UMEÅ UNIVERSITY **June 27, 2019**
Department of Physics
Research and Development Project in Engineering Physics

Tree stem detection in high resolution ALS data

Rickard Sjödin
rickard.sjodin@outlook.com

Supervisors:
Mattias Nystöm
Johan Holmgren

Abstract

Thanks to recent advancements in LIDAR technology, it is now possible to get detailed information about forests with an accuracy that can compete with conventional methods. The sensors can be worn by humans, put on vehicles and can recently even be put on drones.

This report investigates the possibility to automatically detect tree stems in high-resolution, low-height, airborne laser scanned data (ALS), collected by a helicopter. The developed algorithm is projecting the data set onto the horizontal plane and filters the low-density areas. This filtered plane is then segmented into clusters based on a small distance, and each cluster is checked to see if it has the expected properties of a stem. This is done in multiple directions in order to find additional stems standing at an angle. The algorithm showed promising results and this approach could potentially be useful in this type of data.

Contents

1 Introduction

There has for a long time existed a demand to know detailed information about forests. For instance, a buyer of a forest area wants to know exactly what they are buying, and a seller wants to know what they are selling. Large forestry companies wants to know in which shape their forest is in, and how much wood they can get from certain areas. This makes information about properties such as stem diameter, tree height, stem shape and others valuable to obtain. The problem however, is that collecting this information takes a lot of time and effort with conventional methods. Some properties are so impractical to obtain that they are omitted completely, while others are at best rough estimations of the truth.

Thanks to recent advancements in LIDAR technology, it is now a lot easier to obtain detailed information about forests, with an accuracy that can compete with the conventional methods. The sensors can be worn by humans, put on vehicles and can recently even be put on drones. The resulting 3D point cloud data is then fed to an algorithm that extracts the wanted information, which means that software plays a vital part in the reliability of these methods.

This report investigates the possibility to automatically classify stem points in high-resolution, low-height, airborne laser scanned data (ALS), collected by a helicopter. A similar algorithm could potentially be used for data collected by drones. Stem detection is often a first step in the information extraction procedure, since other properties such as diameter, stem form and total wood volume could be obtained from it. It can also be a way to combat shaky data. The fact that stems are approximately smooth could be utilized as a baseline for stabilizing algorithms.

The focus of this investigation is limited to tall trees (>18 m) with clear stems, i.e. stems that are not dominated by thick branches. The reason behind this is that branches blocks the incoming laser signal resulting in fewer actual stem hits, and it could be hard even for a human to see which data points belongs to a stem and which are branch hits. This algorithm is therefore actively trying to avoid "stem" hits found in trees with thick branches since they are unreliable.

2 Materials and Methods

2.1 Data

The data was collected in december 2018 at test site Remningstorp in south-western Sweden. The LIDAR sensor used, Riegl LMS Q680i, was mounted on a helicopter with an 15° angle along the flight direction. Traditionally the sensors are mounted with no angle, i.e. pointing straight down, but having the sensor angled made it possible to get clearer stem hits. The flight route was a square pattern with some overlap, and can be seen in figure 1. The overlapped data were used during the development of the algorithm.

Each point in the raw point cloud data contains information about GPS-time, intensity and ground classification. An example forest area from the raw data without points classified as ground can be seen in figure 2.

A summary from the data acquisition can be seen in table 1 below.

Table 1: Summary from the data acquisition.

Parameter	Value
Flight altitude (above ground)	150 m / 70 m
Beam divergence	$\leq$ 0.5 mrad
Pulse repetition frequency	200 kHz
Scan frequency	90 Hz
Wave length	1550 nm
Pulse length	0.9 m
Scan type	Rotating polygon mirror
Scan width (across flight dir.)	$\pm$ 30°
Scan angle (along flight dir.)	15°
Average point density, all returns	1420 m^{-2}
Average point density, first returns	867 m^{-2}

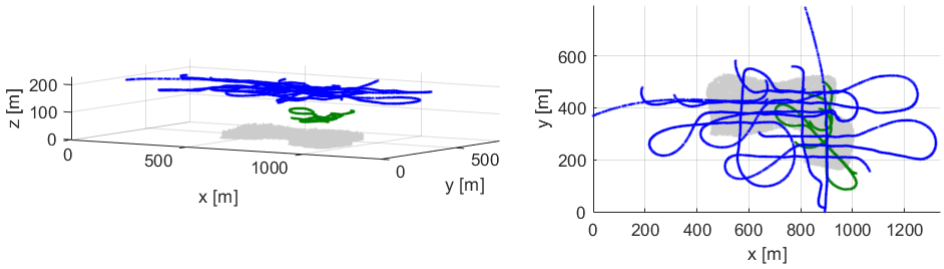

Figure 1: The complete scanned forest area (data from all flight lines merged) and the flight path of the helicopter during the data acquisition. The scanned forest area is seen in grey, while the blue and green lines are the flight paths from 150 m and 70 m respectively.

