Eötvös Loránd University

Faculty of Informatics

Dept. of Software Technology and Methodology

Tree Segmentation using ALS Point Clouds

Supervisor:
Cserép Máté
Assistant Lecturer

Author:
Erdődi Gergely
Computer Science
for Autonomous Systems MSc

Budapest, 2024

Contents

Chapter 1

Introduction

Tree segmentation can be utilized to solve many different challenges in geoinformatics, but one of the most important may be analyzing the changes in the area covered by vegetation. Change detection plays an important role in addressing environmental challenges that influence urban life and planning. Urban expansion and industrialization have significant impacts on the area covered by vegetation, thus necessitating continuous monitoring and analysis to ensure optimal values. This study focuses on advancing the efficiency of 2D-based tree segmentation by integrating 3D point cloud data, specifically leveraging the AHN3 point clouds.

The existing methodologies for change detection and land cover classification, are often reliant on multispectral satellite imagery and aerial photography and face limitations, particularly when applied to larger urban and suburban areas. LiDAR, although a less widespread alternative due to cost and availability constraints, offers a unique advantage by generating 3D point clouds. However, the comparison of such point clouds from different epochs presents algorithmic challenges, including variations in point density and spatial location.

This research addresses these challenges by investigating the causes of false positives in the currently used method in the CloudTools [1] geospatial framework, developed at Eötvös Loránd University. An initial attempt using a morphological operation-based approach was deemed unsuitable. Subsequently, a moving window method was developed, leveraging both 2D data to find false positives in the result.

Moving beyond, this study explores the integration of 3D point cloud data into the segmentation process. This involves researching and implementing various techniques, including the utilization of pre-trained neural networks. The objective is to enhance the accuracy of tree segmentation by making use of the additional information provided by the 3D data, thus mitigating false positives.

A significant contribution of this work is the application and comparison of pre-trained neural networks for segmenting point cloud data. This step involves an evaluation of the performance of these neural networks in detecting and classifying trees, with a particular focus on their ability to reduce false positives. The integration of advanced machine learning techniques aims to further refine the accuracy and efficiency of the segmentation process.

By combining insights from 2D-based and 3D-based methods and incorporating machine learning approaches, this research not only identifies the limitations of current methodologies but also offers a potential solution for addressing false positives in large-scale tree segmentation in urban environments. The findings contribute not only to the improvement of existing techniques but also to the broader understanding of automated processing for diverse and expansive urban and suburban landscapes.

The structure of the thesis is organized in the following way. Chapter 2 is an overview of the LiDAR technology, its use cases, and related research regarding current tree detection methods. Chapter 3 describes the methodology for assessing the problem of tree detection and the different insights gained while conducting experiments. Chapter 4 describes how the different approaches were implemented. In Chapter 5 the achieved results are presented. Finally, in Chapter 6 I summarize my work.

Chapter 2

Related work

Light Detection and Ranging (LiDAR) has emerged as a revolutionary active remote sensing technology, transforming the way we measure distances between sensors and target surfaces. Pioneered in the 1960s, LiDAR gained significance for its role in mapping the Moon during space exploration. Referred to as laser scanning, LiDAR is capable of utilizing diverse light types, including ultraviolet, near-infrared, and visible light. Its applications span across disciplines, from geodesy and geography to autonomous driving, establishing itself as a pivotal tool in high-resolution mapping.

2.1 Data acquisition

The utilization of LiDAR broadly falls into two primary categories: airborne and terrestrial, selected based on considerations such as data objectives, target dimensions, and detection costs.

2.1.1 Airborne LiDAR

Airborne Laser Scanning (ALS) employs sensors attached to flying aircraft, crafting a 3D point cloud model of the target area. ALS can be of two main types, one is the multispectral full waveform, which captures the complete backscattered signal, enabling detailed characterization of complex vegetation and terrain structures, and the other is the discrete return, which acquires individual reflection points, optimizing ground mapping and elevation modeling precision in topographic studies. Scanning, typically perpendicular to the flight direction, excels in differentiating and filtering vegetation, contributing significantly to studies focused on non-vegetation object filtration. Recent advancements have not only enhanced the technology but also contributed to a notable reduction in hardware costs, making ALS more accessible for various applications across industries.

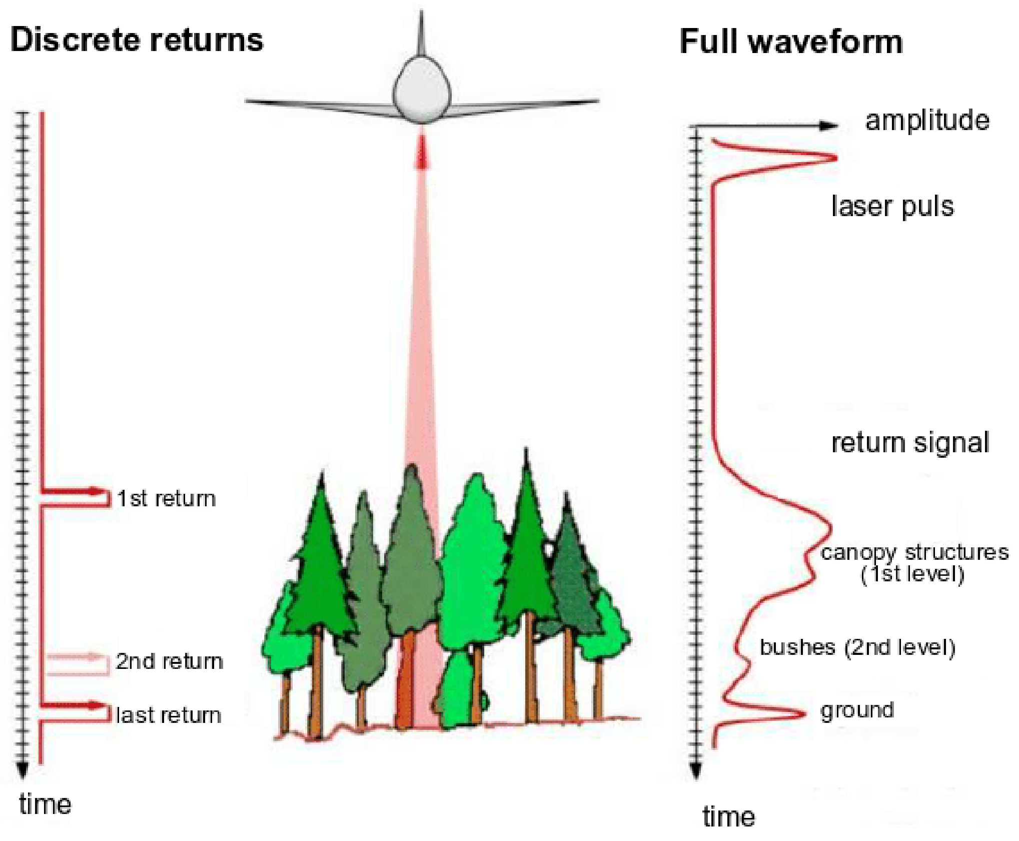

Figure 2.1: Types of Airborne Laser Scanning (ALS) [2]

2.1.2 Terrestrial LiDAR

Terrestrial Laser Scanning (TLS) is realized with sensors positioned on the Earth's surface. Stationary TLS gathers data from a fixed point, aligning point clouds with 2D images to construct realistic 3D models. It finds extensive use in surveying, particularly in construction settings where precise measurements are critical.

Mobile Terrestrial Laser Scanning (MTS) is when sensors are placed on moving vehicles for 3D modeling, which can be used to provide the necessary data for autonomous driving systems about the surrounding environment.

ALS encompasses multispectral full-waveform and discrete return LiDAR. Multispectral lidar captures laser pulses across multiple spectral bands, acquiring detailed waveform data for each pulse. This allows for spectral decomposition and comprehensive analysis of surface features and vegetation structure.

In contrast, discrete return LiDAR records specific points of reflection, emphasizing the precise timing of laser pulse returns. This approach is tailored for accurate elevation modeling and ground mapping, as it directly measures the distance between the sensor and the reflecting surface at distinct points.

2.2 Data formats and types in LiDAR processing

LiDAR technology generates voluminous data, necessitating efficient storage and processing formats. Various file formats cater to specific needs, encompassing Digital Elevation Models (DEM), Digital Surface Models (DSM), LAS, and PCL files.

2.2.1 Digital Elevation Model

An invaluable asset in using LiDAR data for geoprocessing, Digital Elevation Models (DEM) represent a 3D representation of the Earth's surface, capturing elevations at discrete points. This raster format is often stored in GeoTIFF or ASCII and provides a bare-earth model, essential for terrain analysis, hydrology, and landform characterization, also it plays an important role in topographic studies.

2.2.2 Digital Surface Model

Digital Surface Models (DSM) are a form of DEM that incorporates surface features such as buildings, vegetation, and infrastructure. Capturing both natural and man-made elements, DSMs provide a comprehensive representation of above-ground structures. DSMs are often stored in raster formats like GeoTIFF, facilitating analyses in urban planning, forestry, and infrastructure development.

2.2.3 Digital Terrain Model

Digital Terrain Models (DTM) represent the bare-earth surface by removing the above-ground features such as buildings, vegetation, and other structures, providing a detailed representation of the ground topography.

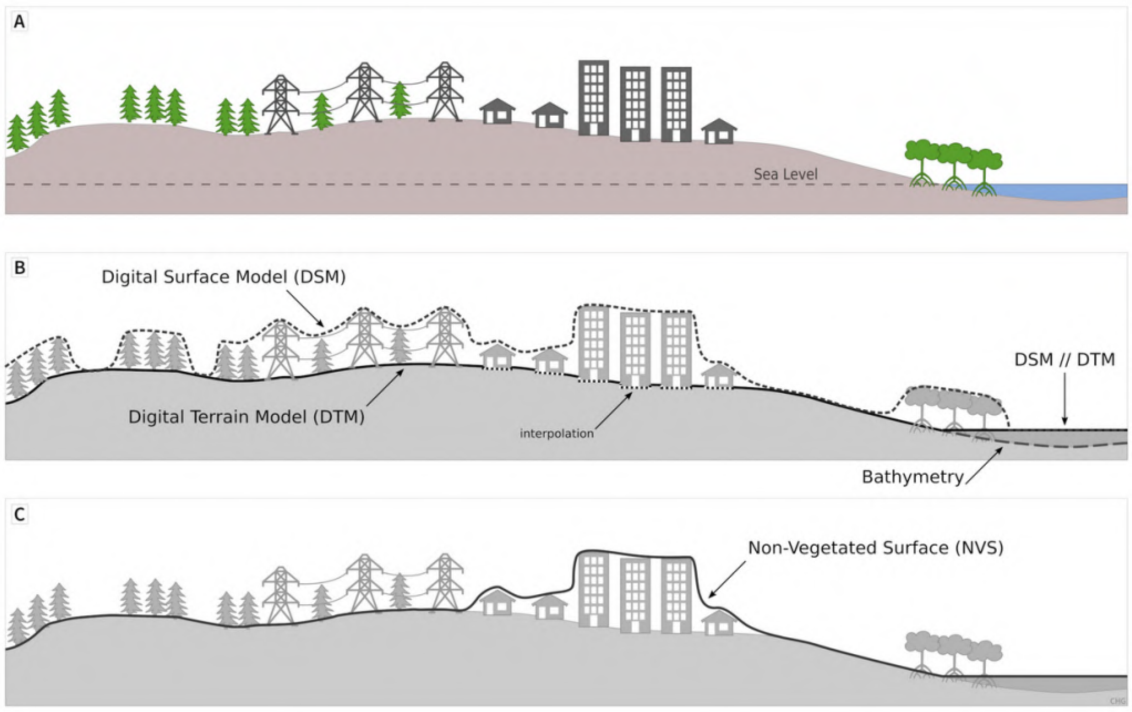
A
Sea Level
B
Digital Surface Model (DSM)
Digital Terrain Model (DTM)
interpolation
DSM // DTM
Bathymetry
C
Non-Vegetated Surface (NVS)

Figure 2.2: Types of Digital Elevation models [3]

2.2.4 LAS format

LAS files represent a standardized binary format for storing LiDAR point cloud data. Developed by the American Society for Photogrammetry and Remote Sensing (ASPRS), LAS files efficiently store millions of LiDAR points with attributes like X, Y, and Z coordinates, intensity, and classification. LAS format facilitates interoperability among different LiDAR processing software. It supports point cloud manipulation, enabling tasks such as filtering, classification, and feature extraction.

