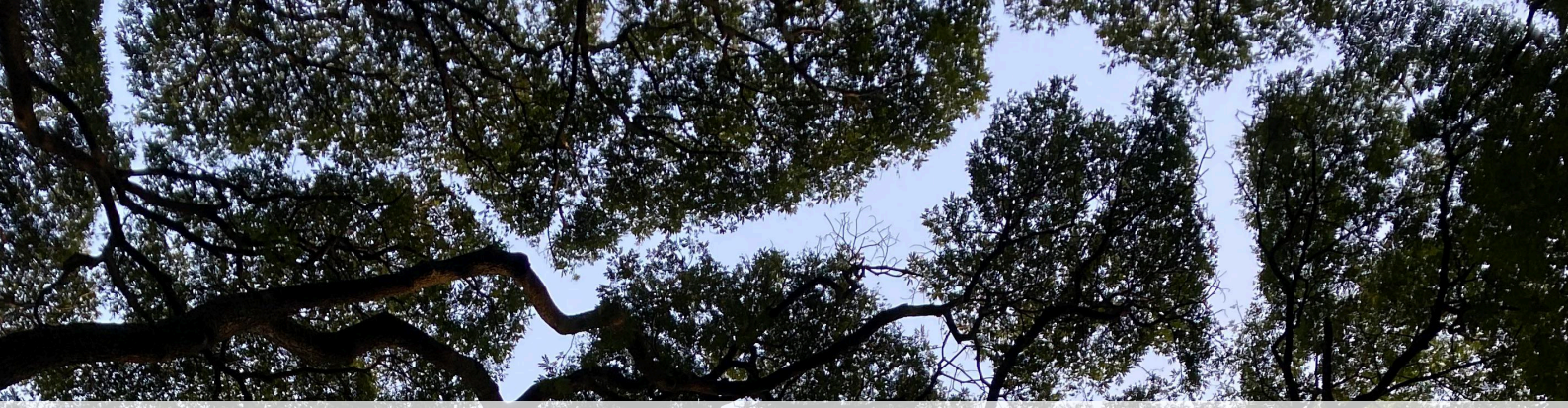


November 2026

Automatic tree reconstruction for CFD simulations

GEOIT1501: Synthesis Project

Henryk Gujda
5399769

Simon Deuten
5600243

Anna Scherrenburg
5212723



TU Delft Delft University of Technology

Haskoning
Enhancing Society Together

***Automatic tree reconstruction for CFD simulations* (2026)**

Henryk Gujda (5399769) Simon Deuten (5600243) Anna Scherrenburg (5212723)

The work in this synthesis project was carried out in the:



3D geoinformation group
Delft University of Technology

Supervisors: Clara García-Sánchez, Themis Vargiemezis

In collaboration with:



Advanced Technology and Research group
Haskoning Industry & Buildings sector

Consultants: Frank van Gool, Kjeld Teunissen

This work is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/.

Abstract

Urban computational fluid dynamics (CFD) models used for pedestrian wind comfort assessment typically omit vegetation, despite trees having a measurable effect on wind flow. With the Dutch NEN8100 pedestrian wind comfort guideline under review, there is growing interest in automated, scalable methods for incorporating tree geometry into urban CFD workflows. However, reconstructing individual trees from airborne LiDAR remains challenging: point clouds are collected top-down and frequently contain incomplete or overlapping canopy returns, while existing segmentation and reconstruction pipelines have not been systematically evaluated for CFD compatibility or scalability.

This research extends an existing tree reconstruction pipeline (`Tree4CFD`) along three stages: individual tree segmentation, tree reconstruction, and scalability through an accessible interface. For segmentation, a hybrid approach combining a 2D canopy-height-model (CHM) coarse pass with 3D multi-directional analysis of under-segmented clusters was implemented and benchmarked against the original watershed-only method, using a sensitivity analysis and grid search to tune parameters across densely, moderately, and sparsely built-up areas. This is later combined with a variable window for finding tree peaks in the CHM. For reconstruction, LoD2 and LoD3 tree geometries were compared on computational efficiency, geometric validity (via `val3dity`), and mesh simplification using `CGAL` alpha wrapping. Finally, an interactive web interface was developed, allowing users to define an area of interest, select desired output parameters, and automatically trigger data retrieval, segmentation, and reconstruction, producing georeferenced OBJ geometries ready for integration into frameworks such as `City4CFD`.

Results indicate that cell size, smooth sigma, and minimum peak distance are the most influential segmentation parameters. The multi-directional refinement resulted in a minimal increase in $F1$-score (average of 1.68%), showing that improvement opportunities are possible but need further evaluation. Reconstruction time was found not to be a bottleneck, with negligible differences between LoD2 and LoD3 processing times, and `CGAL` alpha wrapping produced watertight, substantially simplified meshes suitable for downstream simulation. The developed interface demonstrates that the full pipeline, from AOI selection to downloadable 3D geometry, can be executed without manual intervention, supporting its practical use by stakeholders such as Haskoning.

Acknowledgements

We would like to thank our supervisors, Dr. Clara García-Sánchez and Themis Vargiemezis of the 3D Geoinformation group at TU Delft, for their guidance throughout this synthesis project. Their feedback on our methodology and willingness to discuss open questions as the research scope evolved were invaluable in shaping this work.

We are also grateful to Frank van Gool and Kjeld Teunissen from Haskoning for their support as our industry contacts, and for giving us the freedom to explore and experiment within this field without being constrained by rigid predefined requirements. This openness allowed us to approach the research question from multiple angles and arrive at the scope presented in this report.

Contents

1. Introduction **1**
- 1.1. Necessity of Accurate Wind Flow Modelling ... 1
- 1.2. Challenges of Modelling Trees ... 1
- 1.3. Report Outline ... 2

2. Problem Definition **3**
- 2.1. Current State of Urban Tree Modelling ... 3
- 2.2. Project Summary ... 3
- 2.3. Research Question ... 4
- 2.4. Research Limitations ... 4
- 2.5. Relevant Prior Knowledge ... 5

3. Theory and Context **6**
- 3.1. Individual Tree Segmentation ... 6
- 3.2. Tree Reconstruction Algorithms ... 8

4. Data Collection and Availability **11**
- 4.1. Point Cloud Data ... 11
- 4.2. Satellite Imagery ... 11
- 4.3. Segmentation ... 11
- 4.4. 3D BAG ... 14

5. Research Methodology **15**
- 5.1. Individual Tree Segmentation ... 15
- 5.2. Tree Reconstruction ... 17
- 5.3. Scalability and Framework Integration ... 17

6. Results **19**
- 6.1. Segmentation ... 19
- 6.2. Reconstruction ... 25
- 6.3. Interactive User Interface ... 27

7. Discussion **30**
- 7.1. Segmentation ... 30
- 7.2. Reconstruction ... 32
- 7.3. Interface ... 33

8. Conclusion **34**
- 8.1. Subquestion 1 ... 34
- 8.2. Subquestion 2 ... 34
- 8.3. Subquestion 3 ... 35

References **36**

A AI statement **37**

B Contributors **38**
- B.1 Team members ... 38
- B.2 Roles ... 39
- B.3 Client ... 40
- B.4 Task Distribution Table ... 40

C Tree inventories for each area **41**
- C.1 Amsterdam ... 41

C.2 Utrecht 41
C.3 Delft 42

D Sensitivity analysis results **43**

List of figures

Figure 1.1 Effect of trees on wind speed. 1
Figure 3.1 Marker-controlled watershed 7
Figure 3.2 Different LoDs in trees 8
Figure 3.3 Procedural modelling - LoD2 tree 9
Figure 3.4 Convex hull vs Alpha shape 9
Figure 3.5 Issues with tree reconstruction in LoD3 10
Figure 4.1 Boxplots of NDVI features of the original areas 11
Figure 4.2 Boxplots of NDVI features of the new areas 12
Figure 4.3 Urban density areas for Amsterdam, Delft and Utrecht. 12
Figure 5.1 Differentiation in shapes between correctly segmented and under-segmented patches 15
Figure 5.2 Multi-directional crown projection 16
Figure 5.3 Example of polygon splitting. 16
Figure 6.1 Interpolated heat maps of the tweaked segmentation parameters 20
Figure 6.2 Box plots showing the spread of the optimal parameters per urban density class. 22
Figure 6.3 Interpolated heat maps of the segmentation parameters after variable window implementation. 22
Figure 6.4 Averaged $F1$-scores for different segmentation implementations.s 23
Figure 6.5 Tree inventory vs. detected trees for AOI in Amsterdam. 24
Figure 6.6 Municipal trees and detected trees in different cities 24
Figure 6.7 Reconstruction time of trees at different LoDs. 25
Figure 6.8 Reconstructed mesh before and after alpha wrapping 26
Figure 6.9 Reconstructed forest trees before and after alpha wrapping 27
Figure 6.10 Interface options 28
Figure 6.11 Interface results 29
Figure 6.12 3D Preview in the interface 29
Figure 7.1 Incorrect annotation of bushes and overlapping canopies 30
Figure 7.2 Wall reconstructed as tree 31
Figure 7.3 Cranes in point cloud 31
Figure 7.4 An input polygon that was splitted in four parts, resulting in splitted segments along these lines. 32
Figure 7.5 Reconstructed mesh of a dense forest area using the Marching Cubes algorithm. 33
Figure 7.6 Wrapped reconstruction of a dense forest area. 33
Figure 3.1 Tree inventories for Amsterdam 41
Figure 3.2 Tree inventories for Utrecht 41
Figure 3.3 Tree inventories for Delft 42

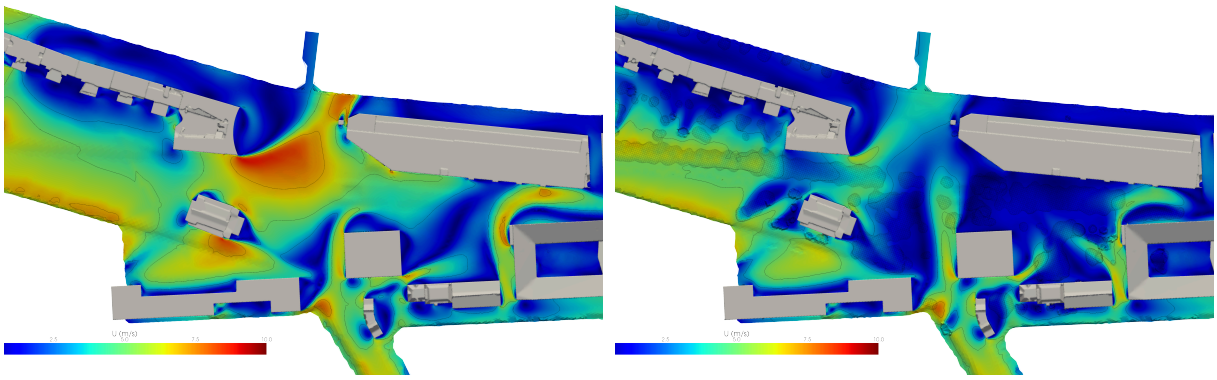
List of tables

Table 5.1 Different parameters per segmentation stage. 17
Table 6.1 Tweaked parameters with their corresponding ranges. 19
Table 6.2 Tweaked parameters with their corresponding ranges. 20
Table 6.3 Optimal segmentation parameters after grid search. 21
Table 6.4 Optimal parameters for the modified segmentation algorithm. 22
Table 6.5 Recall and precision scores for the different implementations. 23
Table 2.1 Task distribution between the team members 40

1. Introduction

1.1. Necessity of Accurate Wind Flow Modelling

Urban environments face growing microclimatic challenges, including urban heat islands, wind microclimates around high-rise developments, and street canyon ventilation issues (Buccolieri et al., 2018; Maison et al., 2022). Rapid urbanisation has concentrated over 93% of populations in built environments, intensifying urban microclimate challenges such as elevated air pollution, heat accumulation, and altered airflow patterns (Alting, 2025). Within dense urban fabrics, building morphology significantly accelerates local wind speeds at pedestrian level, a simulation of this is shown in Figure 1.1 (van Druenen, 2025). To ensure outdoor safety and comfort around new developments, municipal planning authorities mandate aerodynamic wind engineering assessments (van Druenen, 2025). Microscale wind patterns also govern outdoor thermal environments and the mitigations of the Urban Heat Island effect (Fu et al., 2024). To simulate these complex environmental interactions, urban digital twins are necessary to compute CFD simulations (Alting, 2025).



No trees With trees - summer condition

Figure 1.1: Wind simulation on Javaeiland in Amsterdam with and without trees included. Data provided by Haskoning.

In the Netherlands, the Dutch pedestrian wind climate standard NEN 8100 provides a standardised framework for these evaluations, combining local aerodynamic data from Computational Fluid Dynamics (CFD) or wind tunnel experiments with long-term meteorological statistics to determine the probability of exceeding specific wind speed thresholds for both wind discomfort and wind danger (van Druenen, 2025). The NEN 8100 is currently being modernised, including revisions to the inclusion of vegetation effects. To perform these realistic wind comfort evaluation in accordance with NEN 8100, engineering consultancies such as Haskoning require accurate 3D representations of urban vegetation within their microclimate modelling pipelines (Fu et al., 2024).

1.2. Challenges of Modelling Trees

3D city models have become more and more advanced. In the Netherlands, datasets like 3DBAG provide automated, highly detailed models of the national building stock (Peters et al., 2022). However, vegetation remains a missing link in urban digital twins. Where trees are included, they are typically represented as bulk porosity fields or simplified primitives (Alting, 2025). Running CFD does not require manually resolving every leaf and twig, but it does demand topologically consistent, watertight 3D geometries that reflect crown envelope volumes and porosities (Alting, 2025; Fu et al., 2024). To enable rapid urban microclimate studies, the primary engineering challenge is building a fast, automated, and scalable 3D tree reconstruction tool capable of converting open-access airborne laser scanning (ALS) point clouds into simulation-ready 3D assets.

A central question in vegetative wind flow simulation is determining the appropriate geometric level of detail (LoD) required to accurately capture airflow patterns without incurring unnecessary computational costs (Alting, 2025; Fu et al., 2024). Tree LoD frameworks distinguish between simple porous blocks (LoD1), porous crowns with explicit solid trunks (LoD2), and high-fidelity branch or crown structures (LoD3).
Another aspect is the placement of trunks. While practical engineering workflows at consultancies like Haskoning typically group closely positioned trees rather than requiring individual trunk locations, accounting for the structural presence and aerodynamic effects of trunks remains important for accurate wind flow modelling. Therefore, this work also focuses on the segmentation of individual trees from point clouds to isolate and analyse individual crown and trunk structures. Because of time constraints, no actual CFD simulations will be performed. By bypassing heavy CFD execution, the assessment centres on benchmarking performance, comparing memory usage, file sizes, and processing times across the different geometric and parametric abstraction strategies.

1.3. Report Outline

This project report first outlines the existing methods and challenges that lead us to identify the research gaps through formulation of research questions. Then in Chapter 3 the theory and context of this research direction including upcoming legal standardisation in the Netherlands. Chapter 4 explains the utilised data sources and Chapter 5 proposes a research methodology. The resulting outcomes of the experiments are presented in Chapter 6 and critically analysed and evaluated in Chapter 7, finally concluding with synthesised findings and research recommendations in the Chapter 8. The code base created in this project can be found on GitHub

2. Problem Definition

2.1. Current State of Urban Tree Modelling

In this research, two existing pipelines for automatic tree reconstruction are utilised. The first is the `Tree4CFD` pipeline developed by Themis Vargiemezis, which processes airborne LiDAR point clouds to generate 3D tree geometries for integration into the `City4CFD` framework. The second is the `CFTree` pipeline developed by Noah Alting, which also processes airborne LiDAR data to produce 3D tree models. Both pipelines aim to automate the extraction of tree geometries from point cloud data, but they differ in their segmentation and reconstruction approaches. The `Tree4CFD` pipeline employs a marker-controlled watershed segmentation approach, while the `CFTree` pipeline uses a modified TreeSeparation algorithm. The reconstruction methods also differ, with `Tree4CFD` focusing on lightweight mesh generation via gaussian marching cubes and `CFTree` employing an alpha-wrap reconstruction technique. While both of the approaches are valid they lead to differing results which this project aims to evaluate in a quantitative and qualitative manner.