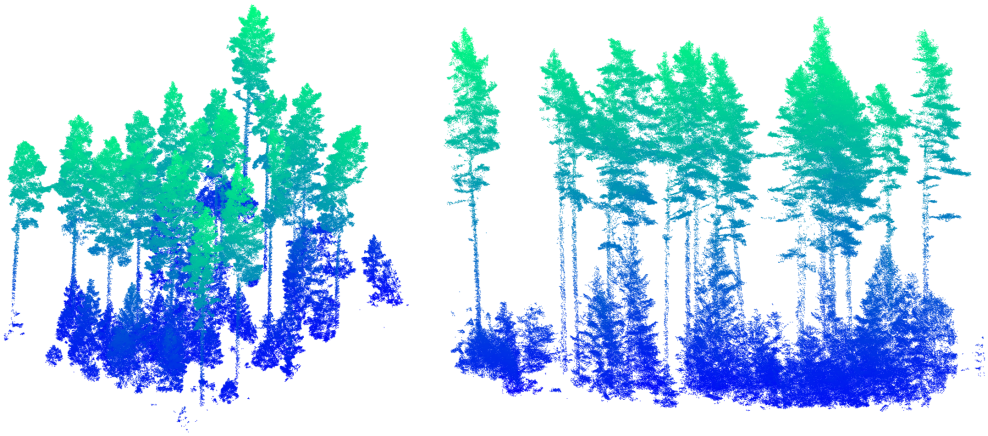

Figure 2: A small example area of the forest where data points classified as ground are omitted.

2.2 Software

The algorithm was implemented in MATLAB. Two functions from the standard library "computer vision toolbox" [1] was used, namely `pcdenoise` [2] and `pcsegment` [3]. The first, `pcdenoise`, were slightly modified and the

changes are described in section 2.3.4. Furthermore, a non-standard library "Digital Forestry Toolbox" [4] were used to load the LAS-files into MATLAB.

2.3 Tree Stem Detection Algorithm

In this section, the algorithm for stem detection will be explained in detail. Each part of the algorithm will be visualized using appropriate examples for the circumstance.

2.3.1 Overview

An overview of the algorithm can be seen in algorithm 1.

Algorithm 1 Stem detection algorithm

```
1: Pre-process data set
2: for each direction vector in direction vector set do
3:     Rotate data set to align with direction vector
4:     Project data set to the horizontal plane
5:     Remove data points that are far away from their neighbours
6:     Cluster based on distance
7:     for each cluster in cluster set do
8:         Revert cluster to 3D space
9:         if cluster properties not equals stem properties then
10:            Remove cluster
11:    Classify data points in saved clusters as stem
12: Add all classified points from all directions
13: Cluster classified points based on distance
14: for each cluster in cluster set do
15:    Perform RANSAC
```

2.3.2 Pre-Processing Data

In the pre-processing phase, the fact that stem hits usually have a higher intensity than branch hits is utilized. Every data point with an intensity

below a threshold is removed along with data points classified as ground.

In figure 3 it is shown how two example trees were changed from these filters. The intensity threshold was set to 700. As seen in the figure, many non-stem hits were removed. Unfortunately, some stem points were also removed. This is handled in a later stage of the algorithm.

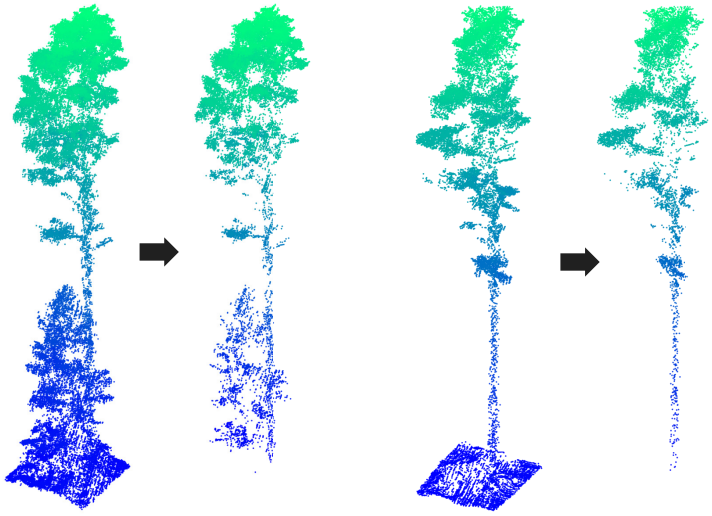

Figure 3: Two examples of the effect from the pre-processing phase, where ground and low intensity points (below 700) were removed. As can be seen in the example to the left, many data points were removed from the small spruce. However, a few stem hits were also removed.

