Report - Tree Count

Mario van Rooij (11607521), Mehmet Ege Arkin (11604808), Ramon Bussing (2559766)
Group 22

I. Introduction

Since climate change is currently one of the larger global challenges that humanity is facing, tools to monitor variables that contribute to climate change are therefore of uttermost importance. Since trees play a role in climate change, the goal of this project is to count the number of trees in the Netherlands based on the Actueel Hoogtebestand Nederland (AHN3) data set. The AHN3 data set contains point-cloud lidar data from aerial scanning with a plane or helicopter of the entirety of the Netherlands. Point-cloud data consists of points in x-y-z space where the laser of the Lidar bounced of, and some properties per point such as laser bounces and incomplete pre-labeled point categories. The points x, y and z are Rijksdriehoeks coordinates, with z being the altitude. This dataset is very large so it is also important to use methods to speed up the data processing. Ultimately we will use a deep learning model to detect trees in this data, and with this count the trees in the Netherlands.

II. Methods and pipeline

A. Data

As stated before, the data consists of pointclouds, of which a 3d map of Amsterdam and surroundings can be looked at in figure 4 or on the interactive website [15]. The points contain a number of incomplete classifications which can be found on the ArcGis website [11].

Besides the pointclouds, the dataset also contains a number of other parameters. All the parameters per point are: 'X', 'Y', 'Z', 'intensity', 'return_number', 'number_of_returns', 'scan_direction_flag', 'edge_of_flight_line', 'classification', 'synthetic', 'key_point', 'withheld', 'scan_angle_rank', 'user_data', 'point_source_id', 'gps_time'.

Parameters to note are X, Y, Z , which are used in the algorithm. X, Y, and Z are indicators of the x- ,y-, and z-coordinates in the Rijksdriehoeks coordinates. The Rijksdriehoeks coordinates represent the netherlands in a flat 2d surface, where the x and y are given in meters. The number of returns is the number of layers returned back to the lidar equipment from a single laser pulse. And the return number corresponds to the layer that specific point belongs to. This is an important feature for trees since they will reflect the laser on different levels. The documentation of the data-sets tells us that a classification of 5 means high vegetation, however it was observed that in the "Veluwe" (an area in the Netherlands with a wide variety of vegetation) in the AHN3 dataset, no points were labeled as such, so it could not be used to our advantage.

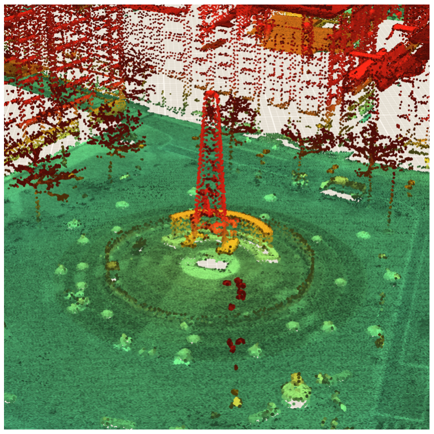
Fig. 1: The monument on the Dam square in AHN3, visualized by ArcGIS [3]

1) AHN2 and AHN3 Datasets: The AHN2 and AHN3 are point cloud datasets that represent a height mapping of the Netherlands. The dataset consists of compressed LAZ files that have to be uncompressed as workable LAS files which contain point cloud data. This data is a group of points that individually represent (x, y, z) coordinates. The AHN2 dataset, spanning 5 years of data between 2007-2012, is a height map of the Netherlands with an average of 8 points in each square meter. It consists of LAZ files (size of 1.6 TB) which are divided over 143 tiles representing the surface area of the Netherlands. The AHN3 dataset is a newer version which spans 2013-2018. The compressed dataset has a size of 2.4 TB (1375 files in total).