2.2. Project Summary

Bridging the gap between raw, unstructured point data and downstream microclimate simulation frameworks introduces computational and geometric trade-offs. Despite the practical demand for 3D vegetation in urban digital twins, current automated tree reconstruction workflows face three primary bottlenecks:

2.2.1. Tree Segmentation Scalability and Accuracy

Accurately detecting individual crown envelopes and trunk positions from unstructured airborne point clouds remains a challenge (Alting, 2025). Municipal tree inventories can be incomplete, out-of-date or non-existent, making validation and segmentation difficult. Consequently, existing point cloud pipelines struggle when municipal ground truth is absent. Comparing raster-based segmentation approaches against 3D point cloud algorithms is useful for determining robust, inventory independent segmentation methods.

2.2.2. Tree Reconstruction Scalability

While existing pipelines demonstrate the feasibility of generating mesh representations such as alpha-wrapped crowns (Alting, 2025), geometric extraction can scale poorly over large urban tiles. To enable city-wide deployment, lightweight reconstruction and volumetric approximation techniques must be systematically benchmarked to optimise memory usage, processing runtime, and geometry generation across extensive datasets.

2.2.3. Accessibility and Pipeline Interface

Current pipelines typically require executing complex scripts and manually handling massive point cloud datasets. This creates a technical barrier for end-users, such as urban planners or CFD engineers, who simply need an easy way to acquire reconstructed tree geometries for an area of interest. Currently, a user-friendly interface is missing that bridges the gap between heavy point cloud extraction pipelines and the on-demand retrieval of simulation-ready 3D models. To make these pipelines practically useful, there must be an accessible method to select a region, trigger the workflows, and retrieve the results without interacting with the underlying code.

2.3. Research Question

The central focus of the research is on the scalability of methods used to segment and reconstruct urban trees based on the computational time. Different levels of detail can capture varying aspects of trees geometries. We will therefore investigate whether increasing these differences justify computational effort. CFTree currently takes around 13 hours to process the entirety of Amsterdam, therefore gains on computational time can increase scalability of the existing pipeline. To examine the differences between different segmentation and reconstruction methods, the following central research question will be investigated:

How does the choice of airborne-LiDAR segmentation and reconstruction algorithms affect the scalability, geometric quality, and CFD compatibility of urban tree models?

This will help identify a suitable balance between geometric detail and computational efficiency for integrating point cloud derived trees into urban CFD workflows. The question is split into the following subsequent questions.

1. *How do individual tree segmentation methods affect the accuracy, scalability, and robustness of isolating single trees from airborne LiDAR data?*

Studies comparing individual tree segmentation methods show that accuracy depends on crown density, species composition, point-cloud characteristics, season, and algorithm parameters (Ma et al., 2022). For example, dense and overlapping deciduous crowns are repeatedly identified as difficult cases, while different methods can produce different tree counts and height distributions on the same area.

2. *How do different reconstruction algorithms compare in computational efficiency (processing speed, memory, scalability) and geometric quality (watertightness, completeness, complexity) when generating LoD3 tree geometries from city-scale airborne LiDAR data?*

Reconstruction speed for large urban areas is dependent on the choice of algorithm and increases linearly with the number of trees. Therefore, choosing a good reconstruction algorithm is important to find the best balance between performance and computational time. The robustness of each of the algorithms depends on the input given. Some algorithms lead to clean geometries on sparse data, while other might include holes, overlapping faces and other failures in the geometry.

3. *How can the tree reconstruction pipeline be integrated into an interactive user interface to facilitate the accessible, on-demand acquisition of 3D tree geometries?*

To evaluate the practical usability of the pipeline, it must be wrapped into an intuitive interface where users can easily define an area of interest and retrieve the generated geometries. This sub-question tests the effectiveness of bridging complex point-cloud processing with end-user accessibility. The interface must abstract away the computational heavy lifting while ensuring that the retrieved models retain geometric validity, proper geo-referencing, and structural compatibility with downstream frameworks like `City4CFD`, all with minimal manual intervention.

2.4. Research Limitations

While this research highlights several valuable directions for urban vegetation modelling, clear boundaries are set regarding scope, computational resources, data availability, and project duration.

First, the core objective centres on geometric extraction, volumetric reconstruction, and algorithmic benchmarking rather than end-to-end fluid dynamics simulations. Given the steep technical learning curve required for numerical CFD solvers, simulating aerodynamic behaviour is outside our current scope; our focus remains on generating simulation-ready geometries and parameterising crown volumes. Similarly, while deep learning architectures for species classification or generative modelling

present compelling results, project time constraints necessitate focusing on established, accessible point cloud processing methods and geometric pipelines.

Second, data availability and structural compatibility pose practical challenges. Evaluating algorithmic scalability across diverse urban morphologies benefits from international point cloud data; however, past experience demonstrates that cross-border datasets frequently exhibit inconsistent point densities, differing acquisition standards, and format variations that complicate pipeline integration. Therefore, this research will focus mostly on the Netherlands. Furthermore, validation is inherently constrained by the scarcity of accurate, high-resolution municipal tree registers. Without verified ground truth, different ways to perform comparative evaluations must be found.

2.5. Relevant Prior Knowledge

As second year students of MSc Geomatics, we have experience with point-cloud analysis, mesh reconstruction and geo-data acquisition. Together with master level knowledge of remote sensing, `Python` and `C++` workflow development is foundational in how we approach the project and understand prior research. In our academic and working careers, we have individually experienced working in a team on a project with assigned roles and responsibilities. Therefore, starting the project and setting up a planning is not new and is done without problems.

3. Theory and Context

To contextualise the development and benchmarking of the automated reconstruction pipeline, this chapter consists of a literature review of the existing scientific research related to the topic. Section 3.1 outlines different methods for tree segmentation and Section 3.2 describes methodologies for individual tree reconstruction. Although having a correctly classified point-cloud is important for filtering tree points, this research will focus on the segmentation and reconstruction of trees, and not on improving the classification of the point-cloud data.

3.1. Individual Tree Segmentation

The accurate detection and delineation of individual trees from airborne LiDAR point clouds is a foundational prerequisite for forest inventory, biomass estimation, and urban microclimate modelling (Liu et al., 2024; Yang et al., 2020). Over the past two decades, individual tree segmentation algorithms have evolved significantly, progressing from 2D raster-based image processing to direct 3D point-cloud operation, deep learning frameworks, and multi-stage hybrid pipelines.

3.1.1. Performance Metrics

In order to compare the performance of different segmentation methods, a few performance metrics are typically calculated; Precision, Recall and $F1$-score. In the context of tree segmentation, these metrics are defined as follows:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\ F1 &= 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \end{aligned} \tag{3.1}$$

TP = True Positives, FP = False Positives, FN = False Negatives.

TP, FP, and FN can be calculated by comparing the segmentation results with a reference dataset (ground truth). A true positive is a correctly detected tree (a detection having an inventory tree within 2.5m), a false positive is a detected tree that does not exist in the ground truth, and a false negative is a tree in the ground truth that was not detected. Each inventory tree can only be linked to one detection, and the other way around.

3.1.2. Segmentation Methodologies

Existing individual tree segmentation techniques can be classified into three primary frameworks: raster-based canopy height model (CHM) methods, 3D point cloud-based methods, and deep learning-based approaches (Liu et al., 2024; Yang et al., 2020).

CHM (raster) based methods

CHM-based methods simplify the 3D point cloud by interpolating elevation values into a continuous 2D height surface grid, framing individual tree extraction as a 2D image processing problem (Mu et al., 2020; Yang et al., 2020). Treetops are identified by finding local height maxima on the raster grid, which then serve as seed points to delineate tree crown boundaries across height gradients using standard image processing principles such as region growing, template matching, or watershed transforms (Figure 3.1) (Liu et al., 2024). Because these methods process 2D grid cells rather than unstructured 3D point sets, they are computationally fast and efficient for large-scale forest stands with distinct crown boundaries (Yang et al., 2020).

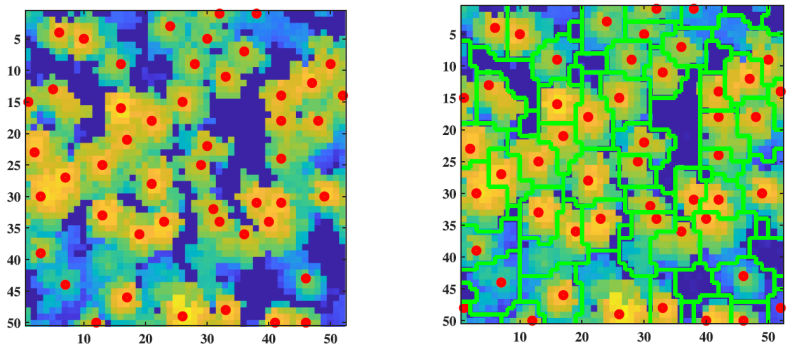


Figure 3.1: Tree plot with identified local maxima in a CHM for marker-controlled watershed segmentation of trees (Yang et al., 2020)

Point-cloud-based methods

Point-cloud-based methods operate directly within raw 3D coordinate space, avoiding 2D rasterization to preserve native structural details (Mu et al., 2020). By exploiting spatial relationships, local point density gradients, and 3D geometric distributions, these approaches group laser returns into individual tree instances using spatial clustering, graph-cut optimisation, or horizontal layer slicing techniques (Liu et al., 2024). Operating directly in 3D preserves native vertical foliage distributions and subdominant canopy structures, but these methods typically involve higher computational complexity and require explicit parameter tuning (Yang et al., 2020).

Deep Learning-based methods

Deep learning frameworks utilise neural network feature encoders to automatically learn spatial and geometric representations for individual tree detection and instance segmentation. These algorithms process either rasterized 2D multi-channel images using Convolutional Neural Networks (CNNs), or operate directly on 3D voxelised point representations using point-level network architectures (Liu et al., 2024). While deep learning models demonstrate strong performance in controlled environments, Liu et al. (2024) noted that they require massive hand-labelled training datasets, generalise poorly across different forest types or species; often lose 3D spatial features when rasterising point clouds into 2D images slices, and often struggles to generalise across diverse forest stands or species with irregular canopy forms. Therefore, this method will not be used in this research, as the focus is on developing a scalable and generalisable pipeline for urban tree segmentation and reconstruction.

3.1.3. Hybrid Method

Liu et al. (2024) conducted a comparative experiment benchmarking three distinct workflows across six forest plots of varying structural complexity;

- **Marker-controlled watershed algorithm**: Operating purely on 2D CHM rasters, the watershed algorithm was extremely fast, but suffered from noticeable under-segmentation, achieving an F_1 score of 0.837 and a precision of 0.849.
- **3D spectral clustering**: Operating directly on 3D point clouds, spectral clustering required extensive iterative super-voxel calculations, making it computationally slow while suffering from severe over-segmentation ($F_1 = 0.636$ and precision = 0.837)
- **Hybrid method**: By combining a 2D marker watershed coarse pass with multi-directional 3D vertical profile analysis and spectral clustering on under-segmented patches, the hybrid approach achieved the highest overall accuracy (F_1-score $= 0.892$, precision $= 0.937$) while maintaining high computational efficiency.

It is important to note that this research was done on homogeneous forest areas, making segmentation a less complex task compared to the variable urban fabrics.

3.1.4. Existing Segmentation Pipelines

`Tree4CFD` uses a marker-controlled watershed method to segment the point cloud into individual trees. The watershed algorithm is applied to a Canopy Height Model (CHM) raster derived from the 3D point cloud. Local maxima in the CHM are detected as tree tops and used as markers for the watershed segmentation. After segmentation, a series of filters are applied to remove non-tree segments based on minimum height, minimum number of points, and OpenStreetMap building footprints.

In the `CFTree` pipeline, the core segmentation process relies on a compiled `C++` binary named TreeSeparation to process the point cloud data. It separates individual trees based on three predefined parameters passed from the `Python` wrapper: a search radius of 2.5 meters, a vertical resolution of 1.5 meters, and a minimum threshold of 3 points per tree.

3.2. Tree Reconstruction Algorithms

Three-dimensional tree reconstruction algorithms convert spatial data into digital 3D models suitable for visualisation and environmental simulations. In the context of CFD and urban microclimate modelling, reconstruction methods must balance visual fidelity, topological integrity, physical parameter representation, and computational scalability (Alting, 2025; Fu et al., 2024).

3.2.1. Level of Detail (LoD) in Trees

To represent vegetation within 3D city models, tree geometries are abstracted into different Level of Detail (LoD) representations (Alting, 2025). LoD0 represents the most basic level of detail in the classification, however it can produce no benefit to 3D CFD simulations as that abstraction level exists only in a 2-dimensional plane. LoD1 is the coarsest 3-dimensional abstraction level where the canopy is represented as a simple prismatic porous block, cylinder, or bounding box. LoD2 is more detailed and consists of predefined parametric primitives. These can be dynamically scaled in height and crown width extracted from the bounding dimensions of individual tree point clusters (Alting, 2025; Fu et al., 2024). LoD3 represents the most detailed abstraction level, where explicit outer crown forms are modelled. The different abstraction levels are visualised in Figure 3.2.

LoD2 can capture relevant macroscale effects of vegetation on wind flow while avoiding the higher geometric and meshing complexity associated with more more detailed representations. Research of Fu et al. (2024) shows that differences in wind velocity magnitude between LoD2 and LoD3 trees are generally smaller than 5%, while the difference between LoD1 and LoD3 can be up to 30%. This indicates that both LoD2 and LoD3 are sufficient for urban microclimate simulations.

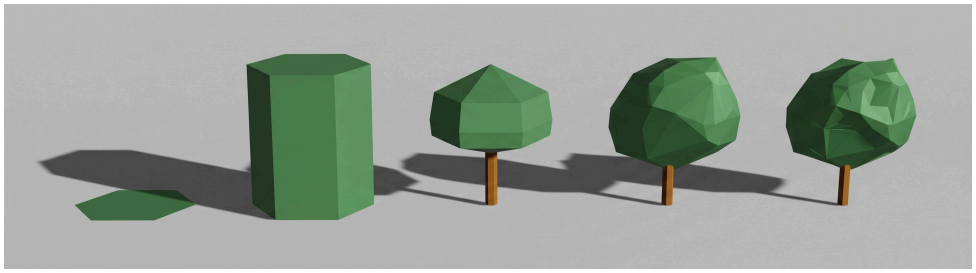


Figure 3.2: Reconstructed tree models of different LoD classifications (de Groot, 2020) (*from left to right: LoD0, LoD1, LoD2, LoD3.1, LoD3.2*)

A tree can be generated at LoD2 using a procedural modelling approach, where a predefined parametric tree icon is fitted to the point cloud data. The icon is defined by a set of parameters that describe the tree's geometry, such as tree top, periphery, and crown base (Figure 3.3). These parameters are extracted from the point cloud data and used to fit the icon to the tree's actual dimensions (Biljecki et al., 2015).

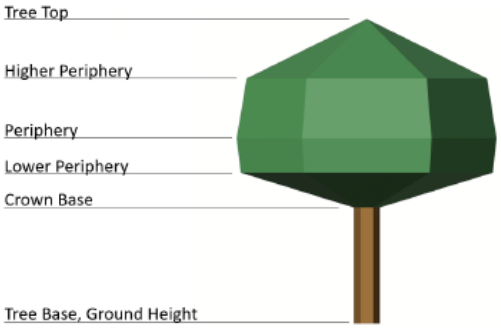


Figure 3.3: A tree represented at LoD2 and the main parameters for procedural modelling indicated (de Groot, 2020)

When explicit outer crown forms (LoD3) are needed, three main surface wrapping approaches can be employed:

Classic 3D alpha shapes

Alpha shapes generalise the convex hull by carving out concavities controlled by a radius parameter α, visualised in Figure 3.4. Decreasing α allows the surface to follow tight canopy concavities. However, when applied to sparse or non-uniform airborne LiDAR data, classical alpha shapes can produce open boundaries, self-intersecting faces, or non-manifold edges (de Groot, 2020).

