3DFin: 3D Forest inventory

DOCUMENTATION

Version date: 05 May 2023

3DFin: Forest Inventory Copyright (C) 2023 Carlos Cabo

3DFin has been developed at the Centre of Wildfire Research of Swansea University (UK) in collaboration with the Research Institute of Biodiversity (CSIC, Spain) and the Department of Mining Exploitation of the University of Oviedo (Spain).

Authors

Carlos Cabo (carloscabo.uniovi@gmail.com). Permanent academic position at the Department of Mining Exploitation, University of Oviedo and Honorary Appointment at Science and Engineering Faculty, Swansea University. Research fields: Spatial analysis, cartography, geomatics.

Diego Laino (diegolainor@gmail.com). PhD student in Natural Resources Engineering at Department of Mining Exploitation, University of Oviedo. Assist. Researcher at Centre for Wildfire Research, Swansea University. Research fields: deep learning, remote sensing, forestry.

Covadonga Prendes (cprendes@cetemas.es). Forest engineer and researcher at CETEMAS (Forest and Wood Technology Research Centre Foundation). Geomatics research group. Research fields: LiDAR, sustainable forestry development, spatial analysis.

Cristina Santin (c.santin@csic.es). Research fellow at the Research Institute of Biodiversity (CSIC-University of Oviedo - Principality of Asturias, Spain) and Honorary Assoc. Professor at the Biosciences Department of Swansea University. Research fields: environmental impacts of wildfires.

Stefan Doerr (s.doerr@swansea.ac.uk). Full Professor at the Geography Department, Swansea University and Director of its Centre for Wildfire Research. Editor-in-Chief: International Journal of Wildland Fire. Research fields: wildfires, landscape carbon dynamics, soils, water quality, ecosystem services.

Celestino Ordonez (ordonezcelestino@uniovi.es). Full professor at Department of Mining Exploitation, University of Oviedo. Main researcher at GEOGRAPH research group. Research fields: Spatial analysis, laser scanning, photogrammetry.

Tadas Nikonovas (tadas.nikonovas@swansea.ac.uk). Office Researcher at Centre for Wildfire Research, Swansea University. Research fields: Global fire activity, atmospheric emissions, fire occurrence modelling.

Summary

Welcome to the documentation of 3DFin: 3D Forest Inventory!

3DFin is a program that is intended to be simple and intuitive. It is designed to allow users to easily modify input parameters, adapting the algorithm that it's ran behind the scenes to better fit the particularities of their forest plot; even if they are unsure of what combination of parameters they should input, the program offers a selection of default values that fit a typical forest plot (yet users shouldn't expect them to work a miracle right off the bat if the forest plot is really complex!).

It also features some figures that illustrate key parts of the algorithm, to help the user draw a mental image of the process, as well as info boxes, that intend to deliver short but concise explanations of what each input parameter does.

After all initial configurations are set, the user may select a point cloud and run the program: the small console attached to the program will shortly start to display information about what things are being done, allowing the user to double check if the selected options were introduced as intended.

In this document, you will find an USER MANUAL, that contains detailed instructions on how to use the program; a description of the ALGORITHM, which hopefully will boost user's capabilities to maximize the performance of the algorithm in their forest plots; a description of INPUTS and OUTPUTS, so that the program is being used as intended; and, finally, REFERENCES, which point out to the original manuscripts which this program is based on. After all, although 3DFin intends to be as easy to use 'as it gets', it's built upon advanced scientific research.

Contents

User manual

General usage

TABS

The main structure of the program is based on tabs, and (currently) there are four of them:

- *Basic tab*
- *Advanced tab*
- *Expert tab*
- *About tab*

To switch from one tab to another, simply click the tab you want to switch to at the top of the program.
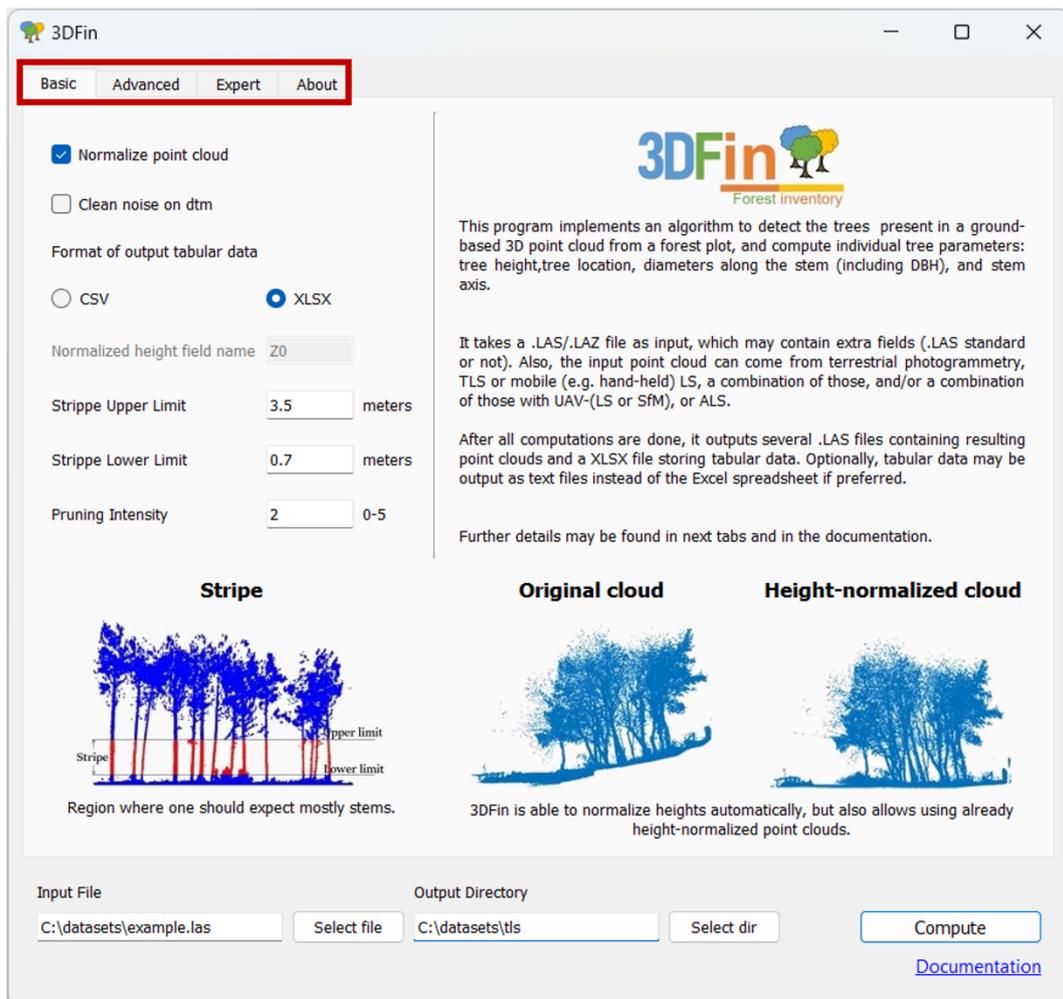
Figure 1: 3DFin tabs. There are four of them: 'Basic', 'Advanced', 'Expert' and 'About.