2.2.5 LASzip compression

LASzip compression (LAZ) is commonly employed to mitigate the challenges of large LAS files. LAZ reduces file sizes without compromising data integrity, enhancing storage efficiency and speeding up data transfer. LAZ retains the LAS structure, ensuring compatibility with LAS-supported software. It serves as a practical solution for managing extensive LiDAR datasets, especially when storage and bandwidth are limiting factors.

2.2.6 Point Cloud Library format

PCL files serve as a versatile format within the Point Cloud Library, an open-source project dedicated to 2D and 3D image processing. PCL files encapsulate point cloud data, supporting various data structures and algorithms for LiDAR processing.

2.3 Dataset

The AHN (Actueel Hoogtebestand Nederland) dataset is integral to our urban LiDAR research, providing a detailed view of the topography in urban areas across the Netherlands. Managed by the Dutch National Mapping Agency (PDOK), specifically AHN3, it serves as a critical resource for our scientific investigation.

Utilizing aerial LiDAR scanning, AHN delivers a detailed depiction of the Dutch urban landscape, offering essential elevation information for geospatial analysis, urban planning, and environmental monitoring.

AHN has a point density of 6-10 points/m^2 through airborne LiDAR scanning. This is lower than the point densities used in current urban segmentation models (typically 1000 to 2000 points/m^2). The dataset also has a random error of up to 15 cm. The lower point density poses a challenge, necessitating the exploration of upscaling methods to align with current urban segmentation model requirements. As we address these challenges, the AHN dataset remains crucial for refining and enhancing our approaches for efficient and accurate tree segmentation.

In the subsequent section, I will discuss methodologies used to extract meaningful insights from point clouds and DEMs.

2.4 Tree detection

Tree detection can be accomplished by many different methods, but the main differentiating factor is the type of data used. One is the approach based on Canopy Height Models which is calculated by subtracting the DTM From a DSM thus getting a good approximation of the trees' locations and heights, methods using CHM are usually fast because the data is 2D, and the size and complexity of the data on which the algorithms are relatively small.

The other method uses raw Lidar Data obtained by terrestrial or aerial scanning, these methods have much more data available, having the potential to produce much more accurate results, but the drawback is that the processing algorithms are much more complex, and the runtimes are much larger since the size of the lidar data is much more complex and voluptuous than 2D data. In the following section, I will present different existing variants of these methods and discuss their potential to be used for accomplishing this study's goal, to reliably discern trees from other objects in an urban environment.

2.4.1 CHM-based approaches

One widely adopted method in the realm of Canopy Height Models (CHMs) involves the local maxima technique, as detailed by Koch et al. [4]. In their study, the authors leverage laser scanning data to automatically delineate individual trees in deciduous and mixed temperate forests. The workflow includes the application of a local maximum filter to identify possible treetops in rasterized laser data. Following this, tree crowns are delineated through a combination of a pouring algorithm, knowledge-based assumptions on tree shapes, and final detection of crown edges by searching vectors starting from the treetops. However, it's essential to note that while the segmentation algorithm performs well for coniferous stands, challenges arise in dense stands of deciduous trees, where crown merging tendencies become more prominent.

Another noteworthy CHM-based approach is the watershed-based method, initially proposed by Beucher and Lantuéjoul in 1979 [5]. This method, rooted in contour detection, identifies edges in an image and fills basins or minimums, creating distinct segments that represent potential tree locations [6]. Yang et al. (2020) adapt the watershed algorithm for individual tree segmentation using airborne LiDAR point clouds. The method combines the watershed algorithm with three-dimensional spatial distribution analysis to achieve the delineation of individual trees.

Morphological operations, were first used for tree detection by Andersen et al. [7], their workflow involved opening and closing operations to refine the separation and outline of potential trees. In their work, they applied these operations to a LIDAR-based canopy surface model, leading to a clearer separation and more accurate outlines of potential trees.

The curvature-based method, utilizing a Gaussian filter based on dominant scales of tree crowns [8], represents another avenue for tree identification. This method enhances precision through scale-aware filtering, wherein the curvature of the point cloud is utilized to identify potential tree locations. Subsequently, a Gaussian filter is applied to refine segmentation, resulting in improved accuracy in delineating individual trees [8].

A relatively novel approach involves the use of Full-Waveform Airborne Laser Scanning (FW-ALS) data, specifically for urban tree classification [9, 10]. This method incorporates the entire waveform of the laser pulse, allowing for detailed characterization and classification of urban vegetation. Koma et al. (2016) utilize full-waveform airborne LiDAR data for object-based classification of urban trees at the taxonomic family level, showcasing the potential of this approach in complex urban environments.

2.4.2 Point cloud-based approaches

Point cloud-based algorithms encompass diverse strategies for tree detection. The top-to-bottom method excels in coniferous forests but faces diminished accuracy in broadleaf forests due to asymmetric and random growth patterns [11]. Lu et al. (2014) introduce a bottom-up method based on the intensity and 3D structure of leaf-off LiDAR point cloud data. This method demonstrates greater effectiveness in deciduous broadleaf forests, offering a nuanced approach to tree segmentation based on the intensity and three-dimensional structure of the point cloud data [12].

Density-based algorithms, such as the approach by Rahman and Gorte [13], rely on individual tree detection based on high point densities, the authors implement a density-based algorithm using the densities of high points in high-resolution airborne LiDAR, showcasing effectiveness in detecting individual trees across diverse forest environments. The emphasis lies on leveraging point density patterns to identify potential tree locations, contributing to a robust approach for tree segmentation.

Clustering algorithms play a crucial role in point cloud-based approaches. Ferraz et al.[14] employ the mean shift algorithm for the unsupervised segmentation of multi-layered Mediterranean forests. Utilizing ALS data and mean shift clustering, the study provides a robust approach for characterizing different vegetation layers. The mean shift clustering aids in identifying distinct segments within the point cloud, offering valuable insights for forest mapping and fuel estimation.

Multispectral LiDAR data has opened new possibilities for individual tree extraction. Dai et al.[15] investigate the performance of multispectral ALS data for delineating individual trees. Their workflow incorporates mean shift segmentation on different feature spaces, showcasing improved accuracy, especially in dealing with clumped tree segments in dense forests. By integrating multispectral information, the study introduces a nuanced approach to individual tree delineation, considering both geometric and radiometric features.

Maltamo et al.[16] explore the potential of laser scanner data for identifying and quantifying structural characteristics of heterogeneous boreal forests. The study focuses on discrete return laser scanner data, analyzing the height distributions of reflected laser pulses to recognize multi-layered stand structures using a histogram-based thresholding method to quantify understory trees.

2.4.3 Neural Network-based Approaches

Recent advancements in point cloud processing have witnessed the emergence of powerful neural network-based methods, showcasing impressive capabilities in semantic segmentation and classification.

Efficiently designed for semantic segmentation of large-scale point clouds, RandLA-Net by Hu et al. [17] leverages both local and global geometric context, achieving good results in semantic segmentation tasks while demonstrating efficiency and scalability in handling extensive point cloud datasets.

Introducing a flexible convolution for point clouds, KPConv by Thomas et al. [18] enhances the accuracy of previous convolutional neural networks. This adaptability to local structures allows for improved feature extraction and semantic segmentation in large-scale point clouds.

Some of the datasets on which the mentioned neural networks were trained for usage in 3D Segmentation in urban settings will be presented next.

The Paris-Lille-3D dataset [19], presented by Xavier Roynard et al., is a substantial urban point cloud dataset for automatic segmentation and classification. Acquired with the Mobile Laser Scanning (MLS) prototype L3D2, this dataset includes two parts, one in the agglomeration of Lille and one in Paris. The dataset provides 143.1 million points distributed over 1,940 meters, featuring a point density of 1000-2000 points/m^2, offering a relatively accurate representation of urban environments for segmentation and classification tasks.

The SemanticKITTI Dataset [20] serves as a benchmark for evaluating algorithms in semantic scene understanding, featuring 23,201 full 3D scans for training and 20,351 for testing. Based on the odometry dataset of the KITTI Vision Benchmark, it stands as the largest publicly available dataset for semantic segmentation in point clouds, providing rich semantic annotations for LiDAR sequences across diverse urban scenes.

Chapter 3

Methodology

The primary challenge addressed in this methodology revolves around the issue of false positives partly stemming from imperfect overlap between the Digital Terrain Model (DTM) and the Digital Surface Model (DSM). The discrepancy in the alignment of these models results in building edges not being accurately removed during the creation of the Canopy Height Map (CHM). Consequently, these inaccurately identified building edges are then erroneously interpreted as tree crowns, to mitigate this problem, three different approaches were examined.