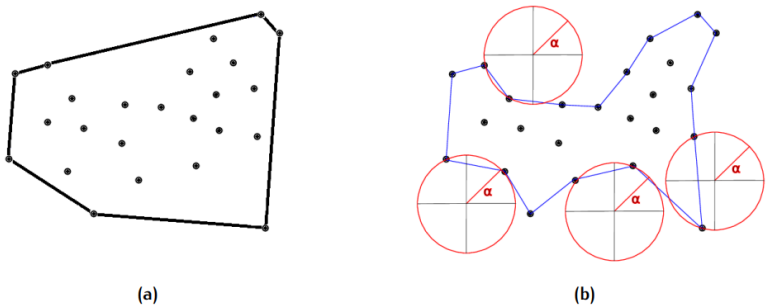


Figure 3.4: Convex hull of a point set (a) vs. alpha shape of the same point set (b) (de Groot, 2020)

3D Alpha Wrapping

The CGAL Alpha Wrapping algorithm (AW3) is used in `CFTree` and enforces manifoldness and watertightness to overcome the topological instabilities of classic alpha shapes. The algorithm shrink-wraps a continuous, triangular surface envelope using feature scale α and tightness δ. Both parameters are normalised and when keeping a fixed ratio between them yields consistent tightness across tree sizes (Alting, 2025).

Gaussian density marching cubes

As implemented in the current `Tree4CFD` pipeline, this approach maps 3D canopy points onto an occupancy voxel grid, applies 3D Gaussian spatial smoothing to obtain a continuous scalar density field, and prunes small disconnected blobs using 26-connectivity. The final watertight surface mesh is extracted from the cleaned density field at a specified isosurface threshold using the Marching Cubes algorithm.

3.2.2. Problems in Reconstruction Methods

While the LoD3 can produce better results in terms of accuracy in simulations and of visuals, it is still crucial to evaluate the geometric quality of the reconstructed tree models and whether the benefits outweigh the computational cost. The reconstruction process also requires consideration of any artefacts possibly present in the output of the segmentation process, as these can cause misrepresentation of the canopy as seen in (b) of Figure 3.5, as well as weighing out the possibility of under- and over-segmentation, as merged canopies can impact the placement of trunks like in (a) of Figure 3.5.

As the models are intended for use in urban CFD workflows, they need to be suitable for further processing and meshing. Rather than testing the models directly in CFD simulations, CFD compatibility will be assessed trough properties of the reconstructed models. This includes properties such as watertightness, level of detail, and geometric accuracy and validity, which can become problematic for automatically segmented point cloud files.



(a) Under-segmentation **(b)** Outliers **(c)** Misclassification

Figure 3.5: Three possible issues that can appear in LoD3 tree reconstruction resulting in weirdly shaped crown geometries (de Groot, 2020)

4. Data Collection and Availability

4.1. Point Cloud Data

The primary data source for this research is the Dutch Airborne LiDAR point-cloud data (AHN). The AHN5 and AHN6 datasets provide high-resolution 3D point clouds of the Netherlands, capturing detailed information about the terrain and vegetation. As both of them only cover a part of the Netherlands, both are necessary to cover the entire study area. Filtering out everything except the vegetation point would be ideal to start segmentation. However, AHN data keeps vegetation points as unclassified and they can therefore not be extracted as a separate set. The resulting set (all unclassified points) also include objects like cars, fences, and building walls.

4.2. Satellite Imagery

Satellite imagery is used to provide additional context and information about the study area. The Sentinel-2 satellite provides high-resolution multispectral imagery, which can be used to derive vegetation indices such as the Normalized Difference Vegetation Index (NDVI). NDVI is a widely used index for assessing vegetation health and density, and it can be used to identify areas with dense tree cover. Additionally, satellite images are used for annotating trees to ensure complete tree inventories used to validate the results.

4.3. Segmentation

4.3.1. Area Selection

Initially, eighteen areas were selected; for each city, two areas per Urban Density Class (UDC) (high, mid, low). The zones were selected using visual inspection through aerial imagery. Densely built-up areas (high) are situated predominantly in the city centres, averagely built up areas (mid) in neighbourhoods around secondary city centres, while the areas categorised as sparsely built-up (low) areas all contain a park and mid/low rise buildings.

Following area selection, various NDVI and building features within the chosen areas were determined in order to systematize the UDC analysis. These include the mean and median NDVI values and building coverage ratio. In Figure 4.1, these features are plotted as a box plot grouped per urban density area. Each dot represents one of the 18 selected areas. After analysis of these results, categorization of 3 areas was adjusted. As visible in the box-plot, the “mid” and “low” boxes are partially overlapping and the “low” boxes are stretched out more than the other categories. Three outlying areas in “mid” or “low” were recategorized or removed completely.

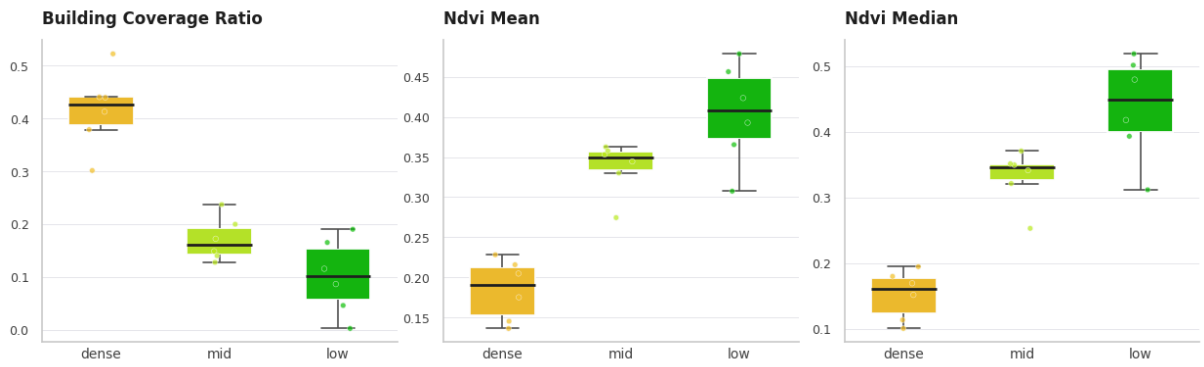


Figure 4.1: Boxplots of the three determined features categorised per urban density area, showing a high variance in “mid” and “low” areas.

The resulting changes are displayed in Figure 4.2. The recategorization resulted in a smaller within-class variance, which helps the classification of the UDCC of a heuristically selected area to be more accurate. The final classified areas can be seen in Figure 4.3.

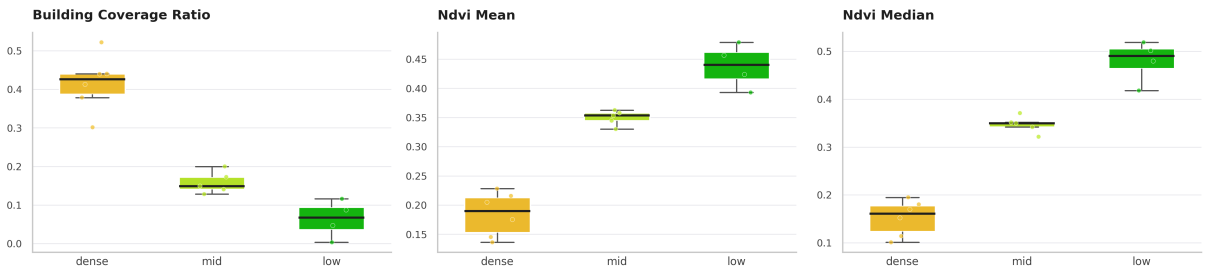
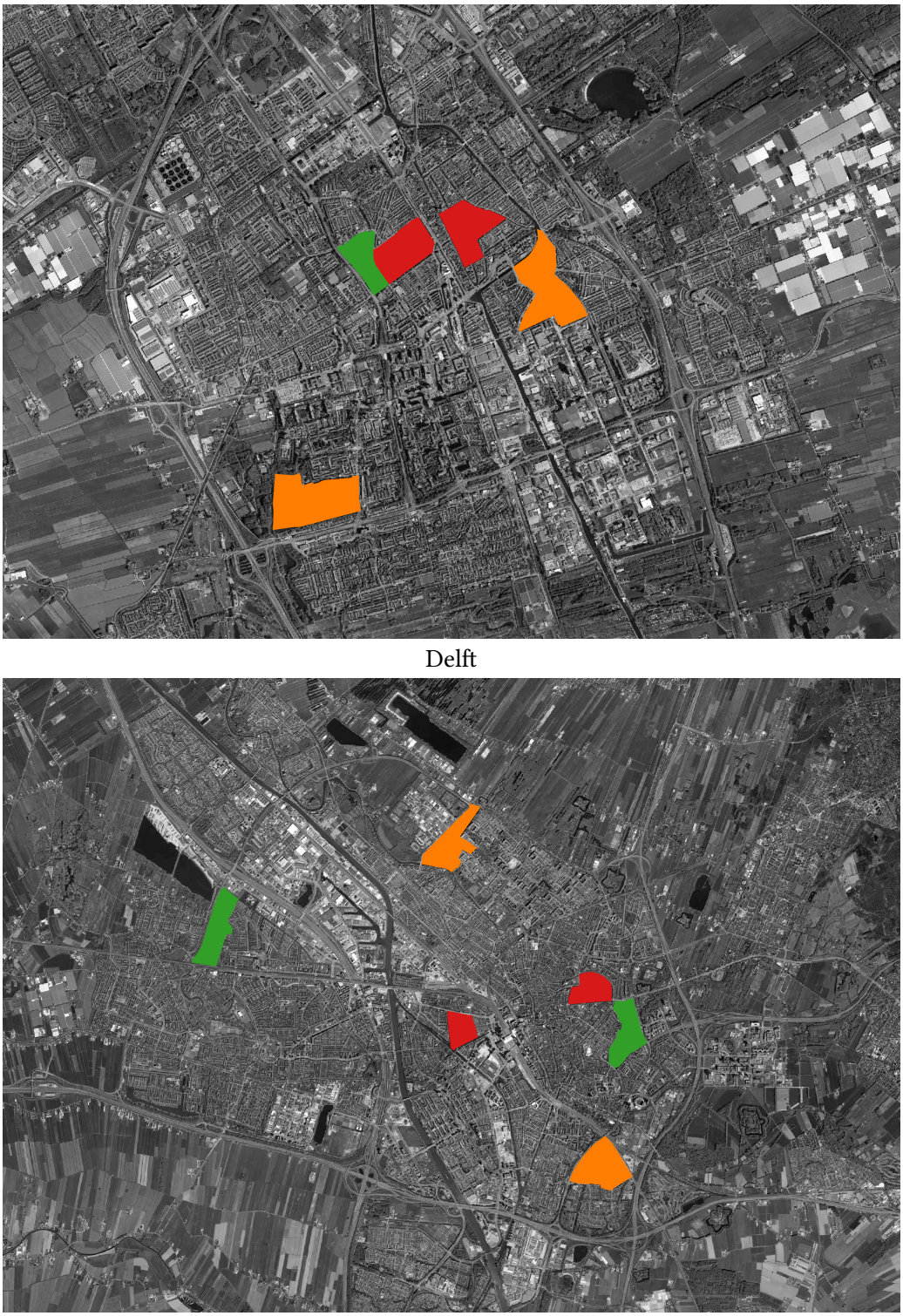


Figure 4.2: Updated boxplots after changing the urban density class of three areas, resulting in more compact boxes.



Amsterdam



Delft

Utrecht

Figure 4.3: Overview of the selected urban density areas in Amsterdam, Delft and Utrecht. The areas are categorised as high (red), mid (orange) and low (green) built up areas.

4.3.2. Tree Inventories

To validate the segmentation, a tree inventory dataset is necessary. Where available, we collected municipal tree inventories to be used as a reference dataset for evaluating the performance of the segmentation method.

The available tree inventories mostly covered trees managed by the municipalities, making it hard

to evaluate results ran on unprocessed data. Two approaches were identified to deal with this: (1) annotation of the missing data in QGIS: adding individual tree points with a recent aerial image as underlay to complete the existing tree inventories, and (2) extraction of point-cloud data using street geometry and buffers to study only public spaces, which would reduce data preparation overhead.
In order to make the final pipeline scalable for complete point cloud tiles, annotation was chosen as the method to proceed with. This way a ground truth per neighbourhood can be established for parameter tuning, after which the finalised segmentation method producing sufficiently accurate results will be scalable; this was done with the initial assumption that the aerial imagery was mostly representative of the pointcloud datasets.

Each of the 17 defined areas was buffered by 0.5m for point-cloud clipping, ensuring that most of the canopies of edge trees are not truncated during data extraction, while the unbuffered boundary is retained for counting detected/registered trees for the actual accuracy comparison.

While for more densely built up areas counting trees was easier, particular attention is given to challenging cases such as dense vegetation and overlapping tree crowns. Manual annotation using QGIS with a recent aerial image as underlay was completed for all case studies, and allowed for visual inspection of existing tree inventory datasets against an aerial imagery ground truth proving that tree inventories were in all cases incomplete. Unfortunately, manually annotating still resulted in non-perfect tree inventories. Comparing the inventories against Street View imagery and actual point cloud data would have resolved this issue, but was too time-consuming within this project. The resulting tree inventories can be found in Appendix C.

4.4. 3D BAG

The building data from 3D BAG is used to provide additional context and information about the study area. By simultaneously retrieving the building and the tree geometries, the output files can be readily used in a complete CFD simulation. In this project this was used instead of `City4CFD` because the 3D BAG provides already reconstructed geometries, instead of generating them on the spot. For this project, that was easier to implement within the given time frame.

5. Research Methodology

To answer our research questions, we will establish an experimental framework focused on automated tree segmentation, reconstruction performance, scalability, and CFD compatibility. Our evaluation prioritises the scalability of the reconstruction pipeline and the topological quality of the resulting 3D geometries for integration into urban microclimate workflows under consistent input and processing conditions.

The methodology is divided into three stages: (1) individual tree segmentation, (2) tree reconstruction, and (3) scalability and accessibility.

5.1. Individual Tree Segmentation

As the current segmentation method needs improvement, different methodologies will be implemented to review their effectiveness. The first step is to include a point cloud based method, which looks at the 3D structure and shape of the clusters. The second improvement will be to use a variable window size in the CHM peak detection.

5.1.1. Point cloud based implementation

As described in Section 3.1, the first improvement in the segmentation of trees will use a hybrid approach. This algorithm can be divided into two stages; (1) 2D CHM coarse pass and (2) multi-directional analysis on multi-tree clusters to further segment them.

2D CHM Coarse Pass

This first phase was already integrated in the `Tree4CFD` pipeline from Themis. It projects the point-cloud to 2D and generates the Canopy Height Model (CHM). After smoothing, local maxima are detected as tree tops from the CHM. These maxima are then used as seeds for the watershed segmentation algorithm. Different filters, like minimum height, minimum amount of points, and OpenStreetMap building footprints, are implemented to remove non-tree segments, which accounts for the imperfect classification of the Dutch point clouds.

Multi-Directional Analysis

After the marker-controlled watershed segmentation, many clusters still contain multiple trees. These clusters are selected using a 2D Principal Component Analysis (PCA). Each segmented point cluster is projected onto the 2D (x, y) plane and the circularity and treetop centredness are extracted. Clusters only containing a single tree will most likely conform to both metrics and can therefore be filtered out, as shown in Figure 5.1.

