3DFin

Forest inventory

Carlos Cabo[1,2], Diego Laino[2], Covadonga Prendes[3], Celestino Ordonez[1], Stefan Doerr[2], Cristina Santin[2,4]

[1]Dept. of Mining Exploitation and Prospecting, University of Oviedo, Principality of Asturias, Spain; [2]Centre for Wildfire Research, Swansea University, Swansea, United Kingdom; [3]Forest and Wood Technology Research Centre Foundation (CETEMAS), Carbayin, Principality of Asturias, Spain; [4]Research Institute of Biodiversity (IMIB; CSIC), University of Oviedo, Principality of Asturias, Spain).

2022-23-12

Contents

Summary

The associated program (3DFIN) implements an algorithm to detect the trees present in a terrestrial 3D point cloud from a forest plot, and compute individual tree parameters: tree height, tree location, diameters along the stem (including DBH), and stem axis. This algorithm is mainly based on rules, although it uses clusterization in some stages.
It may be divided in three main steps:

1. Identification of stems among user-provided horizontal stripe.
2. Tree individualization based on point-to-stems distances.

3. Computation of stem diameter at different section heights.

Algorithm

The algorithm is an updated version of the one presented in (Cabo et al., 2018). It requires a height-normalized point cloud as input, as in Figure 1.

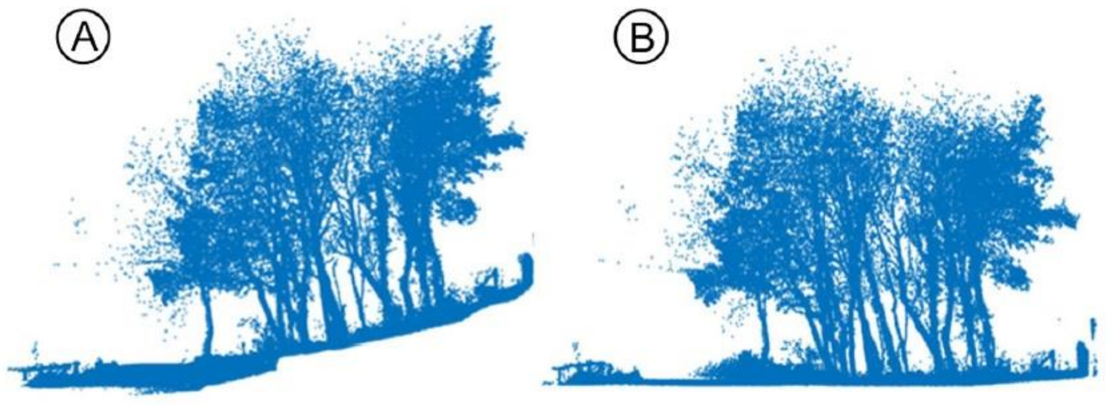

Figure 1: The algorithm requires height-normalized point clouds. A) Original point cloud. B) Height-normalized point cloud. From (Cabo et al., 2018).

1. Identification of stems among user-provided horizontal stripe

In this first step, the user selects a stripe, defined this as a subset of the original cloud delimited by a lower height ($Z_{h(low)}$) and an upper height ($Z_{h(high)}$) , which will narrow down a region where it is expected to only encounter stems. The points within the stripe will be voxelated and their verticality will be computed, based on fixed radius neighbourhoods. Then, they will be filtered based on their verticality value. After this, the remaining points will be clustered using the DBSCAN algorithm (Ester et al., 1996). These procedures will be repeated iteratively a user-defined number of times. At this stage, the potential stems are referred as ‘voxel groups’. Figure 2 illustrates this step of the algorithm.

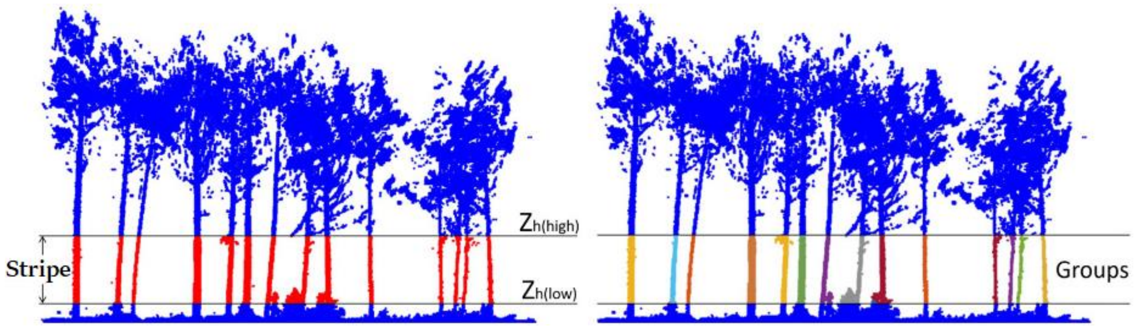

Figure 2: Stripe on the height-normalized point cloud, and candidate voxel groups. Points in the stripe in red, and voxel groups in random colours.