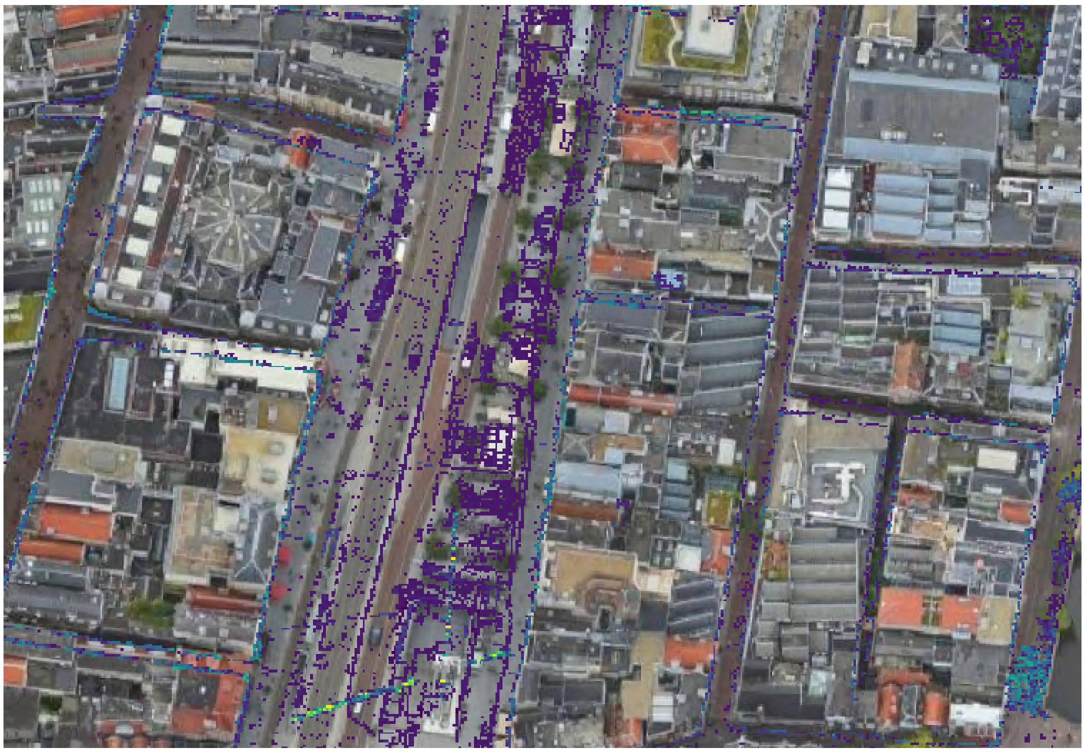

Figure 3.1: Building edges that remained in the CHM causing false positive detections

3.1 Study area

The area on which the different proposed algorithms were tested is in the central part of Amsterdam and has a size of 1.93 km squared. This area was chosen because it contains many different types of landscapes, for example rivers, streets with and without roadside vegetation, parks and buildings with different types of architecture, also there is a publicly available validation dataset which contains all of the trees situated near the main roads.

The point cloud of the area contains approximately 30 million points with an average density of 6-10 points/m^2.

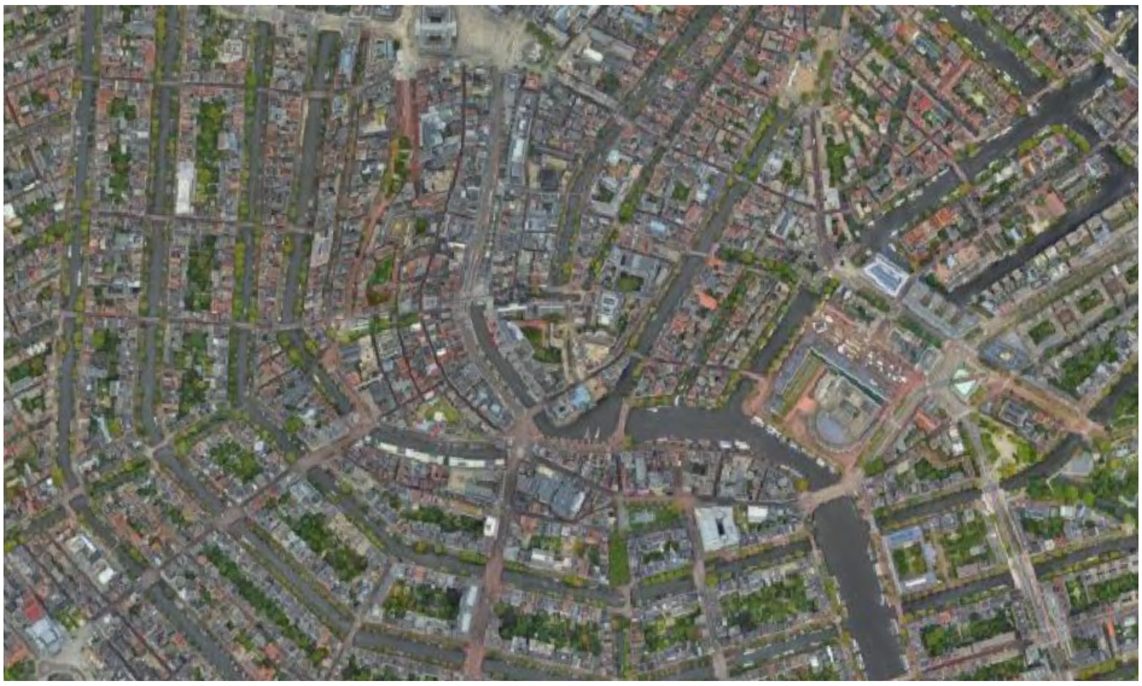

Figure 3.2: Study area in the center of Amsterdam

3.2 CHM-based approach

The first method employs a 2D strategy based on a sliding window technique. In this method, each potential tree seed, identified as a local maximum, undergoes a filtering process. Specifically, points exceeding a predefined threshold or marked as no data in the DTM are considered. The rationale is to count points around the suggested tree seed that might resemble building structures. If a substantial number of nearby points indicate a building-like structure, then the original seed point is excluded from the potential seed points, thus improving the accuracy of the identification process and speeding up the upcoming steps in the workflow.

An additional layer of refinement is introduced by creating a river mask. This mask provides a different version of the DTM, that serves the purpose of differentiating rivers from buildings. Originally both rivers and buildings were represented as no-data points in the DTM. The river mask contains a predefined value in the place of nodata points, which ensures that rivers are not wrongly treated as buildings during the filtering process.

The creation of the River Mask works the following way:

1. For each point in the DSM:
   - If the DSM point lacks data and the corresponding DTM point has data, assign it as nodata.
   - If the DSM point lacks data, assign a predefined constant value.
   - If the DTM has data, keep the value the same.

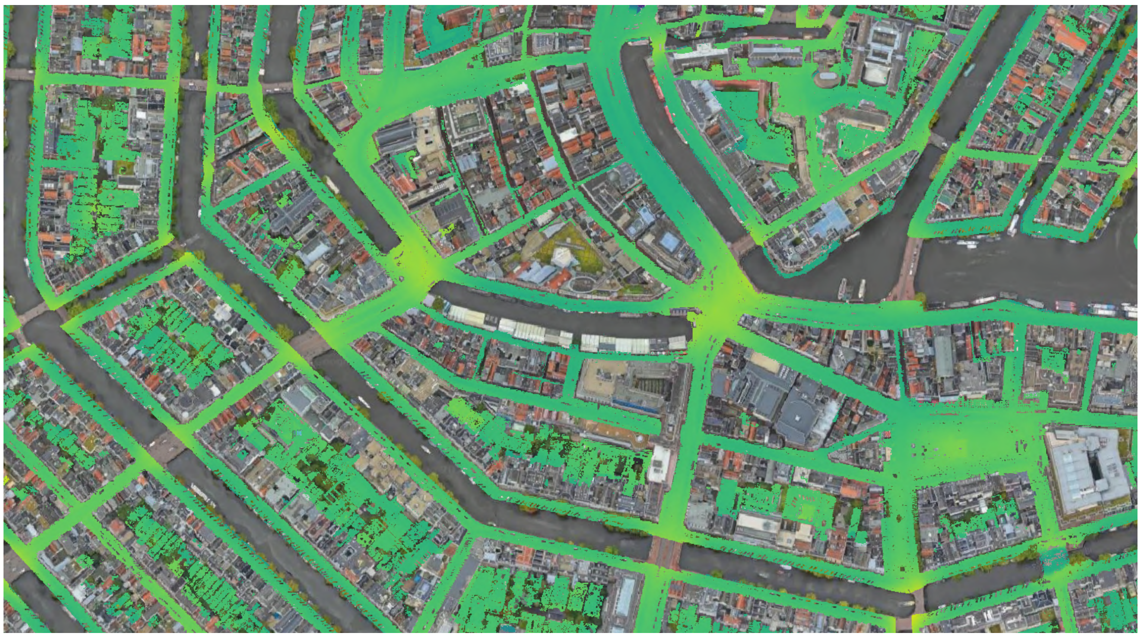

Figure 3.3: Original DTM

Then, in the next phase, the Seed Removal algorithm iterates through each potential tree seed point and examines its local neighborhood in the DSM and the newly created DTM to determine if it resembles a building structure. The key steps are as follows:

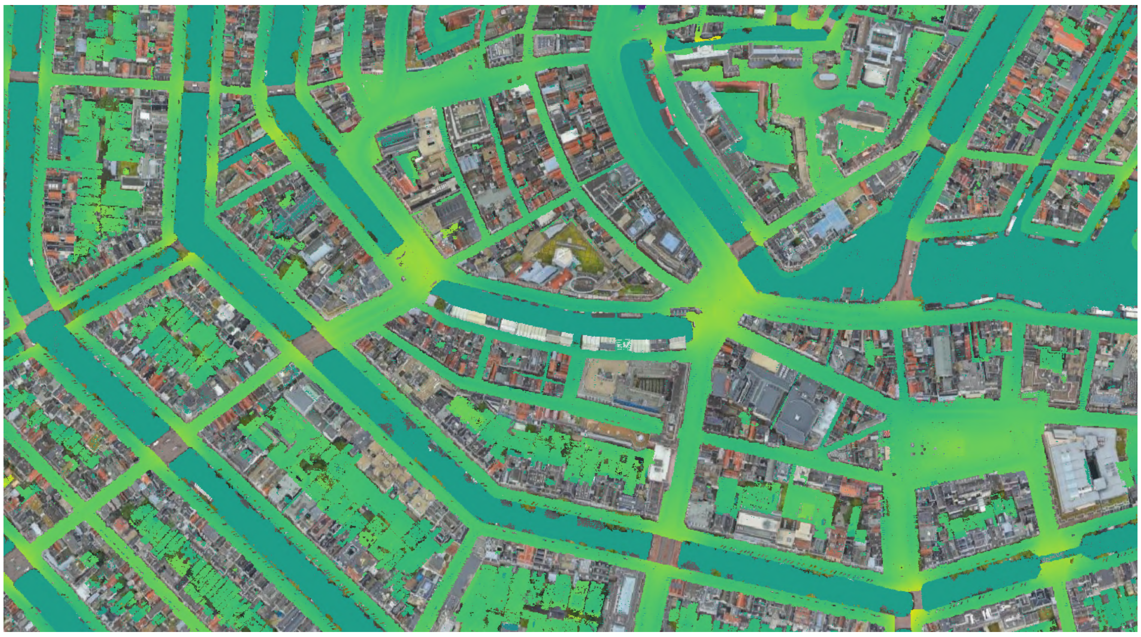

Figure 3.4: New DTM after running the River Mask algorithm

1. For each potential tree seed point in the DSM, identified as a local maximum:
   - Define a window of a specified size around the seed point.
   - Count the number of points within the window that lack data in the DTM and have values above a predefined threshold.
   - If the count exceeds a certain threshold, mark the seed point as likely corresponding to a building and add its index to a removal list.
2. Remove the identified seed points from the list, ensuring that potential building-related points are excluded from further consideration.