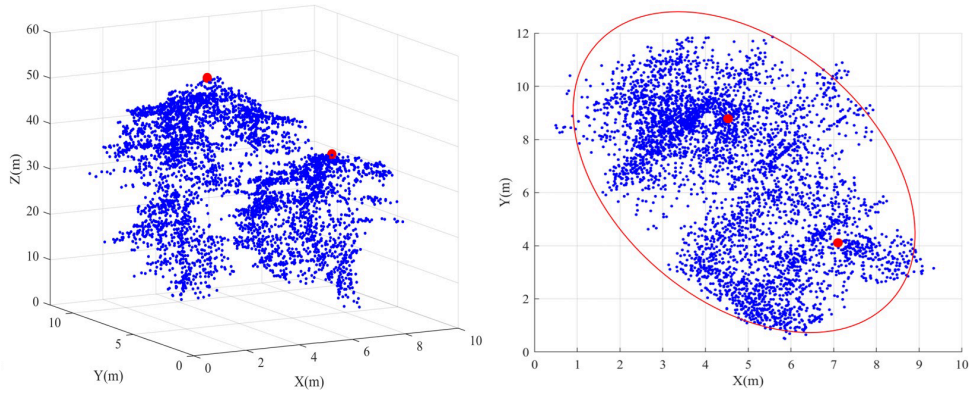


Figure 5.1: The differentiation in shapes is evident between correctly segmented patches and under-segmented patches. The image illustrate an under-segmented patches in a 3D and 2D projection (Liu et al., 2024).

The detected under-segmented clusters are projected along four vertical planes at different angles. At each vertical slice, the point cloud is analysed to detect secondary tree tops, see Figure 5.2. Tree peaks are detected using the `find_peaks` function from `SciPy`. Filters like minimum distance between treetops and margin to the edge of the cluster are implemented to get rid of falsely detected tree tops. The newly detected tree peaks are then used to further segment the cluster. Each point is assigned to the nearest (2D Euclidean distance) treetop.

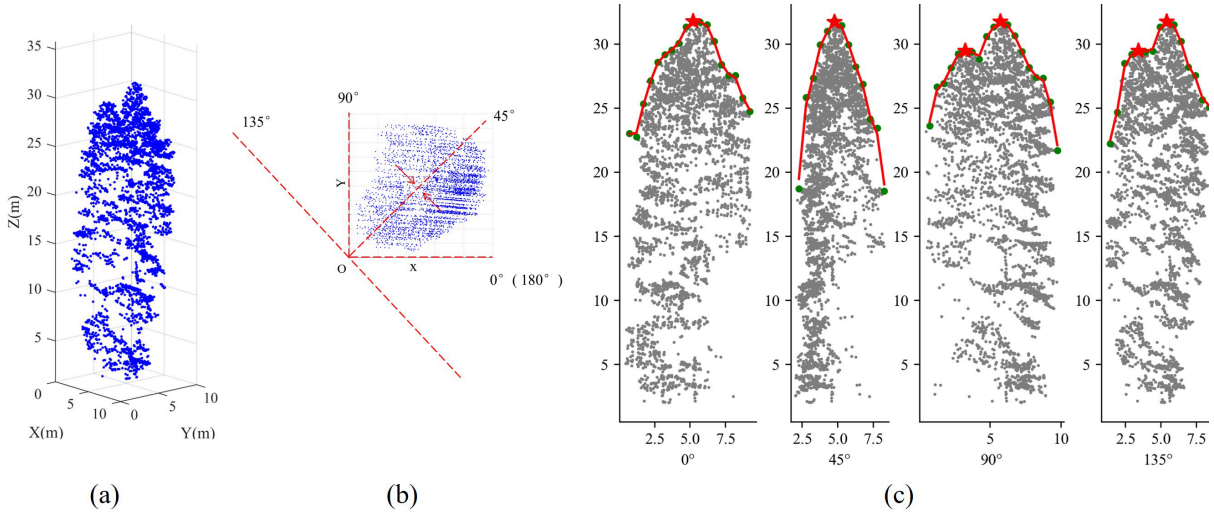


Figure 5.2: Analysis of 2D vertical profiles based on multi-directional crown projection (Liu et al., 2024)

5.1.2. Variable window size

The next improvement that will be made, is implementing a variable window size (minimum peak distance) that is used for detecting the tree tops on the CHM. Heights of treetops and UDC will be used to determine the window size. The different urban density classes use different formulas to determine the detection radius for each height bin. The NDVI mean and median and the building coverage ratio will be calculated for the area and compared to the trained values from the annotated areas in Delft, Utrecht, and Amsterdam. The assigned UDC is the one the values are closest to.

As larger areas can contain multiple UDCs, the selected area will be split into multiple smaller polygons, each with a height and width of roughly 500 meters. The urban density class of each subpolygon is determined, and segmentation is performed separately for each subpolygon. After, everything is combined into one resulting segmentation. An example of splitting and classifying is displayed in Figure 5.3.



Figure 5.3: A selected area in Rotterdam on the left and the subpolygons after splitting on the right. Green polygons are classified as low building density, orange as mid building density.

5.1.3. Parameter tuning

After this segmentation step is implemented, the next step will be to tune the parameters used for each function. The parameters in the segmentation code can be grouped into four categories corresponding to separate stages:

CHM / seed generation	Geometry filtering	Secondary peak detection	Final clustering
Cell size	Minimum height	Maximum elongation	2D Euclidean distance
Smooth sigma	Minimum tree points	Maximum offset ratio	Margin distance
Minimum peak distance		Bin size	

Table 5.1: Different parameters per segmentation stage.

To start off the parameter tuning, a sensitivity analysis will be performed to determine which parameters have the most significant impact on the segmentation performance. Based on this analysis, the three most sensitive parameters were selected for further tuning. A grid search is used to test different combinations of parameter values and identify the optimal configuration. The performance of each parameter combination was evaluated using the $F1$-score, which balances precision and recall, providing a comprehensive measure of segmentation quality.

5.2. Tree Reconstruction

Following successful segmentation, the isolated point clouds are converted into three-dimensional tree geometries. The current `Tree4CFD` pipeline implements reconstruction algorithms for Level of Detail 1, 2, and 3 representations. To determine the most scalable approach for processing large urban areas, these methods will be benchmarked primarily on their computational efficiency.

A secondary metric is the storage footprint of the generated geometries, which must remain manageable for integration into urban microclimate simulations. To mitigate excessive file sizes, optimisation strategies will be investigated. This benchmarking process will analyse the trade-off between geometric detail and processing time, specifically comparing the procedural generation of LoD2 against the outer crown forms of LoD3. The objective is to determine whether the structural advantages of highly complex geometries like the Marching Cubes algorithm justify their additional computational overhead for the intended CFD applications, or if a simplified geometry generated with Alpha wrapping is sufficient.

In the absence of a definitive ground truth for the reconstruction stage, performance evaluation will rely on qualitative assessment and visual inspection. Finally, to ensure topological correctness and strict compatibility with simulation environments, `val3dity` will be utilised to validate the output geometries.

5.3. Scalability and Framework Integration

The final stage of the methodology focuses on the practical usability of the pipeline by developing an accessible, interactive user interface. As outlined in the third sub-question, the goal is to bridge the gap between computationally heavy point-cloud processing and the on-demand acquisition of 3D tree geometries for end-users.

To achieve this, an interactive web application will be developed to abstract the underlying code execution. Ultimately the goal would be to run it on a TU Delft server. The interface will be designed to

allow users to define a specific area of interest by either uploading a polygon or drawing one directly on the map. Different settings for the desired output can be selected before running the pipeline. The application will trigger the complete workflow without requiring any manual intervention. This will include downloading the corresponding AHN tiles, clipping the point cloud to the user-defined boundaries, executing the tree segmentation pipeline and generating the final 3D geometries.

The evaluation of this stage will center on the seamless integration of the processing pipeline into the front-end interface and its effectiveness in facilitating on-demand retrieval of tree geometries, ready for integration into frameworks like `City4CFD`.

6. Results

This chapter presents the outcomes of the 3D tree segmentation and reconstruction pipeline, detailing the algorithm's performance enhancements, geometric validation, and the development of an accessible user interface.

6.1. Segmentation

The segmentation algorithm uses various parameters, each having influence on the results separately. To obtain the most accurate results as possible, the optimal parameter values have to be determined. In theory, the different urban density areas (high, mid, low) use different values for optimal results. For example, high urban density areas have mostly individually standing trees, therefore having a high peak minimum distance will result in less false positives. Tree distribution is more clustered in areas of low urban density, a lower peak minimum distance for those areas theoretically reduces the number of false negatives.

6.1.1. Sensitivity Analysis

To start off the parameter tuning, a sensitivity analysis was performed to determine which parameters have the most influence on the segmentation results. The sensitivity analysis was performed by varying each parameter within a reasonable range while keeping the other parameters fixed at their default values. Table 6.1 describes each parameter and its corresponding default value and range. The performance of the segmentation was evaluated using the $F1$-score, which is a measure of accuracy that considers both precision and recall. The results of the sensitivity analysis can be viewed in Appendix D and show that the most sensitive parameters are smooth sigma, cell size and peak minimum distance.

Parameter	Description	Default value	Range
Cell Size	The raster cell size used when creating the Canopy Height Model	1.0	0.5 - 2.5
Smooth Sigma	The Gaussian blur standard deviation used to smooth small canopy noise before local maxima detection	1.55	1.0 - 3.5
Minimum height	The minimum CHM height threshold to be considered a tree	2.5	1.0 - 3.5
Minimum peak distance	The minimum Euclidean distance required between local maxima on the CHM grid	3.0	1.0 - 6.0
Elongation threshold	The maximum allowed ratio between the major and minor horizontal standard deviations before a segment is flagged as elongated	3.0	1.0 - 6.0
Offset ratio threshold	The maximum allowed ratio of peak-to-centroid offset over the major axis standard deviation before flagging a segment as off-centre	1.5	0.5 - 3.0
Bin size	The horizontal bin width along each projection vector for binning points into a profile	0.5	0.25 - 1.25
Minimum peak distance (secondary)	The minimum horizontal spacing required between peaks along a profile	3.0	2.0 - 5.0

Parameter	Description	Default value	Range
Euclidean distance threshold	The minimum 2D horizontal distance a candidate peak must have from the dominant treetop to be accepted	2.5	1.0 - 3.5
Margin distance threshold	The minimum distance a candidate peak must lie away from the horizontal convex hull boundary of the cluster	1.5	0.5 - 2.5

Table 6.1: Tweaked parameters with their corresponding ranges.

6.1.2. Parameter tuning

3D point cloud method

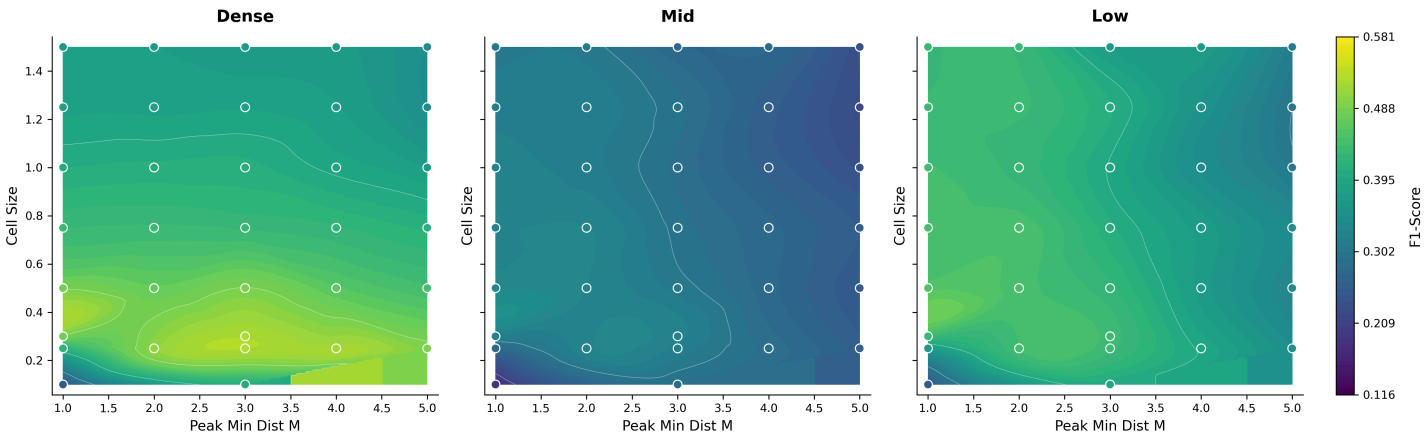
For parameter tuning, a grid search algorithm was applied on the 17 selected areas. With point-clouds covering three cities, Amsterdam, Delft and Utrecht, and three UDCs, a grand variety of files were tested to get results for the optimal parameters per zone. The tuned parameters and their corresponding ranges are shown in Table 6.2. The parameters being tuned are the most sensitive parameters, and the ranges are based on the default values of the already existing algorithm, values around these default values are checked.

Parameter	Minimum value	Maximum value
Smooth Sigma	0.5	2.5
Cell Size	0.1	1.5
Peak Minimum Distance	1.0	5.0

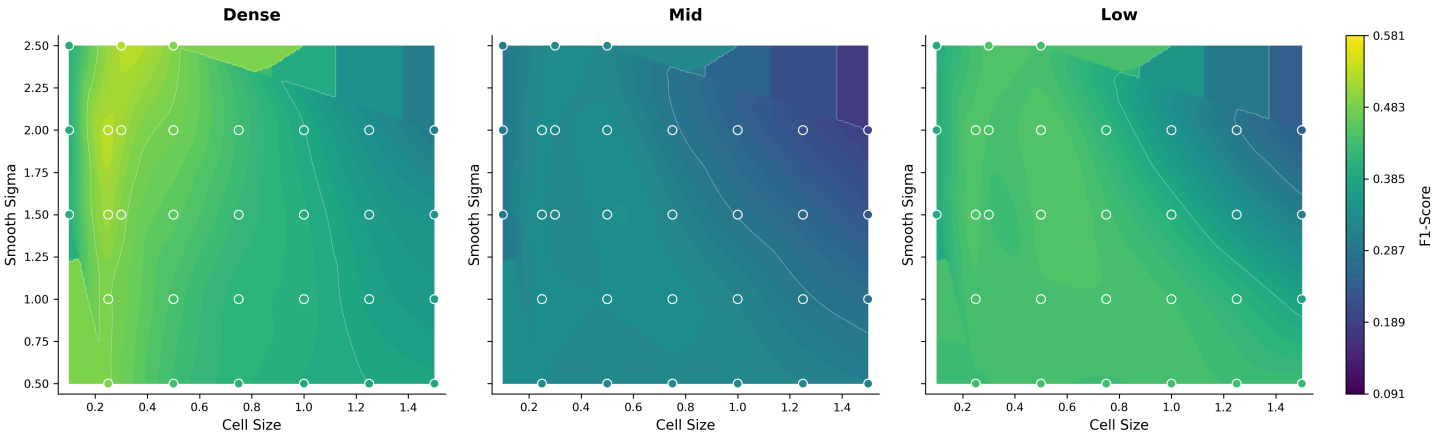
Table 6.2: Tweaked parameters with their corresponding ranges.

In Figure 6.1 the results of the parameter tuning are displayed. The columns correspond to different urban density categories, while the rows represent different pairs of parameters, as visualising all three parameters simultaneously would be impractical. The colour scale in each plot indicates the interpolated $F1$-score for different combinations of the two parameters shown. The third parameter, when not shown in a plot, is set to the value that yields the maximum $F1$-score for each combination of the two displayed parameters.

