carry any attachments. The tram's pantograph slides on the tram cable to feed the tram of electrical power. Section 5.4 will use this information to filter out tram cables in the search for cable attachments. The BGT data is used to identify cables located above tram tracks. The next paragraph describes the classification procedure.

The BGT dataset contains the location of all tram tracks in the Netherlands. Each tram track is stored as a set of (x,y) coordinates forming a line. First, all tram tracks within the boundaries of the point cloud are extracted from the BGT dataset. Subsequently, each tram track line is inflated by 1.25 meters to make sure the complete width of the tram track is covered. Next, we check for each cable cluster if it intersects with one of the inflated tram tracks in terms of (x,y) coordinates. To prevent non-tram cables that stretch over the tracks from being marked as tram cables, a threshold value T_{min} for the minimum height of a cable is introduced. In other words, if a cable intersects the tram track corridor below T_{min}, the cable is classified as a tram cable. An illustration of the classification procedure is shown in Figure 19.

Note, T_{min} is relative to the ground surface. Hence, the z-coordinates of the cables are shifted vertically using the same procedure as in subsection 5.1.2. According to the municipality, cables stretching above tram cables have a minimum height of 9 m because of safety guidelines. The threshold value T_{min} is set to 7 m, which gives enough margin for non-tram cables in case they have an increased sag or differ from industry standards.

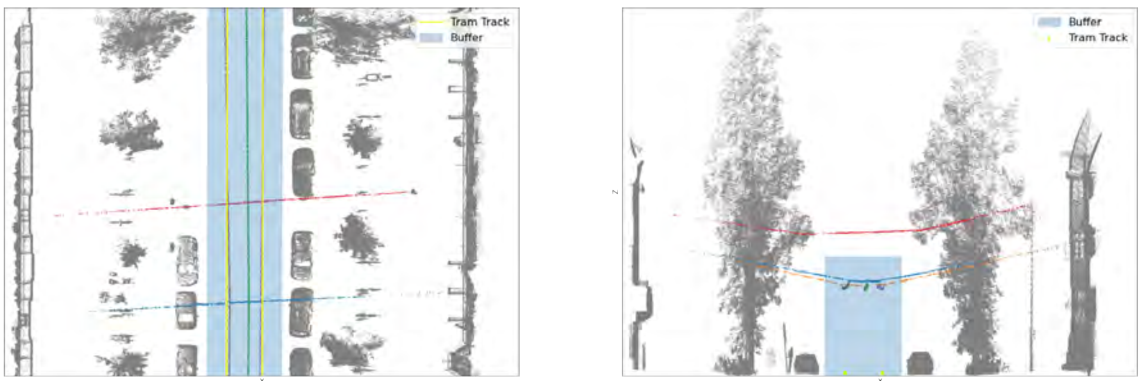


Figure 19: A top-down (left) and side-view (right) illustrations of the classification procedure. Cables in the blue buffer are classified as tram cables.

5.4 Suspended streetlight extraction

In the final and last stage, the identified non-tram cables are analysed for attachments. Subsequently, a bounding box analysis is performed on the extracted attachments to determine if the dimensions fall within the ranges of a streetlight.

We assume cable attachments such as streetlights to be a cluster of points located below the cable due to gravity. Furthermore, the sagging of the cable should include a small bending at the point of connection with the attachment. However, the number of other objects in close proximity to cables makes it a challenging task to distinguish attachments from other objects. For instance, a cable that stretches through a tree can have foliage below the cable that creates a cluster of points below the cable. Another challenge is the loosely fastened power cables attached to suspension cables which create noise directly below the cable and next to the attachments. The cable attachments are detected by dividing the area surrounding the cable into rectangular slices of 0.3x2x1 m (LxHxW) and analysing each individual slice for attachments as shown in Figure 20.

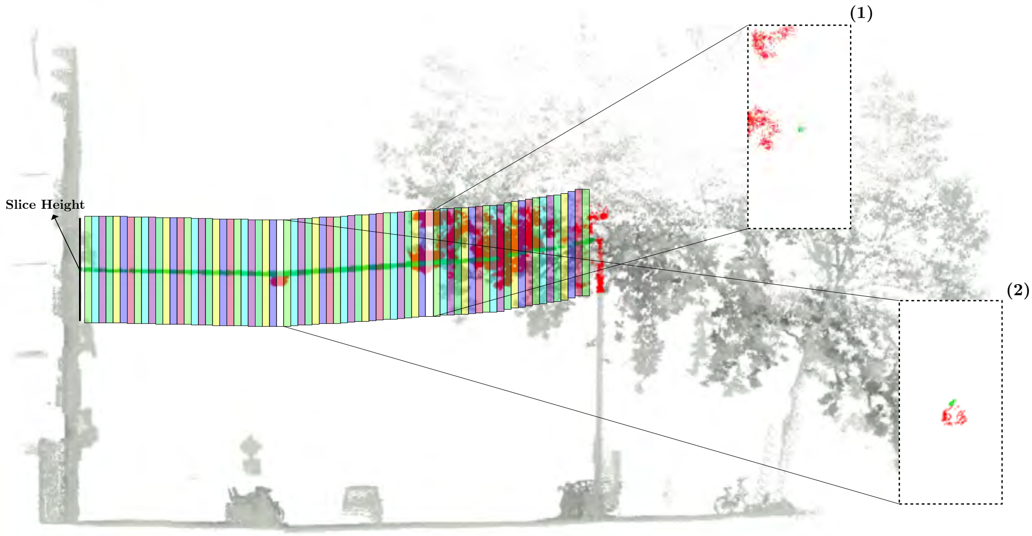


Figure 20: An illustration of the division of the cable surroundings into slices. (1) shows a slice for a part of the cable surounded by tree foliage. (2) shows a slice for a part of the cable where a streetlight is attached.

For each cable slice, we cluster the contained non-cable points based on Euclidean distance. We identify a cluster as an attachment if the following two conditions are met:

1. The center of the bounding box of the cluster is located directly below the cable.
2. The bounding box dimensions of the cluster are within the margins of a suspended streetlight.

These two conditions will help distinguish cable attachments from vegetation and other objects that are in close proximity to cables. The first condition allows us to filter out objects that have a similar shape but are not located directly below the cable. The second condition allows to filter out clusters that are located below the cable but that are too small or big to be considered a streetlight. An illustration of the conditions is shown in Figure 21. In this thesis, the margins of the bounding box dimension of a streetlight are set empirically using the training pools (Section 6.3). The maximum horizontal offset from the cable to the cluster center is set to 15 cm.

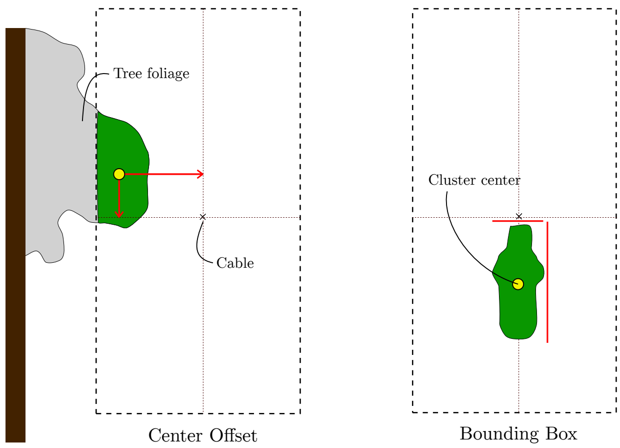


Figure 21: Schematic illustration of the condition that identify attachments in a cable slice.