B. Pipeline

The data is given in .LAZ format and has to be converted to LAS files so it can be used in the classification algorithm, because the algorithm chosen unfortunately cannot read LAZ files. This was done by using LAZmerge, which is part of the LAStools toolkit [8]. This tools allows to decompress the LAZ to LAS files ($6-7\times$ larger), while at the same time splitting up the files into chunks small enough to be processed in the next step.

The next step consists of classifying points as either part of a tree, or not part of a tree. This was done using a variation of a convolutional network, which is able to process point clouds called PointCNN [10].

After this, we use a local maximum algorithm on the points classified to be part of a tree to find where the trees are located. First we filter the points on two criteria. They have to be classified as part of a tree as well as the number of returns being higher than one, which was empricially seen to be true for trees. The filtered data is then streamed using Laspy [13] into a function using the SciPy library [4] called $KDTree.queryball\_tree$ [12], which returns the neighbours within a pointcloud that reside within a certain radius of that point. After this for all those new clouds with neighbours, the local maximum is calculated, and if that local maximum has more than 100 points in a certain neighbourhood (chosen to be $5m$ in our project) it was classified as the centre of a tree. One difficulty was encountered here for the AHN2 dataset, which does not contain a number of returns in the same way AHN3 does. This caused us to run the algorithm on all points which where classified as part of a tree, which significantly, and falsely increases the number of trees.

The tree heights were obtained by looking in the X, Y-neighbourhood of a tree and finding the lowest point within 2 meters. This method proved not to be perfect however since not all trees were perfectly classified and a tree may stand on a hill, or be in valley, which made this method flawed.

C. Clusters

The resulting cluster configurations on the cloud is displayed in 2. Lasmerge is part of the LAStools library. The source of this project can be cloned [9] and compiled, which results in a executable lasmerge file. For the PointCNN to work the following command has to be ran in the Databricks environment specified in Cluster 1:

```
conda install --revision 0
&& conda install -c esri arcgis_learn=1.8.5
```

For Cluster 2 the only installation requiring special attention is Laspy, which has to be installed using optional dependencies

```
pip install laspy[lazrs,laszip]
```

In cluster 1, we set up the data splitting such that Spark could be used by PointCNN, which required a single folder with in it the LAS files. Due to budget constraints we could not run the PointCNN on all the datasets, which would require around 27.000 GPU hours.

As for cluster 2, we managed to write a Spark script that parallelizes the writing of the trees to text-files.

In order to segment trees from their surrounding we opted to use pointCNN [10], as this is currently one of the best performing deep learning models that is able to work with point-cloud data. Key to the good performance of pointCNN is accounting for the irregular and unordered points in the point-cloud data that we have, unlike PointNet++ [14]. Prove of concept is given by ArcGIS, one of the larger geographic information systems, that uses pointCNN to segment trees from their surroundings in point-cloud data [18].

III. Goals

The basic goal of the project is to count the number of trees in the Netherlands and make a density plot of the number of trees per area. The map will be hoverable, telling a person what the density is of the current location of the mouse. If we manage tree classification, we create a similar heat-map where three colors show the three most occurring trees.