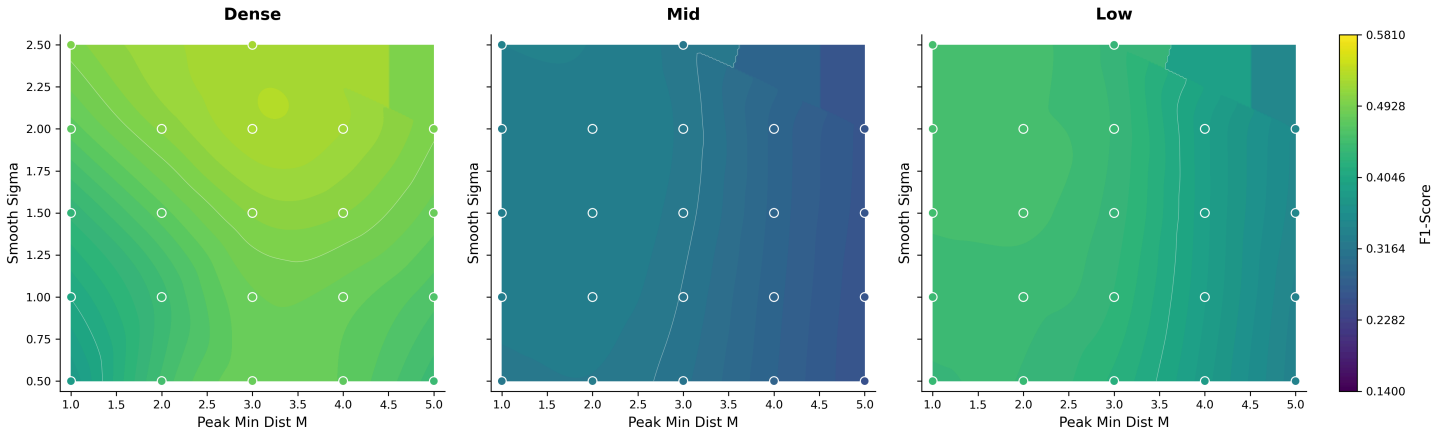
All the values are an average of the different areas corresponding to the UDC. For example, the dense plots show the average results of amsterdam-dense, delft-dense and utrecht-dense.



Cell size vs. Minimum peak distance



Smooth sigma vs. Cell size



Smooth sigma vs. Minimum peak distance

Figure 6.1: Interpolated heat maps for averaged per urban density type for each of the parameter combinations with the color indicating the resulting $F1$-score.

The optimal parameter values per UDC obtained by the grid search can be found in Table 6.3.

Dataset	Cell Size	Smooth Sigma	Minimum Peak Distance	Average F1-score
Dense	0.2706	2.2063	3.0325	0.5601
Mid	0.4578	1.9714	1.1689	0.4903
Low	0.4552	2.2332	1.0000	0.4769

Table 6.3: Optimal parameters for each urban density zone resulting from the grid search parameter tuning process.

The variance of the optimal parameter values per UDC is shown in Figure 6.2. This displays the overlap of the optimal parameters between UDCs and the wide spread within one class. Dense area indicate more consistent optimal parameters, but mid and low have a large variance. Thus, for UDCs to use their own smooth sigma and cell size values, more areas should be annotated and analysed.

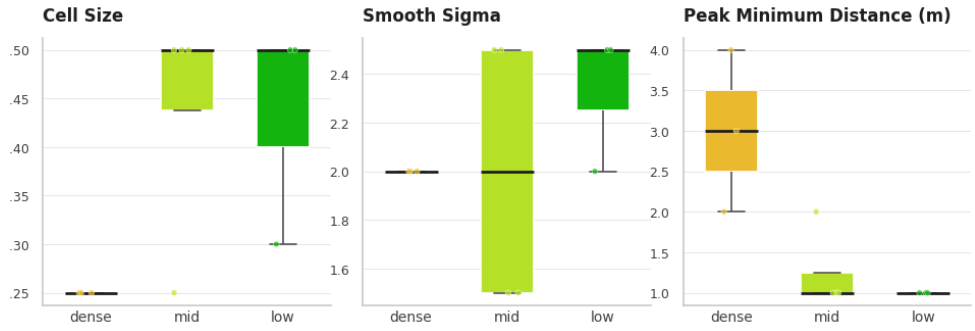


Figure 6.2: Box plots showing the spread of the optimal parameters per urban density class.

Variable Window Size

Because the resulting $F1$-scores are low compared to the literature, the segmentation method was further improved by implementing a variable-window size in the CHM peak search. Again, parameter tuning was performed on the new algorithm, but as the peak minimum distance is based on the urban density class in this algorithm, the parameter was left out. The resulting plots are displayed in Figure 6.3 and the new optimal parameters are in Table 6.4.

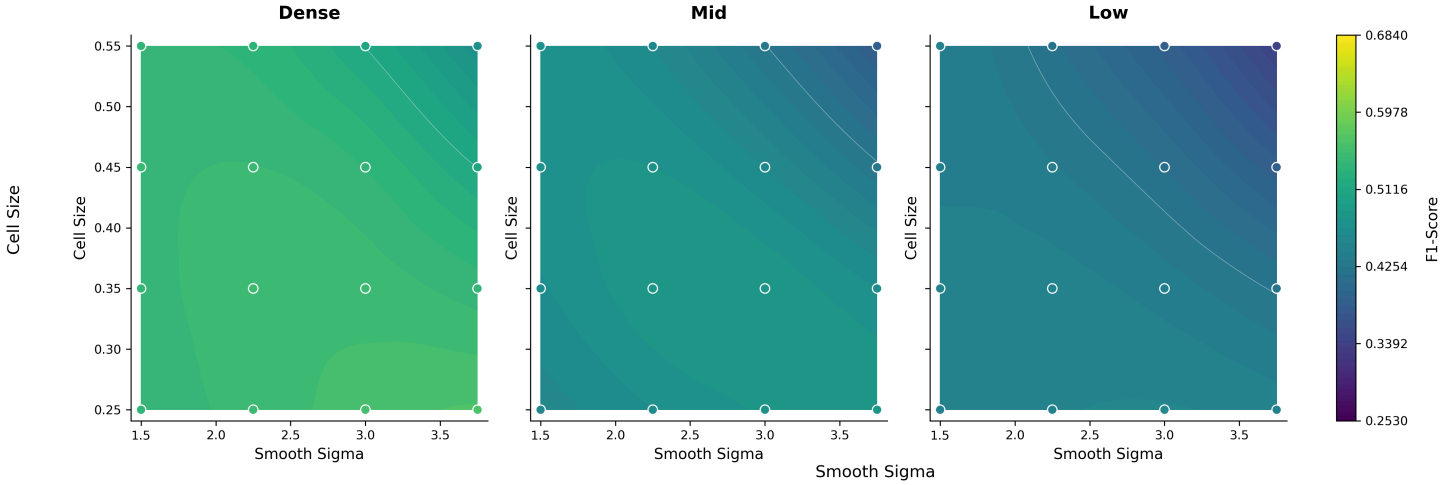


Figure 6.3: Interpolated heat maps for different urban density zones with parameters smooth sigma and cell size after implementation of the variable window size.

Urban density	Cell size	Smooth sigma	Average best F1-score
Dense	0.2822	3.1508	0.5737
Medium	0.3314	3.0401	0.4866
Low	0.3046	2.7932	0.4553

Table 6.4: Optimal parameters for the segmentation algorithm including the variable window size, together with the respective $F1$-score.

6.1.3. Benchmarking the Different Segmentation Methods

Comparing the $F1$-scores between the old and the modified segmentation algorithm implementation tells us whether the results are an improvement or a decrease in performance. Therefore, the different methods were run separately on the 18 areas and the resulting $F1$-scores are displayed in Figure 6.4.

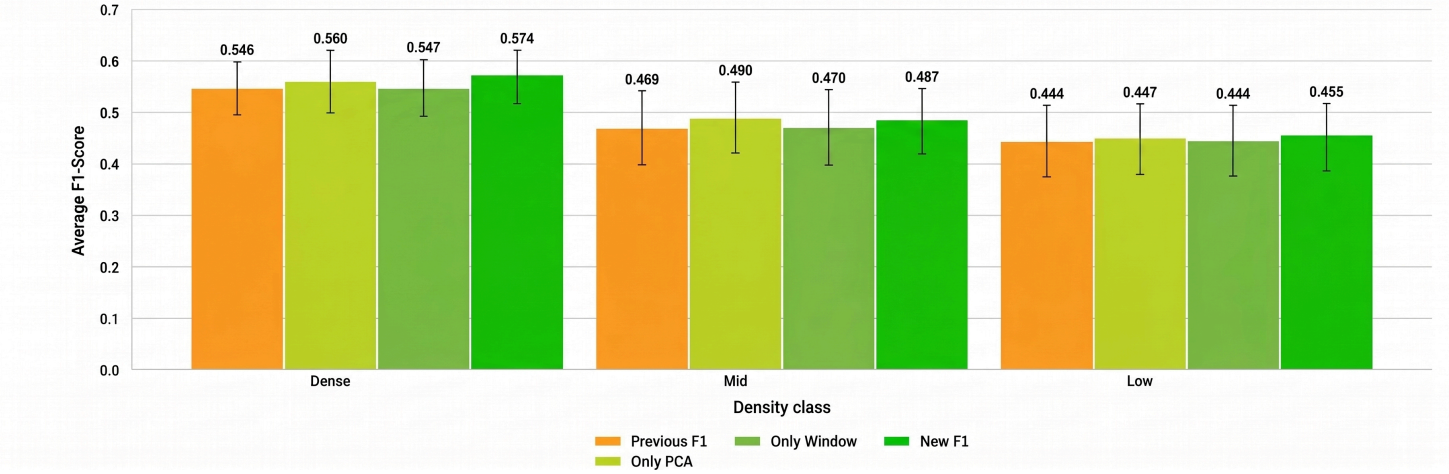


Figure 6.4: Average $F1$-scores per urban density type for the different implementations; original method, only PCA, only variable windows, and the complete method.

For these runs the precision and recall were determined as well. An improvement in precision means there are less false positives (e.g. less oversegmentation), and an improvement in recall means there are less false negatives. The value for the different method combinations can be seen in Table 6.5.

Metric	Old average	New average	Change
Precision	0.409	0.468	+0.59
Recall	0.583	0.531	−0.52

Table 6.5: Average Recall and Precision scores for the different implementations; original method and the complete method.

6.1.4. Verifying methodology on unseen areas

The results were so far only tested on the seventeen selected areas. Next, one AHN tile area in Amsterdam was intensively, manually annotated using both an aerial image and the actual point cloud. This way a "perfect" ground truth was generated that can be used for verification. When running the pipeline for this tile and comparing the result against the inventory, the precision (0.763), recall (0.601), and $F1$-score (0.673) were calculated. The resulting $F1$-value is about 30% higher then the average of the seventeen areas.

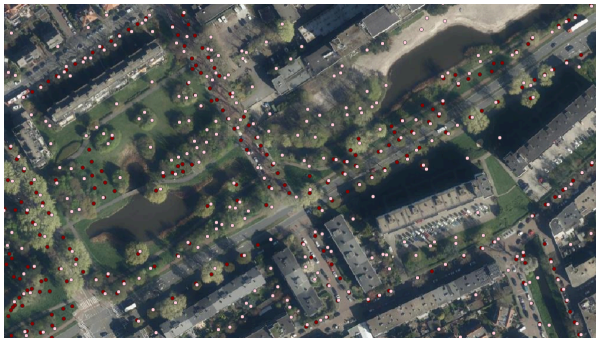
To check whether the method also works on other areas, the pipeline was run on new cities where there is a municipal tree inventory available, but no additional annotation was done. As these inventories are incomplete, we selected areas where most of the trees were covered in the dataset. This way it was still possible to calculate an $F1$-score for each of the areas. The detected trees were also visualised in QGIS to identify the areas that incurred problems. A few zoomed in screenshot are shown in Figure 6.6.



Figure 6.5: Complete tree inventory (red) and detected trees (pink) for one tile in Amsterdam (Javaeiland).



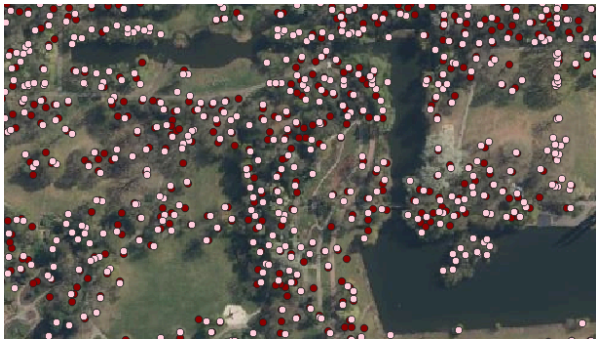
Alkmaar - $F1 = 0.574$



The Hague - $F1 = 0.447$



Breda - $F1 = 0.658$



Eindhoven - $F1 = 0.560$

Figure 6.6: Municipal trees (red) and detected trees (pink) in a zoomed part of different cities including their $F1$-scores

For the Alkmaar area, the results are promising. Mostly small trees (with little points) are missing. In The Hague, most problems come from incomplete inventory. In the park-like area, multiple tree clusters were not included in the inventory and therefore skew the results. Breda shows even better results, with an $F1$ score of 0.658. A few rows of small, newly planted trees are missing, but most trees are correctly identified. In a park in Eindhoven, some trees are oversegmented and tree tops are not at the exact same location as the inventory trees, but overall the trees are segmented relatively well.

It stands out that these $F1$-scores are mostly higher than the ones resulting from the 17 additionally annotated areas.

6.2. Reconstruction

To evaluate the impact of reconstruction on overall processing time, a comparison was made between the reconstruction times of LoD2 and LoD3 models. The results of this comparison are presented in Figure 6.7, which illustrates the time difference between reconstruction processes. The plot shows that the time difference of reconstructing 40.000 trees is only ~15 second and is therefore negligible. This indicates that the reconstruction process is efficient across all LoDs and does not significantly affect the overall processing time leading to the conclusion that the current implementation and choice of reconstruction LoD is not a bottleneck and fits well within the scalability of the pipeline.

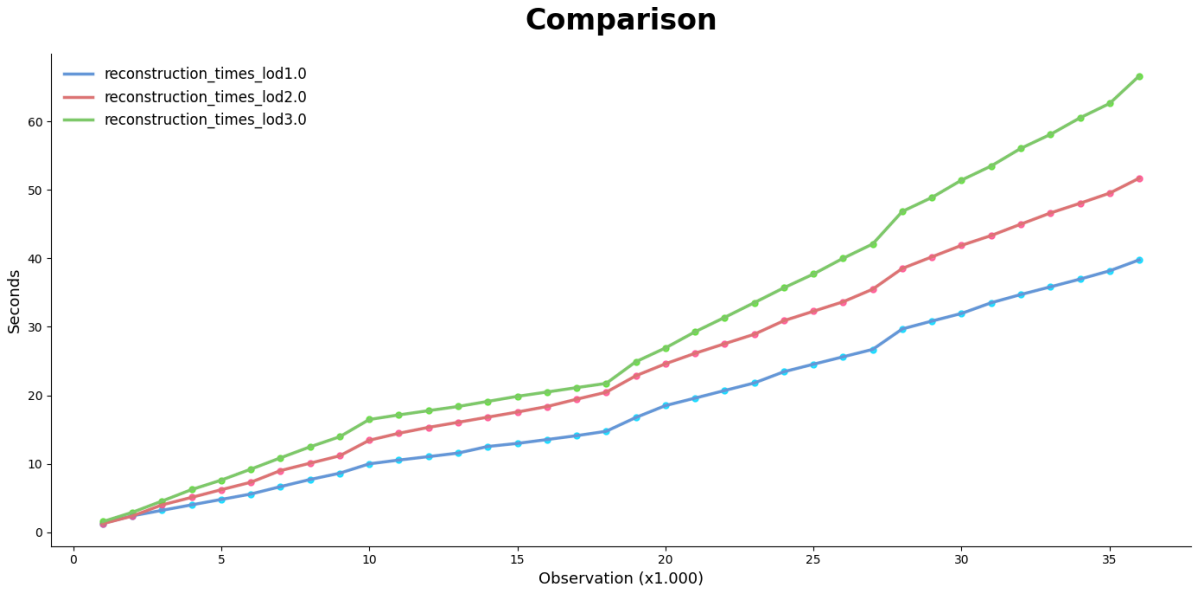


Figure 6.7: Reconstruction time of trees at different LoDs.

6.2.1. Visual Inspection of Reconstructed Trees

Because no ground truth exists or can be created, the reconstructed trees needed to be visually inspected in order to determine the quality of the reconstruction. The visual inspection compared results of the LoD2 and LoD3 reconstruction methods, and led to the conclusion that mis-segmentation of trees is the main cause of any visible errors.