6 Results and Discussion

In this chapter, the results of the proposed pipeline are presented and discussed. The pipeline was implemented in *Python 3.8* and executed on a laptop with 1.7 GHz Intel Core i7 with 8GB RAM. The code is available in the public GitHub repository[1].

6.1 Experimental data

The proposed pipeline is evaluated on a selection of the Amsterdam point cloud dataset. A total of 35 tiles were manually selected so that different types of street scenes with suspending streetlights are represented in the evaluation dataset. We do not prefer a random selection because of the following two reasons. Firstly, a random selection will include too few suspended streetlights, since most of the city has streetlights attached to poles. Secondly, a manual selection allows us to incorporate different types of urban street scenes which makes the pipeline more robust (i.e., narrow streets, road-crossings, old town areas). The ground truth of the selected tiles is manually labelled using *CloudCompare*[2] software. In total, 95 suspending streetlights were present in the evaluation selection.

The 35 tiles were randomly split into two groups: train and test. We consider 25 tiles 'training tiles' and use them to optimise the input parameters of the proposed pipeline. The training tiles are randomly divided into five pools over which we average the results, which allows us to include the error of uncertainty in the measurements. The remaining ten tiles are used to evaluate the pipeline's performance using optimal parameters on unseen data. Table 3 shows the distribution of objects in each pool. We can see each pool has about 26 million points and roughly 14 streetlights. The cable points account for about 0.25% percent of the total points in the pools and streetlight points for about 0.02%. In general, the distribution of the cables and streetlights among the pools is balanced. Finally, we include one tile without suspending streetlights and cables in the test selection to verify the pipeline's robustness to false positives.

6.2 Evaluation

The output of the pipeline is evaluated in terms of efficiency and validity. The validity provides a detailed understanding of the quality of the pipeline's output. The detected points are compared to the ground truth to measure the quality. We evaluate the pipeline's output both point-wise and object-wise. The point-wise quality is measured using precision, recall, and intersection over union (IoU) (Equation 6.1), which are commonly used metrics to evaluate point-wise point cloud labelling. We evaluate the object-wise quality using recall and precision. If the centroid of a detected streetlight is within the bounding box of a true streetlight, the detected streetlight is considered true positive. The computational efficiency will be measured in terms of the time needed to process 1 million points.

$$\text{Precision} = \frac{\text{Total number of true positives}}{\text{Sum of true positives and false positives}} = \frac{TP}{TP + FP} \tag{6.1}$$

[1] https://github.com/Amsterdam-Internships/UPC_Suspended_Streetlight_Extractor

[2] CloudCompare: http://www.cloudcompare.org/

Table 3: Data distribution in evaluation set

Dataset	Points	Cables		Suspending Streetlights		
		Points (#)	Points (%)	Points (#)	Points (%)	Objects (#)
Trainset						
Pool 1	21 584 723	59 461	0.28	3 515	0.02	10
Pool 2	27 102 063	72 055	0.27	5 946	0.02	17
Pool 3	26 124 756	58 247	0.22	4 184	0.02	13
Pool 4	30 090 791	60 737	0.20	6 093	0.02	13
Pool 5	24 672 902	63 278	0.26	3 902	0.02	13
Total	129 575 235	313 785	0.24	23 640	0.02	66
Testset	44 852 411	55 489	0.14	7 900	0.02	29

$$\text{Recall} = \frac{\text{Total number of true positives}}{\text{Sum of true positives and false negatives}} = \frac{TP}{TP + FN} \tag{6.2}$$

$$\text{IoU} = \frac{\text{Total number of true positives}}{\text{Sum of true positives, false positives, and false negatives}} = \frac{TP}{TP + FP + FN} \tag{6.3}$$

6.3 Parameter selection

The output of the pipeline depends on multiple input parameters. Hence, setting the parameters appropriate is essential for good performance. All input parameters of the pipeline are summarised in Table 7. The parameters are set based on either industry standards, literature, empiric analysis, or sensitivity analysis.

The search space reduction stage has input parameters I, H_{min}, and H_{max}. In this thesis, I defines the inflation of the building outlines and is set to 50 cm based on previous work by Bloembergen and Eijgenstein (2021). The separation heights in the vertical segmentation filtering are defined by H_{min} and H_{max}. H_{min} is set to 4 m based on industry standards and H_{max} is set to an increased value of 15 m using empiric analysis on the maximum height of cables in the training selection.

In the cable extraction stage, the candidate cable point selection step has the parameters neighbourhood radius (r), linearity threshold (L_t), and slope threshold (A_t). These parameters are influential and have a lot of impact on the performance because they identify the seed points of the cables. If too few seed points of a cable are recognised, the cable might not be extracted. Using a too-large radius r can decrease the identified cable points and result in missing cables. An inappropriate linearity threshold value L_t can identify non-cable points as seed points for cables and thus increase false positives. For these two parameters, a sensitivity analysis is used to test the impact of different parameter combinations and determine their optimal values.

The sensitivity analysis is performed using a randomised search and validated using the training tiles. For r, we sample between 0.2-1.2 m. For L_t, we sample values between 0.7-0.98. The considered ranges were selected based on preliminary experiments. We prefer a randomised search over a grid search because it is more efficient for lower dimension parameter optimisation (Bergstra and Bengio, 2012). We investigated 64 randomly selected parameter combinations and validate the identified candidate cable points compared to ground truth using average IoU over the training pools. As can be seen in Figure 22, the majority of parameter combinations achieve an average IoU of roughly 60% on the pools. However, the parameter combination of a small radius with a high threshold value achieves low IoU scores. The reason for this could be the sparsity and precision of the data. For a small radius, there are too few points to accurately compute the linearity feature of a point. The best parameter combination is located roughly at $r = 0.5$ and $L_t = 0.9$ and will be used as optimal parameter values. The remaining parameters in the cable extraction stage were set based on trail-and-error and area-defined standards.

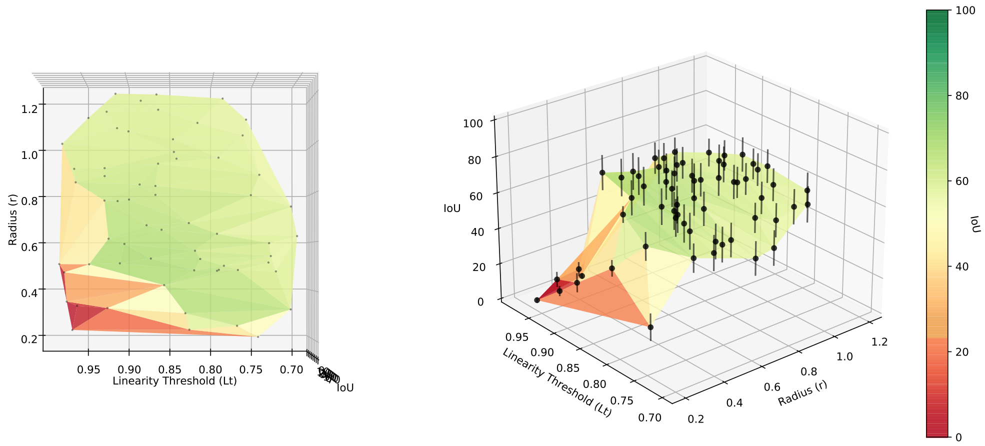


Figure 22: Parameter space for random search on neighbourhood radius (r) and linearity threshold (L_t). The vertical error bars indicate the uncertainty among the 5 training pools.