In summary, this 2D sliding window approach is the initial strategy employed to address the challenges associated with false positives in the current workflow. The methodology aims to enhance the precision of tree crown identification by systematically evaluating and filtering potential seed points.

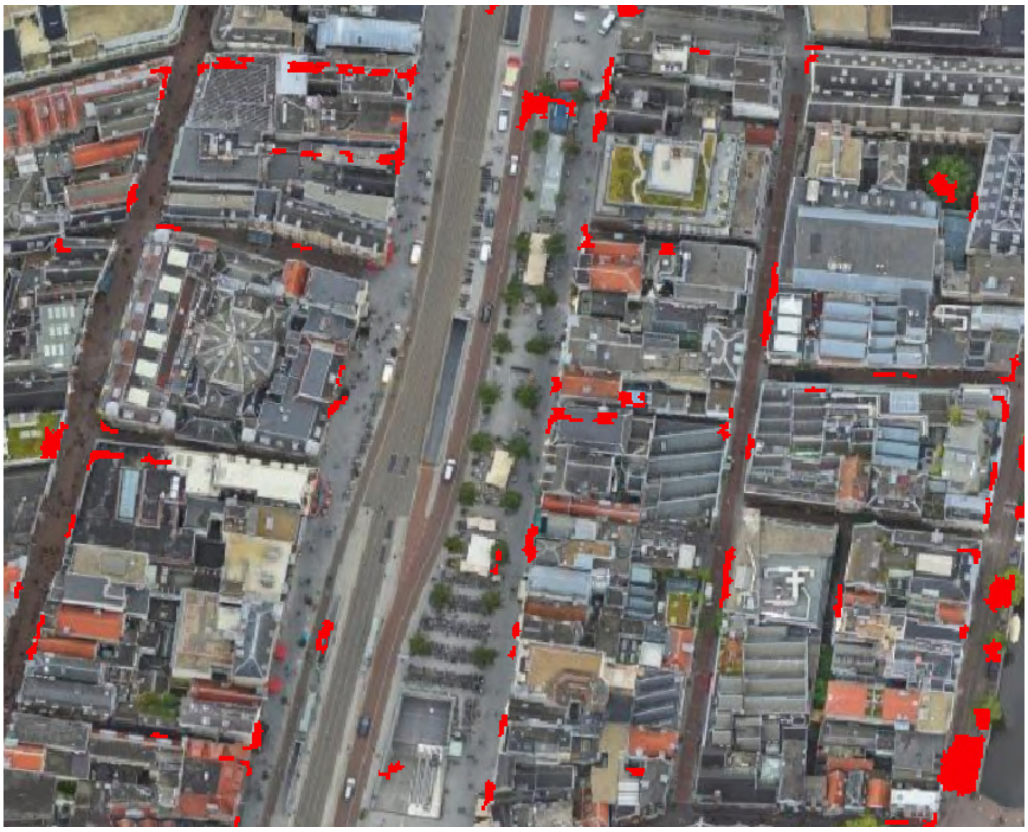

Figure 3.5: Area with many false positive seed points in central Amsterdam

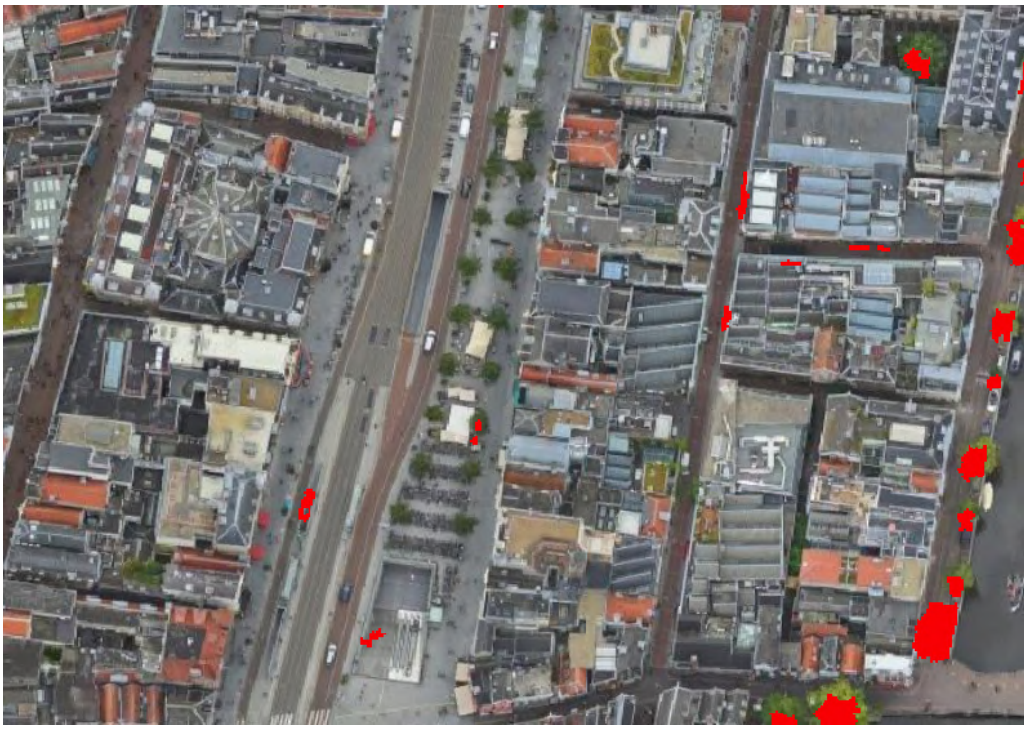

Figure 3.6: Area after running the CHM-based filtering algorithm

3.3 Density-based approach

The second method in our methodology transitions from the 2D sliding window approach to a thorough analysis of the full 3D point cloud. In this density-based approach, the objective is to refine the identification process by examining the average density function of trees within a segmented region around each seed point. The algorithm, implemented using the PCL library, introduces a more nuanced evaluation, focusing on trees' overall shape and density characteristics.

This approach initiates the segmentation of the 3D space surrounding a potential tree seed. The algorithm cuts out a region centered on the seed point and divides it into horizontal bins.

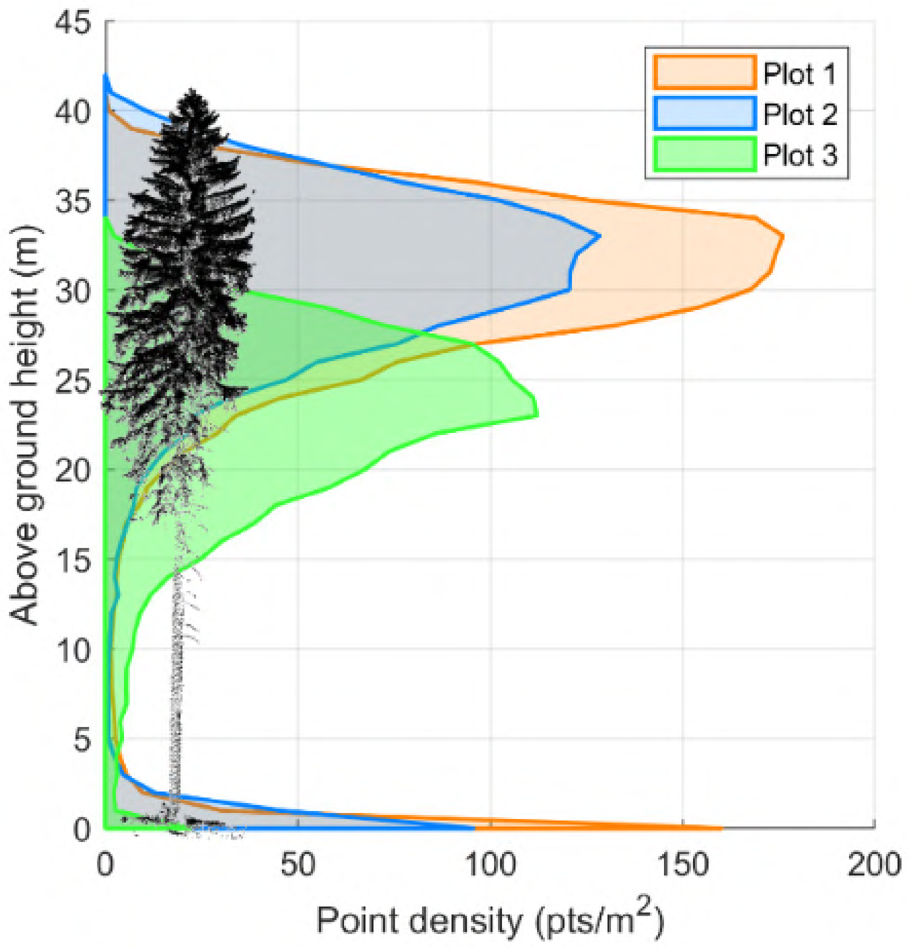

Figure 3.7: Vertical histogram of the point cloud of a tree [21]

Once the horizontal cut is complete, the next step involves the calculation of the average density function within the segmented region. Unlike the 2D method, which primarily considers height information, this approach utilizes the entire 3D point cloud data. By aggregating information from all dimensions, the algorithm captures a more comprehensive representation of the tree's density characteristics.

With the average density function calculated, the algorithm proceeds to assess the cosine similarity of the derived shape. This step evaluates how closely the average shape of the tree aligns with an expected tree shape which was derived from the validation dataset. A predefined threshold for the cosine similarity measure was set, acting as a criterion for identifying false positives.

Finally, based on the cosine similarity assessment, the algorithm performs false-positive removal. If the similarity measure falls below the established threshold, indicating a significant deviation from expected tree shapes, the identified tree is flagged as a false positive and subsequently removed from consideration.

As the trees used for the calculation of the model tree were of different species and heights, the gained average representation wasn't as accurate as needed. For this reason, an alternative method was considered which uses a more simple approach, specifically if the point density in the middle-top region is denser by a specific ratio than the bottom part, it was considered a tree, otherwise, it was discarded.

In summary, the density-based approach leverages the entire 3D point cloud to refine false-positive removal.

3.4 Neural network-based segmentation

The third method leverages state-of-the-art neural network architectures, specifically KPConv and RandLA-Net were chosen to perform semantic segmentation on the specific point clouds, because of their outstanding performance reached on 3D Semantic Segmentation datasets [18]. We can see a comparison of the performance of the most used neural networks in Figure 3.1, based on the table and the networks' availability and ease of use, RandLA-Net, and KPConv(pytorch) was deemed the most suitable.

To perform the segmentation, we adopt a two-step process. Initially, we extract a 3D segment around the potential tree seed with an optional extra area included to give more context to the networks. Subsequently, this segmented point cloud undergoes semantic segmentation through KPConv and RandLA-Net. At first, RandlA-Net was used which was trained on the SemanticKITTI dataset.