6.2.2. Validation via Val3dity

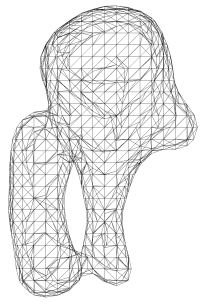
Geometry validation of the reconstructed canopies and trunks was performed using the val3dity tool. The tool checks for geometric errors in 3D models, such as non-manifold edges, self-intersections, and other topological issues. The results of the `val3dity` analysis indicated that some geometry carried errors related to consecutive points and a wrong shell orientation. The reversed normals was easily fixed by reversing the order of the vertices for each of the faces. The default setting for snap tolerance also uses a rather high number that, after setting it to a lower value, fixed most errors. The majority of

the reconstructed trees were free from significant geometric errors, confirming the robustness of the reconstruction process.

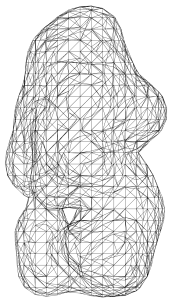
6.2.3. Alpha Wrapping via CGAL

In order to make sure the produced meshes are watertight, valid, and free of issues related to non-manifold edges, the CGAL alpha wrapping algorithm was applied to the point clusters instead of the Marching Cubes algorithm. The alpha wrapping algorithm is a well-established method for generating watertight meshes from point clouds or surface meshes. It works by creating an alpha shape around the input points, which effectively "wraps" the points in a smooth surface, filling in gaps and ensuring that the resulting mesh is manifold and free of holes. The application of the CGAL alpha wrapping algorithm resulted in watertight meshes that are suitable for further analysis and simulation.

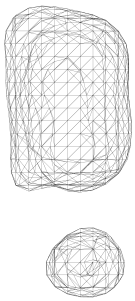
An additional benefit of alpha wrapping is that it can help to smooth out small imperfections in the reconstructed meshes, as well as simplify the resulting geometry which reduced the file size and complexity of the meshes. This is particularily useful for large-scale simulations where computational efficiency is important. Some individual trees are visualised in Figure 6.8. The resulting meshes were inspected and confirmed to be watertight and free of significant geometric errors, ensuring suitability for use in subsequent analyses and simulations.



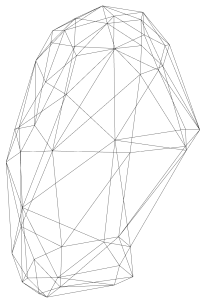
1132 vertices, 2264 faces



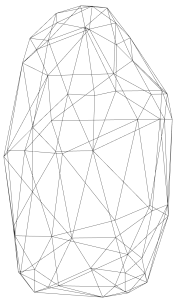
1480 vertices, 2960 faces



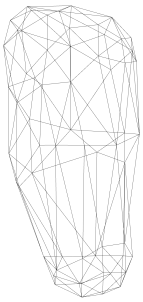
776 vertices, 1544 faces



52 vertices, 100 faces
Weight reduction: 85KB to 5KB



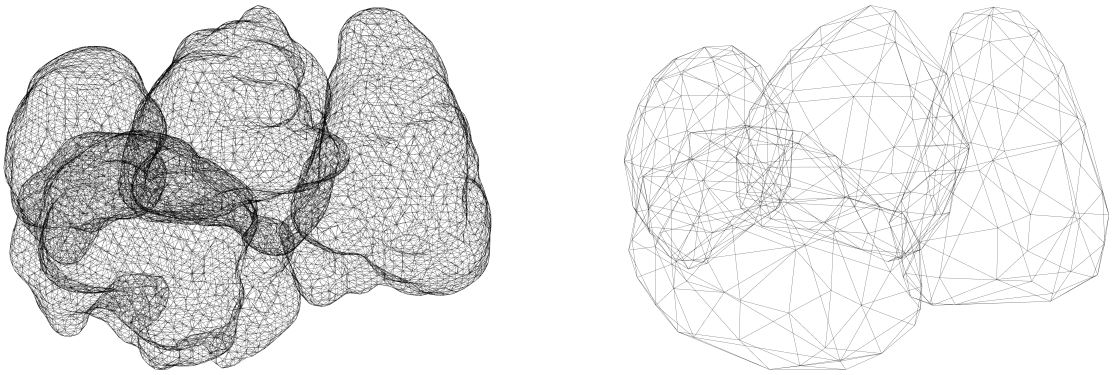
62 vertices, 120 faces
Weight reduction: 113KB to 5KB



61 vertices, 118 faces
Weight reduction: 58KB to 5KB

Figure 6.8: Examples of tree meshes before and after wrapping, both meshes are watertight and valid, but the reduction on vertices via wrapping creates more managable meshes

On these singular trees, alpha wrapping already has a large effect in file size. This is even more important in larger areas or areas with a lot of trees. Figure 6.9 shows multiple trees, where the alpha wrapped mesh significantly reduces the amount of faces and vertices, and therefore also the file size.



14 092 vertices, 28 168 faces 237 vertices, 458 faces

Figure 6.9: A group of trees before (left) and after (right) alpha wrapping with the amount of vertices indicated and file size reduction is 948KB to 15KB

It should be notes that no further research was done to the amount of over-simplification or area expansion that might result from this alpha wrapping.

6.3. Interactive User Interface

To develop an interactive user interface within the project timeframe, the Streamlit and Folium frameworks were utilized. This combination facilitated the rapid prototyping of a streamlined interface while accommodating various configurable parameters. A screenshot of the interface showing the available input parameters is shown in Figure 6.10. The different options are explained in the next sub-section. The interface can be accessed either using the CLI (`streamlit run app.py`) or using the deployed Streamlit webapp.

6.3.1. Input Options

Defining the Area of Interest

Users can define the Area of Interest (AOI) by either drawing a polygon directly on a Leaflet map (utilising an OpenStreetMap basemap) or by uploading an existing GeoJSON file. Drawn polygons are automatically exported as GeoJSON files to serve as input for the pipeline. Providing both options allows for rapid testing as well as the integration of predefined AOIs utilised by external stakeholders.

Tree Level of Detail

The interface permits users to select the desired LoD for the resulting tree geometries. LoD2 models are generated as ellipsoids derived from cluster parameters, ensuring minimal file sizes. Conversely, LoD3 models are generated using the marching cubes algorithm, which extracts the isosurface at a specified point density level. For LoD3, additional refinement options are available:

- **Merge touching crowns:** a simplification process that identifies proximal points post-segmentation and merges adjacent segments into a single unified crown.
- **Alpha wrap:** a geometric simplification and validation method applying the CGAL Alpha Wrap 3 algorithm to point clusters (instead of marching cubes) that guarantees valid geometries. Users can adjust the `offset_ratio` via a slider, where lower values yield simpler geometries.

Building Integration

Users can choose to process and download building geometries concurrently, yielding three distinct output files (crowns, trunk, and buildings). The desired LoD for buildings can also be specified. The pipeline retrieves CityJSON geometries via the 3DBAG API, spatially clips them to the defined AOI, and converts them into the OBJ file format.

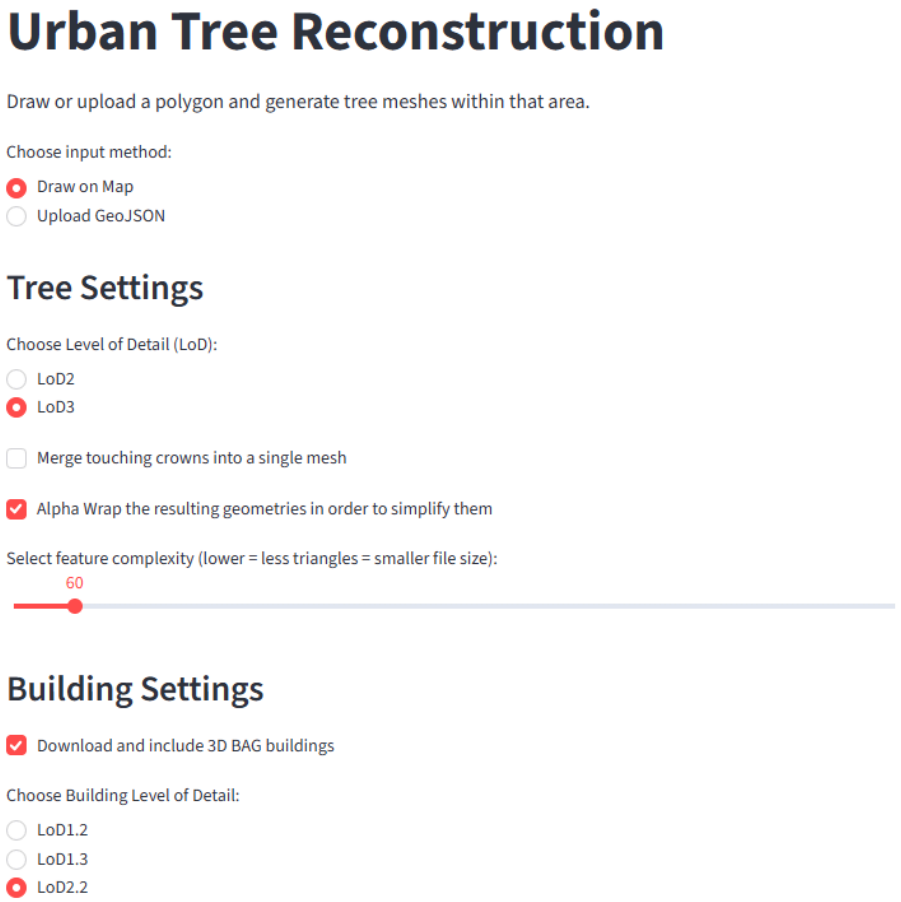


Figure 6.10: Screenshot of the interface showing the different options that can be selected as input for the reconstruction pipeline.

6.3.2. Pipeline Execution

Initiating the pipeline first triggers the data retrieval phase. The system identifies intersecting AHN tile indices and downloads the corresponding point clouds via the API. The algorithm queries both AHN5 and AHN6 datasets to account for partial national coverage. A caching mechanism is implemented to prevent redundant downloads of previously retrieved point clouds. The point clouds are subsequently clipped to the AOI and exported as one, merged LAZ file. A similar clipping process is applied to the 3DBAG data, after which a `config.json` file is generated based on the user's specified input parameters.

Following data acquisition, the core segmentation and reconstruction pipeline is executed. This phase incorporates the previously detailed methodological modifications. These processes are build upon prior implementations by Themis, including Statistical Outlier Removal (SOR) for vegetation cleaning, the filtering of point clusters within building footprints (adapted to utilise BAG data rather than OSM to mitigate timeout errors), and the generation of trunk geometries.

During execution, the interface provides progress updates through a loading indicator. Upon completion, a summary table displays the computational time required for each stage, alongside respective download buttons for the generated OBJ files (Figure 6.11).

Execution Time Summary

Pipeline Step	Duration (Seconds)
1. Download LiDAR Tiles	6.9900
2. Clip & Merge Pointcloud	25.0900
3. Clip & Merge 3D buildings	12.5900
4. Tree Segmentation	13.5300
5. Filter & Clean	3.9100
6. Reconstruct 3D Meshes	38.7000
Total Execution Time	100.8000

Download meshes

Download 3DBAG Geometries (.obj)

Download Meshed Trees (.obj)

Download Trunk Geometries (.obj)

Figure 6.11: Screenshot of the output shown when the entire pipeline has finished. The table shows the execution times for each stage of the process. Below it, download buttons are exposed where the user can download the resulting OBJ files.

Finally, a 3D preview of the resulting OBJ files is rendered at the bottom of the interface, utilising intuitive color coding to distinguish different feature classes, a screenshot of this can be seen in Figure 6.12. This visualisation is restricted to smaller study areas to ensure the file size remains below the 200MB rendering threshold; however, applying the alpha wrap simplification significantly mitigates this limitation.

3D Preview

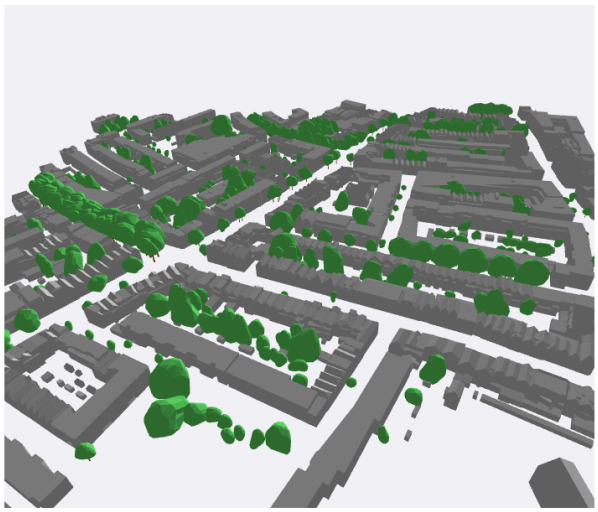


Figure 6.12: Screenshot of the 3D preview shown when the pipeline has finished. It visualises the different obj files that were created in intuitive coloring.

7. Discussion

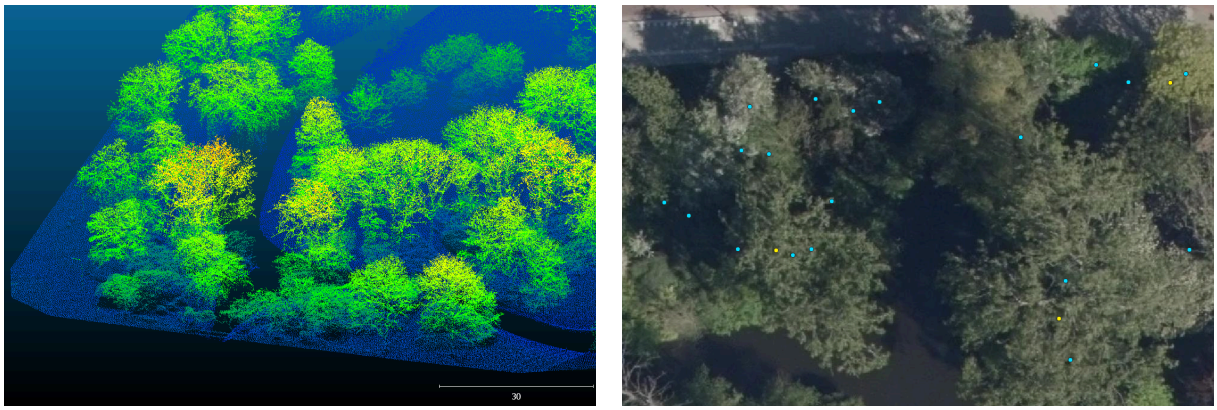
The following discussion is structured along three topics: Segmentation, Reconstruction and Interface.

7.1. Segmentation

Based on the results, we can conclude that the newly implemented segmentation algorithm is only a minimal improvement on the original algorithm. A small grow in most of the $F1$-score values can be seen. While the average recall score decreased, the precision score increased slightly more, resulting in an increase in the $F1$-score. This tells that the newly implemented algorithm focuses more on reducing the number of false positives, while resulting in an increase of false negatives. The choice of algorithm should therefore depend on the intended application of the pipeline, particularly whether the priority lies in achieving more accurate segmentations with fewer false positives or maximizing tree detection by minimizing false negatives.

7.1.1. Tree Inventories

The tree inventories were manually (supplementary) annotated using aerial imagery as reference. This causes mistakes in areas with overlapping crowns and bushes. From aerial imagery it is often hard to identify a bush from a tree and when the canopies are overlapping, it is hard to identify the individual trees. This can lead to a mismatch between the point cloud and the inventory. Another problem may be caused when trees are more recently planted than the point cloud or aerial image was captured. All these problems can lead to a mismatch between the point cloud and the inventory, which influences the evaluation. In Figure 7.1, an example is shown of bushes and overlapping canopies causing mistakes when annotating the inventory.



Point-cloud View Satellite Image View

Figure 7.1: Example location where bushes and overlapping canopies cause incorrect annotation of the inventory. The yellow dots represent detected trees, while the blue dots represent trees from the inventory.

7.1.2. Building Buffer

Because AHN5 and AHN6 include building walls as unclassified, they are not easily separted from vegetation points. Therefore, walls can be reconstructed as trees. To solve this issue, a BAG building footprint filter was implemented, initially with a buffer of 0.5m to ensure no trees on sidewalks were accidentally removed. However, this did not fix cases where overhangs or balconies cause points to be further outside of the building footprint. An example of this is visualised in Figure 7.2. By increasing the buffer distance to 2.0m, which is the minimal distance a tree has to be from a facade according to Dutch law, these “trees” were correctly identified as walls and removed from the results.

