TDK-dolgozat

Anett Fekete

Change Detection of Vegetation Based on LiDAR Data

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPARTMENT OF SOFTWARE TECHNOLOGY AND METHODOLOGY

Author:

Anett Fekete

Computer Science MSc

V. grade

Supervisor:

Máté Cserép

Assistant Lecturer

Budapest, 2019

Contents

Chapter 1

Introduction

The change detection of vegetation is a very important field whose results are used in various practical applications: experts utilize these results during city planning because preserving vegetation in urban environment is getting more and more important. Another corresponding application is in the protection of the environment which has grown to a very serious issue from the viewpoint of climate change.

LiDAR (Light Detection and Ranging) is a fairly new concept among geoinformatical scanning methods. It was first used shortly after the invention of laser, in the first half of the 1960s. LiDAR is a possible alternative for raster scanning although not as widespread due to the high cost of laser scanning. Apart from the ones listed above, point clouds produced by laser scanning are utilized in various fields of science from meteorology to archaeology. There are plenty of options for the application of laser scanning: airborne, terrestrial and mobile scanning are all in practice.

Change detection and landcover classification are well-researched topics for which several manual or semi-automatic methods exist [11, 26]. These algorithms are usually applicable for only small areas. The goal of this research is to define an automatic method of comparing the state of vegetation and determining the changes between individual trees between two epochs which is also applicable to large urban and suburban areas.

During my research, I utilized and combined already existing methods and created a new automatic clustering algorithm to cover the trees in a scanned area. The main purpose of this method is to quantify the changes of trees in an area between scanning epochs. This includes changes in canopy volume, tree height and tree presence. A prototype implementation was also made. This implementation was tested on the *AHN* dataset which covers

The Netherlands.

The rest of the thesis is organized as follows. Chapter 2 gives an overview of the applications of LiDAR and how existing change detection algorithms work. Chapter 3 proposes a description of the test dataset and my methodology for change detection of vegetation. In Chapter 4 I describe the implementation circumstances. Chapter 5 gives an overview of the results of the research. Finally, I conclude and summarize my work in Chapter 6.

Chapter 2

Related work

Light Detection and Ranging (LiDAR) is an active remote sensing system that is used to determine the distance between the target surface and the sensor. This is done by the illumination of the target with pulsed laser light and measuring the reflected pulses. LiDAR was first used in the early 1960s and became generally known after the Apollo 15 mission in 1971 when the method was used to map the surface of the Moon. [15] LiDAR is also known as laser scanning.

LiDAR uses multiple different types of light for illumination, ultraviolet, near infrared and visible. The illuminated targets can be of a wide range of materials, like rocks, rain, clouds, non-metallic fabrics or even chemical compounds. This allows LiDAR to be applied in a broad range of disciplines, such as geodesy, archeology, geography or forestry. The method is mainly used to create high-resolution maps for its applications.

2.1 Data acquisition

The two main types of application are airborne and terrestrial. Multiple viewpoints are taken into account when deciding which type is to be applied, e.g. the purpose of the data, the size of the target, the cost of detection etc.

2.1.1 Airborne LiDAR

As the name suggests, airborne laser scanning (ALS) is when a sensor creates a 3D point cloud model of the target area while attached to a flying aircraft. The scanning is mostly perpendicular to the direction of the flight. By using airborne LiDAR a model of the terrain layout can also be constructed, since vegetation can be easily distinguished and filtered from the data set so trees and other tall plants do not conceal other objects underneath them. This is also useful for purposely filter any object other than vegetation which is an important segment of this study.

2.1.2 Terrestrial LiDAR

When the sensor is placed on the Earth's surface, it's called terrestrial laser scanning. The two main types are stationary (TLS) and mobile (MLS). Stationary terrestrial laser scanned data is collected from a fixed point, hence the point clouds created through scanning can be matched with digital 2D images taken at the same location. Thereby a more realistic 3D model can be made than with other technologies. Terrestrial LiDAR is most common as a survey method, when the exact features (location, height, width) are needed to be measured of the scanned area, e.g. at construction sites.

When a scanner is attached to a moving land vehicle (e.g. a car or a train), it is named mobile terrestrial laser scanning. By utilizing two or more sensors, the method is guaranteed to take all measurements needed to create an adequate 3D model and there is no need to collect individual measurements. By extending such a system with a navigational unit consisting of an integrated global navigation satellite system (GNSS) and an inertial navigation system (INS), a mobile mapping systems (MMS) is formed, capable of recording dense point clouds with high positional accuracy while the sensors are moving. It is also common to further extend an MMS with auxiliary high-resolution raster cameras which create raster imagery synchronously with LiDAR scanning thus adding RGB metadata to the point cloud.

2.1.3 Further methods

Beside classical geospatial applications, the LiDAR technology continuously find its way into further industrial segments, like the operation of autonomous vehicles, e.g. self-driving cars. While professional high-density LiDAR sensors are still quite expensive nowadays, multiple experiments have been carried out on how to manufacture low-cost, but adequate sensors and integrate them with gaming devices, and most recently augmented, virtual and mixed reality applications. The commonly used devices nowadays are Kinect range cameras, smart phones (the most significant try was Project Tango [10, 9] by Google which was terminated in 2017).

2.2 Data formats

Geospatial applications might use raw or preprocessed point clouds. For example, a preprocessing method could be the elimination of false returns. The most common formats of this are LAS, and its compressed version LAZ.

The irregularity of point clouds increase the algorithmic – and thus the computational – complexity of analyzing and comparing point clouds. To address this issue a raster grid can be interpolated based on the original point cloud, named a digital elevation model (DEM). The vertices contain the accumulated height values, calculated by e.g. the inverse distance weighting (IDW) algorithm [21, 14], typically represented in GeoTiff format. This model also reduces the number of data points to analyze, which can be controlled by the grid size of the DEM. For many applications the high density of the point clouds is not necessarily required, yet a more inaccurate, but more easily manageable model is favorable. Digital elevation model types can be further categorized by their contents, as depicted in Fig. 2.1 and described as follows:

Digital elevation model (DEM) is a generally used phrase for models constructed from point clouds. It is a regular grid that contains altitude values in its grid points. DEM is a popular data format which facilitates the utilization of raw point clouds.

Digital surface model (DSM) is a format that contains every object (buildings, vegetation, powerlines etc.) that was captured by LiDAR in the point cloud.

Digital terrain model (DTM) is a model of the bare surface of the Earth. Unlike DSM, objects are eliminated from the model. When considering vegetation, DTM typically contains surface height to count returns from the ground. In the case of buildings, the height values are either interpolated from the neighbouring values or get an extremal value, a so called *nodata* denotation.

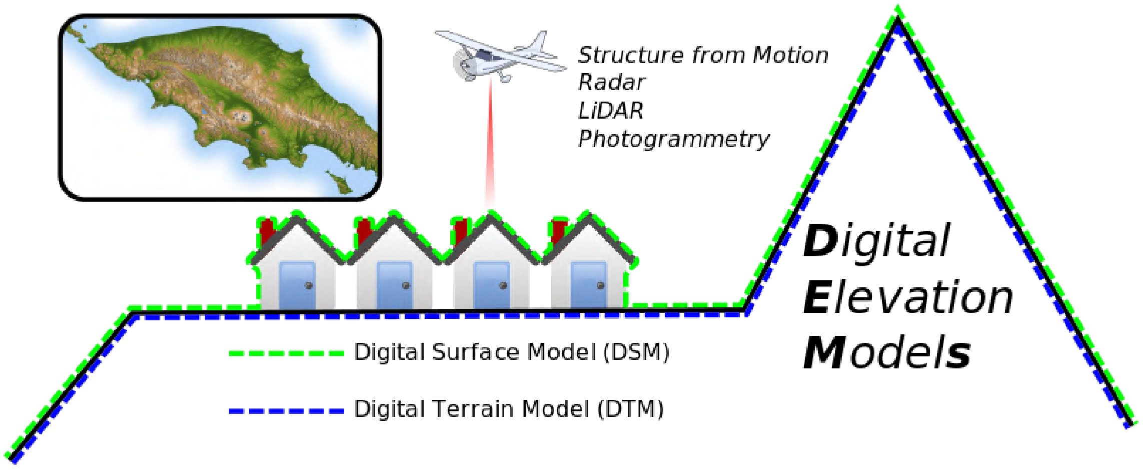

Figure 2.1: The difference between DEM, DSM és DTM.
Source: `https://www.wikipedia.org/`

2.3 Classification of land coverage

In geospatial terms, classification is the arrangement of objects into smaller groups (classes) based on their attributes and relationships. The main classes for the topic of this study are urban objects and vegetation. The latter being important in terms of this paper, this means distinguishing living and lifeless objects, such as rocks and bodies of water. Living beings, plants can be ranked by certain qualities, e.g. type, height, or in a more specific case, canopy density of trees, etc.

Rasterized satellite images and particularly the multispectral imagery are still the main data source since they are cheap and easy to obtain - compared to LiDAR. Several satellites orbits the Earth for the continuous acquisition of satellite imagery, the most important ones are the Landsat and SPOT satellites. A rasterized image contains a grid that encodes the geographic data in the pixel values and locations. A multispectral image contains image data in specific wavelength ranges across the electromagnetic spectrum, including visible, infrared and ultra-violet light. These images do not make feature type identification easy, that is why the collected data needs to be classified at first, followed by various data enhancement techniques.

LiDAR on the other hand, although useful in classification, is expensive and limited, meaning there are no comprehensive LiDAR records of the Earth, yet it is still a promising research topic, as it is getting cheaper and more widespread. One of the greatest advantages of LiDAR is that instead of examining only images, it offers the opportunity to compare 3D point clouds which not only makes classification, but filtering temporary objects like people and vehicles, and change detection way easier than 2D imagery. On the other hand, the comparison of point clouds is definitely more complex algorithmically compared to raster images since the points are not likely to be located in the same spatial point when examining multiple point clouds from different epochs. In fact, significant differences may also occur in the quality and density of the same point cloud which might make processing even more difficult.

In their paper, Antonarakis et al. [1] describe two models to distinguish natural and planted forestry, one of which includes ground hits while the other one does not. The former relies on the use of bimodal distribution skewness and kurtosis models. A skewness value and a kurtosis value was chosen to define which pixels belonged to a natural forest, and which ones belonged to a planted one. This method also proved to be useful in determining if the tree was younger or mature. The other referred method was created using kurtosis and skewness layers that were not influenced by the ground or the ground vegetation, thus making the corresponding values different. A percentage of canopy hits model (PCM) was also taken into account for the sake of a more accurate classification.

Bellakaout et al. [2] mention a classification method that identifies and utilizes different contour types by which objects can be classified. The paper describes four different object classes that contain terrestrial objects identified by contours: superior contour, inferior contour, uniform surface and non-uniform surface. These classes allow the extraction of soil, vegetation, buildings and roads.

Song et al. [22] describe a study whose aim was to evaluate the use of LiDAR data in land cover classification. The key of this method is to classify objects based on the intensity of reflection. The point clouds were converted into grid form by the IDW and the Kriging interpolation methods. The conversion created some noise in the dataset that had to be filtered. Afterwards, the acquired intensity data was ready to be divided into four classes: grass, tree, asphalt road, and house roof.

2.4 Change detection in point clouds

First and foremost, the goal of change detection is to identify changes in the examined area that occurred between the epochs when the datasets were formed. As mentioned above, this is not always an easy task; it is seldom possible to repeat individual point measurements.

2.4.1 Binary and quantifiable changes

There are two distinct types of change detection: binary and quantifiable change. The former is a simple approach, looking for only the existence of a change in the scene. Its result is mostly a binary map where no change is indicated by 0, change is indicated by 1. Quantifiable change, on the other hand, is in search of a more complex answer to non-binary questions that are targeting the ways the change went down. Vosselman et. al [24] were the first to separate the two types, naming the binary type as the actual change detection and calling the other one deformation analysis.

Binary methods

- $2\frac{1}{2}$ dimensional visibility maps: The dimension of the point cloud can be reduced to $2\frac{1}{2}$D if the observation happens from a fixed scanner position. This way, objects can appear, disappear, and move in the point cloud, revealing contingent changes in the scene. As Vosselman et al. [24] describe citing two examples, a spherical coordinate system is used to determine if there are any changes in the point clouds by transforming the dataset that was captured later (and perhaps from a different but fixed stand-point) into the coordinate system. Afterwards, points from the reference cloud are looked for in the new cloud; the point can be present and represent or not represent an object at its location or can be invisible because of occlusion.

- Direct DEM comparison: This method applies a simple subtraction to digital elevation models, followed by corrective movements of the results to eliminate errors caused by misregistration. Two example usages were described by Vosselman et al. [24]. It can be used in both urban areas and nature. In addition, it is worth noting that direct DEM comparison is a decent quantifiable method as well, since exact height and volume changes can be measured by comparing DEMs.

Quantifiable methods

- Pointwise deformation analysis: Most approaches offer methods for change detection that are based on the DEMs constructed from point clouds but there are also existing methods that examine raw point clouds in favor of achieving more accurate results. Butkiewicz et al. [3] describe a change detection method in their paper that find deformations in urban environment, both vegetation and buildings over time using raw LiDAR point clouds. They calculate bounds for what could be scanning error or geologic variation and compare the distance of points in different scans. This method is capable of detecting changes in individual and grouped objects as well.

- Object-oriented deformation analysis: Mentioned by Lindenbergh et al. [13], this method takes advantage of the pattern that man-made objects are mostly constructed by geometric shapes like planes and cylinders. Although these shapes are easily recognizable by only a couple of points, the cloud still consists of hundreds of thousands to millions of points. This redundancy might be used to determine every facet of change in the scene the dataset represents.

2.4.2 Change detection of vegetation

Change detection of vegetation has become a highly important issue in our days since the development or destruction of the ecosystem is determinant from the point of view of urban life, including life conditions and urban planning. Expansion of the urban area and industrialization both mean a great impact to the growth of vegetation, therefore the changes should be continuously monitored and analyzed.

LiDAR proves to be an effective technology for the monitoring of changes in vegetation which is mostly represented by irregularly distributed points in the dataset. This irregularity is helpful when the aim is to monitor changes in vegetation since, as mentioned before, artificial objects are most likely constructed by geometrically regular hulls. LiDAR-based monitoring is also efficient because the laser beams easily penetrate through leaves and the canopy of trees. The denser the point cloud, the easier it is to detect changes in height and land coverage.