Methods	Scannet	Sem3D	S3DIS	PL3D
Pointnet	-	-	41.1	-
Pointnet++	33.9	-	-	-
SnapNet	-	59.1	-	-
SPLATNet	39.3	-	-	-
SegCloud	-	61.3	48.9	-
RF MSSF	-	62.7	49.8	56.3
Eff3DConv	-	-	51.8	-
TangentConv	43.8	-	52.6	-
MSDVN	-	65.3	54.7	66.9
RSNet	-	-	56.5	-
FCPN	44.7	-	-	-
PointCNN	45.8	-	57.3	-
PCNN	49.8	-	-	-
SPGraph	-	73.2	58.0	-
ParamConv	-	-	58.3	-
SubSparseCNN	72.5	-	-	-
RandLA-Net(torch)	-	76.0	70.9	70
KPConv rigid	68.6	74.6	65.4	72.3
KPConv deform	68.4	73.1	67.1	75.9

Table 3.1: Performance comparison of various neural networks on different 3D segmentation datasets (mIoU) [18, 17].

The analysis revealed many instances where the network misidentified points as poles (black), particularly at specific height ranges, and also misclassified building points as vegetation, which interferes severely with the process of classifying the point clouds as trees or non-trees.

After further testing, it was concluded that the reason for misclassifying points at specific heights was the specifics of the architecture of the RandlA-Net as when testing with the KpConv neural the misclassification to the pole class was not present. The reason for the error is most probably that the RandlA-Net network has a bias towards the pole class at specific height intervals, thus another network had to be chosen to improve the accuracy of the segmentation.

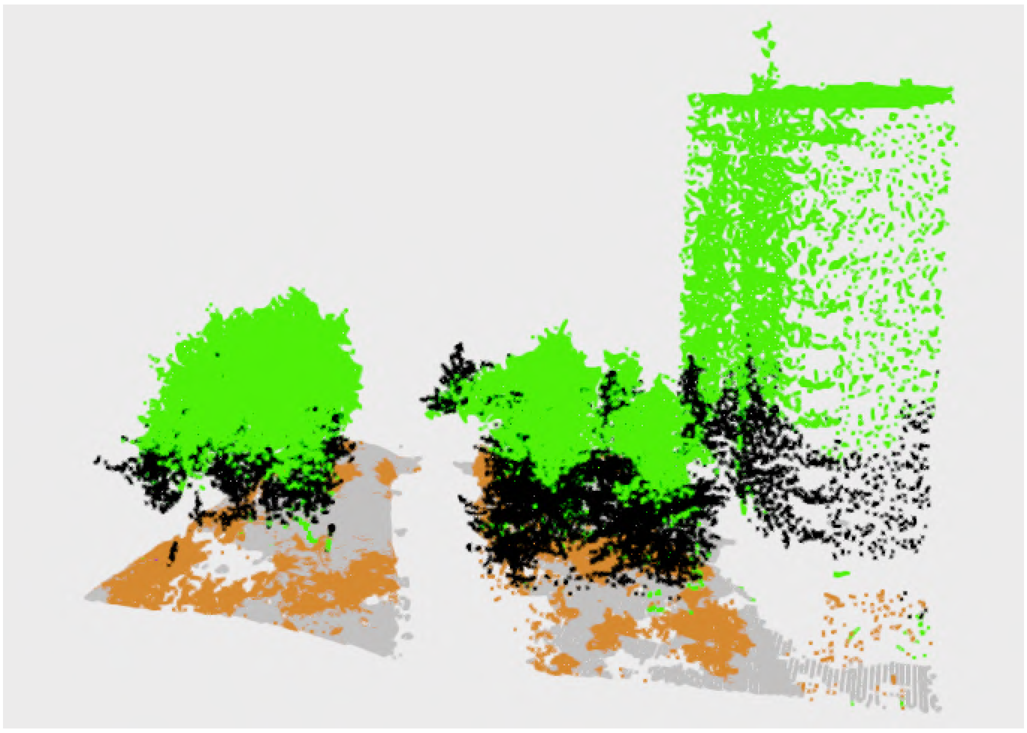

Figure 3.8: Segmentation Results with RandlANet

For this reason, the next choice was the KpConv neural network, which promised similar or better accuracy based on different 3D Segmentation datasets as seen in Table 3.1. The results of the segmentation using KpConv were substantially better, as the pole class was eliminated from the predictions, but the building facades were still largely classified as vegetation, furthermore using the Paris-Lille-3D dataset instead of the Semantic-Kitti increased the accuracy and the speed of the inference.

After further experimentation, it was concluded that the main factor causing the misclassification of many points, was the large discrepancy between the dataset that the network was trained on, and the dataset on which it was currently used, the AHN3 containing 6-9 points/m^2 and the Paris-Lille-3D which can contain close to 2000 points/m^2.

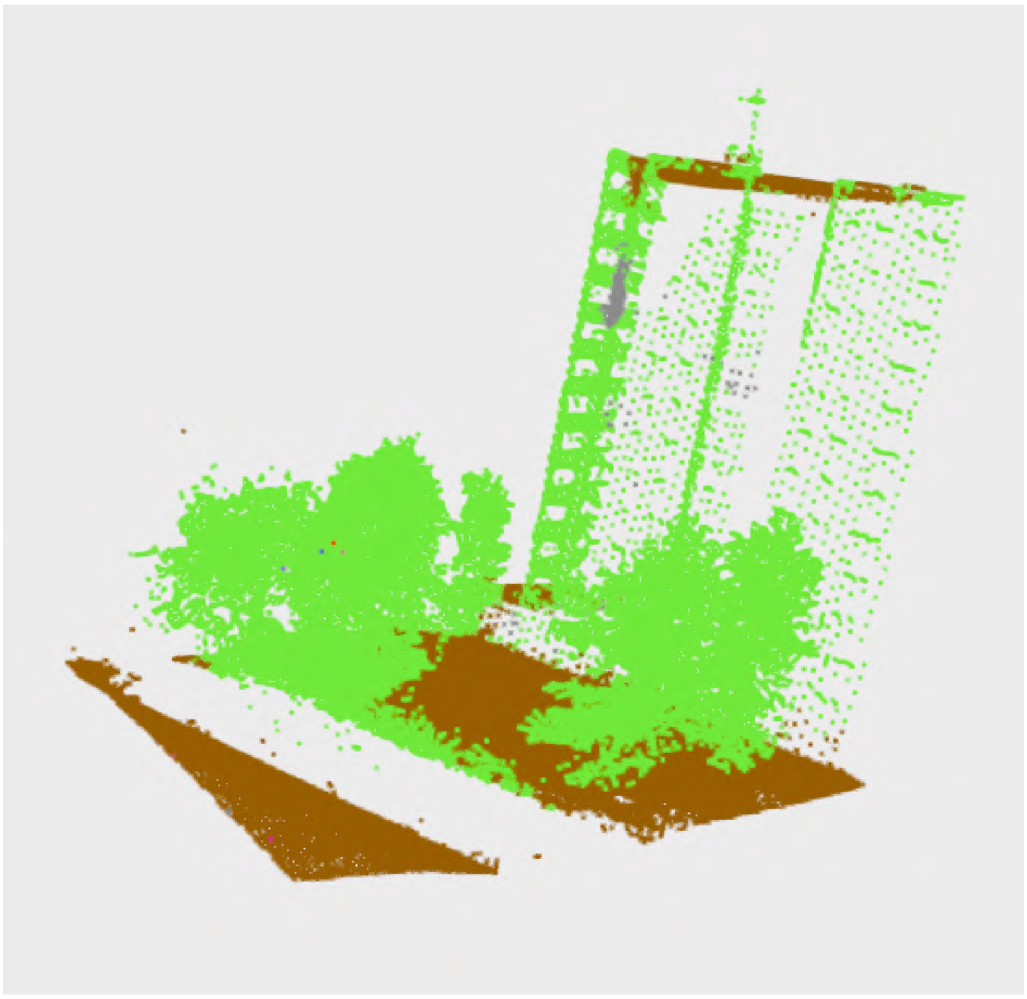
Figure 3.9: Segmentation Results with KpConv with no upscaling

To compensate for the low point density, an upsampling method was applied using the PCL library. Specifically, the local plane of each input point was sampled using a uniform random distribution such that the density of points was constant throughout the point cloud.

With the application of upsampling, we increased the average point density from 6-9 to 50 points/m^2. After running the segmentation process again, we can see a visible improvement in the accuracy of the segmentation, namely the sides of the buildings are starting to be classified correctly, but only partially.

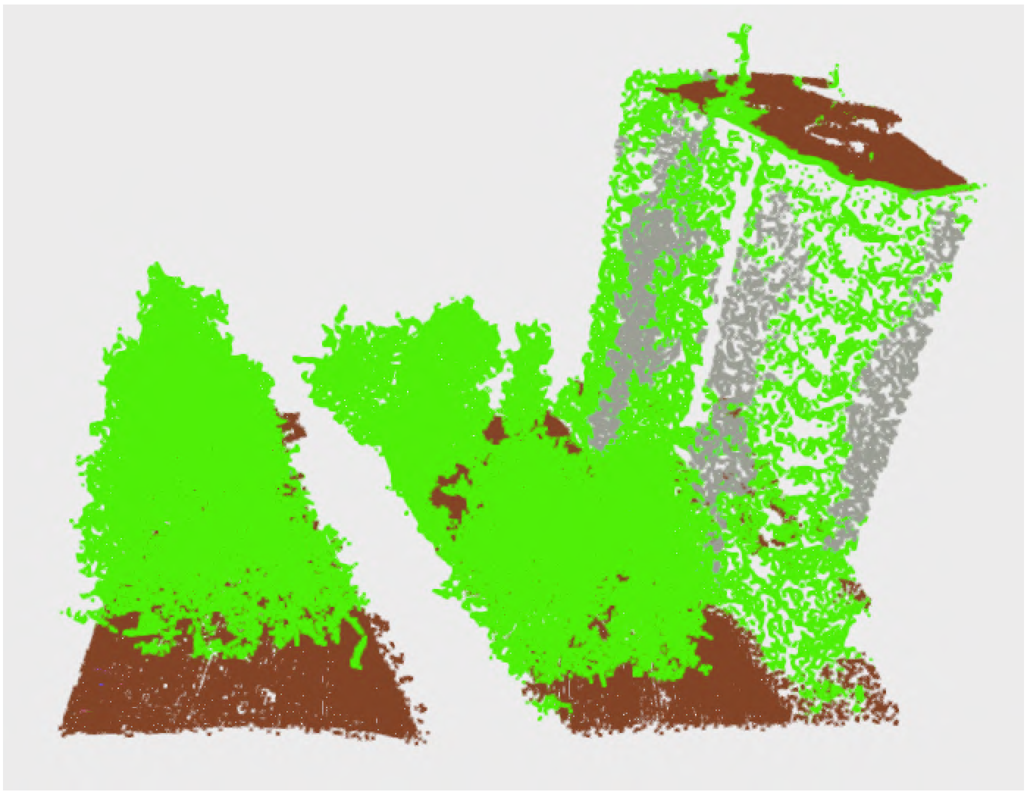

Figure 3.10: Segmentation Results with KpConv upscaled to 50 points/m^2

To further improve the accuracy of the network, an even higher point density was needed, so the density was increased to 100 points/m^2. At this level we can see a major improvement in the recognition of the building class, almost all of the points in this scenario were correctly classified.