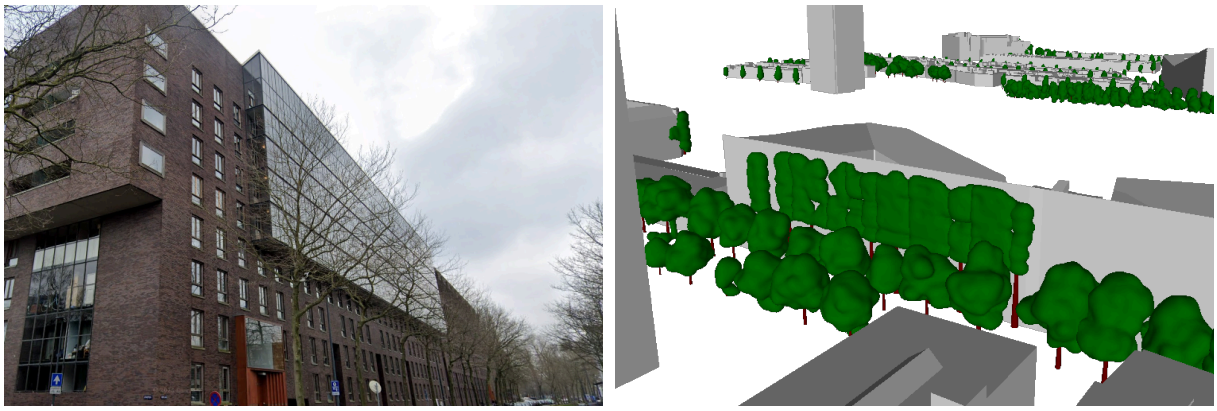


Figure 7.2: Google Street View image of a building with a overhang that is not in the BAG building footprint (left) and the reconstructed trees resulting from these points (right).

7.1.3. Pole-like Segments

The existing code-base already included a filtering of pole and wall-like segments, but when running the pipeline we discovered that this stage took quite a long time. The filtering included computing different tree cluster statistics like the crown diameter and the aspect. Because our research focusses on scalability, this large computational time needed for filtering was not desireable. As the new segmentation implementation already includes a PCA of the cluster, it was the obvious choice to use that for filtering as well. The computational time dropped significantly and visual inspection tells us that it still works well.

7.1.4. Unpredictable Elements

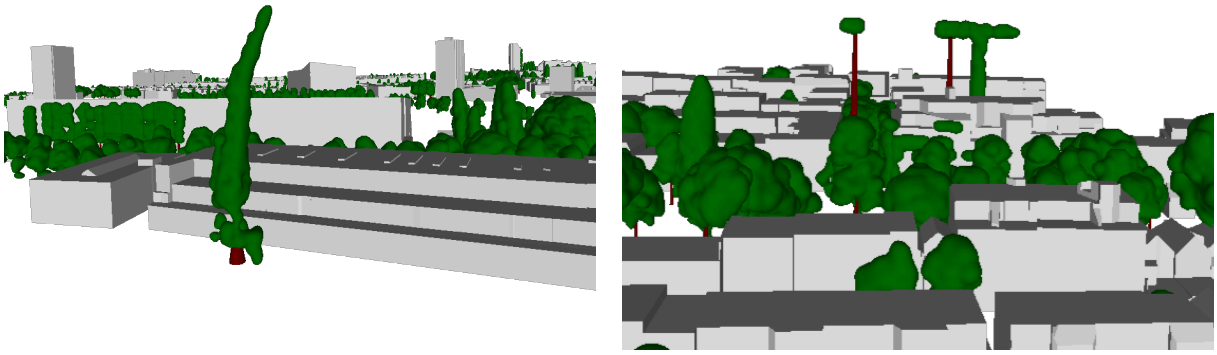
When capturing the AHN point cloud, the LiDAR sensor also captures other elements in the environment, such as cranes. These elements are not part of the tree inventory and can lead to misclassification. In Figure 7.3, two cranes are present in the point cloud and are classified as 'other'. However, due to their density, they are still identified as trees. This misclassification affects the accuracy of tree detection. A filter that was applied to solve part of this problem, is a maximum height of the lowest point in the segment. This filter removes segments that are hanging too high to be trees, preventing very high trunk to be reconstructed. The filter is not perfect and can still lead to misclassification of cranes as trees.



Harbour Crane in Amsterdam



Building Crane in Rotterdam



Reconstructed Crane in Amsterdam Reconstructed Crane in Rotterdam

Figure 7.3: Cranes present in the point cloud which are also classified as 'other'. Because the density is relatively high, they are still identified as trees.

7.1.5. Polygon splitting

One issue that may explain the lower $F1$-scores when only the variable window is added, is the fact that the point cloud is splitted exactly along the lines of the polygon split. This can result in single trees being present in multiple sub-polygons, which are not merged together. In Figure 7.4 an example case of this is visualised. The horizontal and vertical split lines are visible where the clusters are divided along a perfectly straight line. There are multiple locations when a tree peak is detected very close to and on both sides of this split line. More trees are now detected than there are in reality, causing the $F1$-score to be lower.

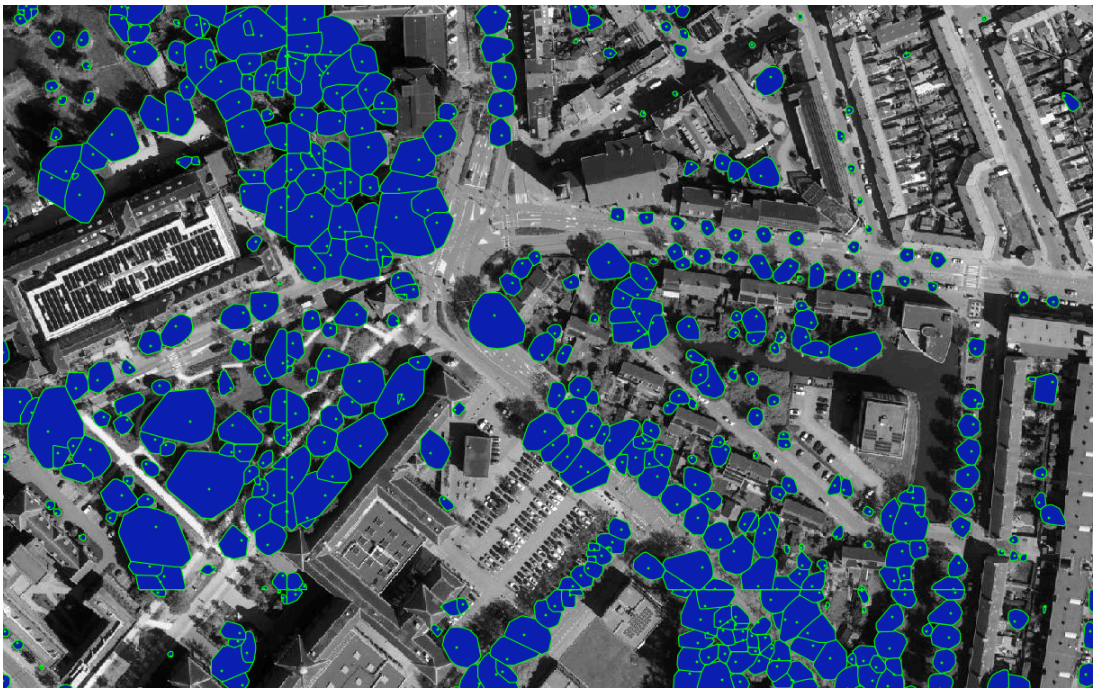


Figure 7.4: An input polygon that was splitted in four parts, resulting in splitted segments along these lines. The green dots represent the tree peaks and the blue areas the convex hull of the points beloning to it.

7.2. Reconstruction

The existing reconstruction pipeline already worked very well. The main bottleneck identified was the file size, especially when reconstructing trees on larger areas. Therefore, alpha wrap was implemented

which has a large effect on file size reduction. It does also smooth and simplify the resulting mesh, usually making it a bit bigger. This might be a problem for CFD simulations, but due to time constraints we were not able to look further into this.

7.2.1. Bushes

Vegetation includes various plant types, such as bushes. These types are difficult to detect and filter out and will therefore be part in the reconstructed mesh. This results in tree crowns reaching to the ground, or tree crowns with multiple components. While this is not the intended results, the reconstructed mesh can be kept, as bushes have a similar effect on wind flow as tree crowns. In Figure 7.5 and Figure 7.6, the reconstruction of a dense forest area can be seen, where the bushes let the resulting crowns reach the ground and have multiple components. While they do not directly resemble greenery, they provide a good approximation for areas needed for approximations as tree canopies.

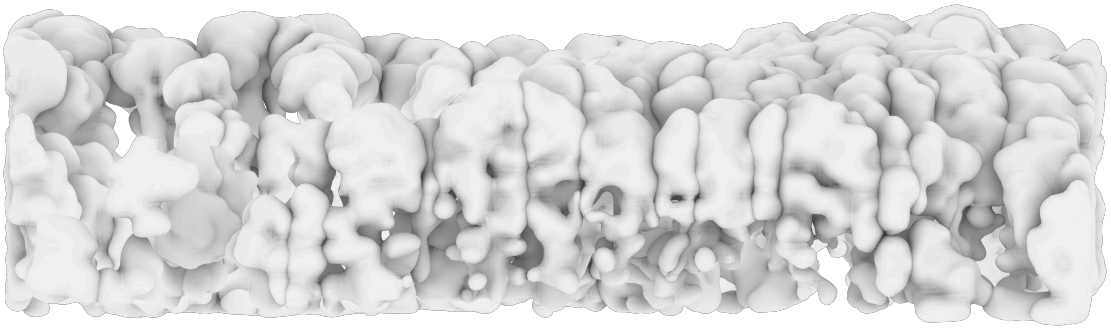


Figure 7.5: Reconstructed mesh of a dense forest area using the Marching Cubes algorithm.

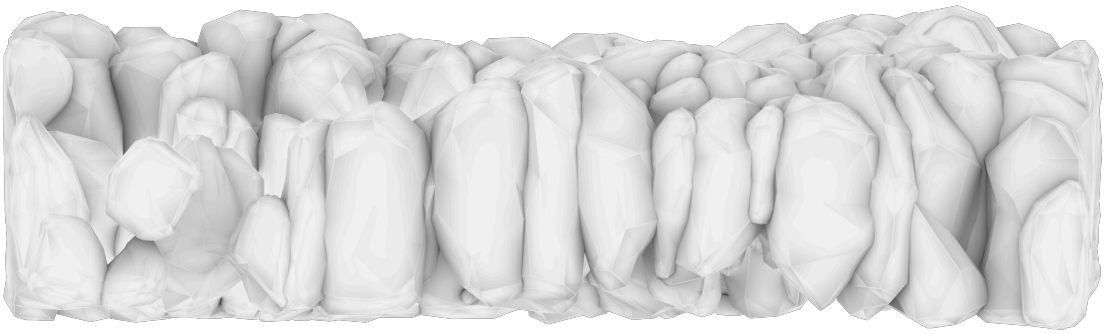


Figure 7.6: Wrapped reconstruction of a dense forest area.

7.3. Interface

The provision of such a tool has been tested and praised by professionals working directly with CFD simulations within the built environment, and directly adresses pain-points related to greenery retrieval.While it can produce geometry out of misclassified or missegmented elements such as cranes in ports, it is a good effort at providing simulation ready geometry previously unexplored in a scalable way.

Although the interface has multiple functions, there are different aspects on which it can still be improved. Mainly it should be made to run on a different server, possible a TU Delft server. Right now it is using the Streamlit server, which can be a bottleneck when trying to run it on big areas as the server only has a limited amount of space available.

Another improvement that can be made is the inclusion of the ground. As this is a necessary input in CFD simulations, this would make the interface more complete.

Lastly, as a goal of this project is to be combined with the `City4CFD` framework, the building generation should be done using this codebase, instead of readily downloading them from 3D BAG.

8. Conclusion

This research was set out to answer the main question; *How does the choice of airborne-LiDAR segmentation and reconstruction algorithms affect the scalability, geometric quality, and CFD compatibility of urban tree models?*. Conclusively, the choice of airborne-LiDAR segmentation and reconstruction algorithms primarily dictates storage requirements, pipeline architecture, and ground-truth reliance, while consistently delivering valid geometries suitable for microclimate simulations.

Scalability
Processing speed is not significantly bottlenecked by the reconstruction's Level of Detail, but storage scalability is algorithm-dependent, with Alpha-Wrap yielding much smaller, more manageable file sizes than Marching Cubes. While the segmentation algorithm scales effectively to unseen cities, true computational scalability for the pipeline requires dedicated high-performance server hosting to handle dense point clouds without memory limitations.

Geometric quality
The overall accuracy of the isolated trees is dependent on the quality of the municipal inventories, as imperfect reference data can insinuate under- or oversegmentation, skewing the parameters. However, the reconstruction algorithms themselves successfully produce complete, watertight meshes with only minor, easily correctable artifacts.

CFD compatability
The generated 3D tree geometries are robust and CFD-ready. Regardless of the specific reconstruction algorithm used (Alpha-Wrap or Marching Cubes), the resulting meshes maintain topological validity, ensuring that minor geometric artifacts do not impede the stability or performance of aerodynamic microclimate simulations.

8.1. Subquestion 1

How do individual tree segmentation methods affect the accuracy, scalability, and robustness of isolating single trees from airborne LiDAR data?

The proposed refinement shows a marginal improvement in the $F1$-score across the 17 evaluation areas. While precision increased, the associated recall loss led to lower balanced detection performance, indicating a tendency toward undersegmentation where fewer trees are detected than are physically present. However, applying the algorithm to unseen urban areas (Alkmaar, The Hague, Breda, and Eindhoven) generally resulted in higher $F1$-scores than those of the initial evaluation areas. Because these unseen areas relied solely on standard municipal tree inventories, this discrepancy suggests that lower performance metrics may be partially attributed to incorrectly or inconsistently annotated reference data. This hypothesis is further supported by the performance metrics achieved in Amsterdam, where a manually verified, complete inventory was utilised. Future research should prioritise the curation of accurate, standardised tree inventories for training and validation, specifically focussing on the exclusion of small trees and understory bushes to prevent ambiguous ground-truth matching.

8.2. Subquestion 2

How do different reconstruction algorithms compare in computational efficiency (processing speed, memory, scalability) and geometric quality (watertightness, completeness, complexity) when generating LoD3 tree geometries from city-scale airborne LiDAR data?

Regarding computational efficiency, processing times do not significantly fluctuate when changing the Level of Detail of the resulting mesh. Across moderately sized test areas, the reconstruction time differs by only a few seconds, indicating that LoD selection should be driven by storage capacity

and application requirements rather than time constraints. Notably, the Marching Cubes algorithms generates meshes with considerably larger file sizes compared to both the CGAL Alpha wrap algorithm and the LoD2 reconstructions. Geometrically, the resulting models have proven to be robust and CFD-ready, yielding valid, watertight meshes. Although some minor artifacts may occur, this would not decrease the performance of a CFD simulation. Furthermore, such artifacts can easily be corrected manually when necessary. Future research should explore alternative methodologies for LoD2 generation to further optimize geometric lightweighting, alongside the investigation of trunk reconstruction algorithms that clip the trunk at ground level.

8.3. Subquestion 3

How can the tree reconstruction pipeline be integrated into an interactive user interface to facilitate the accessible, on-demand acquisition of 3D tree geometries?

The tree reconstruction pipeline was successfully integrated into an interactive web application, enabling accessible, on-demand geometry generation. However, the current deployment relies on either local execution or Streamlit hosting, both of which introduce computational bottlenecks and memory limitations when processing dense AHN point clouds and generating complex meshes. To enhance scalability, professional utility, and accessibility for external research groups or industry partners, the application requires a more robust hosting architecture. Future iterations should focus on migrating the pipeline to a dedicated, high-performance computing environment, such as the TU Delft institutional servers. This transition would not only alleviate current computational constraints but also provide a stable, centralized platform for broader multidisciplinary application.