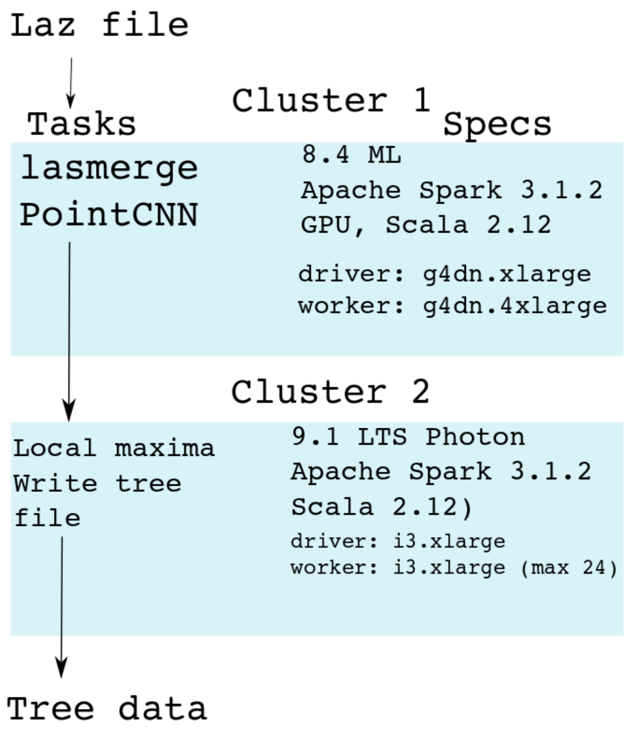

Fig. 2: The configurations of the clusters that were used in this project. Both clusters are created in the Databricks environment [17]. A complete description on how to setup Cluster 1 is given at the start of the results section. A LAZ file is unzipped, split, and classified in Cluster 1 after which the local maxima algorithm is performed by cluster 2, which also writes the data (x, y, z coordinated of the trees) to the S3 server.

If these goals are achieved, we would like to work on showing the time dependence by using the older dataset AHN2. This would be visible in a similar heat-map, where tree increase would be shown in green and tree decrease in red.

In addition to this, an example image of segmented trees (figure 4) such as created by Arno Timmer would be highly illustrative of our achievement [2].

IV. Results

We first show how the local maxima algorithm classifies a single chunk of a las file. The local maximum algorithm is only given the points that were classified by the PointCNN as part of a tree. The local maximum algorithm then looks for local maxima in those points, which cannot be closer than α meters from each other, where α is a tunable parameter. A sample result is shown in Figure 3, 4 and 5. It can be seen that inside the clusters the local maxima are classified as the tree centre. It can also be seen that on the right side of the picture some points where only a single point is classified to be part of a tree are classified as a local tree max (thus being counted towards the tree count).

For part of Amsterdam we made a histogram of the tree heights. This is displayed in Figure 6. There were a lot of mistakes in this finding however, since some buildings, up to