Basic, *Advanced* and *Expert* tabs contain the main types of features of the program, which could be listed as follows:

1) **Check boxes**. They allow the user to modify the parameters of the algorithm.

2) **Entry boxes**. They allow the user to modify the parameters of the algorithm.

3) **Buttons**. They perform some action when clicked on; the cursor will change to a manicule with its index finger pointing up when it is placed over one of them, indicating the user that they can activate the button if they click there.

4) **Radio buttons**. They allow the user to modify the parameters of the algorithm.

5) **Text labels**. They describe either the parameters or general information of the algorithm.

6) **Figures**. They illustrate key steps of the algorithm itself.
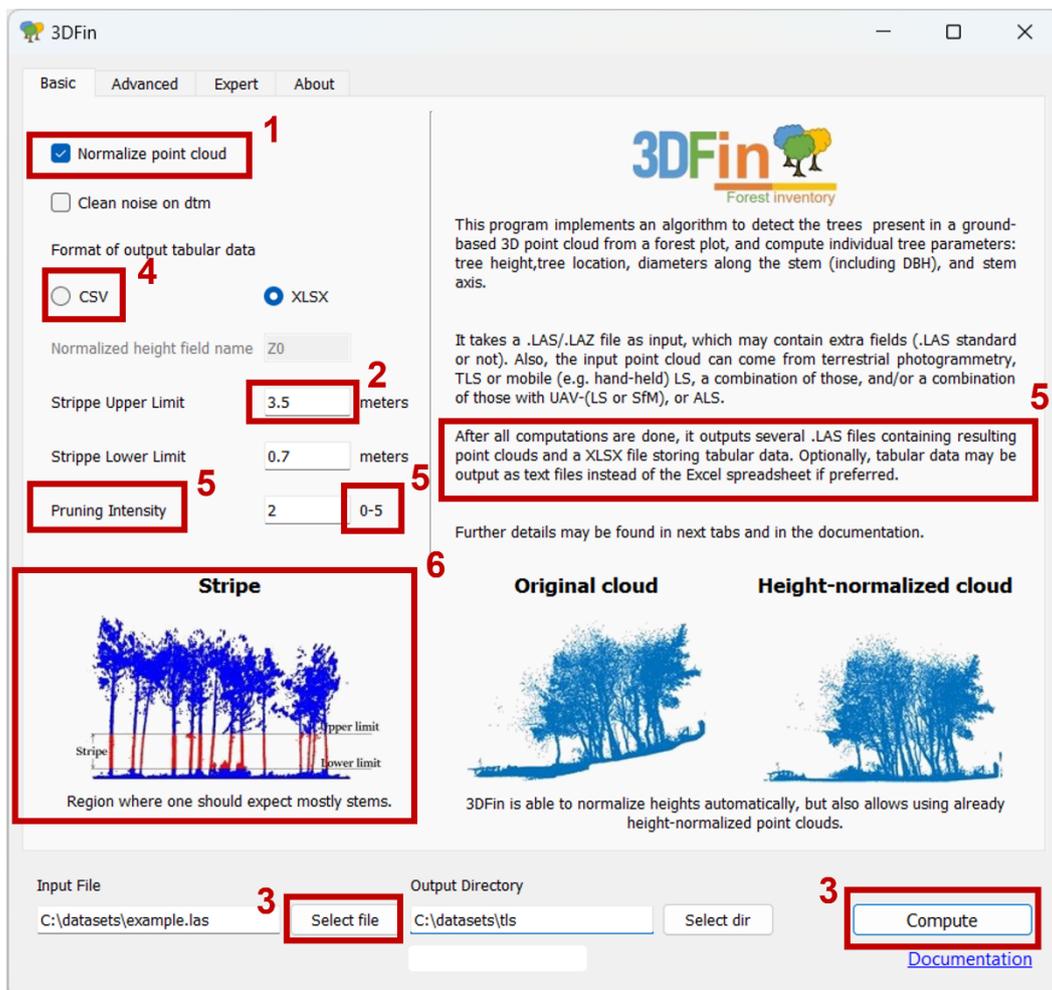
Figure 2: Some examples of 3DFin main features. 1) Radio buttons 2) Entry boxes 3) Buttons 4) Tooltips 5) Text labels 6) Figures.

The first three types of them allow the user to interact with the program; the last three features, on the contrary, just have the mission to display information to the user. It is important to note that every radio button and entry box comes with a default value and that those will be the

parameters passed to the algorithm if left unchanged; however, these may be modified by the user if desired.

MAIN BUTTONS

There are four common buttons to all tabs: these are the **Select file** button, **Select dir** button, the **Documentation** button, and the **Compute** button.
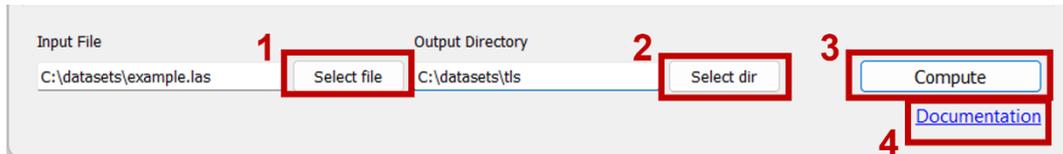

Figure 3: The main buttons of the program. They are common to all tabs. 1) Select file button 2) Select directory button 3) Compute button 4) Documentation button.