References

Alting, N. P. (2025, October). *From point Clouds to Porous Crowns: A scalable approach for CFD-Ready urban tree reconstruction.*

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, *4*(4), 2842–2889. https://doi.org/10.3390/ijgi4042842

Buccolieri, R., Santiago, J.-L., Rivas, E., & Sáanchez, B. (2018). Reprint of: Review on urban tree modelling in CFD simulations: Aerodynamic, deposition and thermal effects. *Urban Forestry & Urban Greening*, *37*, 56–64. https://doi.org/10.1016/j.ufug.2018.07.004

de Groot, G. J. (2020,). *Automatic construction of 3D tree models in multiple levels of detail from airborne LiDAR data*. Delft University of Technology.

Fu, R., Pađen, I., & García-Sánchez, C. (2024). Should we care about the level of detail in trees when running urban microscale simulations?. *Sustainable Cities and Society*, *101*, 105143. https://doi.org/10.1016/j.scs.2023.105143

Liu, Y., Chen, D., Fu, S., Mathiopoulos, P. T., Sui, M., Na, J., & Peethambaran, J. (2024). Segmentation of individual tree points by combining Marker-Controlled watershed segmentation and spectral clustering optimization. *Remote Sensing*, *16*(4), 610. https://doi.org/10.3390/rs16040610

Ma, K., Chen, Z., Fu, L., Tian, W., Jiang, F., Yi, J., Du, Z., & Sun, H. (2022). Performance and Sensitivity of Individual Tree Segmentation Methods for UAV-LiDAR in Multiple Forest Types. *Remote Sensing*, *14*(2), 298. https://doi.org/10.3390/rs14020298

Maison, A., Flageul, C., Carissimo, B., Wang, Y., Tuzet, A., & Sartelet, K. (2022). Parameterizing the aerodynamic effect of trees in street canyons for the street network model MUNICH using the CFD model Code_Saturne. *Atmospheric Chemistry and Physics*, *22*(14), 9369–9388. https://doi.org/10.5194/acp-22-9369-2022

Mu, Y., Zhou, G., & Wang, H. (2020). Canopy LiDAR Point Cloud Data K-means Clustering Watershed Segmentation Method. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *VI-3/W1-2020*, 67–73. https://doi.org/10.5194/isprs-annals-vi-3-w1-2020-67-2020

Peters, R., Dukai, B., Vitalis, S., Van Liempt, J., & Stoter, J. (2022). Automated 3D reconstruction of LOD2 and LOD1 models for all 10 million buildings of the Netherlands. *Photogrammetric Engineering & Remote Sensing*, *88*(3), 165–170. https://doi.org/10.14358/pers.21-00032r2

van Druenen, T. (2025, July). *Revisiting wind climate statistics for pedestrian wind comfort: Updating the NEN 8100 standard.* https://research.tue.nl/en/publications/revisiting-wind-climate-statistics-for-pedestrian-wind-comfort-up

Yang, J., Kang, Z., Cheng, S., Yang, Z., & Akwensi, P. H. (2020). An individual tree segmentation method based on watershed algorithm and Three-Dimensional spatial distribution analysis from airborne LiDAR point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *13*, 1055–1067. https://doi.org/10.1109/jstars.2020.2979369

A AI statement

Large Language Models were used as support during coding and writing. It helped improve coding efficiency, formulating sentences and checking spelling mistakes. AI was never used to make decisions or to deliver new information without sourcing.

B Contributors

The team for this synthesis project consists of three students of the MSc Geomatics program at the TU Delft under supervision of Clara García-Sánchez and Themistoklis Vargiemezis - both working in the 3D Geoinformation group at the Faculty of Architecture and the Built Environment. Besides, the project was done in consultation with Haskoning's Advanced Technology & Research Group for Industry & Buildings, represented by Frank van Gool and Kjeld Teunissen. This section will introduce each team member and describe their role in the project.

B.1 Team members



Anna Scherrenburg

Nationality: Dutch

Background: BSc Architecture, Urbanism and Building Sciences (Delft University of Technology)



Henryk Gujda

Nationality: Polish

Background: MSc Architecture, Urbanism and Building Sciences (Delft University of Technology)



Simon Deuten

Nationality: Dutch

Background: BSc Applied Mathematics (Delft University of Technology)

B.2 Roles

To work effectively during this project, we divided the primary roles and responsibilities between the team members; Project Manager, Technical Lead, and Research Lead. This division was chosen because it addresses the three primary demands of this project. First, the time limit requires strict scope management and a unified final report, making a project manager important to prevent time crunches. Second, bridging the gap between raw point clouds and CFD solvers involves complex software pipelines; the technical lead ensures this architecture remains robust and reproducible. Finally, the research lead guarantees that our computational efforts do not just produce data, but actually answer the core scientific question.

It is important to note that this division is designed to establish primary responsibilities between the team members into three complementary roles: Project Manager, Technical Lead, and Research Lead. Rather than strictly separating the work into individual parts, the roles provide clear ownership over the project's organisational, technical, and scientific aspects. The team collaborated closely throughout all phases of the project.

B.2.1 Project manager: Anna Scherrenburg

As project manager, Anna was responsible for the team's organisation, timeline, and coordination. Throughout the project, she ensured that the research stayed within scope and that the final report became a cohesive, scientific paper. This was achieved by setting internal deadlines and tracking progress in the Gantt chart, which kept the team on schedule. She created the skeleton and structure of the final report, set deadlines for the draft sections, and filled in specific tasks for each team member. Besides coordinating, Anna contributed directly to the technical work. She took part in data collection and annotation, led the design and implementation of the interface, and implemented the segmentation methods. In the report, she wrote the introduction, problem statement, theory, methodology, interface results and discussion, and took responsibility for the consistency of the report as a whole.

B.2.2 Technical lead: Simon Deuten

The primary concern of the technical lead was making sure that the computational pipeline was robust, reproducible, and bug-free. Simon led the structuring of the code so that it could be reused and extended, and handled the storage of the large datasets. When errors occurred, he led the debugging efforts. He took over the further development of the technical pipeline and was responsible for tuning the segmentation parameters, splitting polygons based on NDVI and benchmarking the segmentation results. In the report, Simon wrote the data chapter and the segmentation results, which documented the technical setup and the algorithm implementations clearly and kept them technically viable.

B.2.3 Research lead: Henryk Gujda

Henryk led the preliminary research and the analysis of results. By establishing the existing baseline and determining which metrics needed to be extracted, he turned the raw outputs of the pipeline into actionable insights. He coordinated the comparative analysis across the different tree representations, their implementation methods and resulting levels of detail, and assessed the scalability of the selected approach. He implemented the first validation scripts and worked on the reconstruction of the tree geometry, evaluating the reconstruction methods and benchmarking them against the baseline. He drew the main conclusions from these results. He also led the presentation, including the design and production of figures, maps and animations. Throughout the project he pushed the team to improve on its results and, through discussion with each member, helped steer the work towards their strengths.

B.3 Client

Haskoning is an independent international engineering and consultancy firm headquartered in Amersfoort, the Netherlands, active in the built and natural environment across more than 25 countries. Its work spans infrastructure, water management, energy, mobility, and urban development, with sustainable and climate-resilient urban design as a recurring focus across these markets.
Within this project, the firm is represented by Frank van Gool and Kjeld Teunissen. In initial meetings, Haskoning did not impose specific technical requirements or constraints on the research direction, instead expressing openness to the team's own scoping of the problem and sharing their current workflows. Their interest lies primarily in the practical applicability of the resulting method: a scalable, automated way of incorporating tree geometry into urban CFD workflows would directly support the kind of climate-resilient urban design and microclimate assessment work the firm carries out for clients, directly informing the interface part of the project.

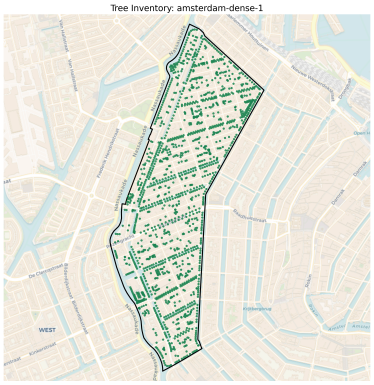
B.4 Task Distribution Table

Student Name	Responsibilities in the report	Responsibilities during the project	Role performed
Anna Scherrenburg	Introduction, Problem Statement, Theory, Methodology, Interface Results, Discussion	Reporting, Interface Design, Segmentation Methods Implementation	Project manager
Henryk Gujda	Abstract, Reconstruction Results, Conclusion	Reconstruction Improvement, Evaluation, Presentation & Figure Design	Research & Data Lead
Simon Deuten	Data, Segmentation Results	Segmentation Parameter Tuning, NDVI polygon splitting and Benchmarking	Technical Lead

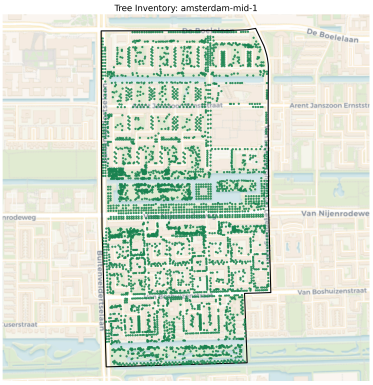
Table B.1: Task distribution between the team members

C Tree inventories for each area

C.1 Amsterdam



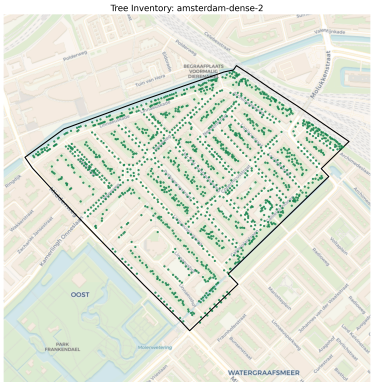
Amsterdam Dense 1



Amsterdam Mid 1



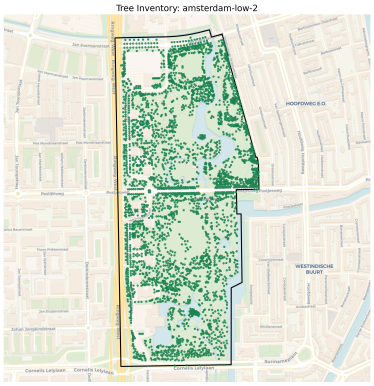
Amsterdam Low 1



Amsterdam Dense 2



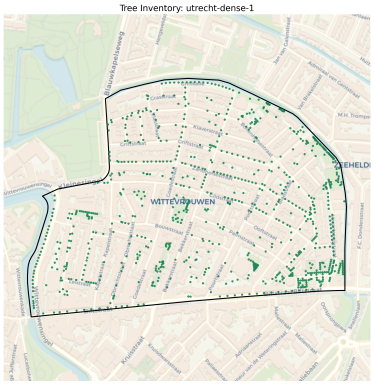
Amsterdam Mid 2



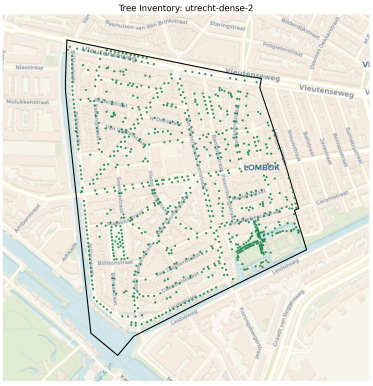
Amsterdam Low 2

Figure C.1: Tree inventory for the different selected areas in Amsterdam being the municipal tree inventory combined with manually added tree locations based on aerial images

C.2 Utrecht



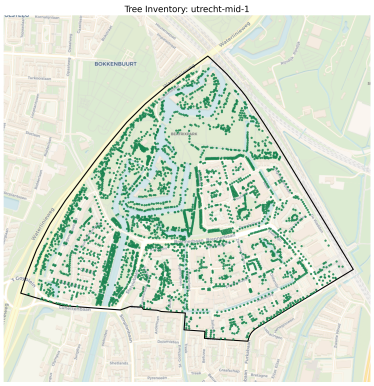
Utrecht Dense 1



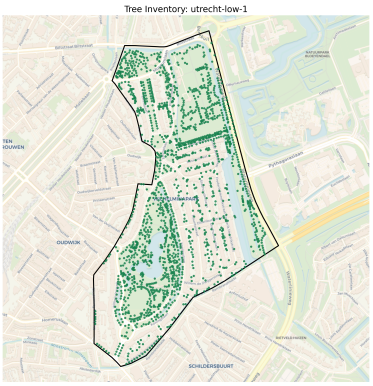
Utrecht Dense 2



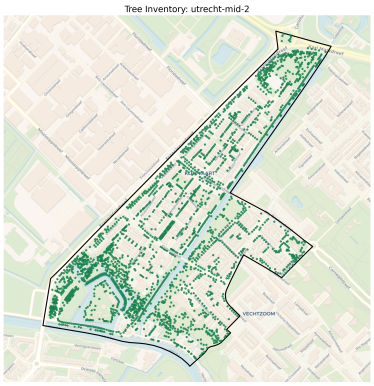
Utrecht Low 1



Utrecht Mid 1



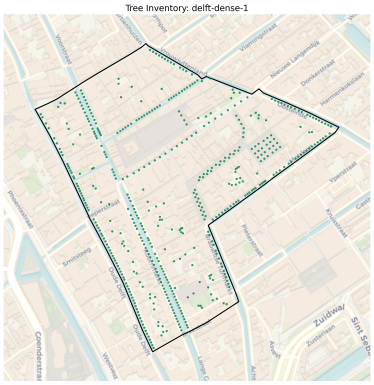
Utrecht Mid 3



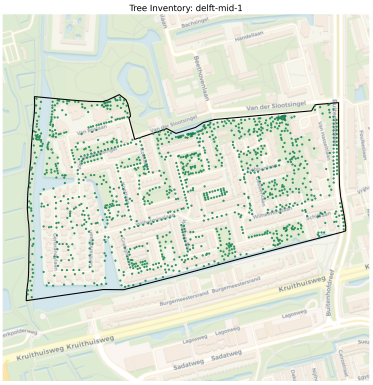
Utrecht Mid 2

Figure C.2: Tree inventory for the different selected areas in Utrecht being the municipal tree inventory combined with manually added tree locations based on aerial images

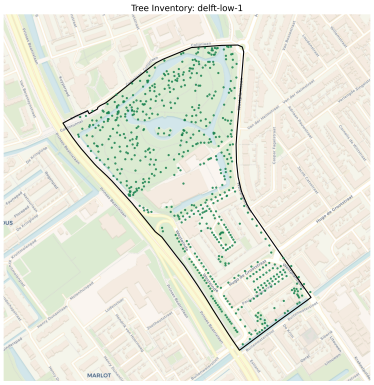
C.3 Delft



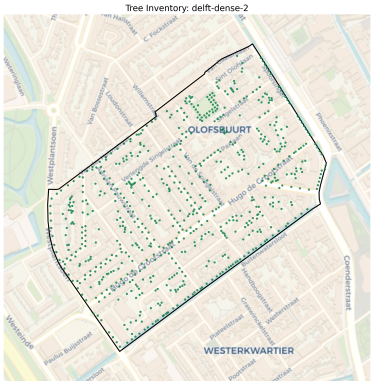
Delft Dense 1



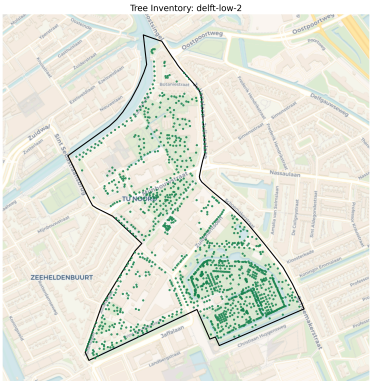
Delft Mid 1



Delft Low 1



Delft Dense 2



Delft Mid 2

Figure C.3: Tree inventory for the different selected areas in Delft being the municipal tree inventory combined with manually added tree locations based on aerial images

D Sensitivity analysis results