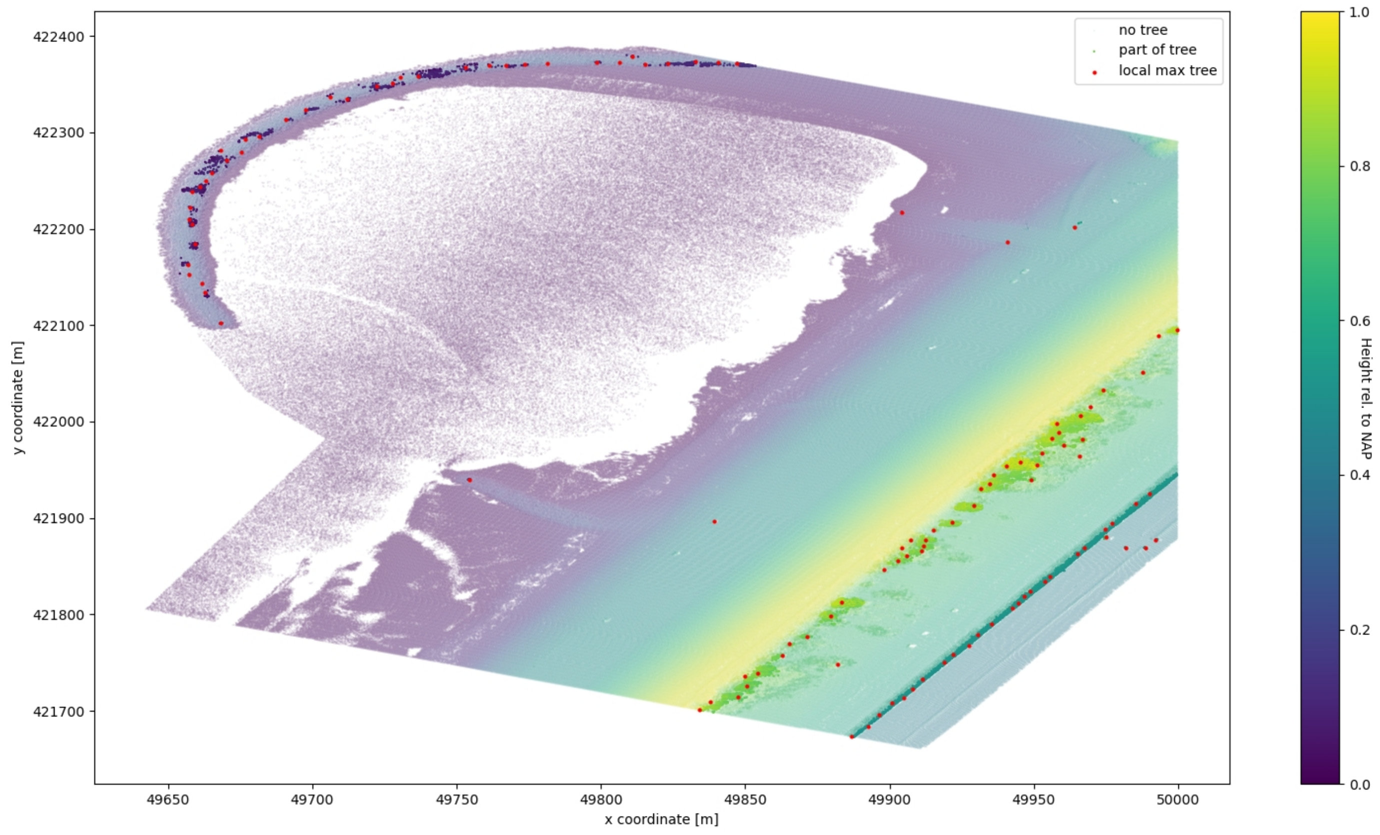

Fig. 3: Spatial plot of a single chunk of a las file (1.000.000 points). The points not belonging to a tree are plotted with a high transparency, while the points belonging to trees have a lower transparency. Lastly, the points which are classified to be the local maximum of a tree cluster are shown in red. The color of the tree and non-tree points are based on the height, where 0 indicates NAP [7].

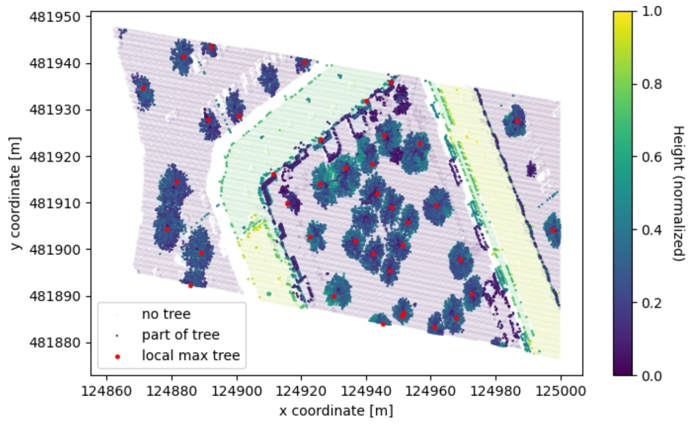

Fig. 4: Spatial plot of a single chunk of a las file (100.000 points). The points not belonging to a tree are plotted with a high transparency, while the points belonging to trees have a lower transparency. Lastly, the points which are classified to be the local maximum of a tree cluster are shown in red. The color of the tree and non-tree points are based on the height, where 0 indicates the lowest of the chunk and 1 the highest.