2.3.3 Direction Vectors

This section summarises the loop beginning at step 2 in algorithm 1. A more detailed description of the procedures encapsulated by the loop are described in section 2.3.4 and 2.3.5.

A core principle of the algorithm is the ability to detect tree stems standing at an angle in addition with the purely vertical stems. This is controlled by the use of direction vectors. As a beginning step, the complete data set is rotated to align with the first direction vector in the direction vector set. See figure 4 as an example. The algorithm proceeds to find stem points in the z-direction (vertically), and then moves on to the rotate the data set along

the next direction vector. The stem points found from the next direction are merged with the previous direction, and the algorithm proceeds until all the direction vectors have been used. In the end of this loop, if a data point has been labeled as a stem point in any direction, it will be labeled in the result. The sequential steps after the loop are described later in section 2.3.6.

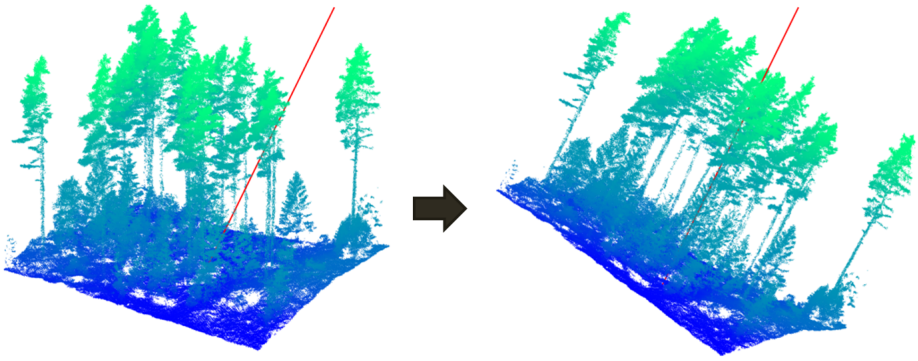

Figure 4: Forest area aligning with a direction vector. The vector is illustrated as a red line.

To avoid missing some angled trees, it is a good idea to have the directions be separated by a constant angle with equal spacing. This relates to the problem of spacing nodes equidistantly on a sphere. The algorithm used to generate direction vectors in this project can be found in [5]. An example of direction vectors can be seen in figure 5. In this example, the maximum angle is 15 degrees with 13 vectors in total. Typically it is wise to have >100 vectors with >20 degrees maximum angle to make sure that no trees are missed.

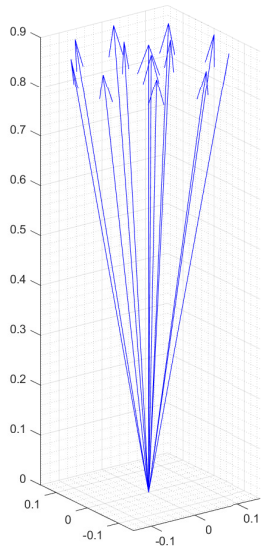

Figure 5: Example of 13 direction vectors, generated by the algorithm described in [5].

2.3.4 Projection and Denoise

This section corresponds to step 4-5 in algorithm 1. When the data set has been rotated to align with a direction vector, the next step of the algorithm is to project the data points onto the horizontal plane. The idea is that tall stems aligned with the z-direction will be dense areas in the projected 2D space. Right after the projection, the space is filtered by a distance-to-k-nearest-neighbours algorithm [2]. In short, this algorithm loops through every point in the data set and calculates the mean distance to the k nearest neighbours. If this distance is above a certain threshold, the point is removed. In some cases, this threshold is derived from the number of standard deviations from the mean of each point's distance. However, in order for the stem detection algorithm to work well with a variety of directions, the threshold is set as an absolute value. This is how the `pcdenoise` function was changed (mentioned in section 2.2). Figure 6 below shows an example of the procedure described in this section. The distance threshold was set to be 2 mm, and k was set to 40.

Figure 6: An example of projection and denoising of a small forest area. The 3D point cloud is projected onto the horizontal plane, and the 2D space is denoised.

2.3.5 Clustering and Property Analysis

This section corresponds to step 6-12 in algorithm 1. The denoised 2D projection is segmented into small clusters by a method based on distance [3]. The data set is then converted back to 3D.

An example of a segmented tree can be viewed in figure 7. In the example the segmentation distance threshold was set to 5 cm.

Each cluster is then analyzed. The procedure is composed of several independent conditions, each of which makes sure that the current cluster fulfills the expected properties of a tree stem. The first two conditions are:
Too few data points - If the cluster has fewer than n data points, the cluster is considered unreliable and is discarded.
Too short - If the cluster is shorter in the z-direction than certain threshold, the cluster is discarded.