In the final suspended streetlight detection stage, inaccurate threshold values for the dimension of the attachment can result in both false negatives and false positives. We empirically obtain the maximum and minimum bounding box dimensions of the streetlights using the dimensions of the streetlights in the training pools. As can be seen in Figure 23, the height of a suspended streetlight ranges between 0.2 m and 0.9 m, and the width ranges between 0.35 m and 0.95 m. To account for the margin of error we added 5 cm to the bounding box. Note, that there is a lot of variation among the bounding box dimensions of the suspended streetlights in the point cloud Figure 23. Part of the variation is caused by the margin of error of the point cloud, the remaining part is caused by the use of different types of streetlights within the city.

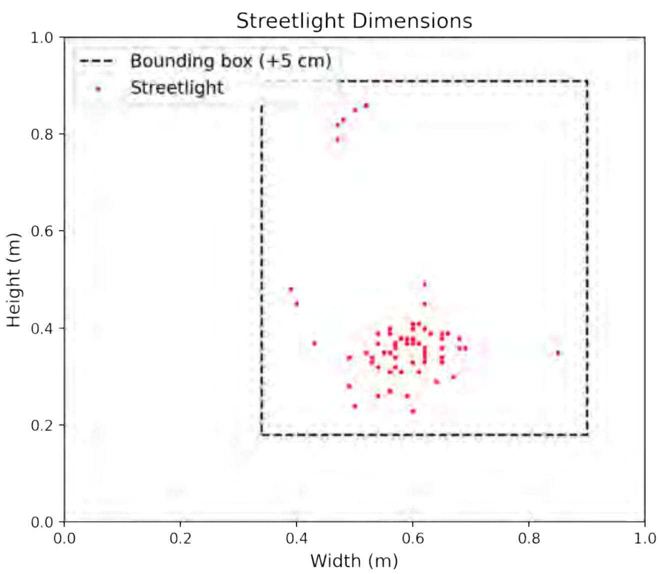


Figure 23: Bounding box dimensions of the streetlights in the training tiles.

6.4 Results

With the above obtained optimal parameters and methods, we analyse the performance of the pipeline on the 10 test tiles. Table 4 shows the results of the pipeline on the test tiles. The pipeline's outputs and the corresponding ground truths are visualised in Figure 29.

Table 4: The quality results of the proposed pipeline on the test tiles

Object Type	**Point-wise**			**Object-wise**		
	IoU (%)	**Precision (%)**	**Recall (%)**	**Precision (%)**	**Recall (%)**	**Total**
Cables	89.80	98.70	90.87	-	-	-
Suspending Streetlights	81.03	98.70	81.9	100.00	82.76	25/29

6.4.1 Cable extraction

In total, we achieve a point-wise IoU of 89.80% for cable points on the test tiles. The precision of the pipeline is 98.70%, which is near-optimal and similar to studies discussed in Chapter 3. The recall of 90.87% on the cable points is a bit lower. The lower recall of the cable points is mainly due to edge cases. An example is shown in Figure 29j in which a cable stretches over the tile's boundaries and is too short to be detected. One approach to overcome this problem is to extend the tile with padding to include valuable information for objects crossing the border of a tile. Another challenge for the pipeline was loose power cables attached to the main suspension cable. These loose cables hang very close to the main suspension cables which makes them difficult to detect. An example where the pipeline fails to extract these cables is shown in Figure 24.

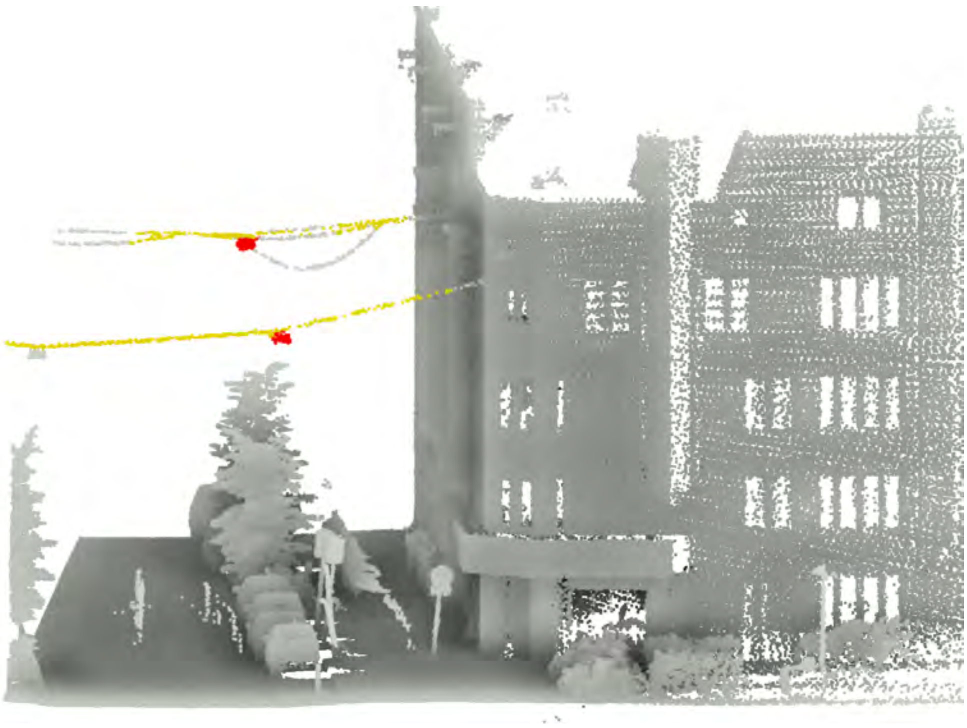


Figure 24: An example of unrecognised suspended streetlight and cable points in tile 2380_9731. The detected suspended streetlight and cable points are shown in red and yellow respectively.

As can be seen in Figure 25, the pipeline successfully extracts the cable part that stretches through the tree foliage. Despite the high precision, the recall for tile 2426_9765 is rather low, 63.14% (Table 8). A visual inspection shows that the relatively low recall is mainly caused by the extremely

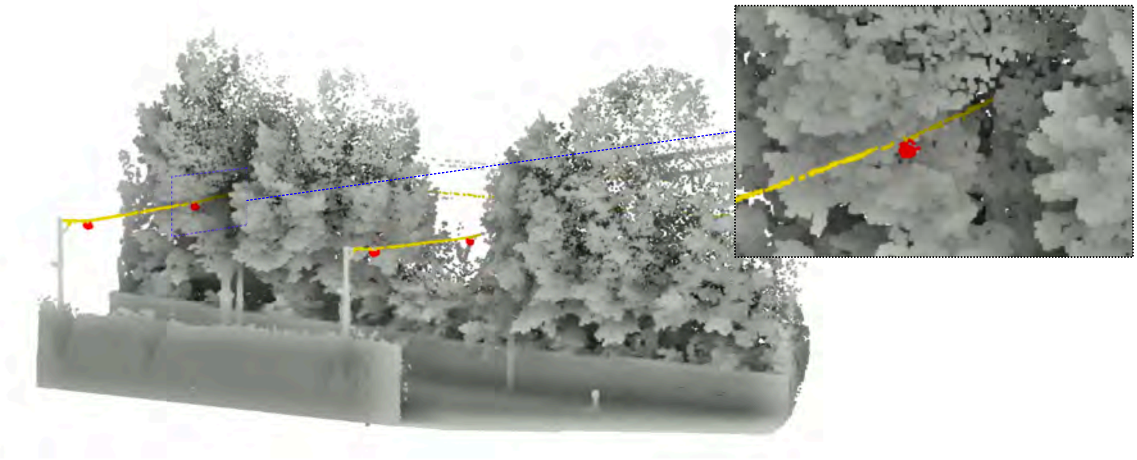


Figure 25: The output result of tile 2426_9765. Detected cables are shown in yellow and suspended streetlights in red.

sparse cables belonging to the occluded railroad corridor next to the road (Figure 29f). If we would ignore the occluded railroad corridor, the recall would be significantly higher.