Specifically, multiple return is useful in the change detection of vegetation. Laser beams gradually grow after leaving the sensor and take the shape of a cone thus they are able to hit multiple objects and produce multiple reflections due to their size. The data earned from multiple returns caused by beam divergence is a great indicator of the general and canopy height of a tree. This phenomenon is depicted by Fig. 2.2.

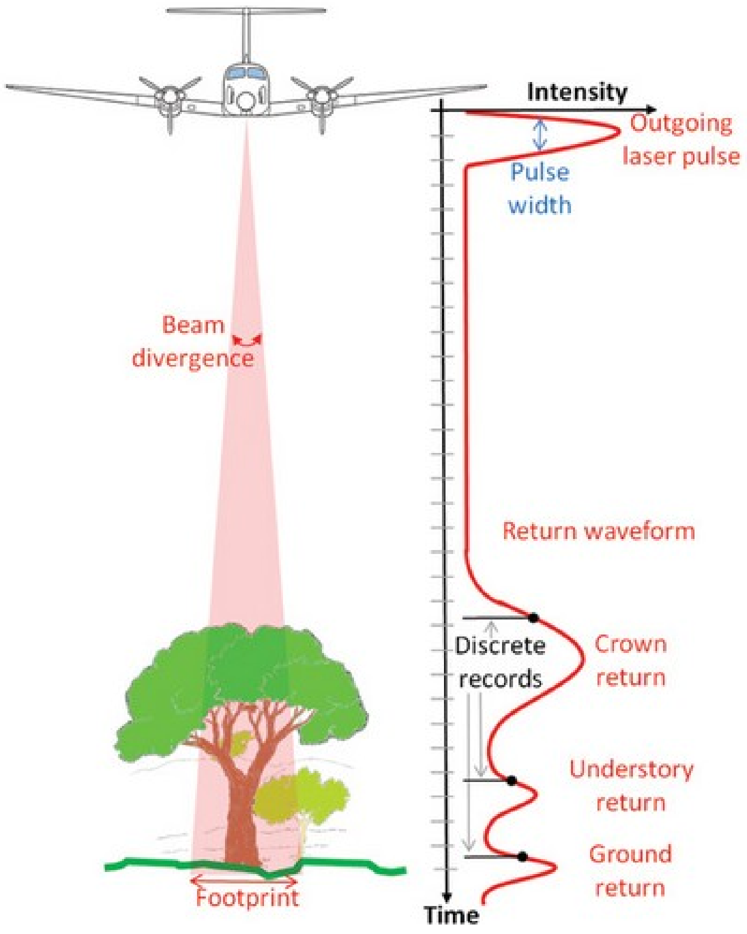

Figure 2.2: Multiple return of laser beams.
Source: Fernandez-Diaz, J. C. Lifting the Canopy Veil - Airborne LiDAR for Archeology of Forested Areas. *Imaging Notes, 26(2).* 2011.

When monitoring specifically forestry, point clouds can also be used to determine if the forest is natural or planted, evergreen or deciduous. In the former case, a certain regular form can be observed in the general irregularity of trees in the point cloud which might be illustrated by density histograms. In the latter case, the naturally wider leaves of deciduous plants have a greater ability to catch and reflect laser beams, making their dataset have a greater radius than that of their evergreen counterparts.

Not only trees and forestry, but the changes in the total biomass of an area might be monitored by analyzing multitemporal laser scanned data. The point clouds can be collected by both terrestrial and airborne laser scanning.

Meyer et al. [16] tried to estimate the biomass change of a 50 ha tropical area. They collected canopy height metrics and used an importance analytical method to evaluate the importance of the variables. They demonstrated the use of spatial scales in biomass estimation and determined that they give more accurate results when used on finer scales.

Another LiDAR based change detection method that was specifically aimed at trees was described by Kaasalainen et al. [12]. In their paper they show a usage of the TIN model to detect quantitative changes in tree growth and litter production. They created a flexible surface model of each tree using the QSM method [19] and compared it to the model created with the TIN model and gave results of 10% accuracy.

In conclusion, it is clear that several different approaches are existing for the change detection of both natural and artificial environment and they provide more or less accurate methods. However, these methods are usually manual or semi-automated and they are not capable to cover large areas.

Chapter 3

Analysis and specification

3.1 Dataset description

In order to test my method, I needed a sample area with diverse landscape (containing both artificial and natural objects) of which there are existing point clouds recorded in different epochs. The study dataset was *Actueel Hoogtebestand Nederland (AHN)*[1], a multitemporal point cloud altimetry archive which covers The Netherlands, 41.526 km^2 per each data acquisition. There are three different datasets in AHN:

- **AHN-1** point cloud was scanned between 1996 and 2003.
- **AHN-2** point cloud was scanned between 2007 and 2012.
- **AHN-3** point cloud acqusition was started in 2014 and is planned to be finished in 2019.

In regards of the accuracy of the points, there are two important parameters to consider:

- *Systematic error:* this represents the accuracy of the entire dataset, the deviation from the actual area. It is hard to detect or sense since it is present in the whole point cloud. This value is maximum $5cm$ for every AHN dataset.
- *Random error:* this is the possible error of a single point independent from every other. Outliers can be removed from raw point clouds through preprocessing. The maximum value of this error is $45cm$ in AHN-1, and $15cm$ in AHN-2 and AHN-3.

[1]Actueel Hoogtebestand Nederland: `http://www.ahn.nl/index.html`

The point density of the datasets [18, 17] also improved over time:

- AHN-1: 0.06-1 point/m^2
- AHN-2 and AHN-3: 6-10 points/m^2

Raw point clouds are extremely large in the dataset [23], as AHN is built up of 1372 tiles each of which covers 31.25 km^2. A single tile is typically over 15 GB (in LAS format). AHN offers preprocessed DEMs with $0.5m$ and $5m$ resolution whose data size is 0.5 GB per tile. In order to spare time and computational cost and attain the most accurate results possible, I used the $0.5m$ resolution DEMs for my research.

AHN-1 has significantly poorer density compared to AHN-2 and AHN-3 due to the less developed technology that existed at the time of scanning, and does not offer $0.5m$ DEM, therefore the focus of landcover classification and change detection was set on the AHN-2 and AHN-3 datasets.

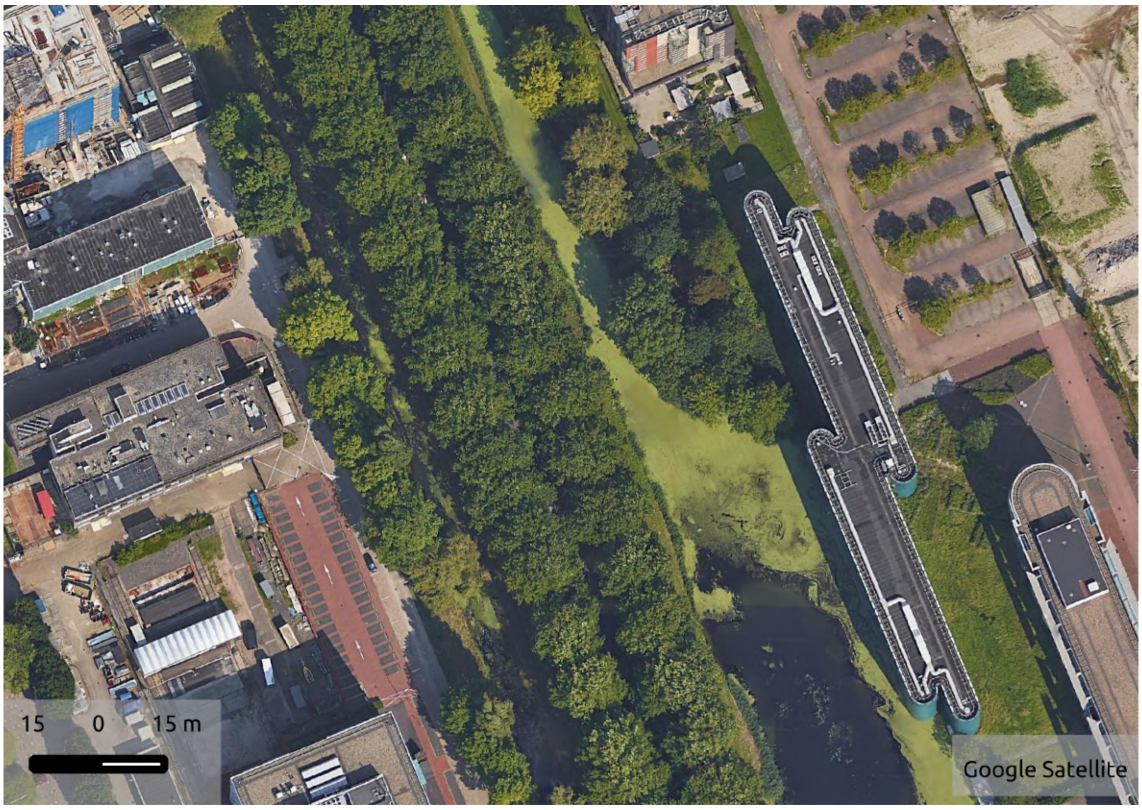

Figure 3.1: Satellite image of the study area.

In order to properly showcase the proposed methodology, a fitting demonstration area had to be selected, which contains urban environment with plenty of vegetation, preferably with considerable changes between the AHN-2 and AHN-3 data acquisitions. The campus of the Delft University of Technology and its surroundings was selected as the testing area, which fulfills these criteria. The city of Delft were scanned in the years 2008 and 2014 for AHN-2 and AHN-3 respectively.

The steps of Sec. 3.2 will be demonstrated on a sample area of the Delft Campus. This is presented with Google Satellite image in Fig. 3.1.

3.2 Methodology

The aim of this research is to provide an algorithm which, when executed on a preprocessed point cloud, produces a cluster map out of the dataset that covers every tree in the area by exactly one cluster. If this is executed on multitemporal point clouds, changes in tree presence, tree height and canopy volume can be calculated. In order to achieve this, the following steps are executed:

1. Produce a canopy height model of the DSM and DTM of the area in the same epoch.
2. Remove excess local maximum points from the CHM.
3. Collect the remaining local maximums.
4. Construct a cluster map from the collection of seed points in which one cluster is equivalent to one tree.
5. Commit morphological opening on the cluster map to erode oulier points.
6. Pair up clusters of the same area in different epochs and seclude trees without a pair in both epochs.
7. Calculate metrics of change in vegetation
   - 7.1. Calculate the height difference of tree pairs.
   - 7.2. Calculate the volume difference of tree pairs and the total volume difference of the epochs.

These steps are going to be described in detail in the following sections.

3.2.1 Producing canopy height models

The canopy height model (CHM) represents the height of individual trees that are present at the examined area. It is constructed by subtracting a digital terrain model (DTM) from a digital surface model (DSM). This process is depicted in Fig. 3.2. As discussed before, a DSM contains a point cloud of the top of the surface thus depicts the terrain and also the natural and man-made environmental elements. On the other hand, a DTM represents the bare-earth surface, this is why their difference provides a good model of the vegetation.

The results of canopy height model production are illustrated by Fig. 3.3 and Fig. 3.4. It can be noted, that the CHM for AHN-2 contains significantly more near ground points (with a canopy height almost 0) compared to the CHM for AHN-3. This is the result of a small divergence between the altimetry values contained by the DTM and the DSM for AHN-2 for the terrain. This issue will be addressed in Sec. 3.2.3.

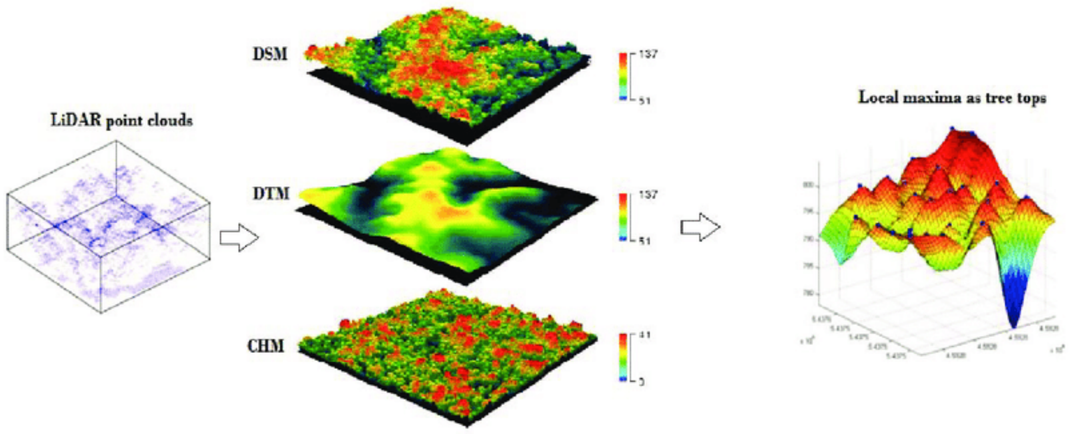

Figure 3.2: Creation of canopy height model from DSM and DTM [25].
Source: `https://www.earthdatascience.org/`

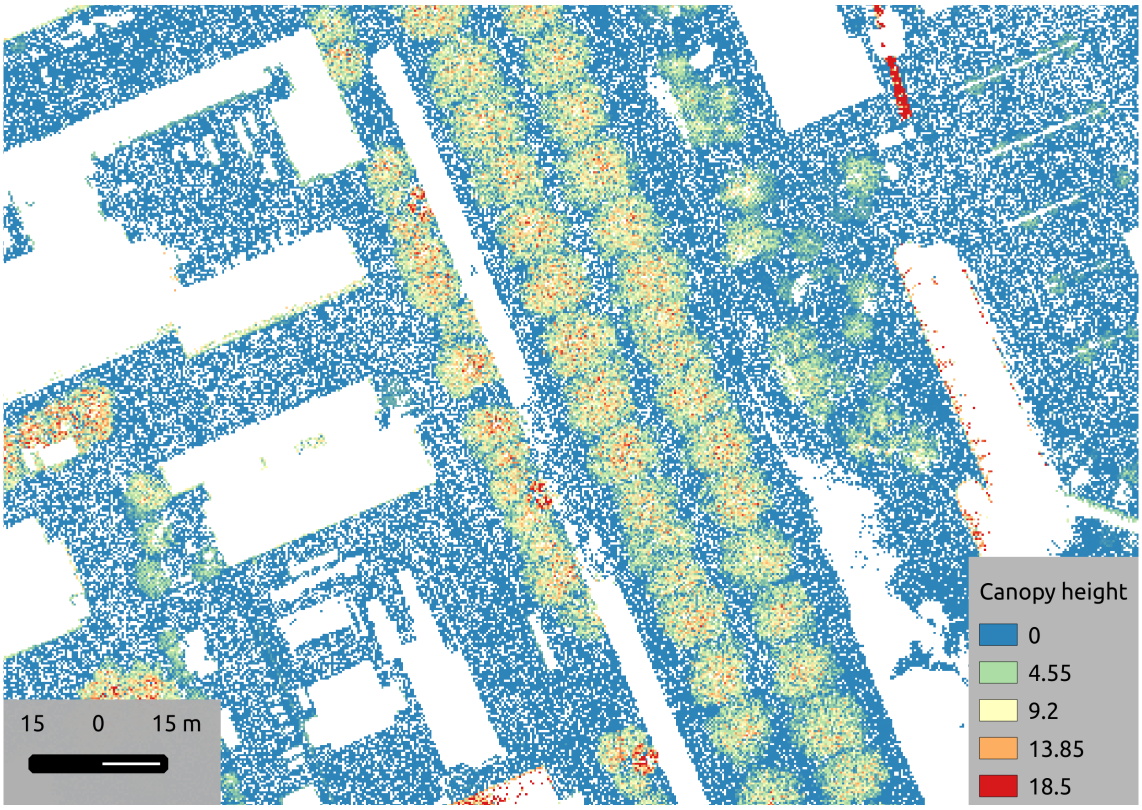

Figure 3.3: AHN-2 canopy height model. All height values are represented in meters.