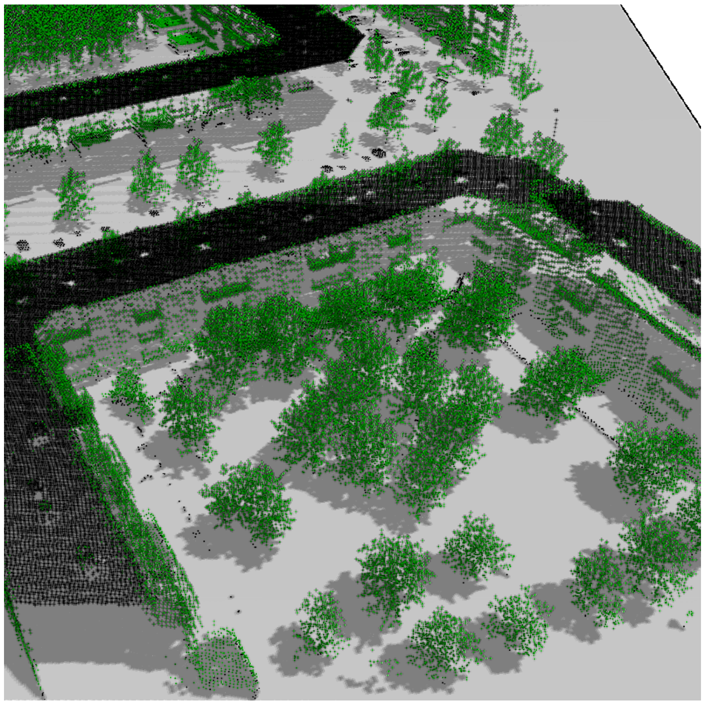

Fig. 5: 3d visualisation of the chunk in figure 4 after point categorisation by pointCNN. Non-tree points are displayed black and tree points are green.

50 m were classified to be a tree. The cutoff was done at 35 m.

We also looked at the sensitivity of our predictions as compared to the municipality data of Amsterdam. The sensitivity is defined as (To BE DEFINED), and can be seen in table

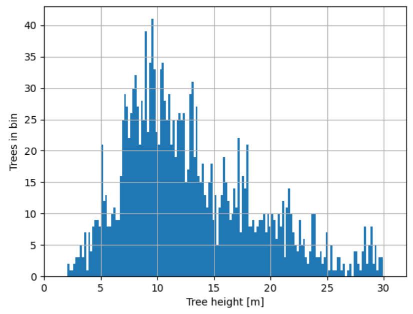

Fig. 6: Histogram of the tree heights in a certain part of Amsterdam. The average height is 12.7 m with a standard deviation of 5.8 m.

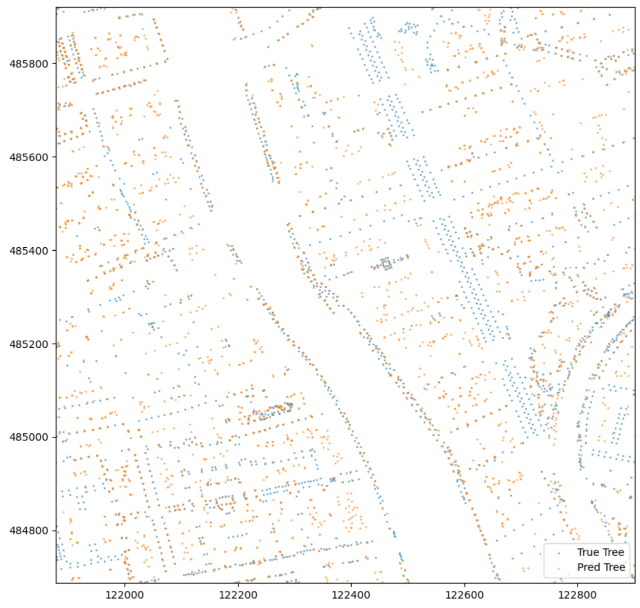

Fig. 7: Spatial plot using Rijksdriehoekcoordinates, showing the spots where the municipality has labeled trees, and where our algorithm has predicted trees. This shows the local maximum algorithm did not detect multiple rows of trees. It was found these trees were planted after the pointcloud was obtained. Not all trees have been labeled by the municipality of Amsterdam.