The high precision and recall indicate our proposed pipeline is performing well in extracting cables from urban point clouds. We can conclude that by first identifying linear points in a reduced point cloud followed by clustering, growing, and re-clustering steps, the pipeline can successfully extract various types of cables from urban point cloud data. The extraction algorithm has the limitation that it fails to detect cables at the edge of the tile and lose cables attached to the main suspension cable.

6.4.2 Suspended streetlight detection

For the point-wise detection of suspended streetlights, the pipeline achieves an 81.90% recall, 98.70% precision and 81.03% IoU (Figure 29). Object-wise, we successfully detect 24 out of 29 streetlights in the test set and do not observe any false positives. The 5 missing streetlights were divided over 3 tiles. 3 of the 5 undetected streetlights are located on the edge of the tiles. An example is shown in Figure 26. As previously discussed, padding can be used to overcome this problem. The other two undetected streetlights are located in tile 2430_9732 and use the same suspension cable. Figure 27 displays the two missed streetlights. We observe that the streetlights are attached to a different type of cable than usual.

The evaluation shows that our proposed algorithm can detect suspended streetlights with high precision in urban point clouds. We did not identify any false positives, which indicates our proposed method is robust and is capable to distinguish cable attachments from other objects in close proximity to cables. However, some streetlights at the borders of the tiles were left undetected. We conclude that by performing a bounding box analysis on vertical slices of the cable's surroundings, suspended streetlights can be detected successfully.

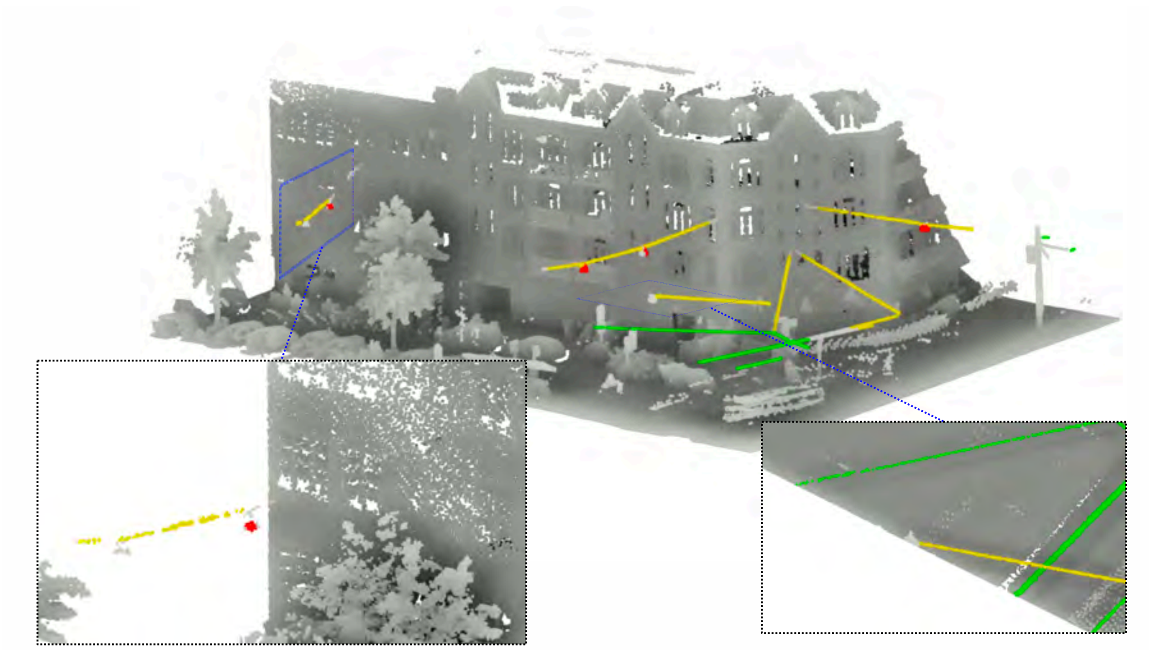


Figure 26: The extraction results of test tile 2381_9742. The two missed suspended streetlights during extraction are highlighted.

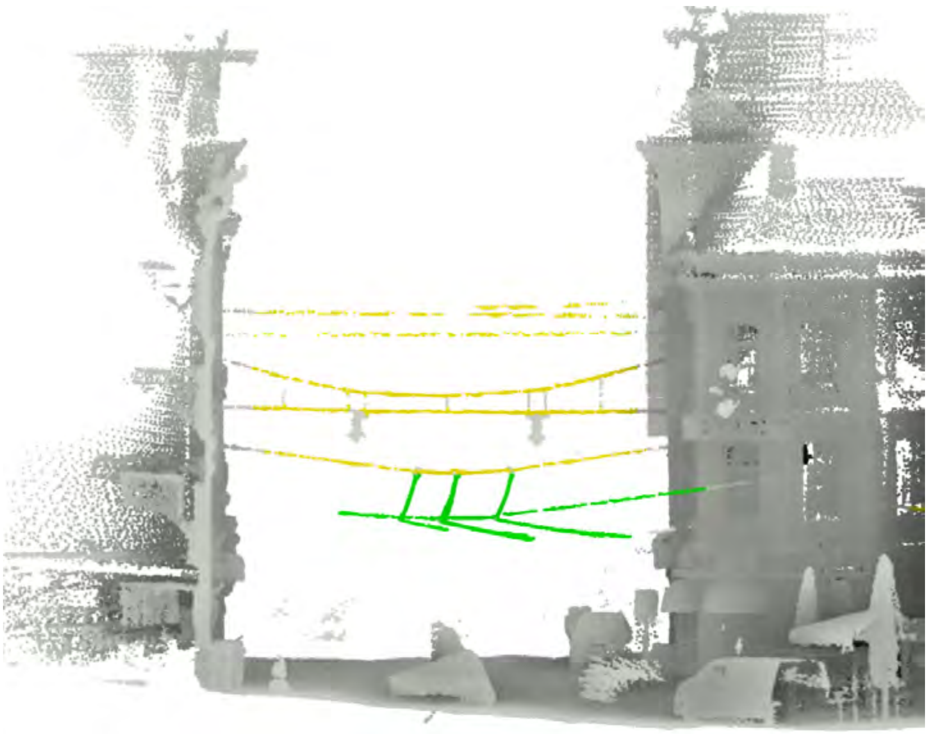


Figure 27: The extraction results of the pipeline for test tile 2430_9732.

6.4.3 Efficiency

The processing times of the pipeline on the test tiles with optimised parameters are shown in Table 5. We can see that the efficiency ranges between 0.31-1.60 million points per second. For most tiles, the vertical segmentation filter and cable extraction steps are the most time-consuming. The least amount of time is spent at the cable classification stage, it was close to 0 for most tiles. This is not surprising, because not all tiles have tram tracks present.

Table 5: Pipeline processing times per stage in seconds with optimal parameters.