The **Select file** button will lead the users to the **Select file** menu when clicked. From there, they may select the point cloud that is to be processed by the program. In a similar fashion, the **Select directory** button will lead the users to the **Select directory** menu, which allows them to choose the folder where the output files should be written. In both cases, the same goal may be achieved by directly typing the path to the file/directory in the respective entry boxes that are right next to each button. On the other hand, the **Compute** button is meant to be clicked once the input file and the output directory have been stablished, and it will start running the algorithm. Lastly, the **Documentation** button is located at the bottom-right corner of the program window, and it is a direct link to the offline version of the documentation (a copy of the document that you are reading right now).
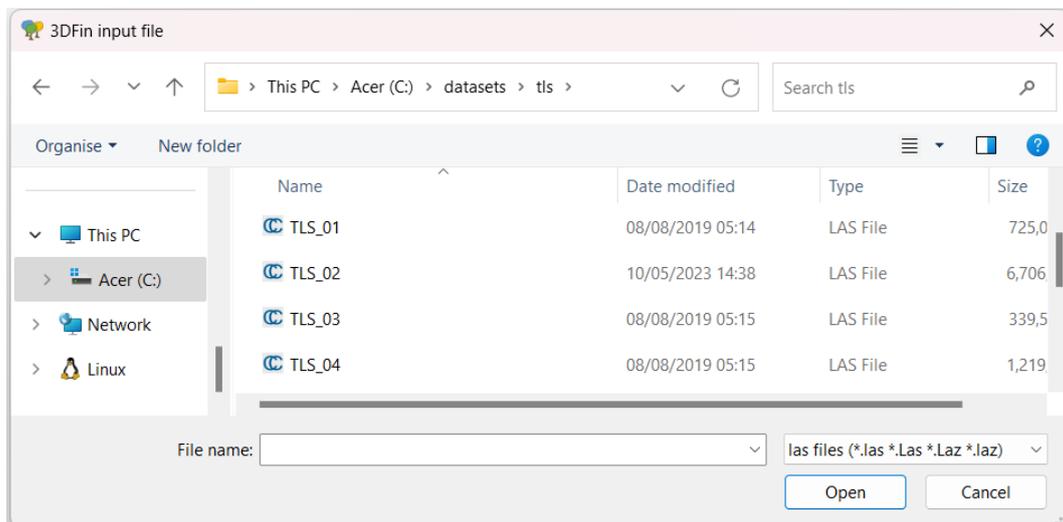

Figure 4: Select file/directory menu. It allows to easily stablish input/output paths.

CONSOLE

3DFin uses a console to display information about its execution while it is running. Once the **Compute** button has been clicked, the user will be able to see the progress of the computations and details about the algorithm's steps.

Figure 5: 3DFin uses a console to display information about the current execution.

EXIT THE PROGRAM

3DFin main window can be closed by pressing the **X** button at the top-right corner at any time. The program window may be minimized as well by pressing the _ button that is also at the top-right corner.

Figure 6: The user can click the X button to close the program.

Once the algorithm has started running, the console will start displaying information about the execution. When the processing of input point cloud has finalised, the user may modify the input file/parameter values/output directory and click the **Compute** again to start another run, or simply close the program.
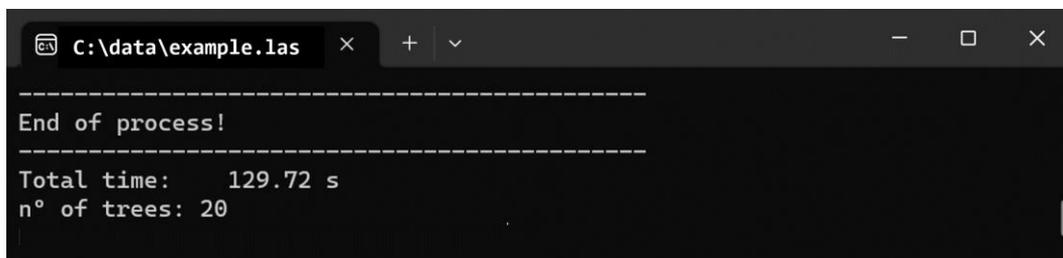
Figure 7: When the processing is over, the user can start another run or abandon the program.

Basic tab

The ***Basic*** tab allows the user to configure the basic parameters: these may be considered as the most fundamental ones and the ones that the user should consider first to modify if any change to the default values was to be done.

It also displays a summary of the main functionalities of 3DFin, as well as information about the format of the files that the program takes as input and the format of the files that are output. Additionally, three figures are displayed, which refer to the steps ***0. Height-normalization of the point cloud*** and ***1. Identification of stems among user-provided horizontal stripe*** of the **Algorithm**.
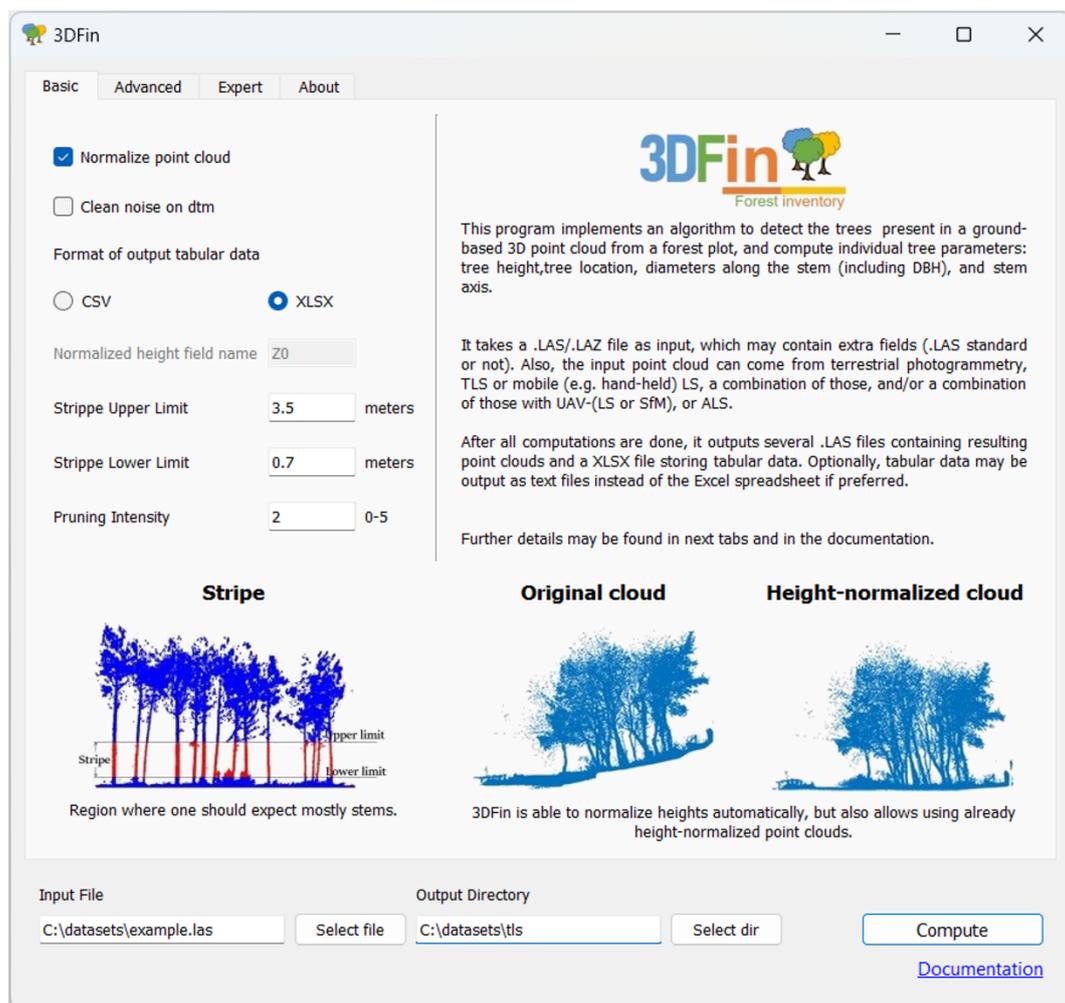

Figure 8: Basic tab.