(TABLE WITH SENSITIVITY). We investigated parts of the map and saw in particular that at the Wibautstraat, there was a strip of trees, displayed in Figure 7, left undetected by our local maximum algorithm. We then investigated when the trees were planted, which was only a couple of years before the dataset had been taken [1], meaning that the clusters of points these trees produced were not large enough to be classified as a tree. Our algorithm also shows an excess of prediction in some places. This may partially be because of errors, but this is also caused by the fact that the municipality has not labeled all trees in the city.

V. Discussion

Starting this project, we aimed to classify different tree species as well as different datasets in order to make a time dependent result. This could not be achieved in the available timeframe, as the basic classification brought with it unexpected problems. Especially installing the packages took a large amount of time due to compatibility issues. Besides this, the classification using the PointCNN is relatively slow. This is also caused by the way the PointCNN is implemented in ArcGis. When given a set of LAS files the PointCNN converts the files to H5 format, after which it performs the classification and writes these files back to LAS files again. This causes a great inefficiency in our pipeline, but could not be avoided (using this method), because there are currently no tools available for converting LAZ files to H5 files. The H5 files themselves are a factor $\approx 10\times$ bigger than the LAS files (150 TB of intermediate files), which might be the reason why the PointCNN runs slow. Another solution to this would be to not use PointCNN, but a regular convolutional neural network [16], which could work on pictures produced by the point cloud data, by making a 2D picture where height would determine color, and using the same ground truth as suggested before.

For the classification of points (tree or non-tree) we used a pre-trained model provided by ArcGis. We first opted to train our own model, but the ArcGis model was sufficient for our task (accuracy of 97.5% [1]). Had we not had this method, we would have trained our own model using a ground truth. In our case, this would have been the tree map of the municipality of Amsterdam. This map provides the location, height, and radius of a lot of trees. By then mapping our point-cloud data onto this map (meaning we see what points are part of a tree in the point-cloud), we could then train the model. This would take us about 10-15 hours depending on the accuracy on a GeForce RTX 3070 GPU. We did not test it on a cluster (m60 Tesla) but it would likely be slower than this.

As for errors in the project, the main one which was observed was introduced by the local maxima algorithm, which would sometimes classify a single point with the high vegetation classification to be a tree. This point (when far enough from other high vegetation point) is a trivial local maximum. It could also be said, however, that this error is introduced by the PointCNN since these points are wrongly classified. One fix for this would be to check around every cluster if there are enough points in the neighbourhood (e.g. > 10). Also, the β parameter increases our model sensitivity but increases the number of false positives as well. We had to guess a value which seemed reasonable, but this should definitely be looked further into.

Another possible error could come from the fact that the AHN2 dataset has a lower point density than the AHN3 dataset. First of all, the classification will be different from the AHN3 dataset, and also the minimum number of points needed to classify something as a tree changes. We accounted for this by requiring the cluster to be smaller exactly by a factor of the reduction of the point density (e.g. if a map with point density $100m^{-2}$ requires minimum of 30 points as a neighbour to be a tree, a map with point density $50m^{-2}$ requires only 15 points as neighbours to be a tree). Another problem is that in AHN3 we could remove the points which had only one return, whereas in AHN2 this was not possible. This increased the