Data	**Building filter**	**Vertical segmentation**	**Cable extraction**	**Cable classification**	**Streetlight detection**	**Total**	**Efficiency (million points/sec.)**
Tile 2380_9731	0.16	2.76	0.50	0.01	0.27	3.70	0.44
Tile 2381_9743	0.40	2.82	1.81	0.10	2.12	7.25	0.62
Tile 2437_9747	0.39	3.51	4.15	0.00	1.16	9.22	0.78
Tile 2430_9732	0.76	2.83	1.72	0.05	3.98	9.35	0.71
Tile 2429_9716	0.38	2.81	1.60	0.03	0.39	5.20	0.82
Tile 2438_9746	0.45	1.55	2.00	0.00	0.84	4.83	0.64
Tile 2416_9745	0.66	2.09	3.97	0.00	1.27	7.99	0.94
Tile 2417_9745	0.54	1.27	0.17	0.00	0.70	2.69	1.60
Tile 2352_9744	0.33	3.72	0.76	0.02	0.49	5.32	0.56
Tile 2426_9765	0.21	4.38	3.79	0.02	0.76	9.15	0.31

The impact of the search space reduction step on the point cloud in terms of considered points for the search is shown in Table 6. The results show that filters remove 90.27-99.97% of the original scanned points, which is significant. Depending on the scene, both filters can remove significant amounts of points. For instance, 61.06% of the points in tile 2430_9732 were removed by the building filter and 38.70% by the vertical segmentation filters, together the filters removed 99.76% of the original scanned points. Note, Tile 2426_9765 has the lowest reduction percentage and almost no building points were removed. If we visually inspect Tile 2426_9765 (Figure 29f), we can see that the scene mainly consists of trees and no buildings. This agrees with the results we see.

Table 6: The effect of the search space reduction filters on the point cloud in terms of points and (percentage).

Data	**No. points**	**Search Space Reduction**		**Reduced Search Space**
		Building Filter	**Vertical Segmentation**	
Tile 2380_9731	1 611 814	555 593 (34.47%)	1 049 925 (65.14%)	6 296 (0.39%)
Tile 2381_9743	4 486 544	1 745 340 (38.90%)	2 654 252 (59.16%)	86 952 (1.94%)
Tile 2437_9747	7 152 363	1 474 253 (20.61%)	5 188 728 (72.55%)	489 382 (6.84%)
Tile 2430_9732	6 682 169	4 080 201 (61.06%)	2 586 078 (38.70%)	15 890 (0.24%)
Tile 2429_9716	4 271 030	2 109 466 (49.39%)	2 145 260 (50.23%)	16 304 (0.38%)
Tile 2438_9746	3 112 487	2 393 677 (76.91%)	683 913 (21.97%)	34 897 (1.12%)
Tile 2416_9745	7 476 139	3 978 106 (53.21%)	3 148 224 (42.11%)	349 809 (4.68%)
Tile 2417_9745	4 297 217	3 226 500 (75.08%)	1 069 376 (24.89%)	1 341 (0.03%)
Tile 2352_9744	2 960 039	457 956 (15.47%)	2 461 027 (83.14%)	41 056 (1.39%)
Tile 2426_9765	2 802 609	48 151 (1.72%)	2 484 529 (88.65%)	269 929 (9.63%)

We conclude that by introducing a building and vertical segmentation filter, the search space of urban point clouds can be reduced by +90%. Furthermore, the efficiency of the pipeline ranges between 0.31-1.60 million points per second, which is reasonable for application on a large-scale point cloud dataset by taking into account the possibility of parallel computing.

6.5 Comparison with existing

The suspended streetlight detection step is a novel approach, and there are no existing methods to compare to. However, a few studies investigated the automated extraction of cables in urban scenes and achieved similar results to ours. Jung et al. (2020) developed an efficient and robust power line extraction pipeline with precision and recall rates of 93.39-96.76% and 82.58-97.65%. They achieved an efficiency of 0.81-1.46 million points per second. Shokri et al. (2021) proposed an algorithm for the automatic extraction of poles and cables from MLS point clouds with precision of 100% and recall of 95.3% on cables, no processing times were reported. Our proposed cable extraction stage achieves precision and recall of 98.7% and 90.87% respectively and has an efficiency of 0.31-1.60 million points per second. Unlike previous research, we mostly focused on urban street scene point clouds, which are more complex due to the various number of objects in close proximity to each other. Furthermore, our research included intersecting cables and cables with attachments, unlike previous studies which mostly focused on catenary cables (i.e., power lines).

6.6 Extensive study

In this section, we test the applicability of our proposed pipeline on large scale. An entire neighbourhood in the city of Amsterdam is processed by our pipeline. The 93 tiles account for 352.7 million points and cover an area of 0.23 km^2, which is about 0.5% of the complete Amsterdam point cloud. The processing took 12 minutes and 12 seconds, including loading and writing the data. The extracted suspended streetlights are shown in Figure 28. We manually compared the output of the pipeline to street view images and saw that apart from two false positives the results were correct. The false positives arose in newly build buildings that were not filtered out. This highlights the limitation of the pipeline of being dependent on the external building outline data.

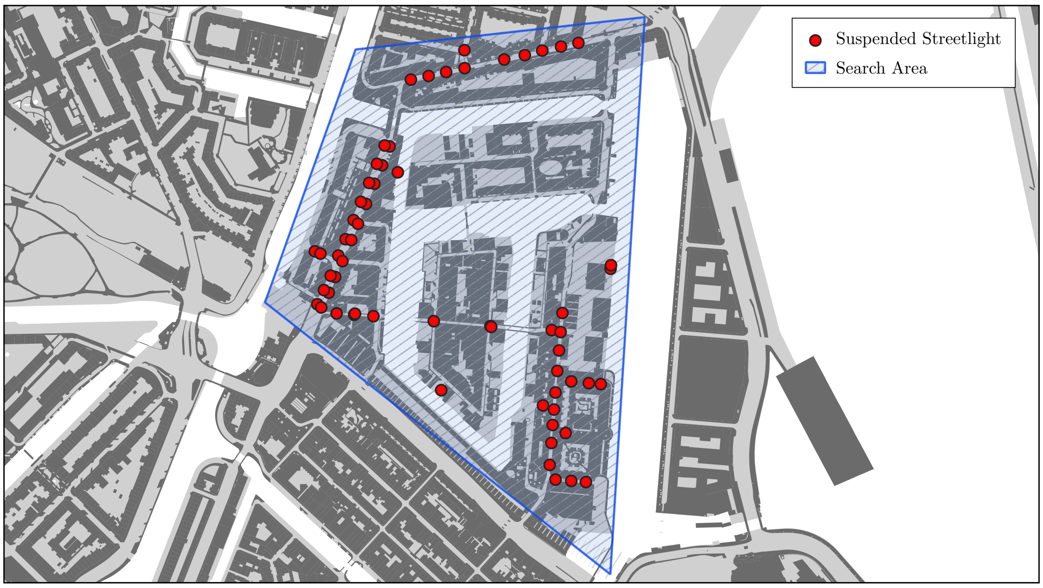


Figure 28: Results of extensive study for the extraction of suspended streetlights for the complete Westelijke Eilanden neighbourhood.

7 Conclusion and Future Works

In this chapter, we set off to answer our main research question and present future work directions.

7.1 Conclusion

In this thesis, a new method for the automatic detection of suspended streetlights in urban point cloud data has been presented. Our method consists of four stages. The first stage reduces the search space using a building and vertical segmentation filter. The filters dramatically decrease the number of points in the point clouds using external data sources. The second stage extracts cables from the reduced point cloud using a point-based analysis followed by clustering, growing, and merging steps. The third stage classifies the previously extracted cables into tram cables and non-tram cables by fusing the location of tram tracks with the point cloud. The fourth and final stage analyses the classified non-tram cables for suspended streetlight attachments based on bounding box conditions. The proposed pipeline obtained IoU, precision and recall rates of 81.03%, 98.70%, and 81.9% for the suspended streetlights points. Object-wise, we managed to detect 24 out of 29 streetlights. In addition, we evaluated the cable extraction stage in terms of point-wise quality and achieved IoU, precision and recall rates of 89.80%, 98.70% and 90.87% respectively. Based on the results we can conclude that suspended streetlights and overhead utility cables can be automatically extracted from urban scene point clouds with high precision and recall.