From here the top of the cluster is cut off by a certain percentage, given by another parameter. The idea is that the top of the clusters almost exclusively contains branch hits, which are undesirable in this context. Right after the cut, another check for **too few data points** is done to quickly get rid of unreliable cases again.

At this stage, it is wise to get back some of the stem points that was removed by the intensity filter in the pre-processing phase. In order to achieve this, a center of the cluster is estimated. This is done by using the RANSAC

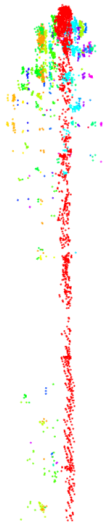

Figure 7: Example of the segmentation of a tree. The data points of the same color are members of the same cluster.

algorithm [6]. RANSAC is used two times in total in the stem detection algorithm (although in different manners) and this one should not be confused with step 15 in algorithm 1. The cluster is projected onto the horizontal plane, and a circle with a certain radius (given by a parameter) is created. The algorithm randomly centers the circle at different points, and the point where the circle has the most points inside it is saved. This algorithm is used to prevent outliers from having an impact.

When the center is estimated, the cluster is converted back to 3D again, and a vertical cylinder is created around the cluster. The cylinder is centered at the estimated center point, and its radius is given by a parameter. Every point inside the cylinder is then added as a part of the cluster. An example of this step where cluster was cut by 35% and the radius of the cylinder was set to 35 cm is shown in figure 8.

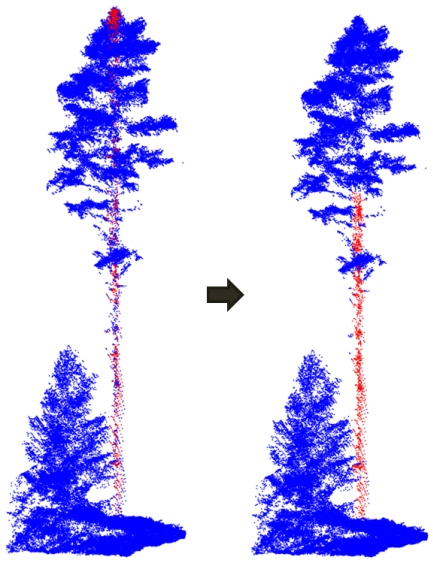

Figure 8: An example of points getting removed from the top of the cluster, and points inside the created cylinder added. The cluster was cut by 35% and the radius of the cylinder was set to 35 cm.

This new cluster is then checked for the following condition:
Large discontinuities in cluster - If the cluster has too many and/or too large discontinuities in the z-direction (vertical direction), it is discarded.

In order to have a fair measure of the discontinuities, a discontinuity index is calculated for each cluster. This is index is given by

$$I_d = \sum_{i=0}^{N-1} \frac{(z_i^d)^2}{h}$$

where h is the height of the cluster. The definition of z_i^d is given by

$$z_i^d \equiv \begin{cases} \frac{z_{i+1}-z_i}{\epsilon}, & \text{if } z_{i+1} - z_i > \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where ϵ is a height threshold and z_i is i:th element in the vector $\mathbf{z}$ containing N values, ordered from lowest to highest. In this project, ϵ was set to 25 cm.

The basis of this condition is the assumption that actual stems have no discontinuities in the z-direction. Since the data is discrete, a height threshold is set for the minimum distance that counts as an discontinuity. This value is given by a parameter and is the same for all clusters. All distances over this threshold are normalized and squared in order to punish one large jump more than two small ones. All these are then summed together, and normalized to the height of the (cut) cluster to give an index describing the general discontinuity of the cluster. Figure 9 shows an illustration of h, ϵ and $z_{i+1} - z_i$ in an example cluster.

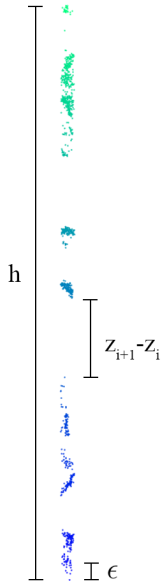

Figure 9: Illustration of h, ϵ and $z_{i+1} - z_i$ in an example cluster.

The last condition that is checked is:
No clear space outside cluster - If the cluster has little or no space outside it, it is discarded.

The cluster is split into small cylinders, each being of height 10 cm with the same radius as the cluster. For each cylinder, another cylinder is created with twice the radius. The amount of data points inside the two cylinders are then compared. If the small cylinder contains more than 80% of the

data points, that part of the cluster is classified as clear. If the number of clear parts for the complete cluster is below a certain threshold (given by a parameter), the cluster is discarded.

This condition is the main counter to trees with thick branches. An example of a cluster that was removed by this condition can be seen in figure 10. The threshold used was 15. The figure shows a clear case where the cluster is not an actual stem and should be removed.