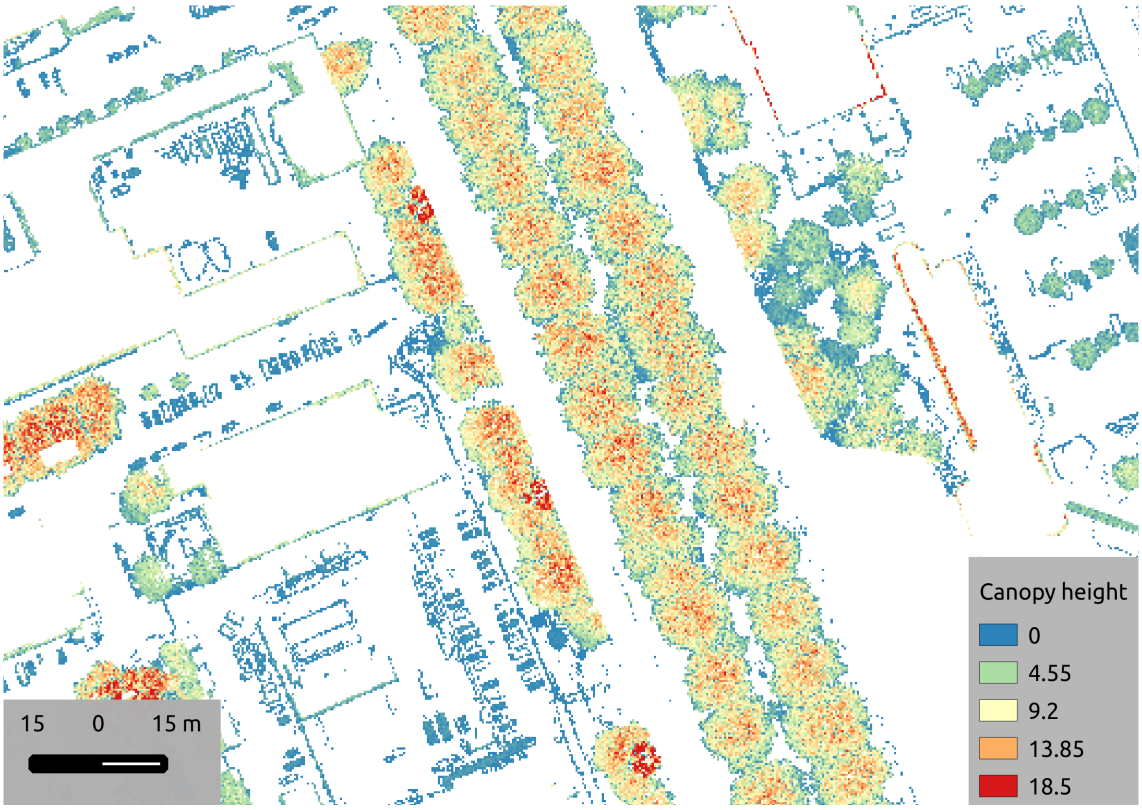

Figure 3.4: AHN-3 canopy height model.

3.2.2 Low-pass filtering

Classification should originate and expand from a *seed point* which is some distinctive point in the CHM. For trees, the obvious choice is to originate a point set - henceforth called *cluster* - from the highest point of the tree. However, a tree might contain multiple "highest points" (which are not necessarily the same height but are surrounded with many lower points). This is why clustering will be done by collecting local maximum points that are the highest points in their surroundings.

Canopy height models are not reliable sources of providing good clustering on their own because they might contain too many local maximum points thus the future classification could result in too many small clusters. This is why the number of local maximum points needs to be reduced by removing several unnecessary peaks. The elimination was done by another sweep-line transformation which used a convolution matrix that "swept" through the CHM and kept all points that are in the close proximity of other points [11].

1. Let v be a sum value and c be a counter value. Define the convolution matrix:

$$\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

2. Sweep a 3×3 window on the CHM. If a point p in the window contains source data s, do one of the following:

   2.1. If p is in a corner of the window, let $v = v + s$ and $c = c + 1$.

   2.2. If p is in the center of the window, let $v = v + s \times 4$ and $c = c + 4$.

   2.3. If p is anywhere else in the window, let $v = v + s \times 2$ and $c = c + 2$.

3. After each iteration, substitute the central point of the window by $v \div c$.

This method corrects the data that would distort the further calculations, e.g. multiple local maximum points in one tree that are too close to each other which would produce a future cluster to be torn into multiple smaller clusters.

The results of low-pass filtering are illustrated in Fig. 3.5 and Fig. 3.6.

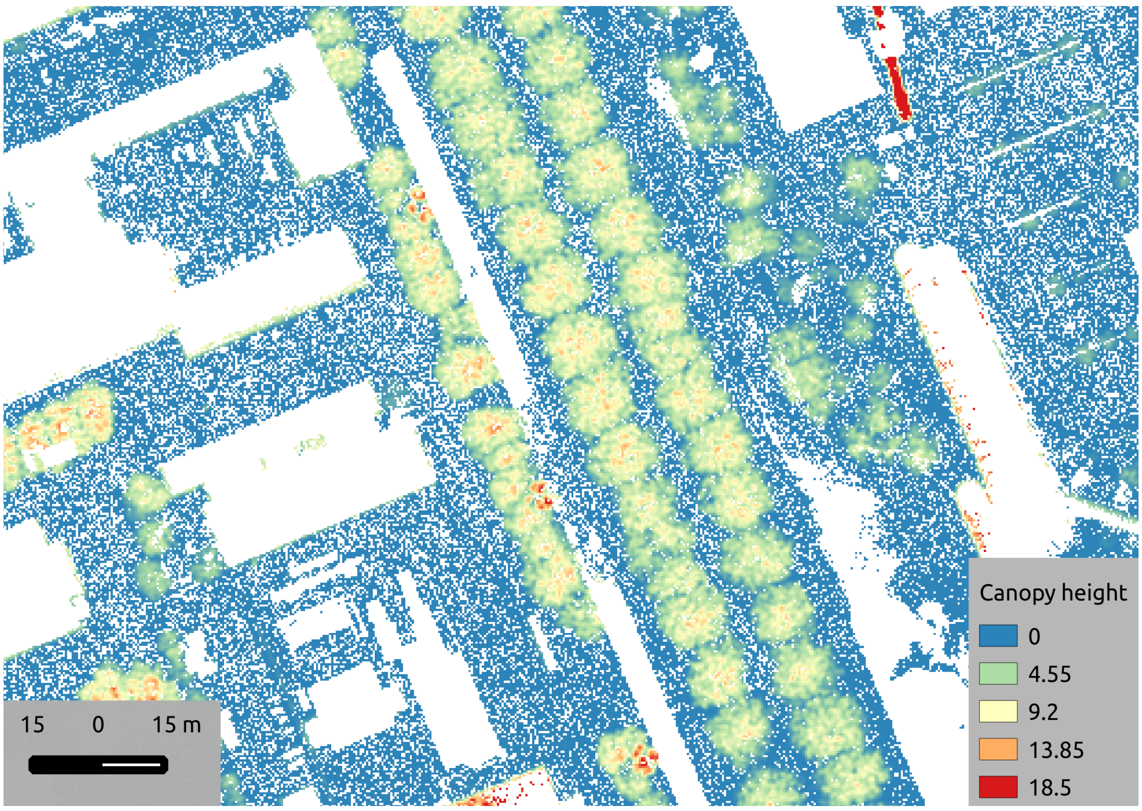

Figure 3.5: Low-pass filtering performed on AHN-2.

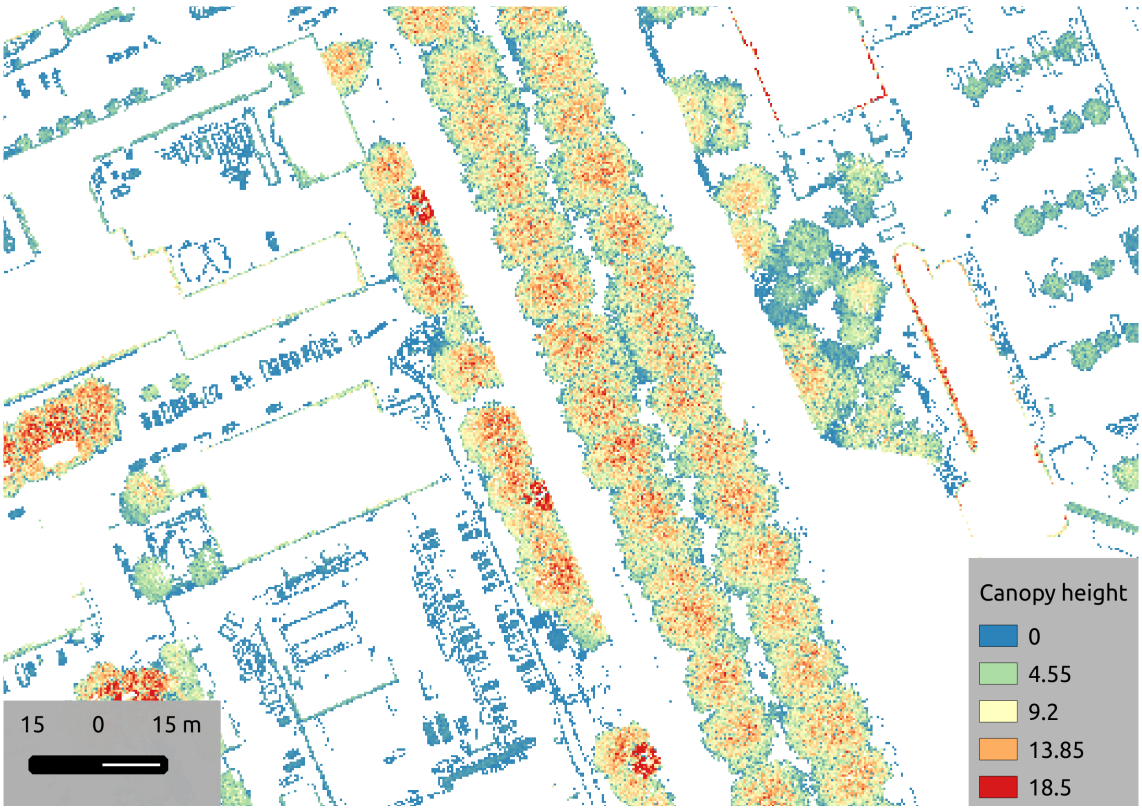

Figure 3.6: Low-pass filtering performed on AHN-3.

3.2.3 Elimination of low points

The main goal of my research is to detect changes in specifically trees among vegetation. The previously filtered canopy height model still contains points that do not belong to trees. These points typically build up vegetation that are shorter than an average tree which means they could be erased from the CHM. In order to have only trees in the CHM, I executed a sweep-line transformation on the model that erased every point below $1.5m$.

The results of the elimination of low points are illustrated in Fig. 3.7 and Fig. 3.8.

Remark. *The constant of 1.5 m may vary depending on the average height of trees in an area.*

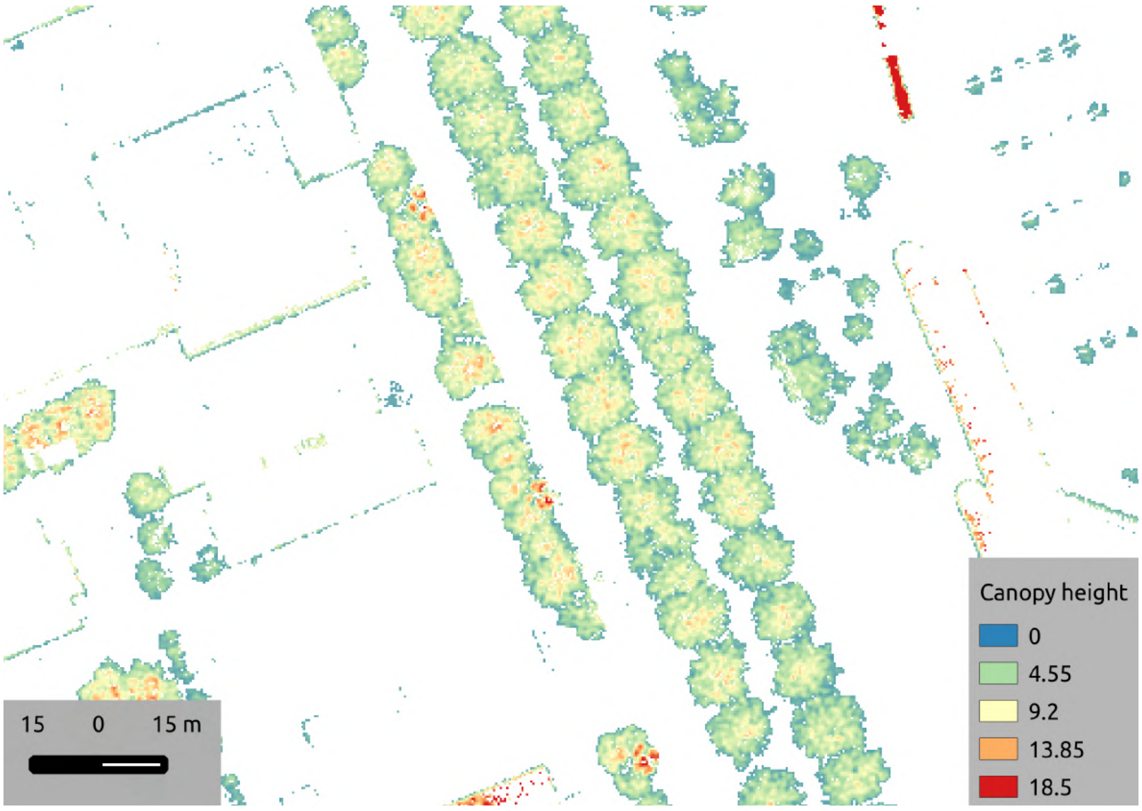

Figure 3.7: Elimination of low points performed on AHN-2.