[1]Click here to open the news article talking about planting the trees

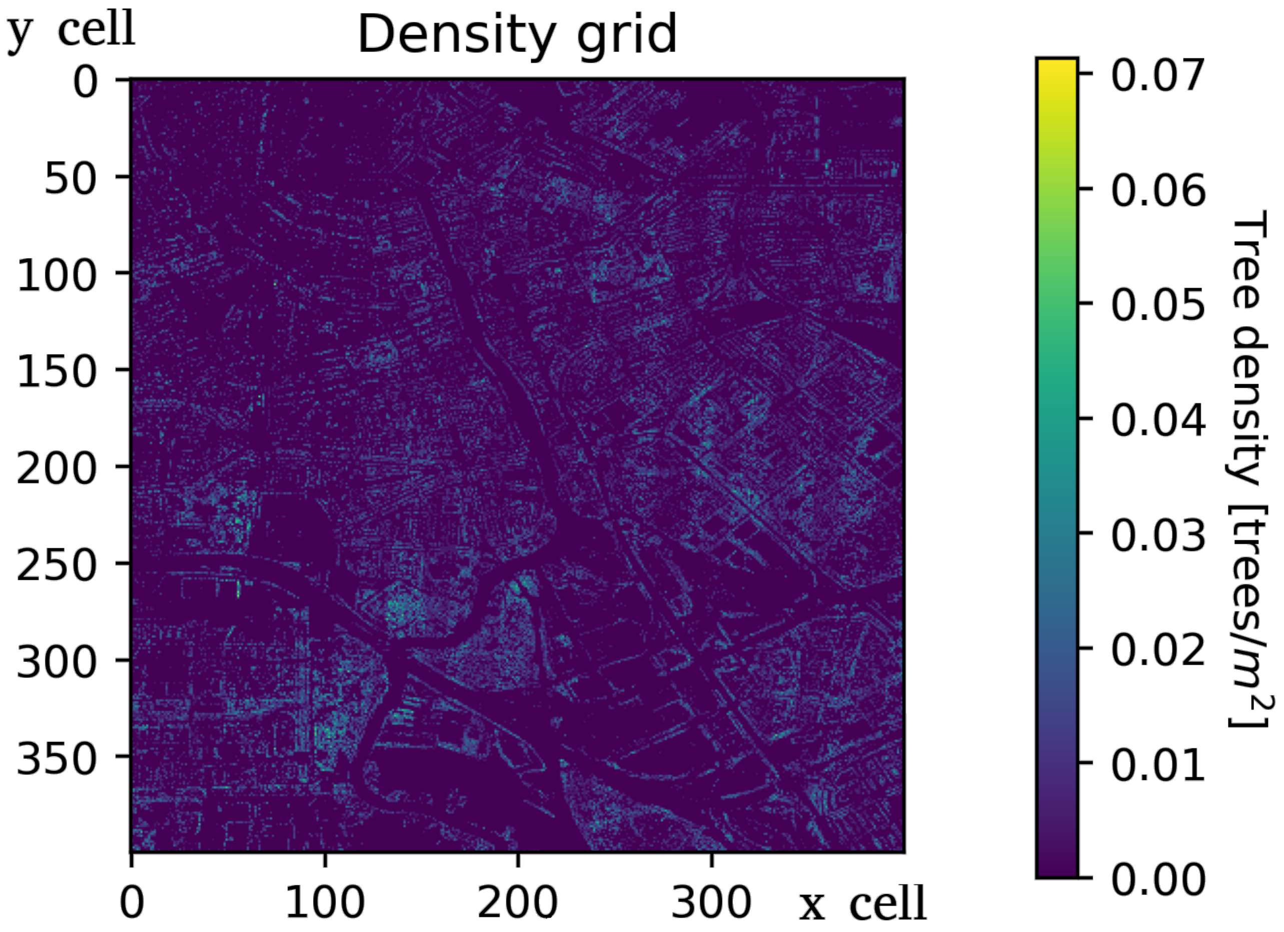

Fig. 8: Spatial plot using the grid cell coordinate system displaying the tree density for different parts of Amsterdam. Some area contain about 7 trees per $100m^{-2}$, whereas others contain 0. The cell number corresponds linearly with the Rijksdriehoekscoordinates of the LAZ file (e.g. x cell 0 is the minimum x value in that LAZ file and x cell max the maximum).

number of points on which our local algorithm was ran on, increasing the chance for a local maximum to be a tree since it has more neighbours when less filtering is applied.