2. Tree individualization based on point-to-stems distances.

Once the voxel groups have been computed and properly peeled-off, they are isolated and enumerated, and then, their axes are identified using PCA (PCA1 direction). These axes will be henceforth considered as stem axes. This allows to group points based on their distance to those axes, thus assigning each point to a tree. This is illustrated in Figure 3.

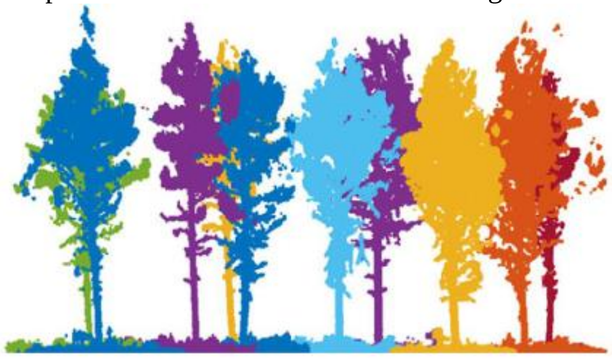

Figure 3: Isolated trees. Note that ground and understory points are assigned as well to the closest axis.

During this step of the algorithm the tree height is computed as well. For this, and, for each tree, the points that are under a certain distance to the stem axis are selected, voxelated again using a higher resolution and clustered with DBSCAN algorithm. From the points that belong to the main cluster (the one that englobes the tree), the highest point is selected, and its height is considered as the tree height. This allows to exclude from the search of the highest point those that could belong to other trees or any noise that happened to be above the tree whilst being scanned. Figure 4 illustrates this.

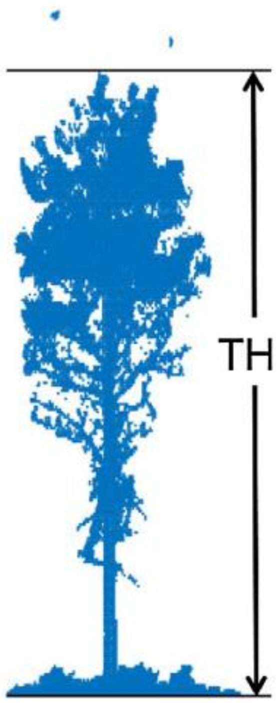

Figure 4: Total tree height (TH) computation. Note that it avoids isolated point clusters that may not belong to the tree.

3. Computation of stem diameter at different section heights.

In this final step a set of heights is defined, which will then be used to measure the stem diameter at different sections around the tree axes. To do so, a slice of points will be selected at every section, and those will be fit a circle by least squares minimization. This procedure is similar as the one proposed in (Prendes et al., 2021)

To ensure robustness, the goodness of fit is checked. What follows is a brief list of all the tests that are performed:

- Number of points inside the circle. This is checked via fitting an **inner circle**
- Percentage of **occupied sectors**
- Size of fitted circle (if it is **radius is too small/big**)
- **Vertical deviation from tree axis** ('outlier probability')

First, a complementary, inner circle is fitted as well, which will be used to check how points are distributed inside the first circle: they are expected to be outside the inner circle, as the scanning should only scan the surface of the stems. Second, the section is divided in several sectors to check if there are points within them (so they are occupied). If there are not enough occupied sectors, the section fails the test, as it is safe to assume it has an abnormal, non-desirable structure. After this, it is checked whether the diameter of the fitted circle is within some boundaries, to discard anomalies. Finally, the vertical deviation from the tree axis is computed for every section and it is used to check possible bad fittings: highly deviated sections are labelled as possible outliers.

On top of all goodness of fit tests, there is a last layer or robustness while computing the diameters. If the first fit is not appropriate, another circle will be fitted to substitute it using only points from the largest cluster in the slice of points, and the goodness of fit will be tested again. Figure 5 illustrates an example of some fitted circles after all tests.