Compared to other methods, the cable extraction stage achieves similar results to approaches presented in related works. Regarding the type of extracted cables, our study area contained cables with attachments which have not been considered in other studies. To the best of our knowledge, we are the first to propose a method that can automatically detect suspended streetlights in point cloud data.

Although our proposed method is capable of detecting suspended streetlights, there are some limitations that should be considered. One of the limitations is that the search space reduction step utilises external data sources. The efficiency of the pipeline depends on the quality and availability of these sources. Another limitation is that the pipeline is vulnerable to error propagation. The quality of the cable extraction stage can affect streetlight detection.

All in all, it has been shown that suspended streetlights and cables can be detected and extracted from urban point clouds with high precision and recall. Furthermore, the low processing time allows the automatic extraction pipeline to be implemented on a large scale. The implementation makes it possible to map suspended streetlights in the city on a regular basis, which was not feasible to do manually. In other words, the proposed pipeline can be used by municipalities to regularly update the streetlight asset and extend the (semi-)automated urban point cloud annotation pipeline.

7.2 Future works

An important direction of future work is to investigate whether the extraction of suspended streetlights and cables is generalisable by a machine learning classifier. Machine learning models require a lot of data to train on. However, the available labelled point cloud data is limited. The proposed pipeline can be used to create a rich labelled dataset to train. Furthermore, the pipeline can be used to extend the already existing automatic point cloud labelling pipeline by Amsterdam with 2 additional classes.

Besides the capability to map suspended streetlights in urban point clouds, the pipeline has some interesting potential applications that can be further investigated. Examples of possible applications within the municipality are:

- A road headroom map for the city. Drivers can consult this map to see what the maximum headroom is for specific routes using the operating height of suspended cables.
- Monitor the state of suspended cables by comparing the sag to previously observed values.
- Hazard analysis for electric-powered tram cables.

Another future work direction is to improve the pipeline in terms of quality and efficiency. The results on processing times indicated that the vertical segmentation filters take up a lot of time. Furthermore, different approaches for the candidate cable point selection can be investigated. For instance, a hybrid combination between k-nearest neighbour and radius search or using a simple rectangular box. For the computation of the eigenvalues, a modified version of the classical PCA can be used like the less outlier-sensitive robust PCA.

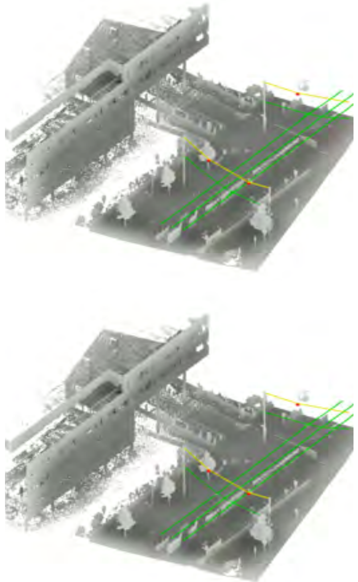
Appendix

Table 7: Summary of parameters used in the pipeline

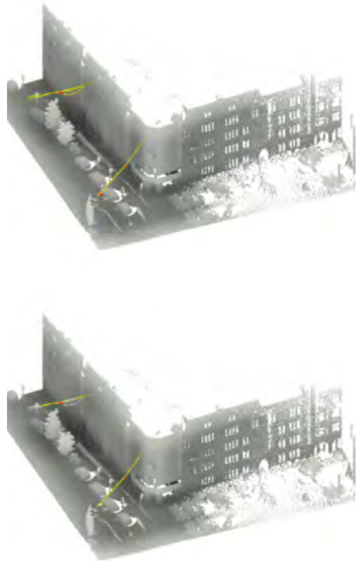
Stage	**Parameter**	**Symbol**
Search space reduction	The inflation of the building outline	I
	Separation height for low-height and medium-height	H_{min}
	Separation height for medium-height and high-height	H_{max}
Cable extraction	Voxel-size	s
	Neighbourhood radius	r
	Minimum distance between clusters	d
	Maximum grow length	G_l
	Maximum grow radius	G_r
Cable classification	The buffer used to inflate tram-tracks	T_w
	Threshold value for the minimum cable height	T_{min}
Streetlight Detection	Minimal streetlight width	SW_{min}
	Maximal streetlight width	SW_{max}
	Minimal streetlight height	SH_{min}
	Maximal streetlight height	SH_{max}

Table 8: The quality results of the proposed pipeline on the 10 test tiles using optimal parameters.

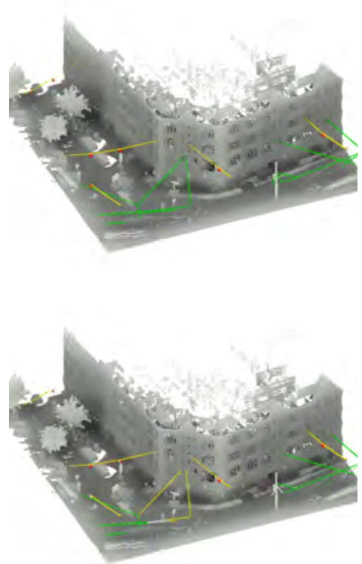
Data	**Object**	**Point-Based**			**Object-Based**		
		IoU	**Precision (%)**	**Recall (%)**	**IoU**	**Precision (%)**	**Recall (%)**
Tile 2381_9743	Cables	82.70	99.61	82.97	-	-	-
	Suspending Streetlights	63.54	97.54	64.57	71.43	100.0	71.43
Tile 2426_9765	Cables	55.22	97.71	63.14	-	-	-
	Suspending Streetlights	90.95	99.54	91.33	100.0	100.0	100.0
Tile 2429_9716	Cables	90.94	95.04	95.48	-	-	-
	Suspending Streetlights	96.17	99.41	96.72	100.0	100.0	100.0
Tile 2438_9746	Cables	72.77	98.31	73.69	-	-	-
	Suspending Streetlights	93.21	98.65	94.41	100.0	100.0	100.0
Tile 2416_9745	Cables	85.22	95.39	88.89	-	-	-
	Suspending Streetlights	95.45	99.9	95.54	100.0	100.0	100.0
Tile 2417_9745	Cables	83.25	98.64	84.22	-	-	-
	Suspending Streetlights	96.88	98.84	97.99	100.0	100.0	100.0
Tile 2352_9744	Cables	97.98	100.0	97.98	-	-	-
	Suspending Streetlights	94.88	95.16	99.69	100.0	100.0	100.0
Tile 2430_9732	Cables	93.03	99.71	93.28	-	-	-
	Suspending Streetlights	60.44	99.27	60.71	50.0	100.0	50.0
Tile 2437_9747	Cables	97.55	99.85	97.69	-	-	-
	Suspending Streetlights	-	-	-	-	-	-
Tile 2380_9731	Cables	40.6	99.22	40.73	-	-	-
	Suspending Streetlights	33.44	98.15	33.65	66.67	100.0	66.67