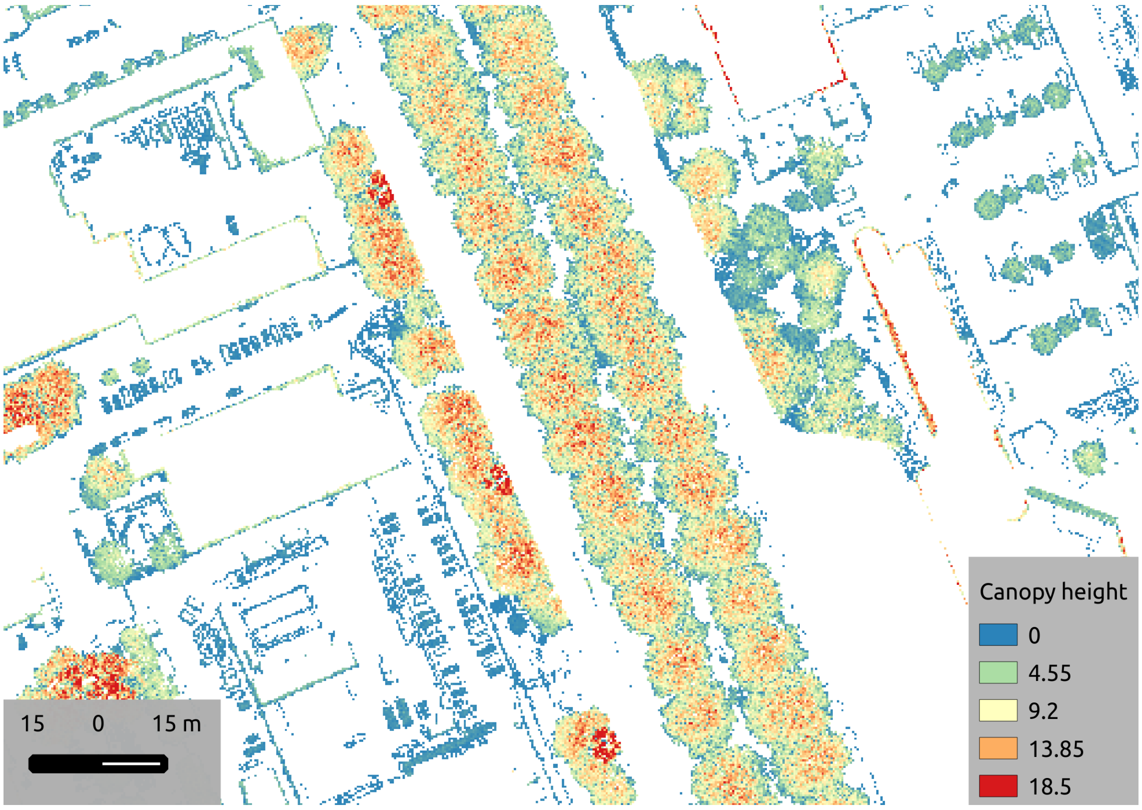

Figure 3.8: Elimination of low points performed on AHN-3.

3.2.4 Counting and collecting local maximum points

My fundamental aim was to create a cluster map that covers the trees. First, I needed the seed points of the prospective clusters. As mentioned before, the most reasonable decision was to set the tree tops as the seed points. This was achieved by executing a sweep-line calculation on the point cloud which ran a 3×3 kernel matrix (*window*) through the points and saved the point to a container if it was located higher than all its neighboring points, meaning it was a local maximum point.

3.2.5 Construction of a cluster map - tree crown segmentation

Once the seed points are collected, the next step is to construct clusters that cover the same tree and extend them by classifying the remaining points. In order to be able to define a cluster map that covers the vegetation tree by tree, 2 constant values need to be defined:

- *Maximum horizontal value,* max_h: the assumed greatest horizontal radius that a tree canopy can reach calculated from its seed point.

- *Maximum vertical value, max_v*: the assumed greatest vertical difference of the seed point and another arbitrary point in a cluster.

In order to give a detailed explanation of the algorithm, the following concepts need to be initiated:

- *Neighbors of a cluster* are the points adjacent to the cluster but not part of it or any other cluster. No other limitations are given when collecting the neighbors of a cluster because they can be used for various purposes that have their own limitations regarding the neighbors so they are further filtered later.
- *Nodata values* are the points in the raster where one or more coordinates are missing. Nodata points can be generated at the creation of DEMs or CHMs when the point is nodata in one or both source datasets. The exact value of nodata is a manually given extremal value, usually a very small negative number is sufficient.

Even though a local maximum-decreasing step was described in Sec. 3.2.2, it is still possible that multiple seed points are present in one single tree. This might occur when a tree consists of multiple local peaks that surround local *valleys* [4]. If two (or more) clusters cover the same tree then they should be merged. In order to determine whether a valley between seed points is space between two trees, or a local valley in a single tree, the ratio r of the tree heights and the depth of the valley has to be calculated. For this calculation, I chose the seed point of the shorter tree, so if $r > 1.0$, then the valley defines separate trees, otherwise it is a local valley in one tree. The possible cluster merge cases are illustrated in Fig. 3.9:

(A) depicts the case where two tall trees are very close. The valley between them is marking that the seed points belong to different trees.

(B) illustrates the case when a tall and a shorter tree are close. This case does not require merging either.

(C) shows that when there is a valley inside a (taller) tree, merging is required.

(D) depicts when two shorter trees are close. This case does not require merging either.

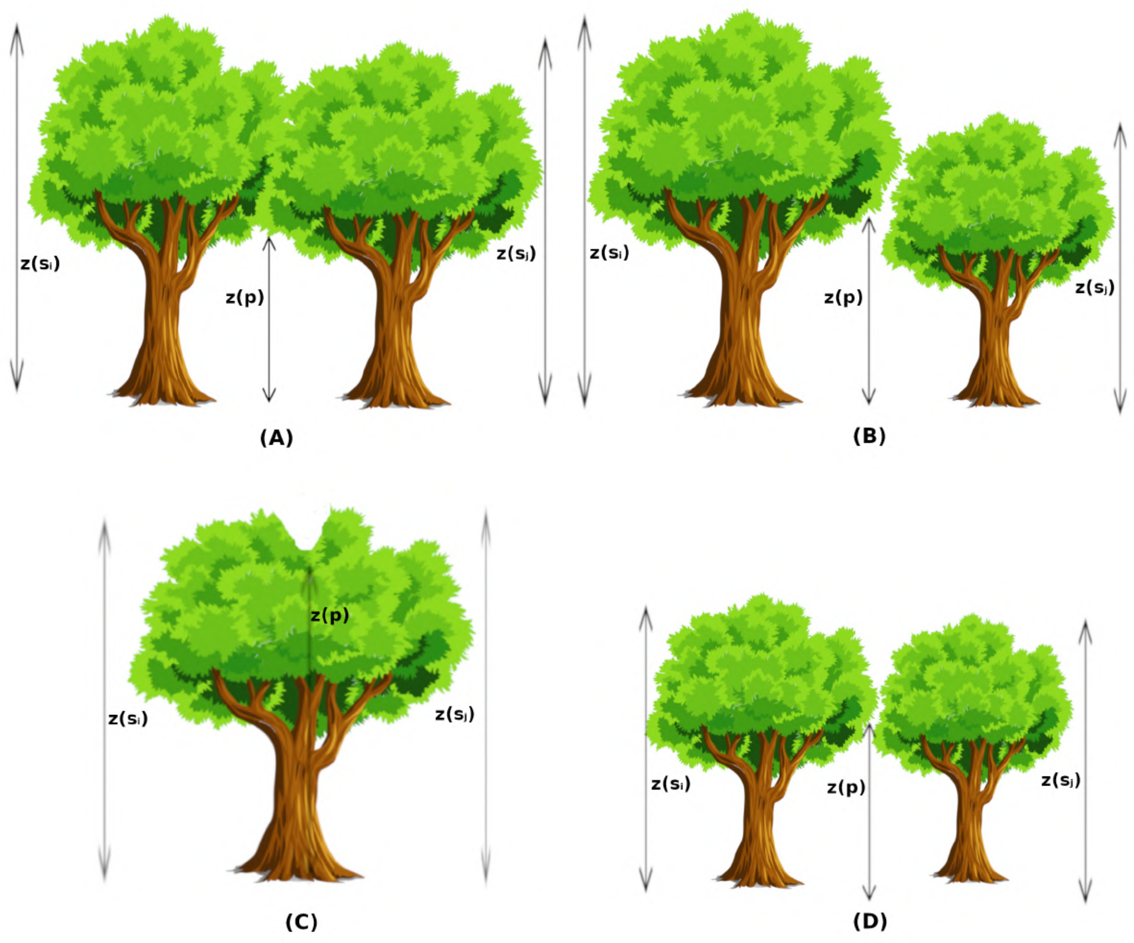

Figure 3.9: The four types of possible cluster merges.

Equipped with the collection of the seed points, the constants and the depth of the valley, the following algorithm is described for the construction of the cluster map:

1. Define the maximum horizontal max_h and vertical max_v radius of one cluster.

2. Construct a cluster for each seed point. Let the set of all clusters be C.

3. Filter the neighbors of every cluster: calculate the horizontal $d_h(p, s)$ and vertical $d_v(p, s)$ distance of a point p to the seed point s. If $d_h(p, s) <= max_h$ and $d_v(p, s) <= max_v$ and p is not a nodata value, then add the point to the the *filtered neighbor set* of the cluster.

4. Take all cluster pairs (c_i, c_j) $(i < j)$ and the height of their seed points $z(s_i)$ and $z(s_j)$. Construct the intersection of their filtered neighbor sets.

5. Take each point p in the intersection (if there is any), and calculate the sum height difference d_z for $z(s_i)$ and $z(s_j)$ and the height of the current point, $z(p)$.

$$d_z = z(s_i) + z(s_j) - 2 \times z(p) \tag{3.1}$$

6. Calculate the ratio r of d_z against $z(s_i)$ and $z(s_j)$.

$$r = \frac{d_z}{min(z(s_i), z(s_j))} \tag{3.2}$$

   If $r < 1.0$ and neither of c_i and c_j are to be merged yet, then list c_i and c_j to be merged.

7. Merge the listed cluster pairs.

8. Expand the clusters by the previously determined neighbors. Do not add points that form intersections twice, nor add points to clusters that no longer exist.

9. Repeat from step 3 until there are no changes made to the cluster map.

At the end of the algorithm, if constants were defined correctly, a cluster map is constructed which covers one tree by one cluster. The results of tree crown segmentation are illustrated in Fig. 3.10 and Fig. 3.11.

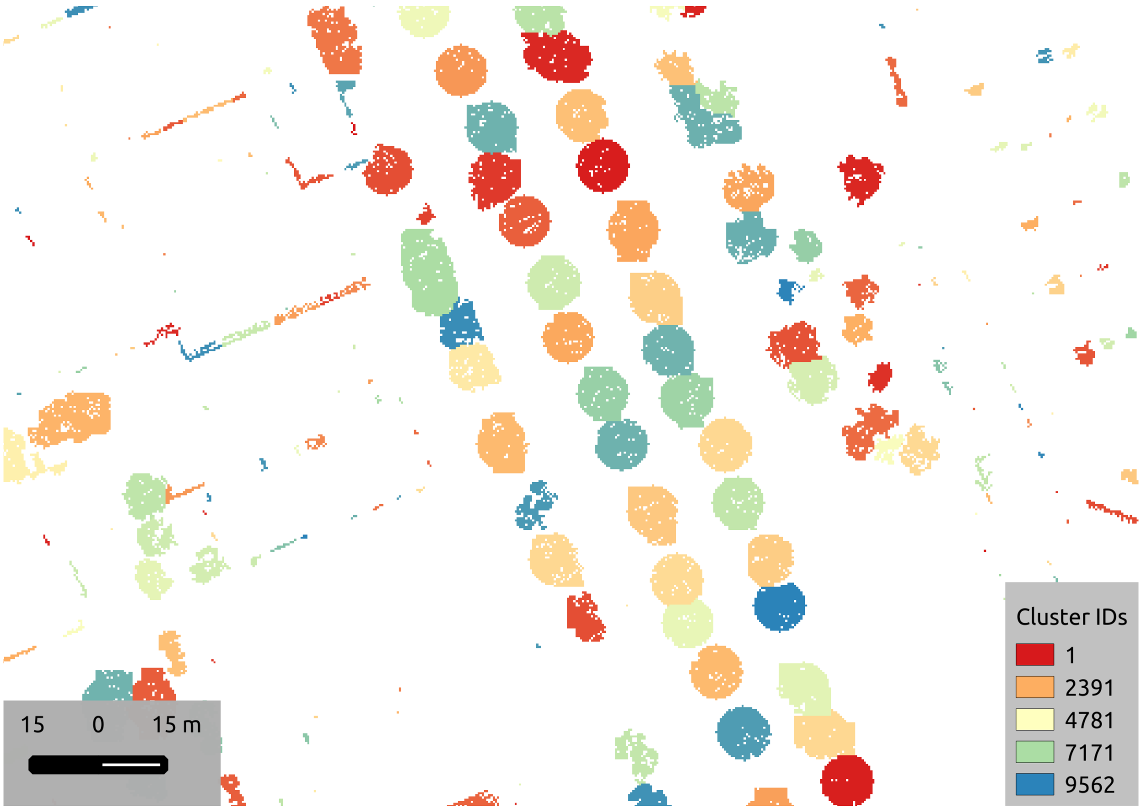

Figure 3.10: Tree crown segmentation performed on AHN-2.