In this tab the user may find two check boxes and one radio button that may be used to prepare the data before and after the algorithm is run.
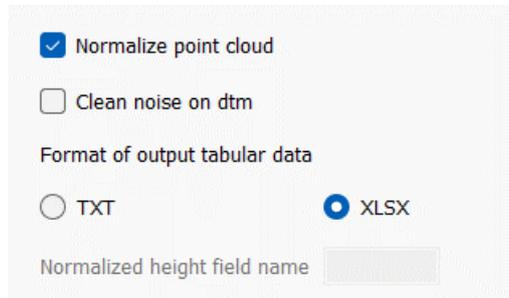
The first check box asks the user if the point cloud is already height-normalized: by default, this option is ticked; this will make the program run the *0. Height-normalization of the point cloud* step of the algorithm. If the user selects unchecks it, the step will be skipped. **From there, it will be assumed that the point cloud has been height-normalized previously, and the user will have to provide a Z_0 field from which the program may read the normalized heights.** If the point cloud has not been height-normalized previously or if you are unsure about it, it is advised that the default option (Normalize point cloud) is selected.
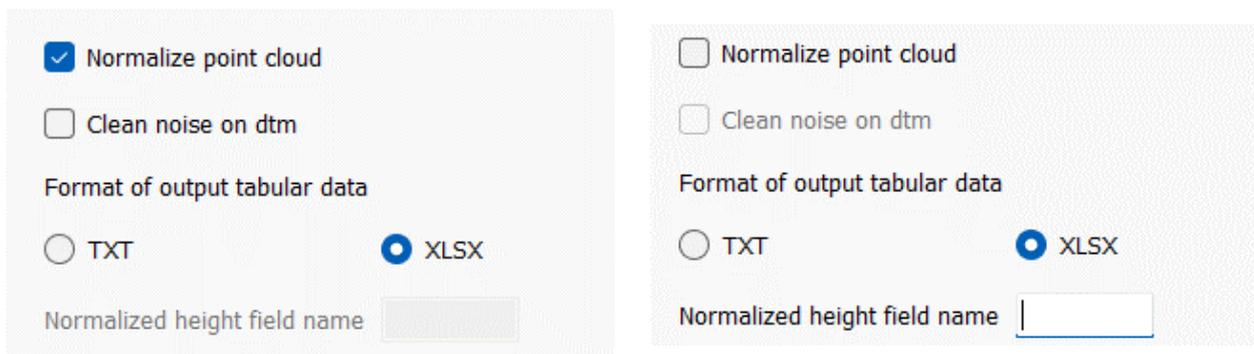
Figure 10: The first check box allows the user to decide whether the step 0 of the algorithm is run or not. If it is indeed run, the user may as well add a denoising step using the second radio button, to ensure the normalized heights are computed correctly. On the other hand, if it is not run, normalized heights will have to be provided by some point cloud field.

Unchecking the box will also disable the second check box, which allows the user to add a denoising step prior the height normalization. If the point cloud is noisy, specially below ground level, the height normalization step might not be applied correctly; checking this box will add a pre-processing step that aims to remove the noise points. However, this action will add some time to the total execution time. If it is not needed, simply let it uncheck.

Lastly, the radio button allows the user to modify how the tabular data is output. More precisely, it enables to either 1) export the tabular data as an Excel (.XLSX) file or 2) export the tabular data as separate text (.TXT) files. XLSX format is the default setting for outputs. These data are the numeric results of the algorithm execution, and they account for several metrics: Diameter at Breast Height (DBH), Total Height (TH), tree location, etc. A more detailed description of the outputs may be consulted in **Outputs** section.

Another important aspect of the *Basic* tab is that, from there, the user can modify three parameters of the algorithm that will affect greatly how the stems are extracted and thus how well the metrics are computed. These parameters are 1) the stripe upper limit, 2) the stripe lower limit and 3) the 'pruning' intensity. The first two are self-explanatory / well documented in *1. Identification of stems among user-provided horizontal stripe*. The pruning intensity parameter takes values between 0 and 5 (strictly integer numbers) and will the program how softly – aggressively should the stripe be denoised in order to look for the stems. Values of 0-2 are recommended for clean forest plots and/or point clouds that are not noisy. Higher values are advisable in cases where the trees have a large number of low branches or especially noisy point clouds. Note that this pruning is performed twice: first during the stem identification within the stripe, and later during the whole stem extraction that occurs in step *2. Tree individualization based on point-to-stems distances.* of the **Algorithm.**

Advanced tab

*The **Advanced** tab gives access to more parametrization of the algorithm (although default parameters are still provided) and focus on steps **2. Tree individualization based on point-to-stems distances.** and **3. Computation of stem diameter at different section heights.** of the **Algorithm.***
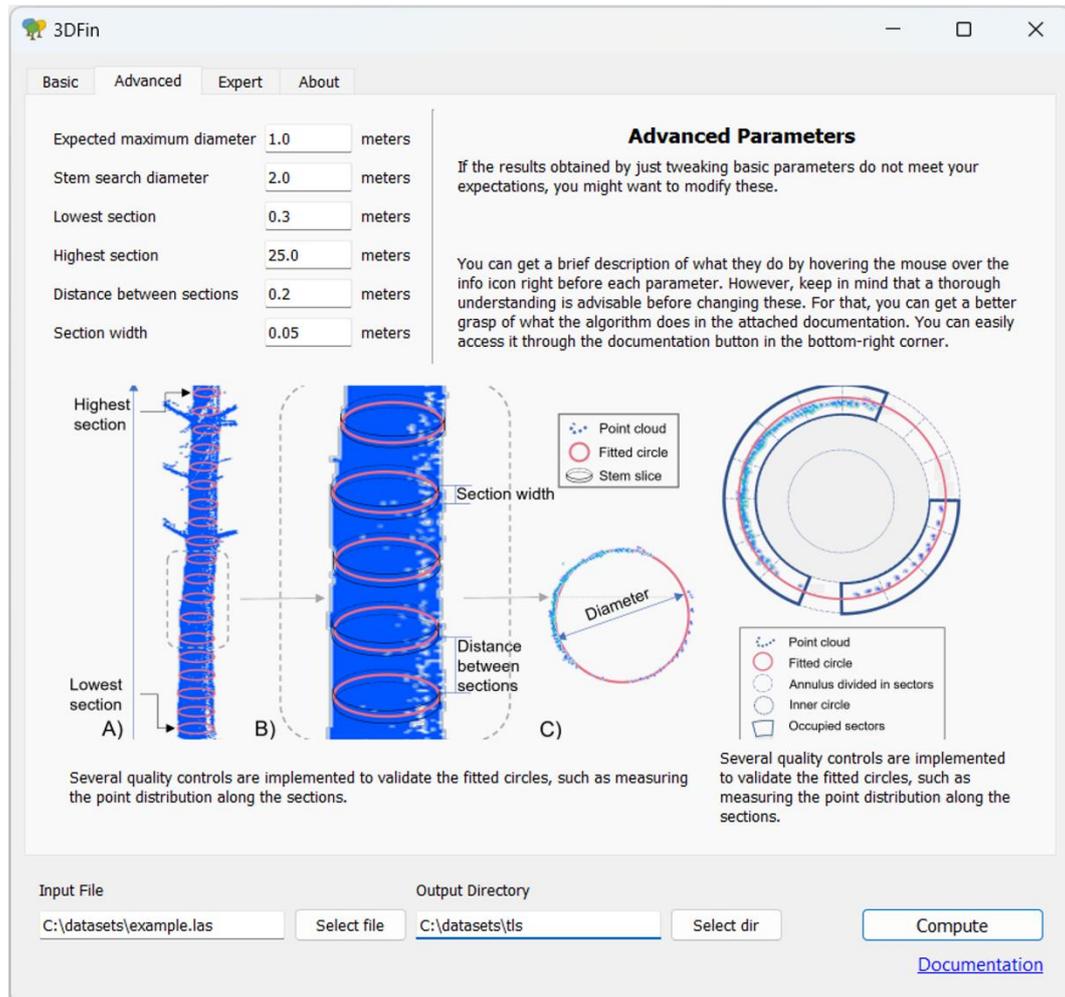
It features six parameters, which are the following:

- *Expected maximum diameter*. As its name states, the user may provide the program with an estimate (in meters) of what is the maximum diameter expected for any stem. This value will be used to discard objects that were wrongly identified by the algorithm as stems, as well as to discard sections that may be too wide.

- *Stem search diameter*. This parameter defines a diameter that will be used during the stem extraction in step 2 of the algorithm. Points within this distance from the stem axis computed during step 1 will be used to identify the complete stems. Even if there is an expected maximum diameter of, i.e., 1 m, a larger margin helps the algorithm to discern what actually is stem versus what is not.

- *Lowest section.* It is self-explanatory: this parameter defines at which height, from ground level, should the sections start to be computed. Please note that the units are meters.

- *Highest section.* In this case, the user can define the height of the last, highest section. It must be a value larger than the *lowest section*.

- *Distance between sections.* This parameter defines the interval at which sections will be computed among the lowest and highest ones.

- *Section width.* This parameter defines the vertical range around each section. All points within each of these ranges will be considered as belonging to its respective section. Note that this value is added to each side of the section, which makes the effective section width double the input value: i.e., if a section's height is 1 m and section width is 0.05 m, then the points regarded as belonging to that section are those with height values within [0.95-1.05] m range.

Expert tab

The last tab containing parameters is the *Expert* tab, which is designed as the 'last line of settings' to adjust the algorithm to the user's point clouds. In any case, if you have come this far and you feel the need to modify some of these parameters, it is advisable that you first do some checks before continuing:

- You have a deep understanding of how the **Algorithm** works.

- The **References** contain detailed and more technical information on the algorithm itself and how several parameters work. It is advisable that they are consulted.

- *dendromatics*, the *Python* package that powers 3DFin, has its own documentation. Reading it could be beneficial as well: https://github.com/3DFin/dendromatics.

- You have first tried to obtain desired results by setting *Basic*/*Advanced* parameters.
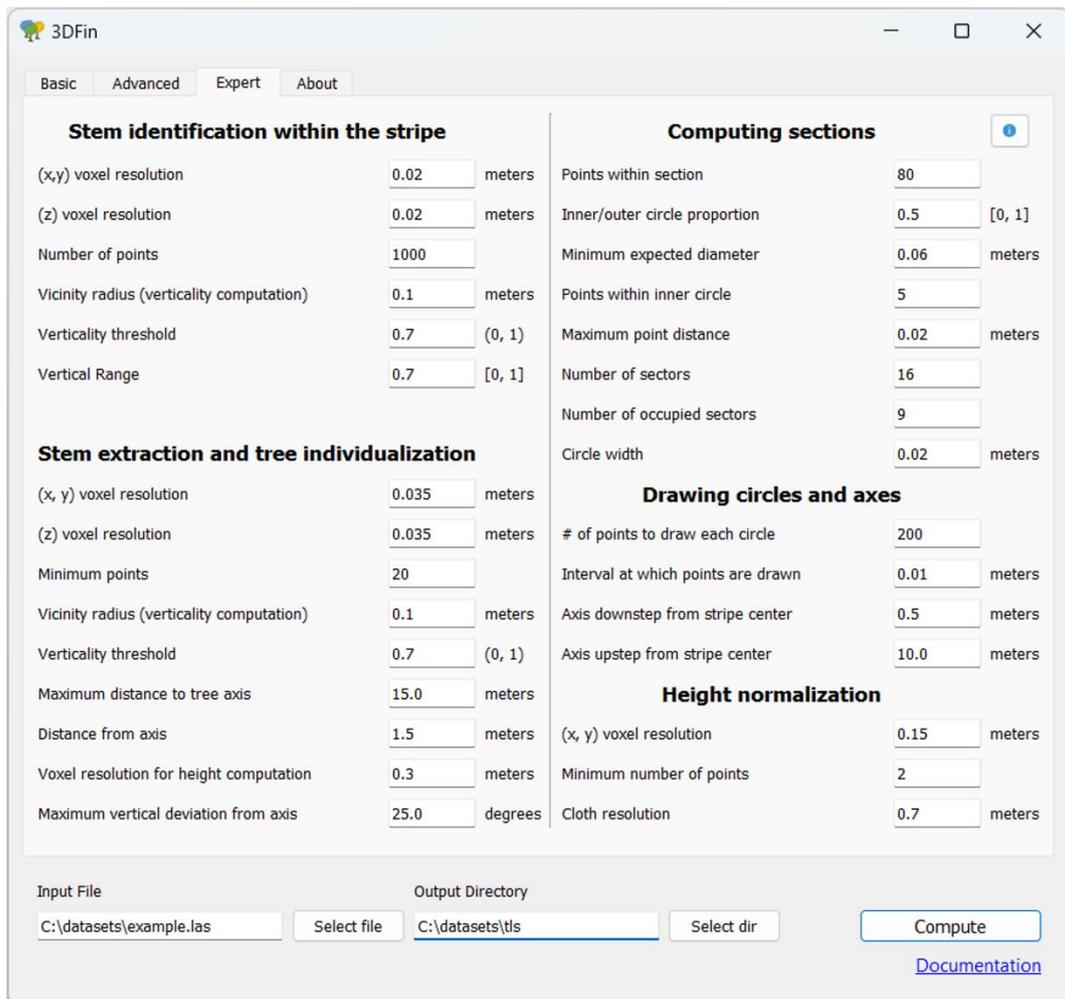
Figure 11: Expert tab. It contains 30 parameters for the most audacious users.