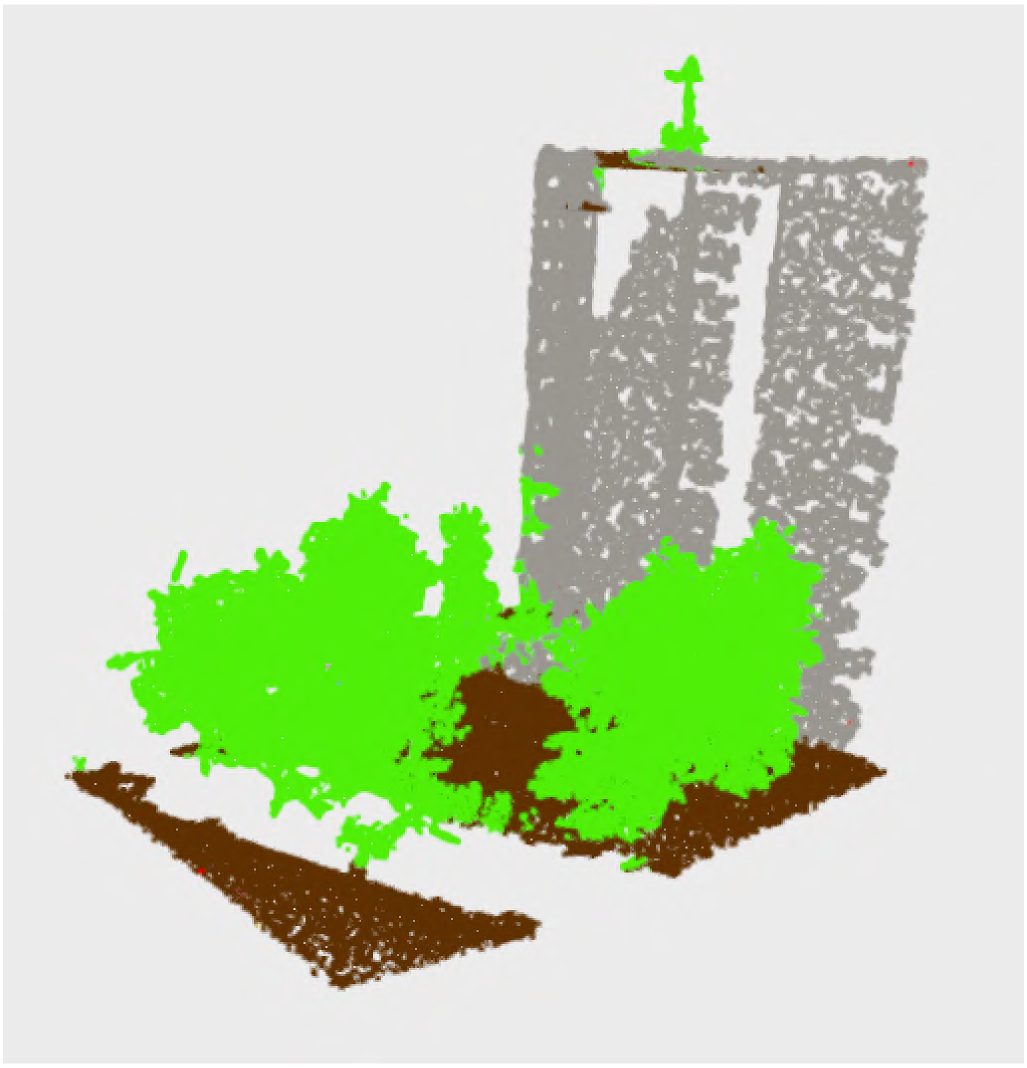

Figure 3.11: Segmentation Results with KpConv upscaled to 100 points/m^2

The area on which the networks were visually examined contained a typical apartment house, which has flat sides and tops so that segmentation can be fairly easily accomplished. However, the urban landscape of Amsterdam contains many different shapes and types of buildings, which cannot be classified in such a manner.

Running the segmentation with the same upsampling factor on a more central area in Amsterdam shows that when the point cloud is sparse because of occlusion or other factors, the results deteriorate accordingly. Also when the shape of the building tops is not flat but triangle-shaped, it causes the network to classify it as vegetation.

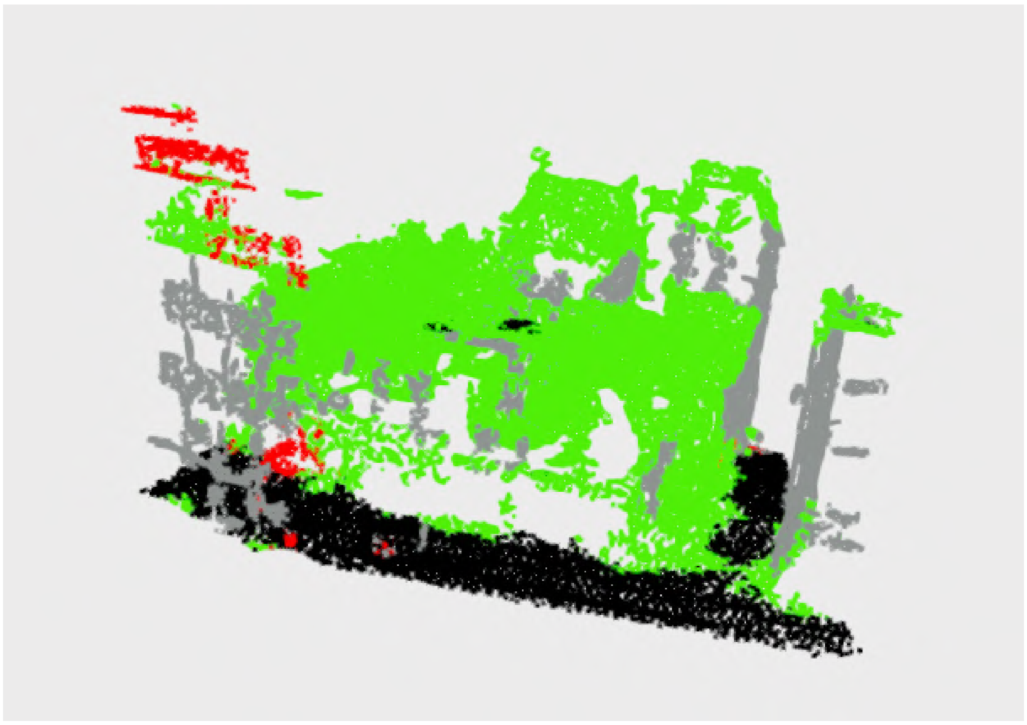

Figure 3.12: Segmentation Results with KpConv upscaled to 100 points/m^2

Further increasing the density up to 1000 points/m^2 seems to partially solve the problem, but some of the building tops are still misclassified.

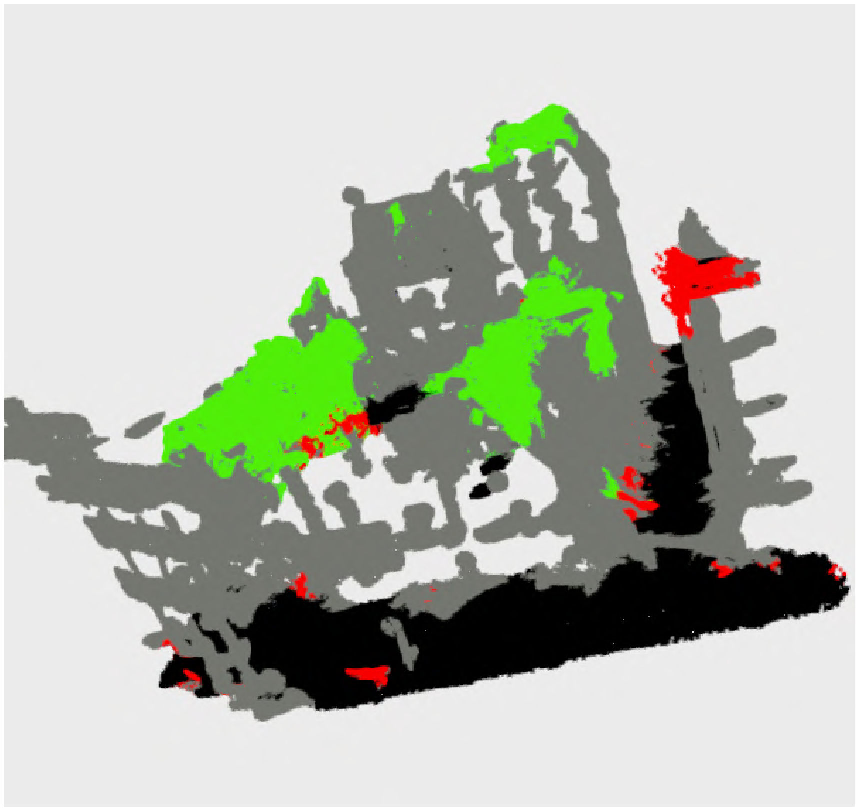

Figure 3.13: Segmentation Results with KpConv upscaled to 1000 points/m^2

From the previous experimentations, it was deduced that with a large enough extra area for greater context and the application of upsampling to a density of around 1000 points/m^2, the segmentation results were suitable for use with ALS point clouds. The gained segmentation results were forwarded to a post-processing step which classifies the results into tree or non-tree, by using different thresholds like the ratio or number of points classified as vegetation, buildings, or ground, this will be discussed further in Chapter 5.4.

Chapter 4

Implementation

The prototype implementations for the methodology described in Sec. 3.1 and 3.2 were carried out in standard C++ as part of the CloudTools geospatial framework. CloudTools – developed at Eötvös Loránd University – aims to create an easily reusable, high-abstraction level, an operation-based software library for raw point cloud and DEM processing.

As DEM files (preprocessed from the original point cloud) were selected as the input for the proposed algorithm, the CloudTools.DEM module was primarily utilized in the framework. CloudTools.DEM depends on the GDAL/OGR geospatial and geoprocessing software library for the input and output management of the spatial data.

The CHM-based and the Density-based approach was implemented in AHN.Vegetation. The neural network-based approach was implemented partly in C++ and partly in Python, as the integration of the neural network into C++ proved to be unfeasible which will be discussed further in Section 4.4.

The implementation follows the design principles and architecture of the CloudTools library. The UML class diagram shown in Figure 4.1 shows how the implementation fits into the CloudTools library.

4.1 CHM-based approach

The `Operation` class serves as the base class of the other operation classes; it defines a general transformation from an arbitrary data type to any other (`Any → Any`). The processing of Digital Elevation Model (DEM) data sets is represented by the `Calculation` class (`DEM → Any`). `DatasetCalculation` also inherits from `Calculation`; here, the complete source datasets for the target area are read and passed to the `computation`, which is then performed in a single call.