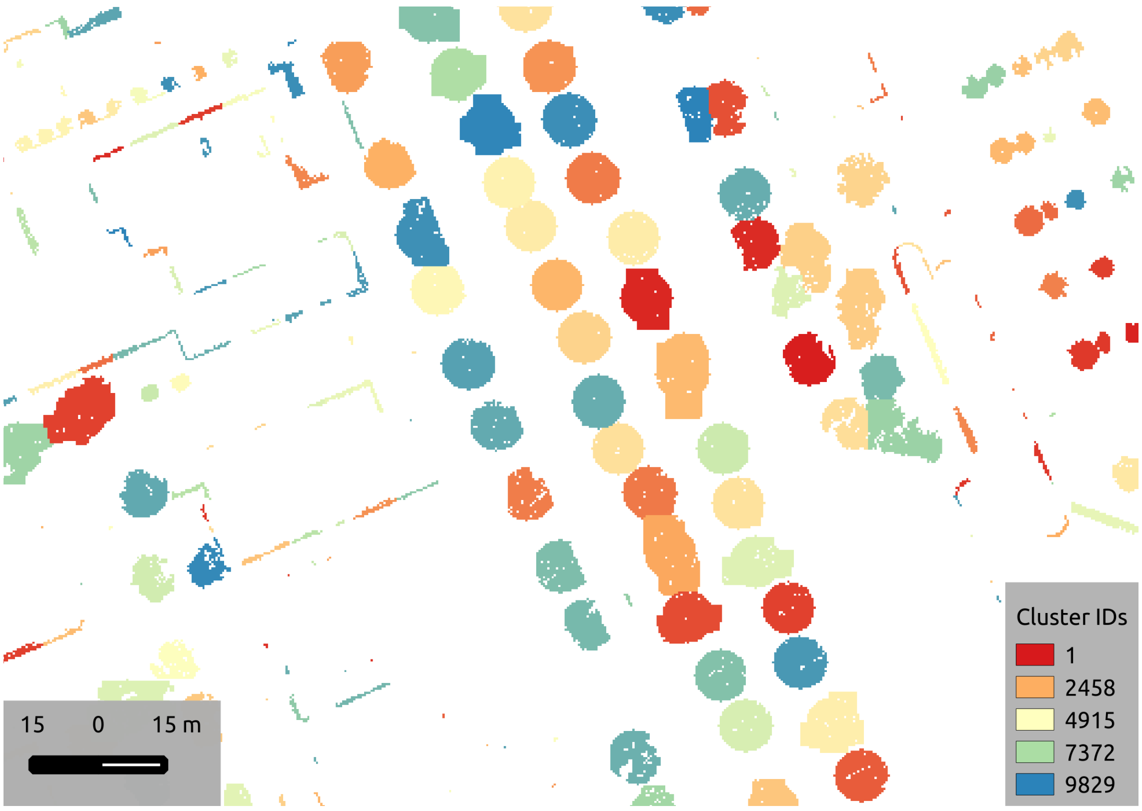

Figure 3.11: Tree crown segmentation performed on AHN-3.

3.2.6 Morphological filtering

Morphological image processing [7, 6, 5] is an image-manipulation method that is suitable for modifying the shape of an image. It inspects a pixel and assigns new value to it according to the other pixel values in its environment. In the case of my research, it relies on the existence of pixels instead of their numerical values, therefore it is especially applicable for the processing of binary images. There are two basic operations:

- *Erosion* removes the small details from a picture while also reducing the size of main regions in the image. It is done by erasing pixels from the image boundaries. Erosion is illustrated in Fig. 3.13.
- *Dilation* expands existing regions of a pixel by adding new pixels to the boundaries of regions. It is used to erase holes and refine the shape of images. Dilation is illustrated in Fig. 3.14.

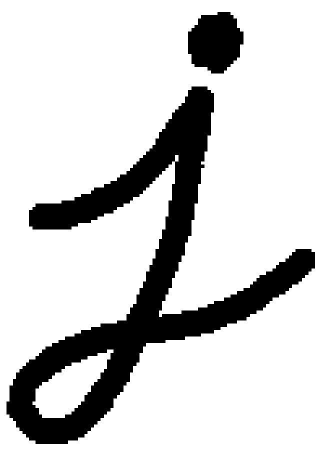
Figure 3.12: Original image.

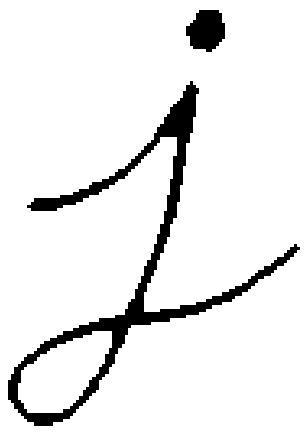
Figure 3.13: Morphological erosion.

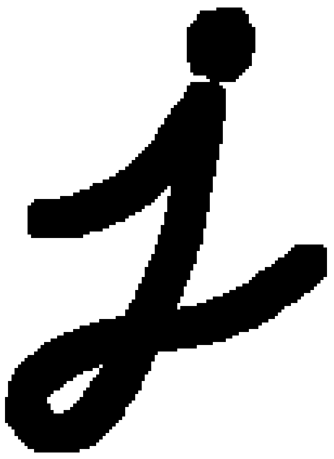
Figure 3.14: Morphological dilation.

Source: `https://docs.opencv.org/`

Erosion and dilation create duality, as they are the exact opposite of one another. They can be used together, thus produce the two fundamental types of morphological filtering:

- *Morphological opening* is executed when erosion is done first followed by dilation.
- *Morphological closing* is executed when dilation is done first followed by erosion.

Morphological opening was performed on the previously constructed clusters the following way:

- *Erosion*: let $Th_{er}(p)$ be a threshold value which determines the amount of neighbors of a p point required to be in the same cluster as p in order for p to remain in the cluster. If $Th_{er}(p) < 6$ (constant may vary), then p is erased from the cluster.
- *Dilation*: let p be a point in the neighboring point set of a cluster, and $Th_{dil}(p)$ be a threshold that determines the number of neighbors of p that are required to be in the same cluster for p to be added to this cluster. If $Th_{dil}(p) > 0$ (constant may vary), then p is merged into the cluster.

Morphological filtering is usually executed multiple times on a dataset in order to make it more accurate and eliminate holes (nodata values) in the cluster map. For these reasons, I execute morphological opening 3 times.

This step was preceded by the removal of small clusters that contain too few points to cover a tree and are most likely the result of DTMs containing tall objects other than trees such as lamp posts or reflecting points on buildings. As a result of the removal, the number of clusters was significantly decreased.

The results of morphological opening are illustrated in Fig. 3.15 and Fig. 3.16.

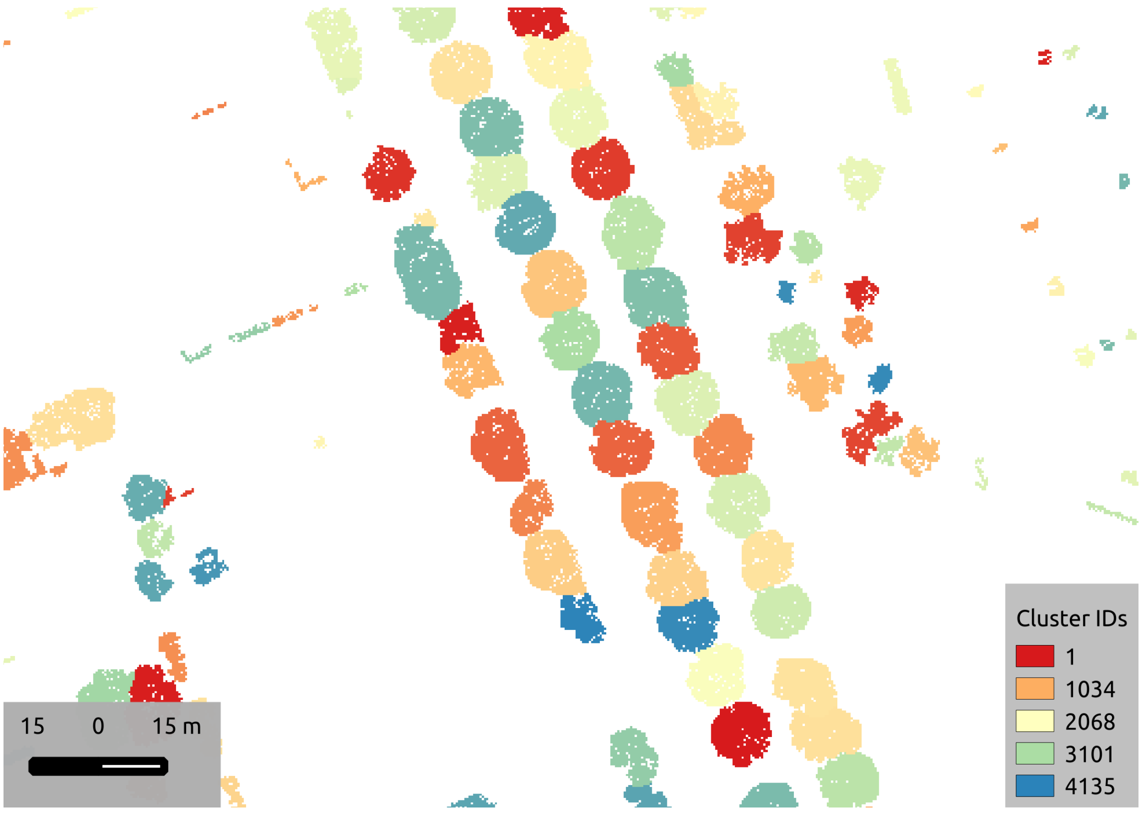

Figure 3.15: Morphological opening performed on AHN-2.

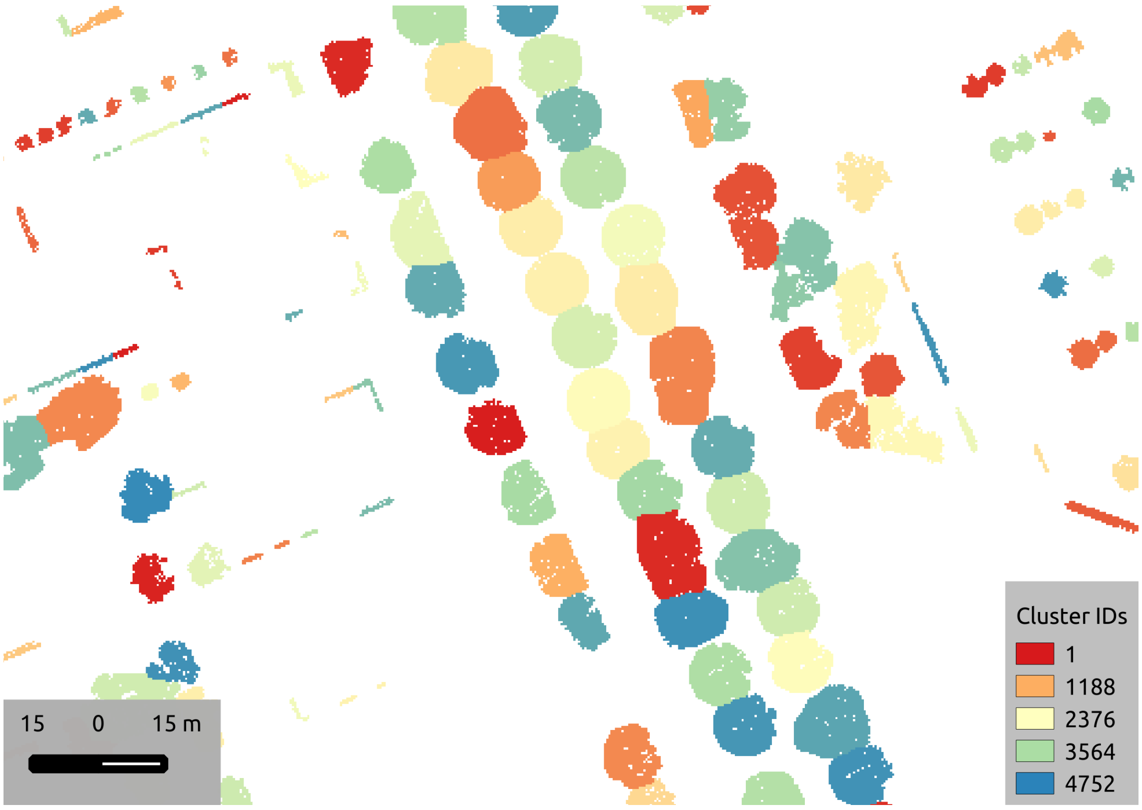

Figure 3.16: Morphological opening performed on AHN-3.

3.2.7 Cluster pairing

After the construction of the cluster map, the next step is pairing up the clusters in the two epochs and determine which trees lack a pair. It is worth noting that the clusters in the two maps generally do not cover each other, as the points collected during the data acquisitions are not located at the same place. This challenge is tackled when a DEM is created but the irregularity of the original points results in clusters covering the same tree not being located at the exact same place even in the DEM. This is why I rather have to determine which cluster in the other map is the closest to the currently examined cluster.

Multiple methods were implemented and tested for pairing:

- The **Hausdorff distance** [20] of two point sets is a *maximin function*: the maximum distance of the cluster to the nearest point of the other cluster. Formally, for sets A and B the Hausdorff-distance can be defined as [8]:

$$h(A, B) = max_{a \in A}(min_{b \in B}(d(a, b))) \tag{3.3}$$

The distance calculation and comparison require very high computational cost due to the high number of points in a cluster. Let k be the number of clusters in the examined AHN-2 cluster map, l be the same in AHN-3, n be the average number of points in an AHN-2 cluster and m be that of in AHN-3. Since the calculation requires the distance of every point in an AHN-2 cluster to be calculated to every point in an AHN-3 cluster and this calculation has to be done to each cluster in both cluster maps, the asymptotic bounds for the calculation of Hausdorff distance is $\theta(n * m * k * l)$.