The parameters are divided in several categories that are intended to help the user know what step of the algorithm will be affected when modifying them. There are five categories:

- *Stem identification within the stripe.* This category directly relates to step *1. Identification of stems among user-provided horizontal stripe* of the **Algorithm.**

- *Stem extraction and tree individualization.* This category It directly relates to step *2. Tree individualization based on point-to-stems distances.*.

- *Computing sections.* This category It directly relates to step *3. Computation of stem diameter at different section heights.*

- *Drawing circles and axes.* This category includes parameters that control how some of the **LAS files** are output. More precisely, '[*original file name*]_circ' and '[*original file name*]_axes' files.

- *Height normalization.* This category directly relates to step *0. Height-normalization of the point cloud*.

It is encouraged as well that the info labels contained in the **Radio** buttons. are read: they display valuable information about each parameter.

About tab

The *About* tab displays information about the software's License, the funding that made it possible and the team behind 3DFin. It actually does not provide any helpful information on how the program works.
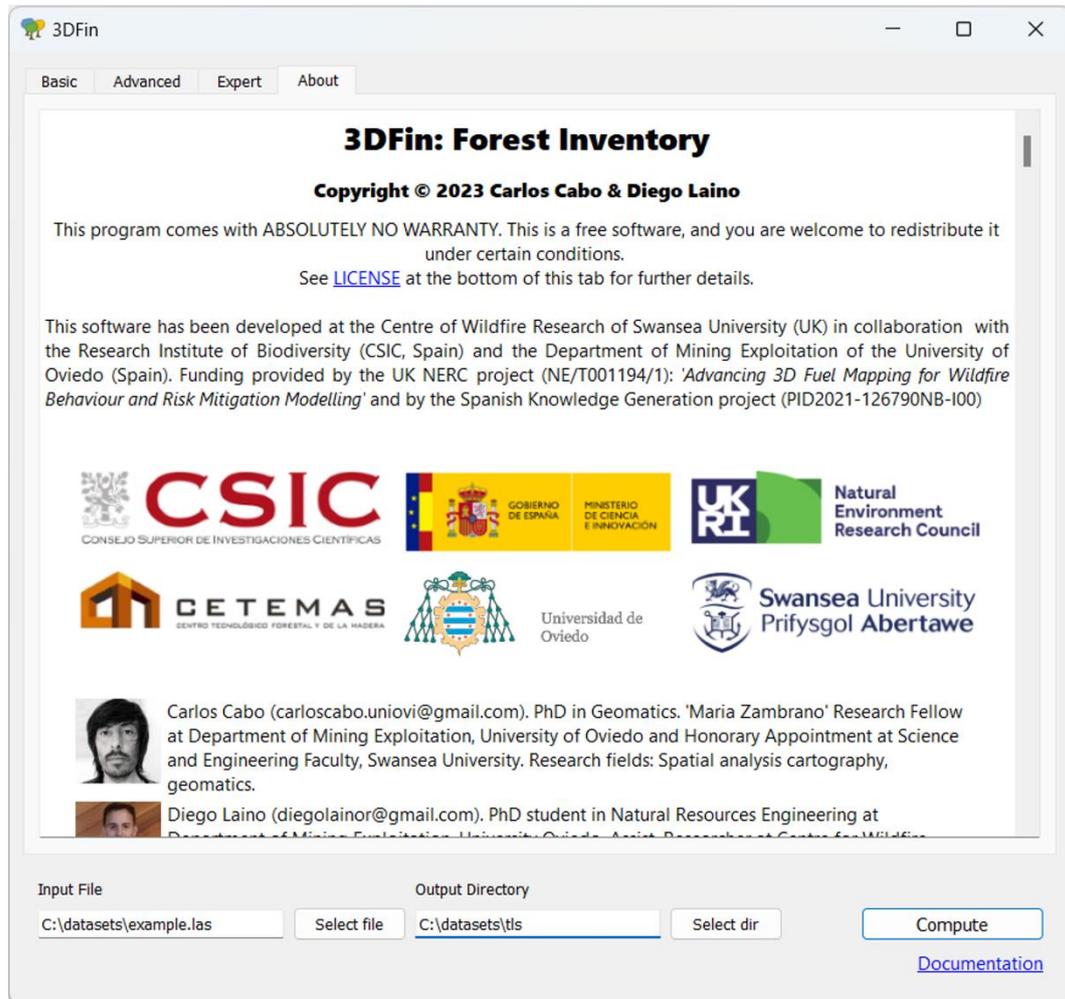
Figure 12: About tab does not contain information on how the program works.

The users may use the scroll bar to see the full team and the *License*, as well as to access the License button: it directs the user straight to the text containing the terms and conditions of the program's license.
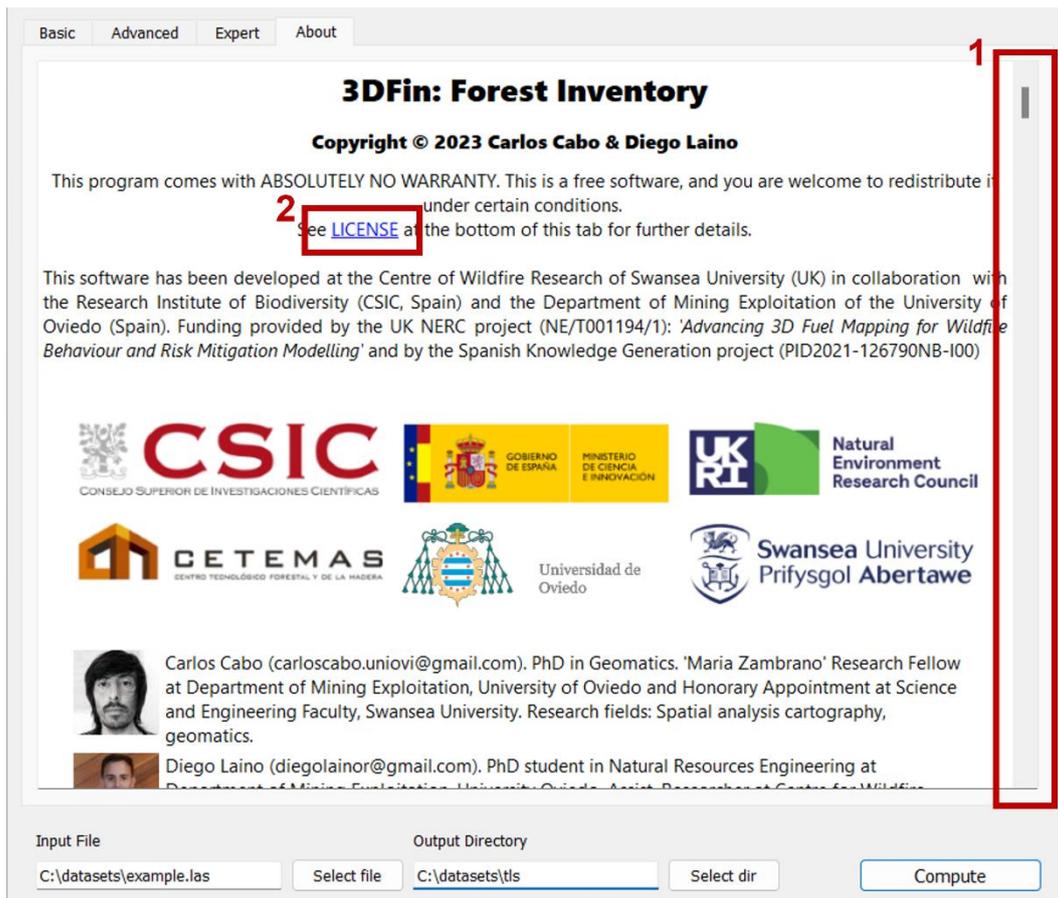
Figure 13: About tab is scrollable. To scroll up or down, simply use the scroll wheel of the mouse or click and hold the scroll bar and move up or down with the mouse. 1) Scrollbar 2) License button.