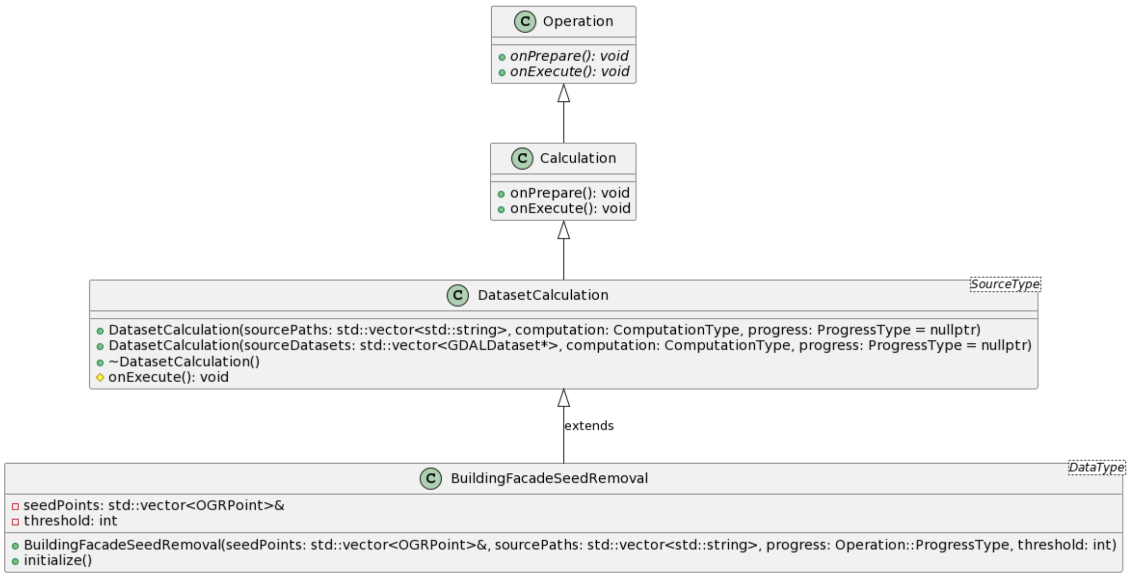
Operation
onPrepare(): void
onExecute(): void
Calculation
onPrepare(): void
onExecute(): void
DatasetCalculation
SourceType
DatasetCalculation(sourcePaths: std::vector<std::string>, computation: ComputationType, progress: ProgressType = nullptr)
DatasetCalculation(sourceDatasets: std::vector<GDALDataset*>, computation: ComputationType, progress: ProgressType = nullptr)
~DatasetCalculation()
onExecute(): void
extends
BuildingFacadeSeedRemoval
DataType
seedPoints: std::vector<OGRPoint>&
threshold: int
BuildingFacadeSeedRemoval(seedPoints: std::vector<OGRPoint>&, sourcePaths: std::vector<std::string>, progress: Operation::ProgressType, threshold: int)
initialize()

Figure 4.1: UML Diagram of the implementation of the CHM-based approach

The specific task of removing false positive seed points close to buildings is encapsulated in the `BuildingFacadeSeedRemoval` class. This class, derived from `DatasetCalculation` takes a vector of seed points, source paths, a progress callback, and a threshold value as parameters during initialization.

The `computation` in the `BuildingFacadeSeedRemoval` class is a window-based analysis, performed on the seed points to identify and remove those that are considered false positives. This is achieved by iterating through the seed points and applying a specified threshold to determine their validity based on the surrounding data as described in Section 3.1.

The full algorithm of the CHM-based approach can be seen below:

Algorithm 1 Building Facade Seed Removal

```
procedure INITIALIZE
    computation ← lambda function
    Input: $x, y, t, ws$                    ▷ Window size = $ws \times 2 + 1$, t=10, ws=3

    function COMPUTATION($x, y$)
        idxs ← empty vector
        $c \leftarrow 0$
        for all point ∈ seedPoints do
            $counter \leftarrow 0$
            $px \leftarrow$ point.getX()
            $py \leftarrow$ point.getY()
            for $i \leftarrow px - ws$ to $px + ws$ do
                for $j \leftarrow py - ws$ to $py + ws$ do
                    if not hasSourceData($0, i, j$) and sourceData($1, i, j$) $> t$ then
                        $counter \leftarrow counter + 1$
                    end if
                end for
            end for
            if $counter >$ threshold then
                idxs.push_back($c$)
            end if
            $c \leftarrow c + 1$
        end for
        for $i \leftarrow$ idxs.size() $- 1$ to 0 do
            seedPoints.erase(seedPoints.begin() + idxs[$i$])
        end for
    end function
end procedure
```

4.2 Density-based approach

The Density-based approach uses a histogram The vertical histogram of potential trees is computed using the `calcHist` function. This function takes a vector of 3D points and information about height and terrain then calculates a histogram with 21 bins representing the distribution of points in the vertical space.

The function iterates through the provided point, adjusts the height considering terrain elevation, and updates the histogram accordingly. The resulting histogram provides information about the vertical distribution of potential trees. The values are then normalized to facilitate comparison.

After calculating the histograms we either filter out the ones that are below a set similarity to the model histogram which was calculated based on the average vertical density function of the trees from the validation dataset or remove the ones that do not fall into a manually specified ratio.

The `calcHist` method used for calculating the vertical histogram can be seen below:

Algorithm 2 Calculate Histogram

```
function CALCHIST(ids, points, height, terrain)
    if height $\leq 1$ then
        return hist
    end if
    hist $\leftarrow$ vector of size 21 initialized with zeros
    binSize $\leftarrow$ height/20
    counter $\leftarrow 0$
    z, idx $\leftarrow 0, 0$
    for all $i \in$ ids do
        z $\leftarrow$ points[$i$][2] $-$ (terrain $+ 0.5$)
        if z $\leq 0$ then
            continue
        end if
        idx $\leftarrow$ round(z/binSize)
        if idx $\geq 0$ and idx $\leq 20$ then
            counter $\leftarrow$ counter $+ 1$
            hist[idx] $\leftarrow$ hist[idx] $+ 1$
        end if
    end for
    return hist
end function
```

4.3 Neural network based approach

The neural network based approach is implemented in such a way that when the C++ workflow reaches the step where potential tree crowns are segmented in the 2D data, we cut out the minimal bounding box which still encompasses the crown. When cut out, the areas are also optionally upscaled using the PCL library.

After upscaling the point clouds are written out into PCL format and then read by the Python script. The Python script runs all the potential trees through the selected network and classifies them into trees or non-trees based on the number and ratio of the classified points.

Following the classification a binary vector of size equal to the potential trees is written into a file which is then read by the C++ workflow, subsequently the points that are deemed false are removed from further calculations.

4.4 Implementation Remarks

The initial plans for the implementation of the neural networks were to find a solution that allows full integration into the existing software in C++. After thorough research, it was concluded that there is no native C++ library that supports 3D Detection and Segmentation, but there were some libraries that promised inference-only solutions in C++ through ONNX [1] or TensorRT [2].

One such library was MMDeploy [3] which promised support for both TensorRT and ONNX with popular 3D models with their own MMDetection3D [4] library, after dedicating a substantial amount of time on manual compilation it became clear that the given support for 3D model conversions is minimal in practice as the ONNX conversion support wasn't entirely finished. Compiling the library in such a way that it allows 3D model conversions was also a considerable challenge, as many different bugs were raised, thus a decision was made to search for a better alternative.

Another option was a more direct approach, converting the Pytorch [5] model to TorchScript then compiling it with Torch-TensorRT [6], after making several attempts

[1]ONNX: `https://onnx.ai/`
[2]TensorRT: `https://developer.nvidia.com/tensorrt`
[3]MMDeploy: `https://github.com/open-mmlab/mmdeploy`
[4]MMDetection3D: `https://github.com/open-mmlab/mmdetection3d`
[5]Pytorch: `https://pytorch.org/`
[6]Torch-TensorRT: `https://github.com/pytorch/TensorRT`

of conversion with different architectures, it was unveiled that 3D model support was lackluster.

Considering the above-discussed findings the integration of the models into C++ was dismissed and a Python-only implementation was chosen namely the OpenML [7] library, as it has support for the most accurate 3D models which are also optionally pretrained.

[7] Open3DML: `https://github.com/openml/OpenML`

Chapter 5

Results

Many of the results were visually presented in Chapter 3, this chapter's main function is to present the performance of the different algorithms, like the accuracy, and the effects of different parameters and thresholds.

All of the results were calculated using a Ryzen 2600X 6-core processor, running at a base clock of 3.6 GHz and a boost clock of 4.25 GHz, and a GTX 1060 6GB graphics card with a base clock of 1.5 GHz and a boost clock of 1.9 GHz.

The validation was carried out on the previously mentioned study area in Chapter 3.1 which covers an area of 1.93 square kilometers.

5.1 CHM-based approach

As we can see in Table 5.1, the method significantly reduces the runtime of the whole workflow because it is run right after the potential tree seeds (local maximums) are found, so all of the further steps have a reduced number of candidates to consider.

The threshold(T) is the main variable that defines how many points have to resemble a building for the seed point to be considered invalid.

The best results were arguably obtained with the threshold set at 20 points, at this level, the number of trees matched was reduced by 2% but there was a 37% decrease in the number of false positives and a 63% decrease in the runtime of the workflow.

	Original	T20	T17	T15
Detected Trees	1718	1461	1355	1307
Ref. Trees Matched	1096	1072	1052	1039
Ref. Trees Failed	211	235	255	268
Match Ratio	83.36%	82.02%	80.49%	79.50%
False Positives	622	389	303	268
Runtime	1505 s	553 s	448 s	375 s

Table 5.1: Performance of the CHM-based approach with different thresholds(T) and a window size of 7.

5.2 Density-based approach

As discussed in Section 3.2 and Section 4.2, 2 different methods were created, one based on a model density function and one based on a manually specified threshold ratio. The ratio-based method seen in Table 5.2 uses the best-performing thresholds in the manual examination, specifically the bins between bins number 7 and 15 have to be cumulatively at least 1.5 times dense as the bins below them.

	Model-based	Ratio-based
Detected Trees	1362	1508
Ref. Trees Matched	878	954
Ref. Trees Failed	429	353
Match Ratio	67.18%	73%
False Positives	484	554
Runtime	7350 s	7323 s

Table 5.2: Performance of the Density-based approaches.

5.3 Neural network based approach

The Neural network based approach's results improved as more upsampling was applied to the original AHN3 dataset as discussed in Section 3.3, so datasets with different densities were created to conduct experiments on the approach. In Table

5.2 we can see the effect of the density of the point clouds on the inference time of the KPConv neural network trained on the Paris-Lille-3D dataset.