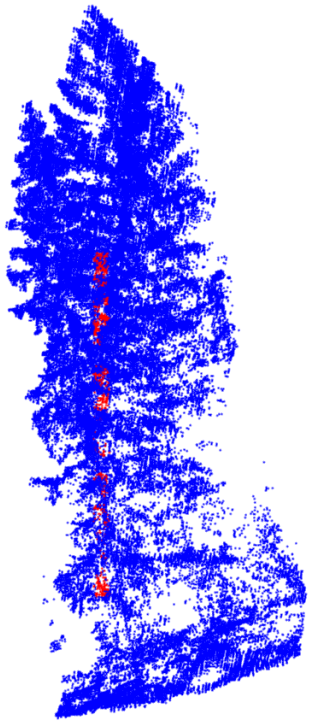

Figure 10: Example of a cluster that was removed from having little to no clear space around it.

If the cluster has survived all the conditions so far, the cluster's data points are classified as stem points.

Once all clusters has been checked for the conditions, the algorithm moves on to the next direction vector, and the procedure from section 2.3.4 is repeated. The results from each direction are continually merged together until

all directions are done.

An example of a forest area after classifications in 113 angles can be seen in figure 11. Note the failed classification at the bottom and top of the stems. The bottom are due to close spruce trees and bushes, while the cause at the top is the crowns.

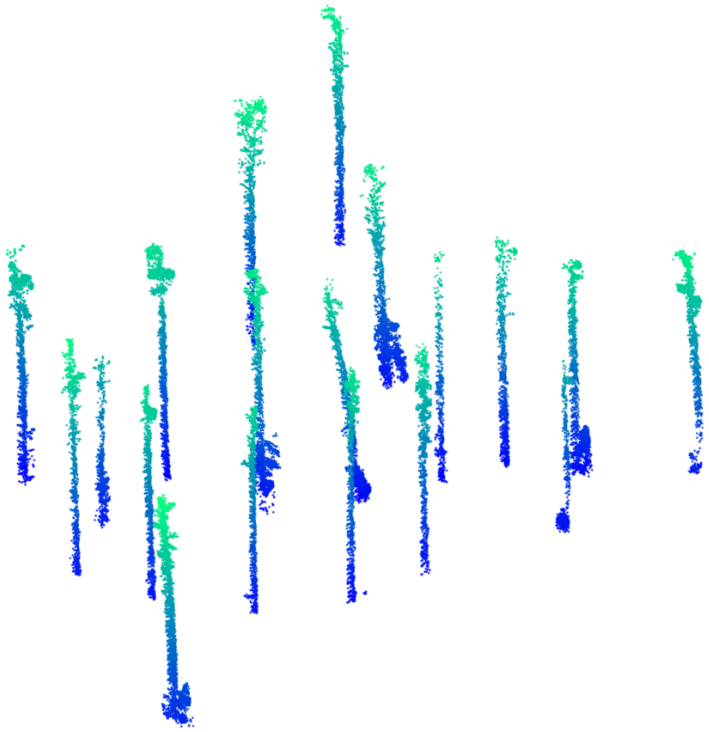

Figure 11: The outcome after merging the results from 113 directions (maximum angle 15°).

2.3.6 Clustering and RANSAC

This section corresponds to step 13-15 in algorithm 1. The resulting classified stem points from all directions are then segmented based on distance in order to isolate each cluster. The RANSAC algorithm [6] is then performed on each one. This time it is done by creating random cylinders over the cluster, and saving the one which have the largest amount of data points inside it. This

will give an estimation of the center and a direction of the stem. A cylinder is then created with the a radius that is given by a parameter, and all points outside it are removed from the classification. This is the last step of the algorithm, and some results can be seen in the next section.

3 Results

The only presentable results are in form of visualization. Therefore, this section consists of a mixture of figures of typical forest areas, as well as some dense areas to check where the limit of the algorithm is. The parameters used to produce the results below are summarized in table 2.

Table 2: Summary of the parameters used in the results.

Parameter	Value
Intensity threshold	700
Denoise distance threshold	2 mm
Denoise number of neighbours	40
Data point threshold	20
Tree height threshold	18 m
Tree top cut percentage	35%
Added points cylinder radius	35 cm
Discontinuity index	1.4
Discontinuity height threshold (ϵ)	25 cm
Amount of clear cluster parts	15
RANSAC cylinder radius	25 cm

The result following the example area from figure 11 can be seen in figure 12, and a mixture of different forest areas can be seen in figure 13.