Remark. *Assuming that $k = l$ and $n = m$ as a simplification, the computational cost can be approximated as $\theta(n^2 * k^2)$.*

In order to decrease the consequent long runtime and great CPU time consumption, a horizontal threshold value Th_h was introduced as the maximal possible distance between two clusters. The examination was limited to check only those AHN-3 clusters that are falling in the range of the examined AHN-2 cluster defined by Th_h. This method eliminates a great deal of unnecessary computation as vegetation does not change its location over time, so if a pair exists for a cluster then that is in the close proximity of the cluster.

- Another approach was pairing up clusters according to their **distance of centroids**. This calculation was also utilized in the previous method for the elimination of unnecessary distance calculation between clusters that are overly far from each other in the first place. On the other hand, the Hausdorff-distance calculation does allow pairing one AHN-3 cluster to multiple others in AHN-2 thus losing partial injectivity and distorting results. This problem was handled as follows.

  1. Define the maximum horizontal distance max_{hc} of any two clusters. Let C_2 be the cluster set of AHN-2 and C_3 be that of AHN-3. Let S be the set of pairs.
  2. Take every c_i $(i \in 1..|C_2|)$ that is not already paired and search for the nearest c_j $(j \in 1..|C_3|)$ by calculating the distance of their centroids where $d_h(c_i, c_j) <= max_{hc}$ and insert the pair into S. Note, that this can result in c_j paired to multiple clusters in C_2.

3. Take each pair from S where $c_j = c_k$ ($k \in 1..|C_3|$), search for the pair where with minimal distance and erase the others from S. This step results in previously paired up clusters from C_2 getting unpaired. For this reason, a new iteration is needed to search for a pair for lone clusters.
4. Repeat from step 2 until no new pairs are found. Hence, the pair set is partially injective.

The results of cluster pairing are illustrated in Fig. 3.17.

Remark. *It is worth mentioning that the Hausdorff distance method could be made partially injective by adding iterations similar to how the other method is done. But since Hausdorff distance is already very costly, new iterations would make it highly ineffective.*

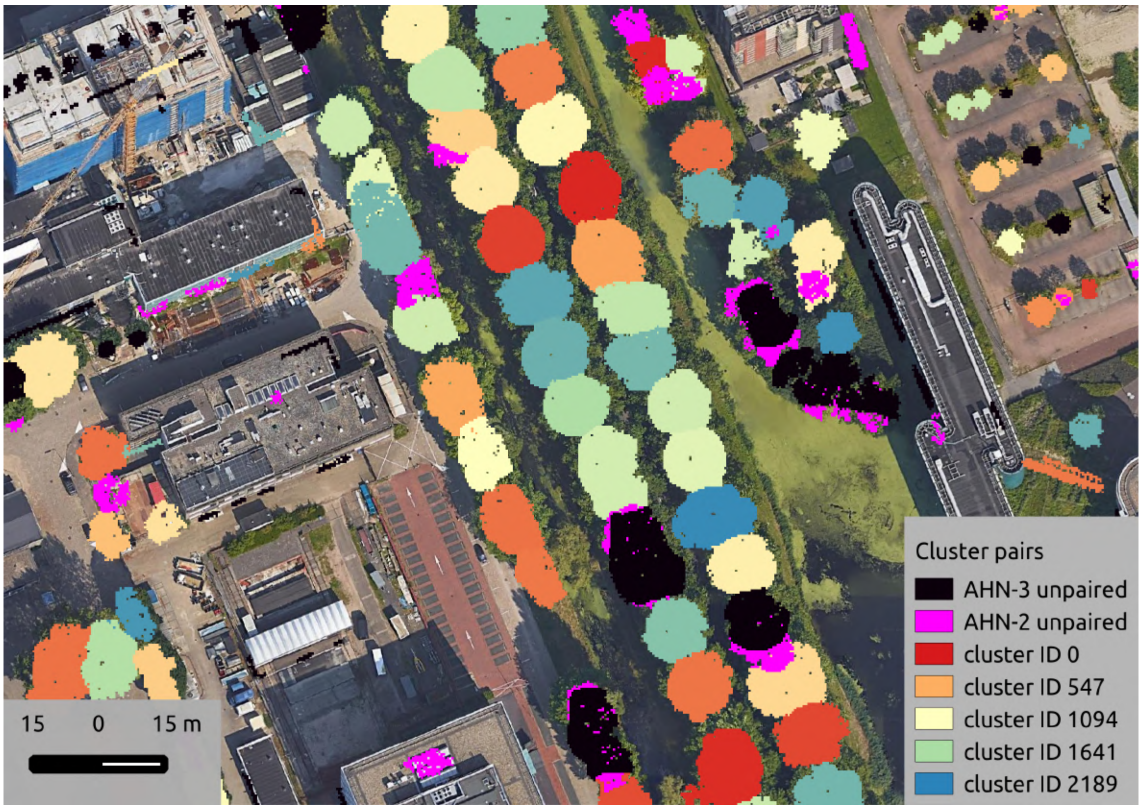

Figure 3.17: Detected paired and unpaired clusters.

Detecting unpaired trees

If the algorithm does not find a pair for a given cluster then it is presumable that if it is present in the first point cloud then the tree it covers was cut between the epochs, and if it is present in the second one, it was planted some time after the first data acquisition. The clusters representing these trees are stored separated from the cluster pairs.

3.2.8 Difference of tree heights

Height differences of individually paired trees and average height difference of an area can be calculated from the data acquired through the previous steps.

- **Individual pairs**: let C_2 be the cluster set of AHN-2 and C_3 be that of AHN-3. Let $z(peak(c_i))$ be the maximum height of a cluster c_i where $c_i \in C_2$, $z(peak(c_j))$ be the maximum height of a cluster c_j where $c_j \in C_3$ and $\delta_h(c_i, c_j)$ the height difference of this cluster pair which is calculated as follows:

  $$\delta_h(c_i, c_j) = z(peak(c_j)) - z(peak(c_i)) \tag{3.4}$$

  If $\delta_h > 0$, then the tree has grown since the first scan, and if $\delta_h < 0$. then the tree has been cut back by some natural or human force. $\delta_h = 0$ is highly unlikely even in the case of trees that have already reached their height limit since the scanning is inconsistent and points do not usually fall to the same exact coordinates in different inspections.

- **Average difference**: This calculation also starts with the collection of the highest points. But in this case I sum up the values of the z coordinates first. As I am aware of the rough number of trees in the area, I am able to divide the sum by this number and get the average height of the trees in the area. This method is executed on both epochs, and extract the former value from the latter, just like in the case of individual trees. If the difference is positive, the trees have grown since the last measurement, and if it is negative, then the average tree height has been decreased either by natural causes or human interference.

  The average calculation should be done by separating the tree pairs from the individual ones. Appearing and disappearing trees could significantly distort the results and their data should be considered outliers. This also means that this statistic cannot be used to determine general conclusions about the vegetation in the area.

3.2.9 Difference between the volume of trees

Once each and every one of the trees in the examined area are located, I am able to calculate their individual and aggregate volume of canopy. This calculation can only be done in seasons when there are actual leaves on the trees since they build up the canopy and provide the volume.

Calculating the volume of a tree is a difficult task if done very accurately since the clusters that represent tree canopies can be considered completely irregular polygons. It would be a very high-demanding task to calculate the exact volume of a cluster regarding computational costs and execution time, so I took advantage of the raster grid instead. Let V_{c_i} be the volume of the cluster c_i, $z(p_n)$ the height of a point and $|c_i|$ the total number of points in the cluster. The computation of V_{c_i} is described by Equation 3.5.

$$V_{c_i} = \sum_{n=1}^{|c_i|} z(p_n) * 0.5^2 \tag{3.5}$$

As mentioned in Sec. 3.1, the real distance between two grid points is $0.5m$ which is why the multiplication is done by 0.5^2. After that, the total volume of the trees in the former epoch is extracted from that of the latter epoch, and similar conclusions can be drawn from the difference as in the case of the changes in tree height.

The previously described method of volume calculation adds redundancy to the total volume by adding the space under the canopy and around the bole, and also a smaller amount around the canopy. However, this does not distort our calculations significantly because our main goal is to determine the overall difference between the volumes in the epochs and the distortion is around the same amount in both scans, hence does not matter when the extraction is executed.

Chapter 4

Implementation

The prototype implementation for the methodology described in Sec. 3.2 was carried out in standard C++11 as part of the *CloudTools* geospatial framework. *CloudTools* – developed at Eötvös Loránd University – aims to create an easily reusable, high-abstraction level, operation-based software library for raw point cloud and DEM processing.

As DEM files (preprocessed from the original point cloud) were selected as the input for the proposed algorithm as discussed in Sec. 3.1 due to computational and storage space efficiency, the *CloudTools.DEM* module was primarily utilized of the framework. *CloudTools.DEM* depends on the *GDAL/OGR*[2] geospatial and geoprocessing software library for the input and out-put management of the spatial data.

The implementation of vegetation segmentation and change detection was implemented in a new module *AHN.Vegetation*.

4.1 Architecture of CloudTools

The *CloudTools.DEM* defines an architecture of DEM processing with an operation-based concept in focus. The module provides various base classes modularized based on their input and output types.

Operation ($Any \rightarrow Any$) is the most general class that serves as the base class of all other operation classes. `Operation` separates the evaluation of a task into 2 phases: i) *preparation*, which is usually a short activity (e.g. opening input data, calculating output size) and ii) *execution*, which carries out the full operation.

[2]GDAL/OGR: `https://www.gdal.org/`

`Operation` is an abstract class with two matching overridable abstract methods: `onPrepare()` and `onExecute()`.

Calculation ($\{DEM\} \rightarrow Any$) inherits directly from `Operation`. It extends the *preparation* phase from `Operation` by opening source data and if there are multiple source datasets (`sourceDatasets`), it also performs validation. (Checks whether all sources have the same pixel resolution, use the same projection system, etc.) The input metadata is read from the source header(s) and then loaded into `sourceMetadata()`. Multiple input sources do not have to cover the exact same area but they must intersect. The intersection will be the *target area* whose metadata is loaded into `targetMetadata()`.

SweepLineCalculation is a subtype of `Calculation`. It completes the *execution* phase by reading all source data. It expects an algorithm (`computation`) in its constructor[3] which is then executed iteratively on a sweeping *window* of the sources.

DatasetCalculation also inherits from `Calculation`, but the complete source datasets for the *target area* are read and passed to the `computation`, which is then performed on a single call.

Transformation ($\{DEM\} \rightarrow DEM$) inherits from `Calculation`. The addition to `Calculation` is that there is a target DEM dataset (`targetDataset`) for the *target area*.

SweepLineTransformation is the correspondent transformation class for `SweepLineCalculation`. It is completed with an output file in which it writes the output data.

DatasetTransformation is the correspondent of `DatasetCalculation` with writing the results into an output file.

[3]The algorithm can be passed as a function pointer, functor or lambda expression.

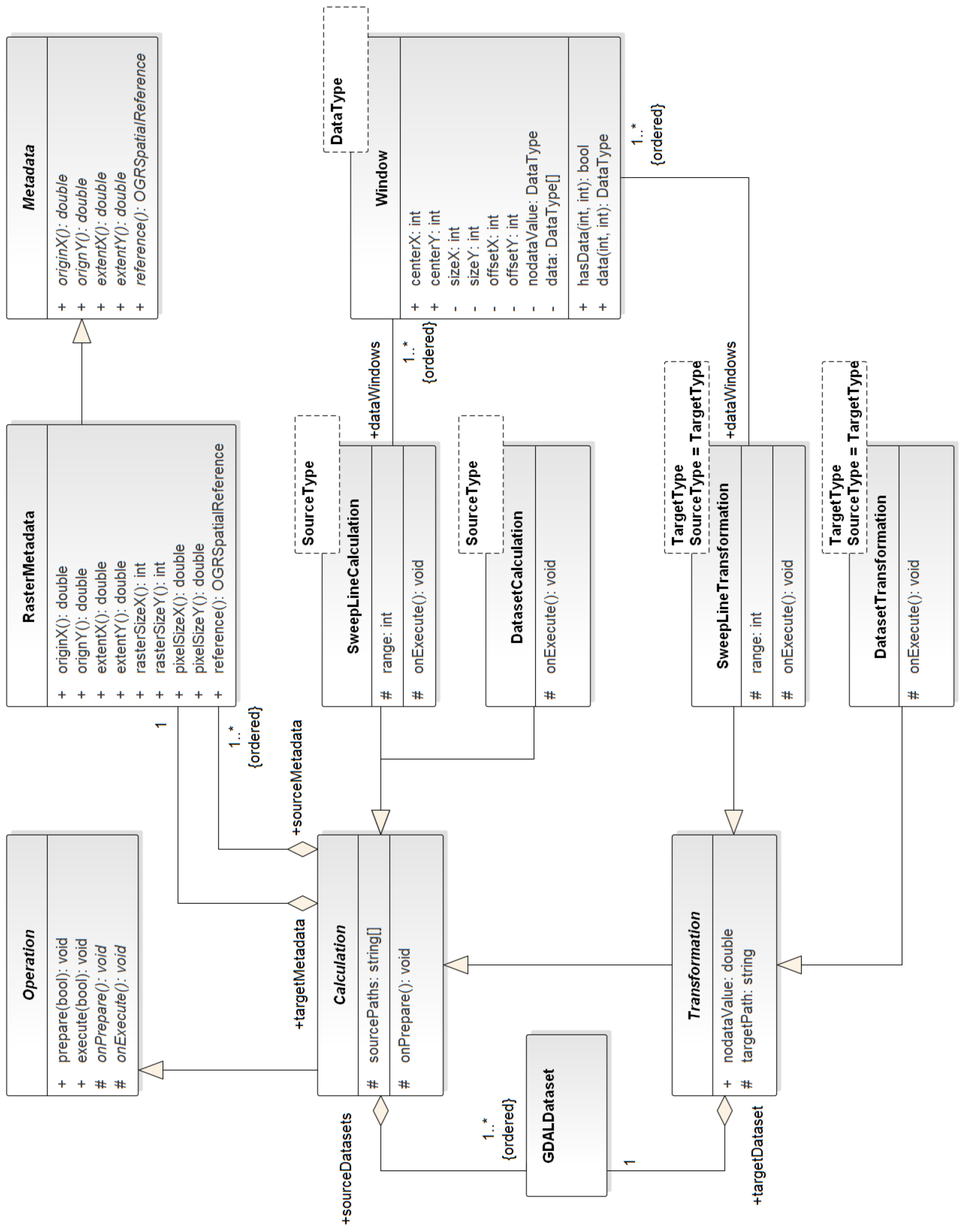

Figure 4.1: UML class diagram of CloudTools' *operation model*.