Density (points/m^2)	Avg Time (s)	Whole Time (s)
6-10	0.40	4536.14
200	0.43	4850.28
500	0.47	6133.43
1000	0.71	9028.01
2000	1.19	15207.19

Table 5.3: Inference times for KPConv with different density point clouds

The classification of the areas under investigation as tree or non-tree using the segmentation results proved to be challenging, as the criteria used in the task obtained poor results, this problem will be discussed further in Chapter 5.4. Because of the large number of different criteria and different combinations of them which all depend on the density of the data, no validation results are included.

The accuracy of the neural network is better shown in Chapter 3.3 as the results are hard to quantify because no validation dataset is available for the study area.

5.4 Discussion

After experimenting with different parameters and areas regarding the filtering of false positive tree detections using neural networks, the following things were deduced:

- **Criteria Selection.** After the segmentation is run, the task is to make a binary classification based on the obtained results. Different criteria were tested, for example, setting a threshold for the ratio of ground and vegetation points, or limits were set for the maximum number of building points or minimum points for the number of points containing vegetation or the combination of those, but the results of the classification were underwhelming. Many trees were removed from the results and only a small amount of false detections were removed, even though visual examination of select areas with different landscapes showed good results. To create a more robust and accurate method, further examination would be needed on the trees that were classified as non-trees or vice-versa, and new, more refined criteria should be developed.

- **Effects of upsampling.** Upsampling the dataset greatly improved the results of the neural networks, but as we increase the density of points we cause a change in the ratio and number of points thus making direct comparison between methods with the same selection criteria but different point densities a difficult task.

Chapter 6

Conclusion

To summarize, in this study I have presented different types of approaches that can assist in the problem of finding and removing false positive tree detections in urban environments. The discussed approaches can be applied to various geoinformatical tasks that utilize DEMs or raw point clouds.

By creating a CHM-based method that removed seed points that were presenting building-like characteristics that could be derived from the DEMs, it was shown that a relatively simple method can help improve results significantly without the need for expensive calculations or a complex workflow. Because of the nature of LiDAR technology, datasets are inherently large and methods that can help improve results without additional overhead can be beneficial.

As density-based algorithms are widely used in forestry, it was important to uncover how their performance translates into an urban setting. After the examination of the algorithms on the AHN3 point cloud, it was evident that the similarity between man-made objects and trees made it hard for them to differentiate between trees and non-trees without more complex heuristics or additional processing steps.

Evaluating the performance and behavior of current state-of-the-art neural networks on datasets with different capture methods and characteristics than the ones they were trained on, gave insight into their potential use in areas other than those they were originally designed for. By applying upscaling to the ALS datasets it was shown that the selected networks have promising adaptability and could be effectively used for processing point clouds captured by different sensors with different output resolutions.

6.1 Future work

The presented approaches can all be improved in different ways, first, I want to address the density-based approach:

- The current method only takes into account the vertical density of the potential trees, an added horizontal density analysis could improve the accuracy of the method.
- The thresholding in this approach was based on the average probability density function of the trees found in the validation dataset, this can lead to inaccuracies, because the trees can be of different heights and species with different point densities. If a dataset were created with the specific characteristics of each species, it would allow researchers to create more accurate heuristics. It would open up many different possibilities for classification, including the training of neural networks to distinguish different objects based on the profile of their density.

As for the neural network based segmentation, the following improvements could be made:

- The neural network based segmentation classifies all of the points in the area where the potential trees could be. After the classification we need to decide by the number and position of classified the points, if there is a tree there, we have to create some kind of heuristic to make a decision. In the current approach, there was a thresholding based on the ratio of points classified as vegetation and the ground or there was a set number of minimum points that had to be present for it to be considered a tree. With a more nuanced heuristic approach or a more complex approach that is combined with the density analysis of the points, the results of the segmentation could be utilized to their fullest potential.
- As the neural networks weren't trained specifically for an ALS dataset, creating a dataset specifically for urban segmentation would allow for the exploration of using transfer learning to improve the accuracy. Also, if the dataset size allows, the networks could be fully trained from start to finish, specifically for use in ALS-based geoinformatical analysis.

- The performance of the chosen neural networks could be improved if they were converted to either TensorRT or ONNX [22] enabling easier integration into C++ and improving inference speed, unfortunately, the compatibility of the 3D Segmentation capable models with these technologies is not yet on an optimal level to accomplish this promptly.

Acknowledgements

I would like to thank my supervisor, Cserép Máté, for his invaluable guidance and mentorship. I also express my gratitude to my girlfriend for her unwavering support, and to my family for their enduring encouragement throughout my academic journey.

Bibliography

[1] Máté Cserép and Roderik Lindenbergh. “Distributed processing of Dutch AHN laser altimetry changes of the built-up area”. In: *International Journal of Applied Earth Observation and Geoinformation* 116 (2023), p. 103174. ISSN: 1569-8432. DOI: `https://doi.org/10.1016/j.jag.2022.103174`. URL: `https://www.sciencedirect.com/science/article/pii/S1569843222003624`.

[2] Vasileios Alexandridis. “Automatic detection of waterbeds in shallow muddy water bodies in the Netherlands using green LiDAR”. PhD thesis. Aug. 2020. DOI: `10.13140/RG.2.2.20014.43840`.

[3] Peter L. Guth et al. “Digital Elevation Models: Terminology and Definitions”. In: *Remote Sensing* 13.18 (2021). ISSN: 2072-4292. DOI: `10.3390/rs13183581`. URL: `https://www.mdpi.com/2072-4292/13/18/3581`.

[4] Barbara Koch, Ursula Heyder, and Holger Weinacker. “Detection of Individual Tree Crowns in Airborne Lidar Data”. In: *Photogrammetric Engineering & Remote Sensing* 72.4 (2006), pp. 357–363. ISSN: 0099-1112. DOI: `doi:10.14358/PERS.72.4.357`. URL: `https://www.ingentaconnect.com/content/asprs/pers/2006/00000072/00000004/art00001`.

[5] Serge Beucher and Christian Lantuéjoul. *Use of Watersheds in Contour Detection.* workshop published. Sept. 1979. URL: `http://cmm.ensmp.fr/~beucher/publi/watershed.pdf`.

[6] Juntao Yang et al. “An Individual Tree Segmentation Method Based on Watershed Algorithm and Three-Dimensional Spatial Distribution Analysis From Airborne LiDAR Point Clouds”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 13 (2020), pp. 1055–1067. DOI: `10.1109/JSTARS.2020.2979369`.

[7] Hans-Erik Andersen, Stephen E. Reutebuch, and Gerard F. Schreuder. “Automated Individual Tree Measurement Through Morphological Analysis of a LIDAR-Based Canopy Surface Model”. In: *Precision Forestry: Proceedings of the First International Precision Forestry Cooperative Symposium*. Available online at `https://www.fs.usda.gov/pnw/olympia/silv/publications/opt/485_AndersenEtal2001.pdf`. Seattle, Washington, USA: Forestry Publishing Company, 2001, pp. 11–22.

[8] Yanhe Shu Yanshan Bian Peng Zou and Ronghuan Yu. “Individual Tree Delineation in Deciduous Forest Areas with LiDAR Point Clouds”. In: *Canadian Journal of Remote Sensing* 40.2 (2014), pp. 152–163. DOI: `10.1080/07038992.2014.943700`. eprint: `https://doi.org/10.1080/07038992.2014.943700`. URL: `https://doi.org/10.1080/07038992.2014.943700`.

[9] Koma, Koenig, and Höfle. “URBAN TREE CLASSIFICATION USING FULL-WAVEFORM AIRBORNE LASER SCANNING”. In: III-3 (June 2016). DOI: `10.5194/isprsannals-iii-3-185-2016`.

[10] Bernhard Höfle, Markus Hollaus, and Julian Hagenauer. “Urban vegetation detection using radiometrically calibrated small-footprint full-waveform airborne LiDAR data”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 67 (2012), pp. 134–147. ISSN: 0924-2716. DOI: `https://doi.org/10.1016/j.isprsjprs.2011.12.003`. URL: `https://www.sciencedirect.com/science/article/pii/S0924271611001560`.

[11] Jonathan Cheung-Wai Chan Kaja Kandare Hans Ole Ørka and Michele Dalponte. “Effects of forest structure and airborne laser scanning point cloud density on 3D delineation of individual tree crowns”. In: *European Journal of Remote Sensing* 49.1 (2016), pp. 337–359. DOI: `10.5721/EuJRS20164919`. eprint: `https://doi.org/10.5721/EuJRS20164919`. URL: `https://doi.org/10.5721/EuJRS20164919`.

[12] Xingcheng Lu et al. “A bottom-up approach to segment individual deciduous trees using leaf-off lidar point cloud data”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 94 (2014), pp. 1–12. ISSN: 0924-2716. DOI: `https://doi.org/10.1016/j.isprsjprs.2014.03.014`. URL: `https://www.sciencedirect.com/science/article/pii/S0924271614000860`.

[13] M. Z. Abd Rahman and Ben G. H. Gorte. "Individual tree detection based on densities of high points of high resolution airborne lidar". In: 2008. URL: https://api.semanticscholar.org/CorpusID:14060134.

[14] António Ferraz et al. "3-D mapping of a multi-layered Mediterranean forest using ALS data". In: *Remote Sensing of Environment* 121 (2012), pp. 210–223. ISSN: 0034-4257. DOI: https://doi.org/10.1016/j.rse.2012.01.020. URL: https://www.sciencedirect.com/science/article/pii/S0034425712000570.

[15] Wenxia Dai et al. "A new method for 3D individual tree extraction using multispectral airborne LiDAR point clouds". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 144 (2018), pp. 400–411. ISSN: 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2018.08.010. URL: https://www.sciencedirect.com/science/article/pii/S0924271618302296.

[16] M. Maltamo et al. "Identifying and quantifying structural characteristics of heterogeneous boreal forests using laser scanner data". In: *Forest Ecology and Management* 216.1 (2005), pp. 41–50. ISSN: 0378-1127. DOI: https://doi.org/10.1016/j.foreco.2005.05.034. URL: https://www.sciencedirect.com/science/article/pii/S037811270500352X.

[17] Qingyong Hu et al. *RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds*. 2020. arXiv: 1911.11236 [cs.CV].

[18] Hugues Thomas et al. *KPConv: Flexible and Deformable Convolution for Point Clouds*. 2019. arXiv: 1904.08889 [cs.CV].

[19] Xavier Roynard, Jean-Emmanuel Deschaud, and François Goulette. *Paris-Lille-3D: a large and high-quality ground truth urban point cloud dataset for automatic segmentation and classification*. 2018. arXiv: 1712.00032 [cs.LG].

[20] J. Behley et al. "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences". In: *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*. 2019.

[21] Karel Kuželka, Martin Slavík, and Peter Surovy. "Very High Density Point Clouds from UAV Laser Scanning for Automatic Tree Stem Detection and Direct Diameter Measurement". In: *Remote Sensing* 12 (Apr. 2020). DOI: 10.3390/rs12081236.

[22] Yuxiao Zhou and Kecheng Yang. "Exploring TensorRT to Improve Real-Time Inference for Deep Learning". In: *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. 2022, pp. 2011–2018. DOI: `10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00299`.

List of Figures

List of Tables

List of Algorithms