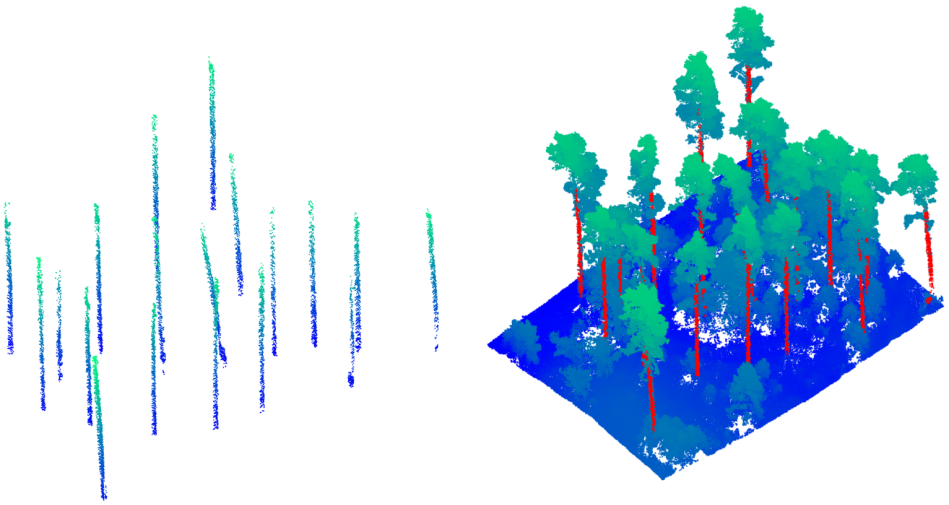

Figure 12: To the left are the points classified as stem by themselves, and to the right are the classified points marked in red.

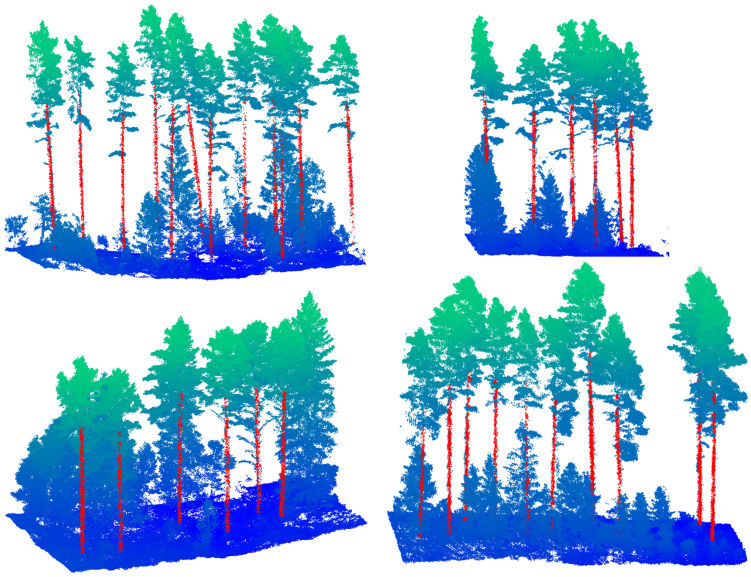

Figure 13: Results from the algorithm in four example forest areas. Points classified as stem are marked in red.

Following below are examples of thicker forest areas, seen in figure 14-16. The density of the forest were so high that a drastically lower intensity threshold (100) had to be used in order to get good results. When the forest density is higher both the total amount of stem points and the intensity of the existing stem points are lowered. This makes the stems harder to detect, and the risk of classifying non-stems is increased.