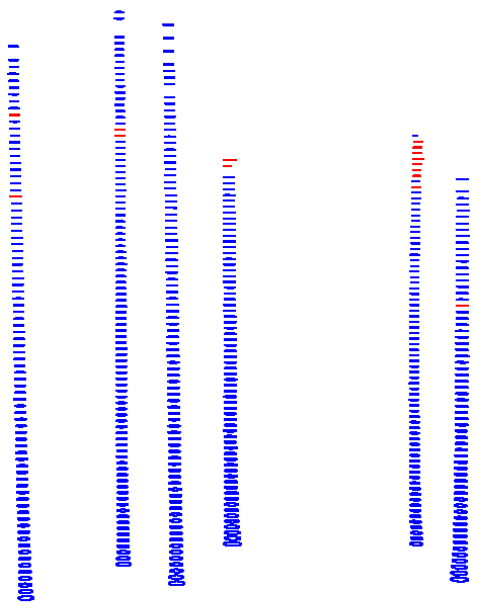

Figure 5: Fitted circles in 6 stems, at sections ranging from 0.3 to a maximum of 25.0 meters, one every 0.2 meters. Blue circles passed all quality tests, while red circles mean the fitting may be unreliable. This may be due to partial scans, non-expected diameter measurements, non-reasonable distribution of points within the section or a high value of tilting.

During this step, besides computing all the diameters at the selected heights, the DBH will be approximated as well (even if BH was not included as one of the selected heights). For this, the section closest to 1.3 m will be used as a proxy, and the DBH will only be computed if there is coherence between that section and the ones around.

Tree location [(x, y) coordinates] is obtained at this step too, either derived from the proxy section (to BH) when appropriate; that is, when it passes all goodness of fit tests and it is coherent, or from the tree axis when not.

Dependencies

The script imports several Python libraries and functions. They can be found listed below:

Libraries

These are libraries that are imported directly:

- os
- sys
- timeit
- tkinter
- configparser
- jakteristics
- laspy
- numpy

Functions

These are libraries from which only specific functions are imported:

- PIL (imports ImageTk and Image)
- copy (imports deepcopy)
- scipy (imports cluster.hierarchy, optimize and spatial.distance_matrix)
- sklearn (imports cluster.DBSCAN and decomposition.PCA)

Inputs

The script takes a .LAS file containing the ground-based 3D point cloud as input. The point cloud must be height normalized. Normalized heights can be contained in the Z coordinate of the point cloud or in an additional field in the .LAS file. If so, the name of that field is used as an input parameter. The parameters may be considered as inputs as well. Please refer to the info boxes in the executable program to get a description of what they do. Further explanations will be added in a future, public code repository.

Outputs

After all computations are complete, the following files are output:

Filenames are: [*original file name*] + [*specific suffix*] + [*.txt* or *.las*]

LAS files

- [*original file name*]_tree_ID_dist_axes: LAS file containing the original point cloud and 2 scalar fields that contain tree IDs and distance to closest axis.
- [*original file name*]_axes: LAS file containing stem axes coordinates and a scalar field that contains tilting (in degrees).
- [*original file name*]_circ: LAS file containing circles (sections) coordinates and several scalar fields: overall quality, outlier probability, diameter, inner circle points, percentage of sector occupancy and tree ID. These allow the user to filter the fitted circles based on their respective values.
- [*original file name*]_stripe: LAS file containing the stems obtained from the stripe during *step 1*.
- [*original file name*]_tree_locator: LAS file containing the tree locators coordinates.
- [*original file name*]_tree_heights: LAS file containing the highest point from each tree.

Text files

- [*original file name*]_dbh_and_heights: Text file containing tree height, tree location and DBH of every tree as tabular data.
- [*original file name*]_X_c: Text file containing the (x) coordinate of the centre of every section of every tree as tabular data.
- [*original file name*]_Y_c: Text file containing the (y) coordinate of the centre of every section of every tree as tabular data.
- [*original file name*]_R: Text file containing the radius of every section of every tree as tabular data.
- [*original file name*]_outliers: Text file containing the ‘outlier probability’ of every section of every tree as tabular data.
- [*original file name*]_sector_perct: Text file containing the sector occupancy of every section of every tree as tabular data.
- [*original file name*]_check_circle: Text file containing the ‘check’ status of every section of every tree as tabular data.
- [*original file name*]_n_points_in: Text file containing the number of points within the inner circle of every section of every tree as tabular data.
- [*original file name*]_sections: Text file containing the sections as a vector.

References

Cabo, C., Ordóñez, C., López-Sánchez, C. A., & Armesto, J. (2018). Automatic dendrometry: Tree detection, tree height and diameter estimation using terrestrial laser scanning. *International Journal of Applied Earth Observation and Geoinformation*, *69*, 164–174. https://doi.org/10.1016/j.jag.2018.01.011

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*. www.aaai.org

Prendes, C., Cabo, C., Ordoñez, C., Majada, J., & Canga, E. (2021). An algorithm for the automatic parametrization of wood volume equations from Terrestrial Laser Scanning point clouds: application in Pinus pinaster. *GIScience and Remote Sensing*, *58*(7), 1130–1150. https://doi.org/10.1080/15481603.2021.1972712