Modifying default parameters

A really useful feature of 3DFin is that the users can modify the value that the parameters hold by default. This way, if there are several point clouds that the user may want to process using the same configuration of parameter values, it will not be needed to modify them each time the program is launched.

This is achievable with a configuration .INI file of certain characteristics, which are listed below:

1. The file must contain the sections and parameters coded following 3DFin structure.

2. This file must be named 3DFinconfig.INI.

3. It must be placed in the same directory as 3DFin.exe (the program itself), as displayed in *figure 14*.

An example file (that contains the very same default values as the 'vanilla' program) may be downloaded from 3DFin's official website https://github.com/3DFin/3DFin, under the /3DFinconfig/ folder.

Name		Date modified	Type	Size
3DFin		26/04/2023 14:36	Application	98,819 KB
3DFinconfig		20/04/2023 10:08	Configuration settings	5 KB

Figure 14: The configuration file must be placed in the same directory as 3DFin program, although there may be more files in this directory.

Algorithm

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 an updated version of the one presented in (Cabo et al., 2018) and is mainly based on rules, although it uses clusterization in some stages.

It may be divided in four main steps:

0. Height-normalization of the point cloud.
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.

The program runs a Python script that imports and makes use of *dendromatics*, a *Python* package whose functionalities implement the algorithm. What follows is a description of the main steps of it.

0. Height-normalization of the point cloud

The algorithm starts with a height-normalized point cloud as input, as in Figure 15. If normalized heights are not provided, the first step **3DFin** will take is to compute them, although the users can easily provide normalized heights computed externally (see *User manual – Basic tab*).
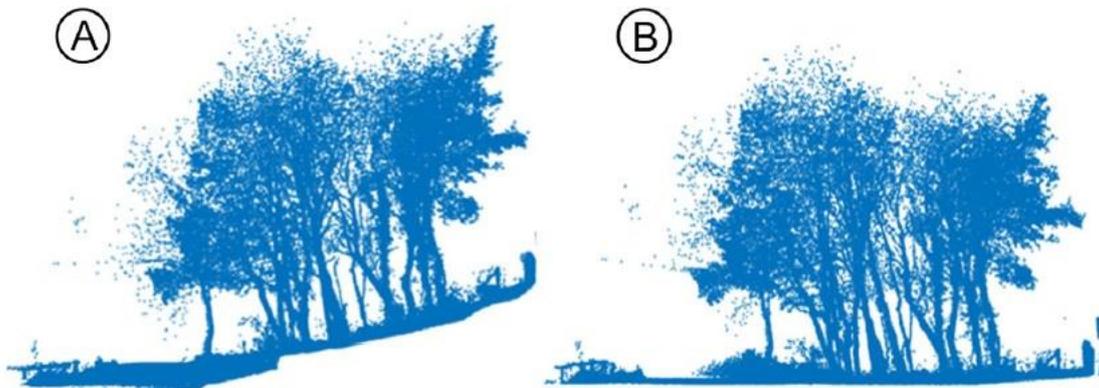
Figure 15: The algorithm first normalizes heights if they are not provided as input A) Original point cloud. B) Height-normalized point cloud. From (Cabo et al., 2018).

If the user opts to normalize the point cloud using **3DFin**, a digital terrain model (DTM) is generated and the normalized heights for each point in the cloud are obtained as the difference between their (z) value and the (z) value of the DTM points that are below.

To generate the DTM itself, a cloth-simulation filter (CSF) as described in (Zhang et al., 2016) is applied to the point cloud. To obtain the DTM points that are below a certain cloud point, a k-d-tree is generated, and the 3 closest DTM points are identified. Then, a weighted average of the (z) value based on the distance to the cloud point is obtained, and the normalized height value is computed as the difference between the (z) value of the cloud point and the weighted average (z) value of the DTM points.

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 16 illustrates this step of the algorithm.
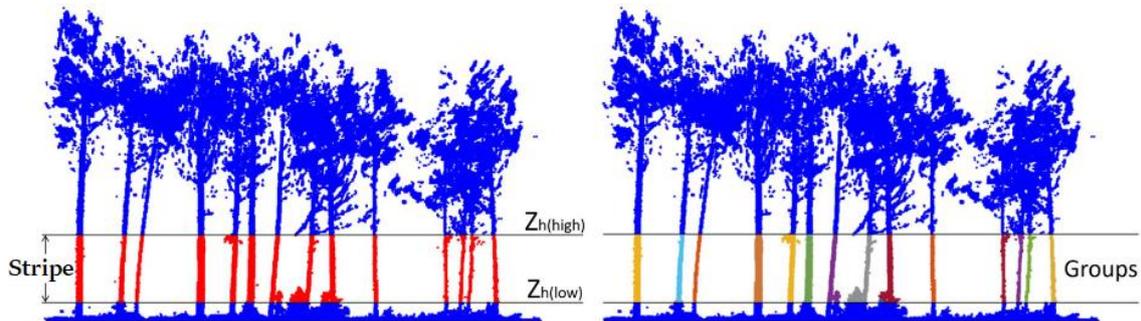
Figure 16: Stripe on the height-normalized point cloud, and candidate voxel groups. Points in the stripe in red, and voxel groups in random colours.

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 17.

Figure 17: 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 18 illustrates this.