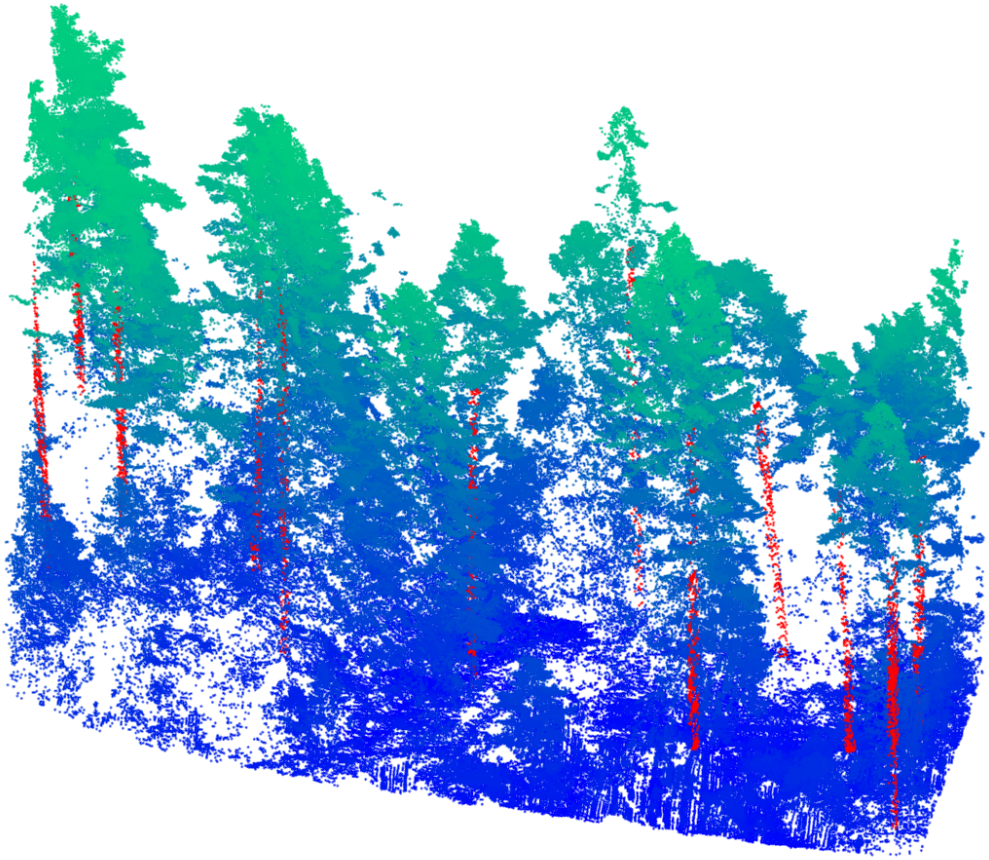

Figure 14: Dense forest area. Points classified as stem are marked in red. It is hard to see how well the algorithm performs based on a 2D image when the forest is this dense, but all clear stems seems to have been found.

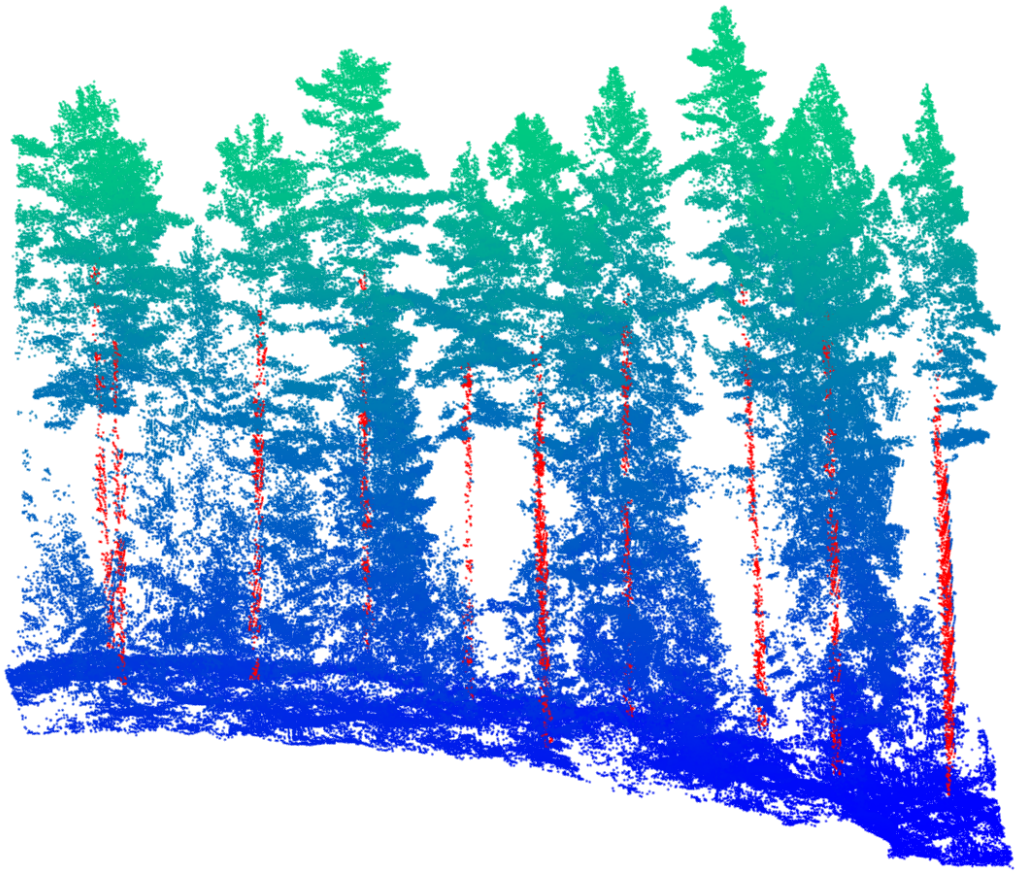

Figure 15: Dense forest area. Points classified as stem are marked in red. Note how few data points the stem in the middle has compared to the far right stem, for example. These are cases where the algorithm can have troubles, however it performed well here.