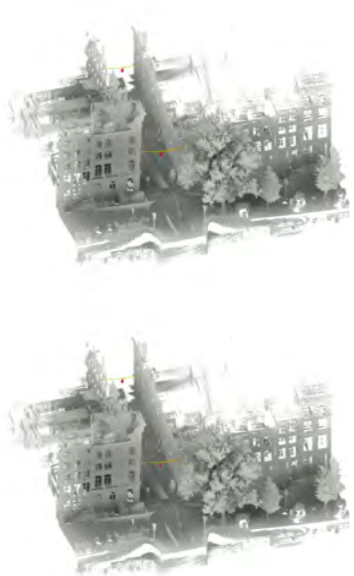
(a) Tile 2352_9744



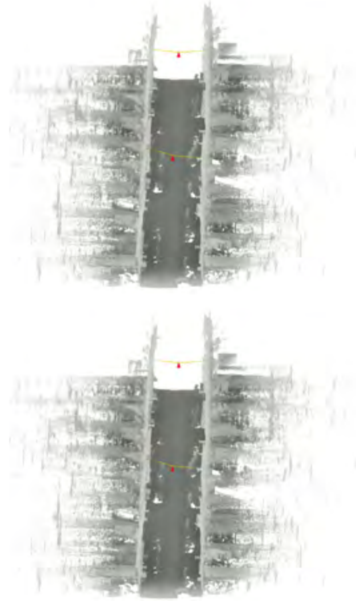
(b) Tile 2380_9731



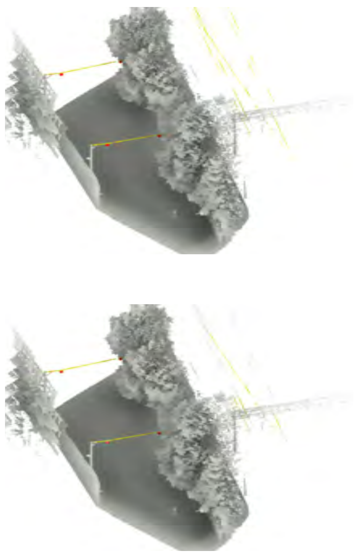
(c) Tile 2381_9743



(d) Tile 2416_9745

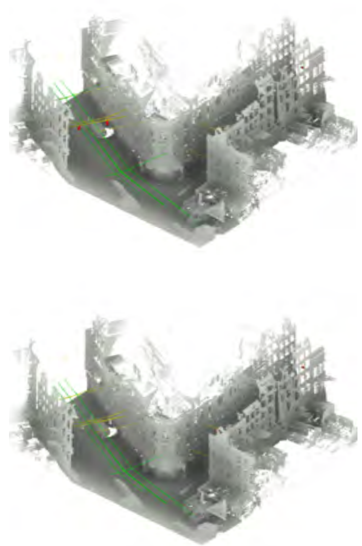


(e) Tile 2417_9745

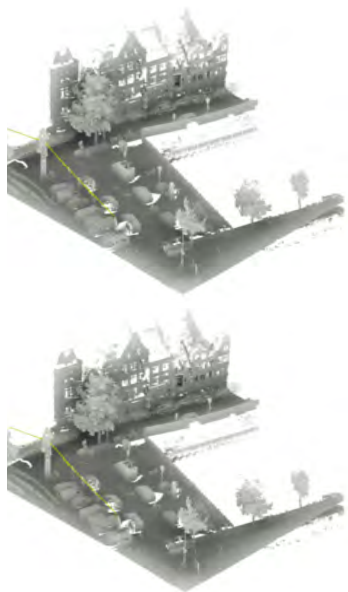


(f) Tile 2426_9765

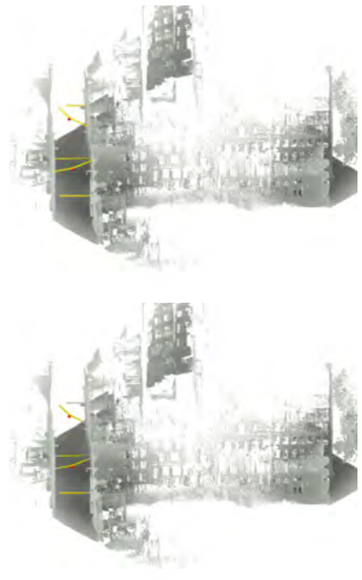
(g) Tile 2429_9716



(h) Tile 2430_9732



(i) Tile 2437_9747



(j) Tile 2438_9746

Figure 29: Extraction results of test set selection using optimal parameters. For each tile, the top image represents the labelled ground truth and the bottom the the output. Each type of object is shown in a different colour: suspended streetlights (red), utility cables (yellow), and tram cables (green).

Bibliography

Arastounia, M. (2017). An enhanced algorithm for concurrent recognition of rail tracks and power cables from terrestrial and airborne lidar point clouds. *Infrastructures*, 2(2):8.

Awrangjeb, M. (2019). Extraction of power line pylons and wires using airborne lidar data at different height levels. *Remote Sensing*, 11(15):1798.

Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305.

Bloembergen, D. and Eijgenstein, C. (2021). Automatic labelling of urban point clouds using data fusion. *arXiv preprint arXiv:2108.13757*.

Cabo, C., Ordóñez, C., López-Sánchez, C. A., and Armesto, J. (2018). Automatic dendrometry: Tree detection, tree height and diameter estimation using terrestrial laser scanning. *International journal of applied earth observation and geoinformation*, 69:164–174.

Che, E., Jung, J., and Olsen, M. J. (2019). Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review. *Sensors*, 19(4):810.

Cheng, L., Tong, L., Wang, Y., and Li, M. (2014). Extraction of urban power lines from vehicle-borne lidar data. *Remote Sensing*, 6(4):3302–3320.

Cserép, M., Hudoba, P., and Vincellér, Z. (2018). Robust railroad cable detection in rural areas from mls point clouds. In *Free and Open Source Software for Geospatial (FOSS4G) Conference Proceedings*, volume 18, page 2.

Fang, H. and Lafarge, F. (2019). Pyramid scene parsing network in 3d: Improving semantic segmentation of point clouds with multi-scale contextual information. *Isprs journal of photogrammetry and remote sensing*, 154:246–258.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395.

Guan, H., Yu, Y., Li, J., Ji, Z., and Zhang, Q. (2016). Extraction of power-transmission lines from vehicle-borne lidar data. *International Journal of Remote Sensing*, 37(1):229–247.

Guo, B., Huang, X., Zhang, F., and Sohn, G. (2015). Classification of airborne laser scanning data using jointboost. *ISPRS Journal of Photogrammetry and Remote Sensing*, 100:71–83.

Hackel, T., Wegner, J. D., and Schindler, K. (2016). Fast semantic segmentation of 3d point clouds with strongly varying density. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, 3:177–184.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., and Markham, A. (2021). Learning semantic segmentation of large-scale point clouds with random sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1.

Husain, A. and Vaishya, R. C. (2019). An automated method for power line points detection from terrestrial lidar data. In *Emerging Technologies in Data Mining and Information Security*, pages 459–472. Springer.

Jung, J., Che, E., Olsen, M. J., and Shafer, K. C. (2020). Automated and efficient powerline extraction from laser scanning data using a voxel-based subsampling with hierarchical approach. *ISPRS Journal of Photogrammetry and Remote Sensing*, 163:343–361.

Kim, H., Sohn, G., et al. (2010). 3d classification of power-line scene from airborne laser scanning data using random forests. *Int. Arch. Photogramm. Remote Sens*, 38:126–132.

Lehtomäki, M., Kukko, A., Matikainen, L., Hyyppä, J., Kaartinen, H., and Jaakkola, A. (2019). Power line mapping technique using all-terrain mobile laser scanning. *Automation in Construction*, 105:102802.