A future project that does not use a deep learning model could use the litree filter [6] from PDAL [5]. This is a model which requires the points to have defined the height above the ground per point. This can be obtained by looking at clusters of points in small areas where the one with the most points is likely the ground height. The average of this large cluster can then be added to the z attribute of the small area of points, which would result in the required height above ground information. In areas where there are hills extra care has to be taken when performing this operation since there is only a ground level on a local level, and so there will be a high variance in height between the biggest clusters.

VI. Conclusion

We conclude this project by classifying points in the AHN2 and AHN3 pointcloud datasets (which are stored on an Amazon S3) of Amsterdam as either ”tree” or ”no tree” using pointCNN, and running a local maxima algorithm to recognize individual trees. We run the algorithms on two Amazon EC2 clusters due to package incompatibility. The data is individual trees with coordinates, which is used to create a density map of Amsterdam (figure 8). Looking at the average height of the trees in (25_GN1) parts of Amsterdam we get a distribution like figure 6. The sensitivity of our model changes depending on β and increases when we increase β as expected, but this results in an increase in false positives. Hence we estimated β to be (BETA VALUE), but this parameter has to be tuned in future projects. Our goal to identify the species of trees and count the number of trees in the Netherlands has unfortunately not been achieved in the available timeframe. This is because of the inefficiency of pointCNN for large data projects.

References

[1] *ArcGis Tree Point Classification*. URL: https : / / www . arcgis . com / home / item . html ? id = 58d77b24469d4f30b5f68973deb65599.

[2] *Bomen herkennen in een 3D puntenwolk*. nl. URL: https: //nl.linkedin.com/pulse/bomen-herkennen-een-3d-puntenwolk-arno-timmer (visited on 10/10/2021).

[3] Bob Booth, Andy Mitchell, et al. *Getting started with ArcGIS*. 2001.

[4] Eli Bressert. “SciPy and NumPy: an overview for developers”. In: (2012).

[5] Howard Butler et al. "PDAL: An open source library for the processing and analysis of point clouds". In: *Computers & Geosciences* 148 (2021), p. 104680.

[6] "Filters.litree". In: *filters.litree - pdal.io* (). URL: https://pdal.io/stages/filters.litree.html.

[7] *Geodetische Infrastructuur en Referentiesystemen*. Oct. 2021. URL: https://www.ncgeo.nl/index.php?option=com_k2&view=itemlist&task=category&id=41:geodetische-infrastructuur-en-referentiesystemen&Itemid=168&lang=nl&limitstart=6.

[8] Ch Hug, P Krzystek, and W Fuchs. "Advanced lidar data processing with LasTools". In: *XXth ISPRS Congress*. 2004, pp. 12–23.

[9] "Lastools". In: *rapidlasso GmbH* (June 2021). URL: https://rapidlasso.com/lastools/.

[10] Yangyan Li et al. "Pointcnn: Convolution on x-transformed points". In: *Advances in neural information processing systems* 31 (2018), pp. 820–830.

[11] *Lidar point classification—Help — ArcGIS Desktop*. URL: https://desktop.arcgis.com/en/arcmap/10.3/manage-data/las-dataset/lidar-point-classification.htm (visited on 10/10/2021).

[12] Songrit Maneewongvatana and David M Mount. "Analysis of approximate nearest neighbor searching with clustered point sets". In: *Data Structures, Near Neighbor Searches, and Methodology* 59 (2002), pp. 105–123.

[13] Florent Poux. "How to automate LiDAR point cloud sub-sampling with Python". In: *Towards Data Science* (2020).

[14] Charles R Qi et al. "Pointnet: Deep learning on point sets for 3d classification and segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 652–660.

[15] *Scene Viewer*. URL: https://www.arcgis.com/home/webscene/viewer.html (visited on 10/10/2021).

[16] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[17] *The data and Ai Company*. Oct. 2021. URL: https://databricks.com/.

[18] Silas Toms and Dara O'Beirne. *ArcPy and ArcGIS*. Packt Publishing Ltd, 2017.