Figure 18: 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).
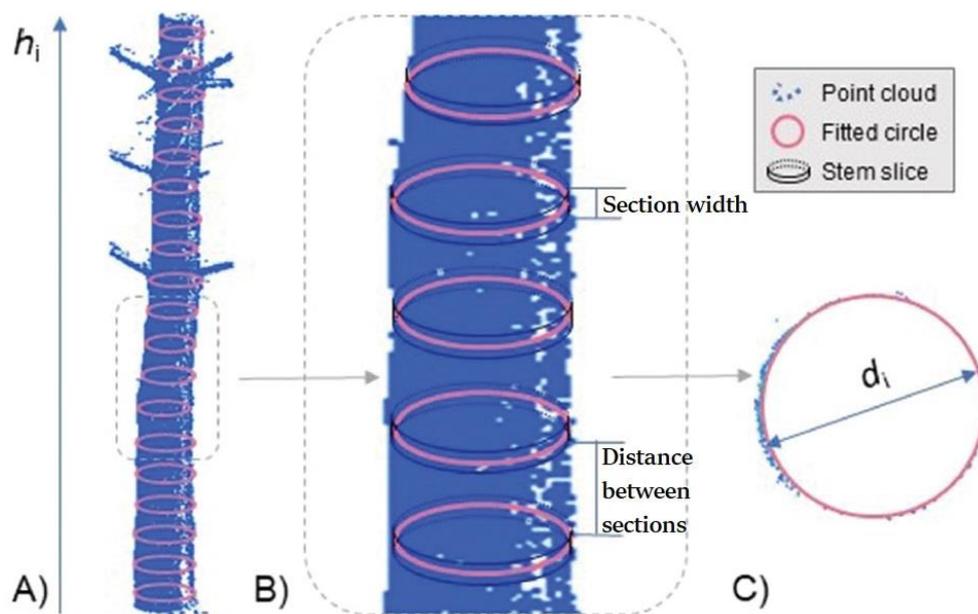

Figure 19: A) Sections along the stem B) Detail of computed sections showing the distance between them and their width C) Circle fitting to the points of a section.

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.
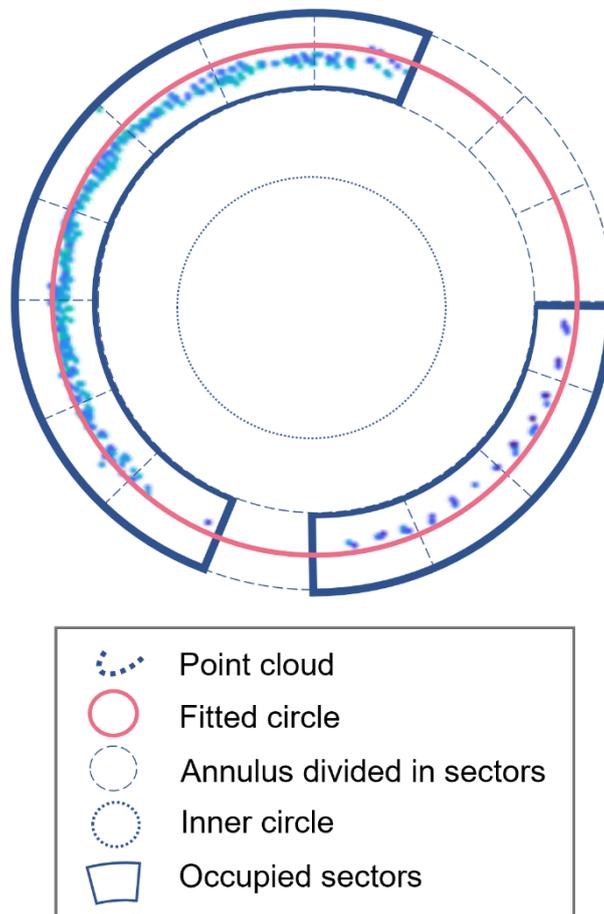
Figure 20: Several quality controls are implemented to validate the fitted circles, such as measuring the point distribution along the sections. Other quality checks include discarding too large/small/deviated sections.

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 21 illustrates an example of some fitted circles after all tests.
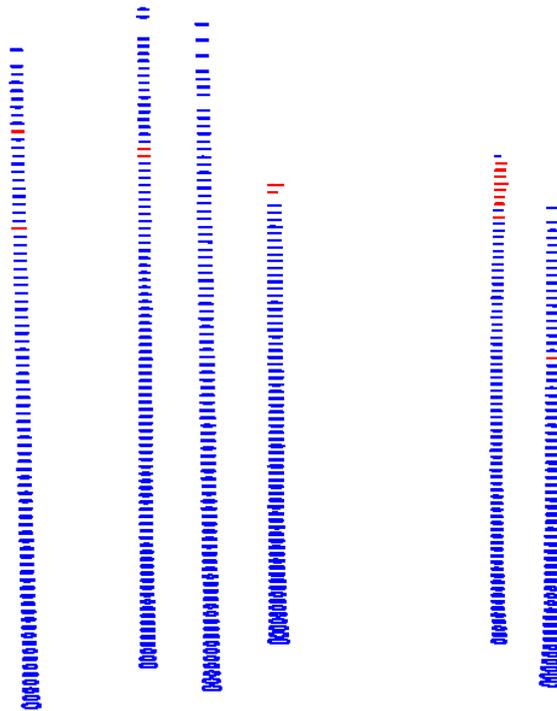
Figure 21: Fitted circles in 6 stems, at sections ranging from 0.3 to a maximum of 25.0 meters, one every 0.2 meters. Blue circles passed all quality tests, while red circles mean the fitting may be unreliable. This may be due to partial scans, non-expected diameter measurements, non-reasonable distribution 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.

Inputs

3DFin takes a LAS file containing the ground-based 3D point cloud as input. It accepts a .LAS/.LAZ file, which may contain extra fields (.LAS standard or not). Also, the input point cloud can come from terrestrial photogrammetry, TLS or mobile (e.g. hand-held) LS, a combination of those, and/or a combination of those with UAV-(LS or SfM), or ALS.

The following LAS versions are accepted:

- 1.2
- 1.3
- 1.4

Outputs

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

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.

Tabular data

Depending on the option selected by the user, the tabular data might be output as a single xlsx file or as several txt files.

XLSX file:

- [*original file name*].xlsx; Excel file containing several sheets:
  - **R**; sheet containing the radius of every section of every tree.
  - **X**; sheet containing the (x) coordinate of the centre of every section of every tree.
  - **Y**; sheet containing the (y) coordinate of the centre of every section of every tree.
  - **Sections**; sheet containing the normalized height of every section.
  - **Q(Overall Quality 0-1)**; sheet containing the overall quality of every section of every tree.
  - **Q1(Outlier Probability)**; sheet containing the 'outlier probability' of every section of every tree.
  - **Q2(Sector Occupancy)**; sheet containing the sector occupancy of every section of every tree.
  - **Q3(Points Inner Circle)**; sheet containing the number of points within the inner circle of every section of every tree.

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

TXT 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.

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

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

Zhang, W., Qi, J., Wan, P., Wang, H., Xie, D., Wang, X., & Yan, G. (2016). An easy-to-use airborne LiDAR data filtering method based on cloth simulation. *Remote Sensing*, *8*(6). https://doi.org/10.3390/rs8060501