Li, F., Lehtomäki, M., Elberink, S. O., Vosselman, G., Kukko, A., Puttonen, E., Chen, Y., and Hyyppä, J. (2019). Semantic segmentation of road furniture in mobile laser scanning data. *ISPRS journal of photogrammetry and remote sensing*, 154:98–113.

Ma, L., Liu, Y., Zhang, X., Ye, Y., Yin, G., and Johnson, B. A. (2019). Deep learning in remote sensing applications: A meta-analysis and review. *ISPRS journal of photogrammetry and remote sensing*, 152:166–177.

Maguya, A. S., Junttila, V., and Kauranne, T. (2014). Algorithm for extracting digital terrain models under forest canopy from airborne lidar data. *Remote Sensing*, 6(7):6524–6548.

Maneewongvatana, S. and Mount, D. M. (1999). Analysis of approximate nearest neighbor searching with clustered point sets. *arXiv preprint cs/9901013*.

Melzer, T. and Briese, C. (2004). Extraction and modeling of power lines from als point clouds.

Munir, N., Awrangjeb, M., and Stantic, B. (2019). An automated method for individual wire extraction from power line corridor using lidar data. In *2019 Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE.

Pu, S., Rutzinger, M., Vosselman, G., and Elberink, S. O. (2011). Recognizing basic structures from mobile laser scanning data for road inventory studies. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6):S28–S39.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv preprint arXiv:1706.02413*.

Qin, X., Wu, G., Ye, X., Huang, L., and Lei, J. (2017). A novel method to reconstruct overhead high-voltage power lines using cable inspection robot lidar data. *Remote sensing*, 9(7):753.

Rusu, R. B., Marton, Z. C., Blodow, N., Dolha, M., and Beetz, M. (2008). Towards 3d point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941.

Safaie, A. H., Rastiveis, H., Shams, A., Sarasua, W. A., and Li, J. (2021). Automated street tree inventory using mobile lidar point clouds based on hough transform and active contours. *ISPRS Journal of Photogrammetry and Remote Sensing*, 174:19–34.

Sánchez-Rodríguez, A., Soilán, M., Cabaleiro, M., and Arias, P. (2019). Automated inspection of railway tunnels' power line using lidar point clouds. *Remote Sensing*, 11(21):2567.

Shan, J. and Toth, C. K. (2018). *Topographic laser ranging and scanning: principles and processing*. CRC press.

Shi, Z., Kang, Z., Lin, Y., Liu, Y., and Chen, W. (2018). Automatic recognition of pole-like objects from mobile laser scanning point clouds. *Remote Sensing*, 10(12):1891.

Shi, Z., Lin, Y., and Li, H. (2020). Extraction of urban power lines and potential hazard analysis from mobile laser scanning point clouds. *International Journal of Remote Sensing*, 41(9):3411–3428.

Shokri, D., Rastiveis, H., Sarasua, W. A., Shams, A., and Homayouni, S. (2021). A robust and efficient method for power lines extraction from mobile lidar point clouds. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 89(3):209–232.

Shokri, D., Rastiveis, H., Shams, A., and Sarasua, W. (2019). Utility poles extraction from mobile lidar data in urban area based on density information. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Suárez-González, A. (2020). Automatic extraction of power cables location in railways using surface lidar systems. *Sensors*, 20:6222.

Tajima, K. and Masuda, H. (2020). Extraction of road-crossing power and communication lines from mobile mapping data. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 5(2).

Tan, W., Qin, N., Ma, L., Li, Y., Du, J., Cai, G., Yang, K., and Li, J. (2020). Toronto-3d: A large-scale mobile lidar dataset for semantic segmentation of urban roadways. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). Kpconv: Flexible and deformable convolution for point clouds. *Proceedings of the IEEE International Conference on Computer Vision*.

Vosselman, G. and Maas, H.-G. (2010). *Airborne and terrestrial laser scanning*. CRC press.

Wang, C. and Glenn, N. F. (2009). Integrating lidar intensity and elevation data for terrain characterization in a forested area. *IEEE Geoscience and Remote Sensing Letters*, 6(3):463–466.

Wang, Y., Chen, Q., Liu, L., Zheng, D., Li, C., and Li, K. (2017). Supervised classification of power lines from airborne lidar data in urban areas. *Remote Sensing*, 9(8):771.

Wu, B., Yu, B., Wu, Q., Yao, S., Zhao, F., Mao, W., and Wu, J. (2017). A graph-based approach for 3d building model reconstruction from airborne lidar point clouds. *Remote Sensing*, 9(1):92.

Wu, B., Yu, B., Yue, W., Shu, S., Tan, W., Hu, C., Huang, Y., Wu, J., and Liu, H. (2013). A voxel-based method for automated identification and morphological parameters estimation of individual street trees from mobile laser scanning data. *Remote Sensing*, 5(2):584–611.

Xu, H., Gossett, N., and Chen, B. (2007). Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics (TOG)*, 26(4):19–es.

Yadav, M. and Chousalkar, C. G. (2017). Extraction of power lines using mobile lidar data of roadway environment. *Remote Sensing Applications: Society and Environment*, 8:258–265.

Yadav, M., Husain, A., Singh, A. K., and Lohani, B. (2015). Pole-shaped object detection using mobile lidar data in rural road environments. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2.

Yan, K., Hu, Q., Wang, H., Huang, X., Li, L., and Ji, S. (2022). Continuous mapping convolution for large-scale point clouds semantic segmentation. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5.

Yang, B., Fang, L., and Li, J. (2013). Semi-automated extraction and delineation of 3d roads of street scene from mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 79:80–93.

Yastikli, N. and Cetin, Z. (2021). Classification of raw lidar point cloud using point-based methods with spatial features for 3d building reconstruction. *Arabian Journal of Geosciences*, 14(3):1–14.

Yen, K. S., Ravani, B., Lasky, T. A., et al. (2011). Lidar for data efficiency. Technical report, Washington (State). Dept. of Transportation. Office of Research and Library ....

Yi, C., Zhang, Y., Wu, Q., Xu, Y., Remil, O., Wei, M., and Wang, J. (2017). Urban building reconstruction from raw lidar point data. *Computer-Aided Design*, 93:1–14.

Yu, Y., Li, J., Guan, H., Wang, C., and Cheng, M. (2012). A marked point process for automated tree detection from mobile laser scanning point cloud data. In *2012 International Conference on Computer Vision in Remote Sensing*, pages 140–145. IEEE.

Yu, Y., Li, J., Guan, H., Wang, C., and Yu, J. (2014). Semiautomated extraction of street light poles from mobile lidar point-clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 53(3):1374–1386.

Yu, Y., Li, J., Guan, H., Wang, C., and Yu, J. (2015). Semiautomated extraction of street light poles from mobile lidar point-clouds. *IEEE Transactions on Geoscience and Remote Sensing*, 53(3):1374–1386.

Zermas, D., Izzat, I., and Papanikolopoulos, N. (2017). Fast segmentation of 3d point clouds: A paradigm on lidar data for autonomous vehicle applications. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5067–5073. IEEE.

Zhang, S., Wang, C., Yang, Z., Chen, Y., and Li, J. (2016a). Automatic railway power line extraction using mobile laser scanning data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci*, 41:615–619.

Zhang, W., Qi, J., Wan, P., Wang, H., Xie, D., Wang, X., and Yan, G. (2016b). An easy-to-use airborne lidar data filtering method based on cloth simulation. *Remote sensing*, 8(6):501.