4.2 Architecture of the prototype implementation

AHN.Vegetation defines the structure of point cloud classification and change detection. The module provides a range of classes derived from the classes of module *CloudTools.DEM*. The classes follow the flow of my algorithm which is described in Chapter 3 and depicted in Fig. 4.2.

Difference is a subtype of `SweepLineTransformation`, capable of calculating the difference of two DEM files with a sweeping window approach (with the window size being 1), thus having low memory requirement. `Difference` was used to calculate the CHM by subtracting DTM from DSM as it is described in Sec. 3.2.1.

MatrixTransformation is a subtype of `SweepLineTransformation`, capable of performing a convolution matrix transformation on a DEM file with a sweeping window approach. `MatrixTransformation` was used to decrease the number of local maximum points with a low-pass filtering as described in Sec. 3.2.2.

ThresholdFilter is a subtype of `SweepLineTransformation`, capable of erasing the points lower or higher than a given threshold from a DEM file with a sweeping window approach. `ThresholdFilter` was used to eradicate the points that are lower than an average tree as described in Sec. 3.2.3.

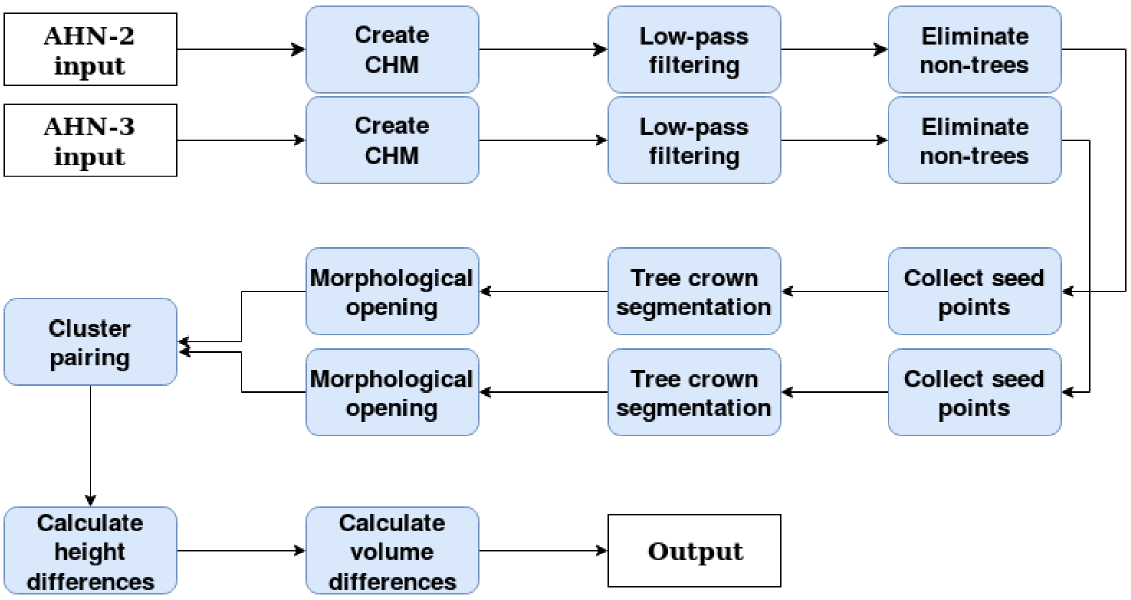

Figure 4.2: Flowchart of the algorithm.

CollectLocalMaximum is a subtype of `SweepLineCalculation`, capable of searching for the local maximum points in a DEM file with a sweeping window approach. `CollectLocalMaximum` was used to collect the local maximum points into a container as described in Sec. 3.2.4.

TreeCrownSegmentation is a subtype of `DatasetCalculation`, capable of constructing a `ClusterMap` of trees from prefiltered CHM produced by the `ThresholdFilter` and the tree top points as seeds provided by the `CollectLocalMaximum`. The `TreeCrownSegmentation` class takes care of constructing, iteratively expanding and merging clusters as described in Sec. 3.2.5.

MorphologyClusterFilter is a subtype of `DatasetCalculation`, capable of performing morphological filtering – both *dilation* and *erosion* – on a provided `ClusterMap` and a CHM dataset covering the affected area. `MorphologyClusterFilter` was used to perform morphological *opening* on the previously built `ClusterMap` as described in Sec. 3.2.6.

HausdorffDistance is a subtype of `Operation`, capable of calculating the Hausdorff distance of clusters in two `ClusterMaps`. The `HausdorffDistance` class was used to pair up the clusters of the `ClusterMaps` constructed from AHN-2 and AHN-3 CHMs as described in Sec. 3.2.7.

CentroidDistance is a subtype of `Operation`, capable of calculating the distance of centroids of clusters in two `ClusterMaps`. The `CentroidDistance` class was used to pair up the clusters of the `ClusterMaps` constructed from AHN-2 and AHN-3 CHMs as described in Sec. 3.2.7.

HeightDifference is a subtype of `Operation`, capable of calculating the height difference of each cluster pair in two `ClusterMaps` as described in Sec. 3.2.8.

VolumeDifference is a subtype of `Operation`, capable of calculating the individual volume difference of each cluster pair in two `ClusterMaps` and their aggregated volume difference as described in Sec. 3.2.9.

Remark. *The steps preceding the pairing of clusters (using either `HausdorffDistance` or `CentroidDistance`) have to be executed twice,* *since there are two input sources (AHN-2 and AHN-3). In order to accelerate execution, these steps of the algorthm can be carried out asynchronously, and the cluster pairing can be initialized once both cluster maps are constructed completely. Fig. 4.2 illustrates this parallelism as well.*

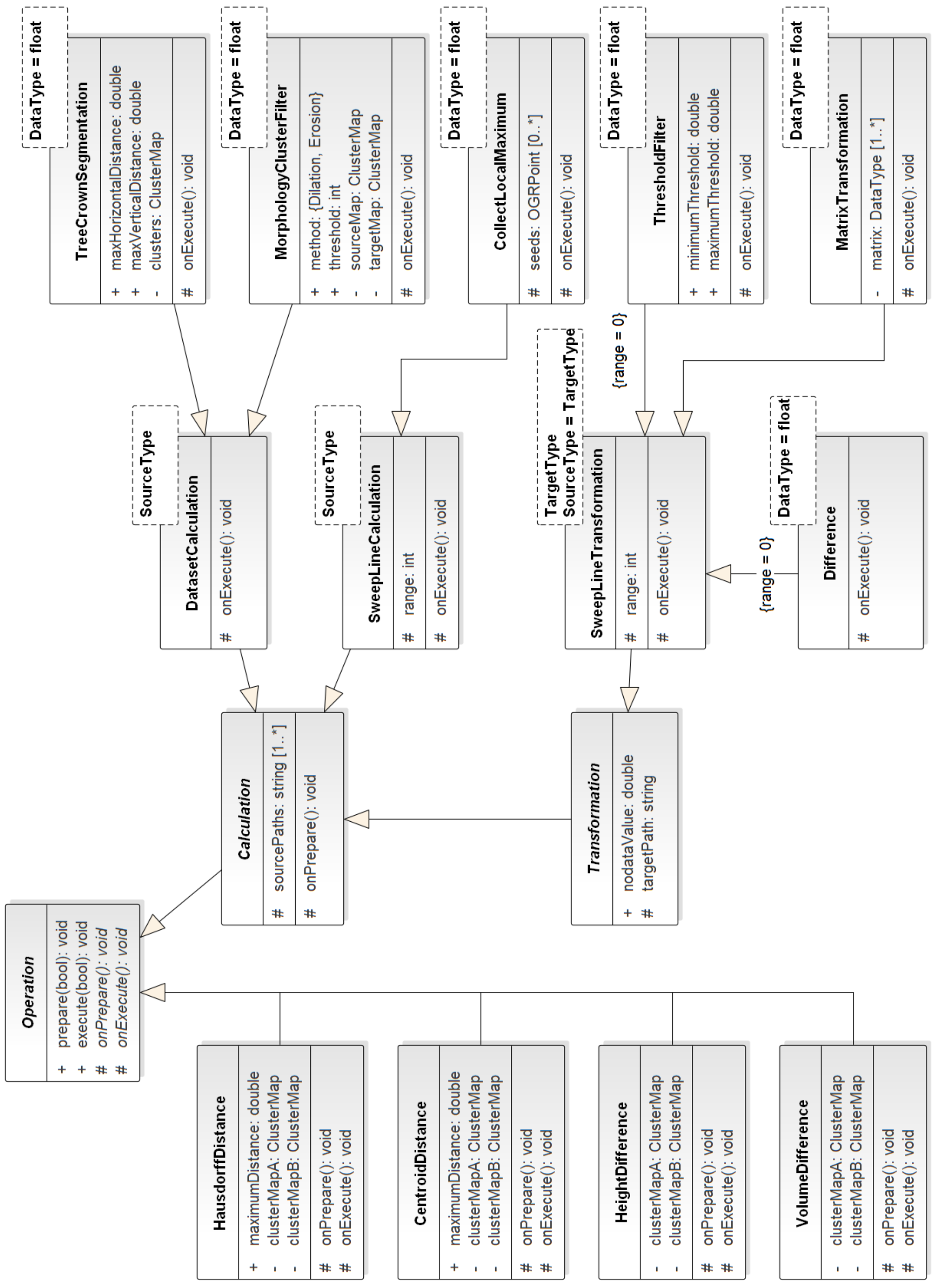

Figure 4.3: UML class diagram of the *AHN.Vegetation* module.

Chapter 5

Results

5.1 Sample territories and performance

Testing of the algorithm was carried out on multiple territories of different size. Table 5.1 contains the name and area of each territory along with performance information for both pairing methods. All tasks were executed sequentially without any parallelization applied on a single CPU core.

Name	Area	Pairing method	Runtime
Single street	0.025 km^2	centroid	2.17 sec
		Hausdorff	125 sec
TU Delft Campus	2.143 km^2	centroid	18 min 10 sec
		Hausdorff	51 min 2 sec
Delft city center	8.64 km^2	centroid	4 h 48 min

Table 5.1: Basic data and runtime information of the sample territories.

Remark. *Increase of runtime is non-linear with the growth of territory size due to the quadratic computational cost mentioned in Sec. 5.2.*

Remark. *Based on the results of testing on the* Single street *and the* TU Delft Campus *sample territories, the centroid distance pairing method turned out to be a better solution in computational complexity.*

5.2 Test results

Table 5.2 contains the results of pairing carried out by different pairing methods for each sample territory.

Sample territory	**Pairing method**	**AHN-2 clusters**	**AHN-3 clusters**	**Pairs**	**Unpaired AHN-2 clusters**	**Unpaired AHN-3 clusters**
Single street	centroid	54	59	40	14	19
	Hausdorff	54	59	41	13	19
TU Delft Campus	centroid	4 135	4 752	2 182	1 953	2 570
	Hausdorff	4 135	4 752	2 251	1 884	2 550
Delft city center	centroid	18 858	20 351	10 319	8 539	10 032

Table 5.2: Results of different pairing methods.

Remark. *The number of paired and unpaired AHN-3 clusters do not add up to the total number due to the not* partially injective *implementation of the Hausdorff distance pairing method as mentioned in Sec. 3.2.7.*

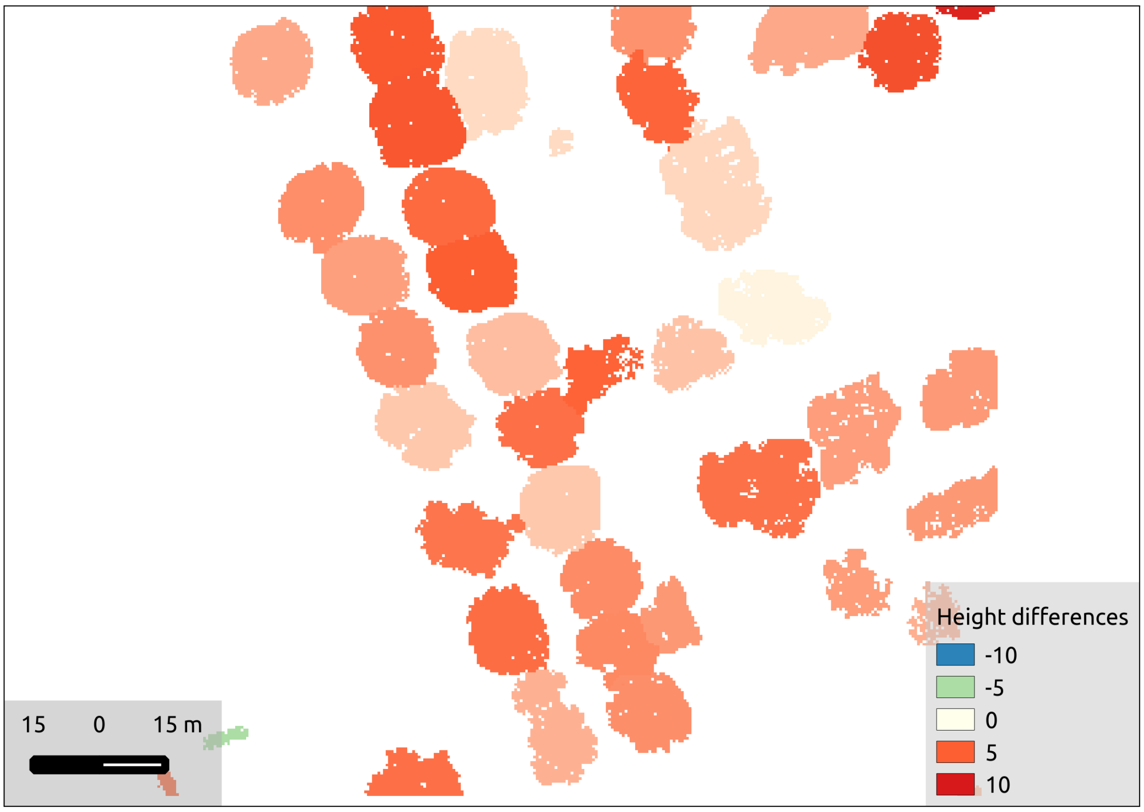

Figure 5.1: Height differences of paired clusters in the *Single street* sample territory.