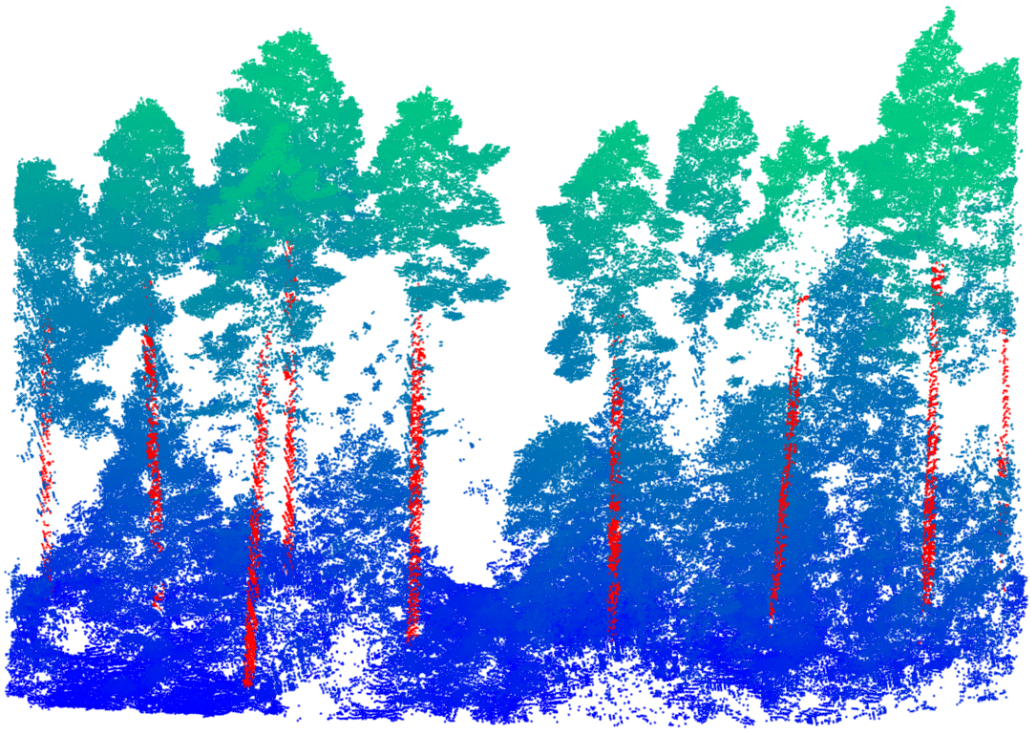

Figure 16: Dense forest area. Points classified as stem are marked in red. Slightly to the right of the middle we see one tree that was wrongly classified. As it turns out, it has very few data points which causes the stem to have large discontinuities.

4 Conclusions

Since there is no robust objective way at this moment to verify the accuracy of the results, it is hard to say how well the algorithm performs. The only available tool is visual examination. Judging from the figures in section 3, most of the tree stems seems to be correctly classified. There is one case where a complete stem is classified as non-stem (figure 16), which arises because the stem has very few data points to begin with.

One way to possibly deal with this problem is to cluster the trees as a beginning step. Even if some stems are hard to find, the tree belonging to the stem could be easier to find thanks to the crown. This segmentation

could therefore yield the information that there is a stem to be found in an area. Instead of removing every cluster that do not meet the properties of a stem (the procedure described in section 2.3.5), the best cluster in the area could be saved.

An important aspect the current algorithm lacks is a dynamic calculation of the tree radii. There has been limited time to investigate reliable methods to calculate this, however, is has been concluded that an estimation based on the mean distance from the center is unreliable because of noisy data and inconsistencies in amount/location of data points. A proper method would probably have to take the roundness of a stem into account to know which parts of the cluster actually is a real stem.

In conclusion, it could be argued that the "project data set $\rightarrow$ cluster analysis" approach is a useful way to classify stems in this type of data. The algorithm shows promising results in finding stems and avoiding miss-classification, even in dense forest areas. The algorithm leaves a lot of room for continually building upon and improving, for example by adding more cluster conditions.

References

[1] MATLAB computer vision toolbox. Available at: `https://se.mathworks.com/products/computer-vision.html` (Accessed 11 June 2019)

[2] MATLAB pcdenoise. Available at: `https://se.mathworks.com/help/vision/ref/pcdenoise.html` (Accessed 11 June 2019)

[3] MATLAB pcsegment. Available at: `https://se.mathworks.com/help/vision/ref/pcsegdist.html` (Accessed 11 June 2019)

[4] Digital Forestry Toolbox. Available at: `https://mparkan.github.io/Digital-Forestry-Toolbox/` (Accessed 11 June 2019)

[5] Deserno, Markus. 2004. How to generate equidistributed points on the surface of a sphere. Max-Planck-Institut für Polymerforschung.

[6] Fisher, M., Bolles, R.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. of the ACM* 24(6), 381 - 395 (1981)