Table A.1 in Appendix A contains the height differences of individual cluster pairs for the *Single street* sample territory. Height differences are also visualized by Fig. 5.1 and Fig. 5.2 for the *Single Street* and the *TU Delft Campus* sample territories respectively.

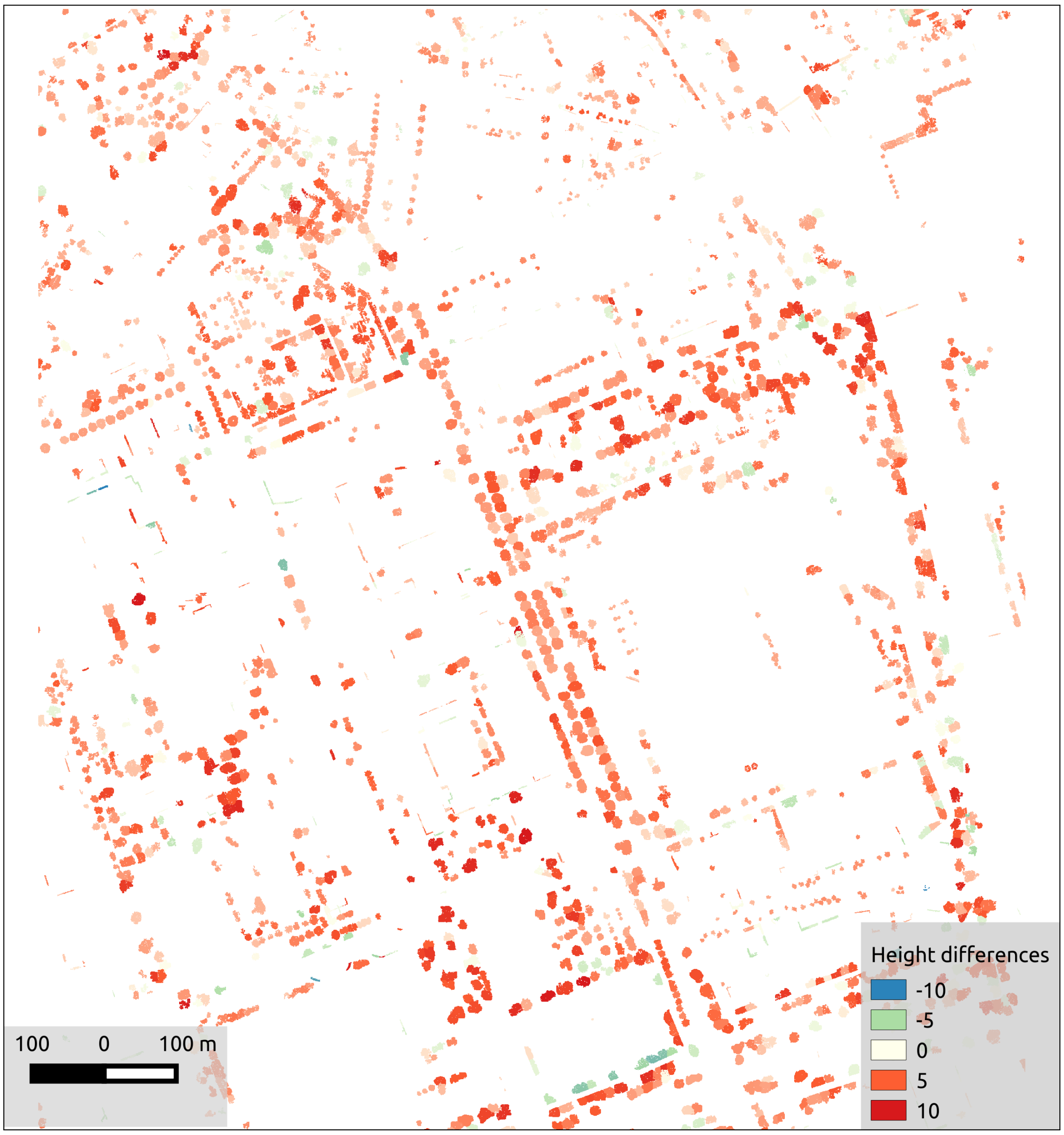

Figure 5.2: Height differences of paired clusters in the *TU Delft Campus* sample territory.

Table 5.3 contains the aggregated volume and total volume difference of both epochs for each sample territory.

Sample territory	Pairing method	AHN-2 volume	AHN-3 volume	Difference
Single street	centroid	39 353 m^3	62 626 m^3	23 272 m^3
	Hausdorff	39 353 m^3	63 075 m^3	23 721 m^3
TU Delft Campus	centroid	1 473 300 m^3	2 350 390 m^3	877 089 m^3
	Hausdorff	1 473 300 m^3	2 384 630 m^3	911 332 m^3
Delft city center	centroid	5 305 280 m^3	8 201 950 m^3	2 896 670 m^3

Table 5.3: Results of volume calculation for different epochs.

Chapter 6

Conclusion and future work

The goal of this research was to develop an automatized, robust algorithm for the change detection of vegetation based on LiDAR data which is applicable for large urban areas. The algorithm works with preprocessed DEM files instead of raw point clouds in order to decrease computational complexity. I construct canopy height models of the DEM files which is then submitted for multiple preprocessing steps (low-pass filtering and elimination of low points). Afterwards, the tree top candidates are collected as seed points of clusters. The clusters are then expanded and merged, if a tree was to be covered by multiple of them. Once I finished processing the sources from both epochs, I am equipped with two cluster maps covering the same area. The clusters of the two maps are paired up, and unpaired clusters are secluded in both maps. Thereby I am able to determine the quantity of removed and planted trees along with those that were present in both scans. Afterwards, I calculate the height and volume differences of individual cluster pairs, and the total volume of both cluster maps which enables the calculation of total volume change. Thus the algorithm is suitable for quantifying changes that occurred between individual LiDAR scans.

The results have shown that the algorithm fulfills the initial expectations. The sample territories covered parts of the city of Delft which is an urban environment with plenty of vegetation. Delft has been exposed to major urban planning between the examined epochs thus creating a good sample territory for detecting planted and removed trees. Cluster construction and pairing gave satisfying representation of the location and scope of trees in the sample territories. Cluster pairing has also provided representative data of the actual quantifiable changes (tree height and volume).

Future work includes the improvement of tree crown segmentation in areas with dense vegetation. The proposed algorithm can fail the segmentation of very close tree crowns, which then introduces further mismatching errors in the pairing phase. Canopy height models also contain occasional nodata points due to scanning errors or faulty DTM generation. The nodata points appear in cluster maps thus distorting results. These points can be corrected and substituted using numeric interpolation based on the surrounding points. The algorithm is still capable of enhancement regarding execution through the parallelization of multiple phases of the algorithm (e.g. tree crown segmentation, pairing of clusters). Pairing methods could also be accelerated by creating indexes for the cluster maps, e.g. a *quadtree* or *R-tree*.

The topic of change detection of vegetation in urban environment still has plenty of potential continuation. From the point of urban planning, it would be very useful to display quantified changes aggregated along administrative areas (cities, districts, etc.). Information like that would help endeavors aiming the protection of the environment and putting climate change under control.

Acknowledgements

This work is supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00001).

Appendix A

Height differences

The following table contains the height differences of individual cluster pairs for the *Single street* sample territory.

AHN-2 cluster ID	**AHN-3 cluster ID**	**Height difference**
17	5	4.32131
20	11	3.30619
22	8	6.60914
24	15	4.39316
27	16	3.75112
28	18	5.39475
29	23	4.80995
33	19	0.848066
38	24	3.73956
40	25	3.80762
44	32	0.0790334
48	46	4.00462
52	45	1.81238
55	43	2.02842
69	52	1.06831
70	53	3.64725
74	59	4.87119
77	68	2.83611

AHN-2 cluster ID	AHN-3 cluster ID	Height difference
79	62	3.66171
83	64	1.78966
85	70	4.95078
87	76	3.00369
89	79	3.65544
90	82	5.58029
92	77	3.80669
94	84	4.10592
97	88	2.77056
99	95	2.4275
101	96	3.52558
105	98	5.07631
107	101	2.93002
109	103	4.30752
111	104	0.168917
113	114	3.3164
116	115	3.73956
127	119	3.5365
130	128	-2.58732
132	127	4.33769
137	132	-0.872801
139	135	4.089

Table A.1: Height differences of individual pairs in the *Single street* sample territory.

Bibliograhpy

[1] A. Antonarakis, K. S. Richards, and J. Brasington. Object-based land cover classification using airborne lidar. *Remote Sensing of environment*, 112(6):2988–2998, 2008.

[2] A Bellakaout, M Cherkaoui, M Ettarid, and A Touzani. Automatic 3d extraction of buildings, vegetation and roads from lidar data. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 41, 2016.

[3] T. Butkiewicz, R. Chang, Z. Wartell, and W. Ribarsky. Visual analysis and semantic exploration of urban lidar change detection. In *Computer Graphics Forum*, volume 27 of number 3, pages 903–910. Wiley Online Library, 2008.

[4] Q. Chen, D. Baldocchi, P. Gong, and M. Kelly. Isolating individual trees in a savanna woodland using small footprint lidar data. *Photogrammetric Engineering & Remote Sensing*, 72(8):923–932, 2006.

[5] P. Delmas. Morphological image processing. `https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm`. Accessed: 2019-04-13.

[6] N. Efford. Morphological image processing. In *Digital Image Processing: A Practical Introduction Using Java*, chapter 11, pages 271–297. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 2000. ISBN: 0201596237.

[7] R. C. Gonzalez and R. E. Woods. Morphological image processing. In *Digital Image Processing (3rd Edition)*, chapter 9, pages 649–710. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006. ISBN: 013168728X.

[8] N. Grégoire and M. Bouillot. Hausdorff distance between convex polygons. `http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html`. Accessed: 2019-04-13.

[9] E. Gülch. Investigations on google tango development kit for personal indoor mapping. *studies*, 1:3, 2016.

[10] J. Hyyppä, J.-P. Virtanen, A. Jaakkola, X. Yu, H. Hyyppä, and X. Liang. Feasibility of google tango and kinect for crowdsourcing forestry information. *Forests*, 9(1), 2018. ISSN: 1999-4907. DOI: `10.3390/f9010006`. URL: `http://www.mdpi.com/1999-4907/9/1/6`.

[11] J. Hyyppa, O. Kelle, M. Lehikoinen, and M. Inkinen. A segmentation-based method to retrieve stem volume estimates from 3-d tree height models produced by laser scanners. *IEEE Transactions on geoscience and remote sensing*, 39(5):969–975, 2001.

[12] S. Kaasalainen, A. Krooks, J. Liski, P. Raumonen, H. Kaartinen, M. Kaasalainen, E. Puttonen, K. Anttila, and R. Mäkipää. Change detection of tree biomass with terrestrial laser scanning and quantitative structure modelling. *Remote Sensing*, 6(5):3906–3922, 2014.

[13] R. Lindenbergh and P. Pietrzyk. Change detection and deformation analysis using static and mobile laser scanning. *Applied Geomatics*, 7(2):65–74, 2015.

[14] G. Y. Lu and D. W. Wong. An adaptive inverse-distance weighting spatial interpolation technique. *Computers & geosciences*, 34(9):1044–1055, 2008.

[15] J. A. McDivitt. Apollo 15 Mission Report. Technical report, NASA, Dec. 1971. URL: `https://www.hq.nasa.gov/alsj/a15/ap15mr.pdf`. Section: 5.12.2 Laser Altimeter.

[16] V Meyer, S. Saatchi, J Chave, J. Dalling, S Bohlman, G. Fricker, C Robinson, M Neumann, and S Hubbell. Detecting tropical forest biomass dynamics from repeated airborne lidar measurements. *Biogeosciences*, 10(8):5421–5438, 2013.

[17] PDOK. Besteksvoorwaarden inwinning landsdekkende dataset AHN2014-2019. Technical report 2.0 final, Dutch National Spatial Data Infrastructure, May 2015. URL: `http://www.ahn.nl/`.

[18] PDOK. Kwaliteitsdocument AHN2. Technical report 1.3 final, Dutch National Spatial Data Infrastructure, May 2013. URL: `http://www.ahn.nl/`.

[19] P. Raumonen, M. Kaasalainen, M. Åkerblom, S. Kaasalainen, H. Kaartinen, M. Vastaranta, M. Holopainen, M. Disney, and P. Lewis. Fast automatic precision tree models from terrestrial laser scanner data. *Remote Sensing*, 5(2):491–520, 2013.

[20] R. T. Rockafellar and R. J.-B. Wets. Set convergence. In *Variational analysis*. Volume 317, chapter 4, pages 108–147. Springer Science & Business Media, 2009. ISBN: 978-3-540-62772-2. DOI: `10.1007/978-3-642-02431-3`.

[21] D. Shepard. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM National Conference*, ACM '68, pages 517–524, New York, NY, USA. ACM, 1968. DOI: `10.1145/800186.810616`. URL: `http://doi.acm.org/10.1145/800186.810616`.

[22] J.-H. Song, S.-H. Han, K. Y. Yu, and Y.-I. Kim. Assessing the possibility of land-cover classification using lidar intensity data. *International archives of photogrammetry remote sensing and spatial information sciences*, 34(3/B):259–262, 2002.

[23] L Swart. How the Up-to-date Height Model of the Netherlands (AHN) became a massive point data cloud. *NCG KNAW*, 17, 2010.

[24] G. Vosselman and H.-G. Maas. *Airborne and terrestrial laser scanning*. CRC, 2010.

[25] L. Wasser. Create a Canopy Height Model With Lidar Data. `https://www.earthdatascience.org/courses/earth-analytics/lidar-raster-data-r/lidar-chm-dem-dsm/`. Accessed: 2019-03-22.

[26] X. Yu, J. Hyyppä, A. Kukko, M. Maltamo, and H. Kaartinen. Change detection techniques for canopy height growth measurements using airborne laser scanner data. *Photogrammetric Engineering & Remote Sensing*, 72(12):1339–1348, 2006.

List of Figures

List of Tables