3DFin: a software for automated 3D Forest Inventories from terrestrial point clouds

Diego Laino[1,2], Carlos Cabo [*3,1], Covadonga Prendes[4], Romain Janvier[5], Celestino Ordonez[3], Tadas Nikonovas[1], Stefan Doerr[1], Cristina Santin[2,1]

[1] Centre for Wildfire Research, Swansea University, Singleton Campus, Swansea SA2 8PP, United Kingdom;
[2] Biodiversity Research Institute, CSIC-University of Oviedo-Principality of Asturias, 33600 Mieres, Asturias, Spain;
[3] Department of Mining Exploitation and Prospecting, University of Oviedo, 33600 Mieres, Asturias, Spain;
[4] Forest and Wood Technology Research Centre Foundation (CETEMAS), Pumarabule, Spain;
[5] Independent researcher, Nancy, France

Correspondence: *carloscabo@uniovi.es*

Keywords

Forestry; Remote sensing; LiDAR; Terrestrial Laser Scanner (TLS); Diameter at Breast Heigh (DBH); photogrammetry; Mobile Laser Scanning (MLS); Opensource Software; Point clouds; Forest inventory

Abstract

Accurate and efficient forest inventories are essential for effective forest management and conservation. The advent of ground-based remote sensing has revolutionized the data acquisition process, enabling detailed and precise 3D measurements of forested areas. Several algorithms and methods have been developed in the last years to automatically derive tree metrics from such terrestrial/ground-based point clouds. However, few attempts have been made to make these automatic tree metrics algorithms accessible to wider audiences by producing software solutions that implement these methods. To fill this major gap, we have developed *3DFin*, a novel free software program designed for user-friendly, automatic forest inventories using ground-based point clouds. *3DFin* empowers users to automatically compute key forest inventory parameters, including tree Total Height (TH), Diameter at Breast Height (DBH), and tree location. To enhance its user-friendliness, the program is open-access, cross-platform, and available as a plugin in *CloudCompare* and *QGIS* as well as a standalone in *Windows*. *3DFin* capabilities have been tested with Terrestrial Laser Scanning (TLS), Mobile Laser Scanning (MLS) and terrestrial photogrammetric point clouds from public repositories across different forest conditions, achieving nearly full completeness and correctness in tree mapping and highly accurate DBH estimations (RMSE < 2 cm, bias < 1 cm)-in most scenarios. In these tests, 3DFin demonstrated remarkable efficiency, with processing times ranging from two to seven minutes per plot. The software is freely available at: https://github.com/3DFin/3DFin.

1. Introduction

Forest inventories (the systematic process of collecting, analysing, and reporting data about the characteristics of forest resources, such as their location, composition, and distribution; Wulder, 2004) play a crucial role in the sustainable management and conservation of forest ecosystems (Ridder, 2010). Traditional forest inventory methods often rely on time-consuming and labour-intensive field surveys, which are limited in their ability to capture detailed spatial information (Van Laar, 2007). In recent years, the utilization of ground-based remote sensing technologies has emerged as a transformative approach for forest inventory tasks. These technologies comprise Light Detection and Ranging (LiDAR), which includes Terrestrial Laser Scanning (TLS) and Mobile Laser Scanning (MLS); as well as the use of terrestrial Photogrammetry (Newnham et al., 2015; Liang et al., 2016). Ground-based technologies generate dense three-dimensional representations of objects on the Earth's surface that can be referred to as "terrestrial point clouds". This term can be deceiving, though, as "terrestrial point clouds" is commonly used exclusively for static laser scanning (i.e., TLS), but should also include other systems, as stated above. For this reason, the term "ground-based point clouds", which has gained popularity over the very last years, will be used in this article to refer to this subset of close-range remote sensing (Liang et al., 2022), as it conveys clearly that there is a diversity of technologies that use a perspective from the ground and that produce point clouds. These ground-based point clouds are typically composed of millions of individual data points, which collectively form a detailed and accurate digital representation of the scanned area or object, capturing detailed geometric information, including the shape, position, and orientation of objects within the scanned area.

TLS and MLS involve using a laser scanner that emits light pulses which measure the distance from the sensor to reflecting surfaces (Dassot et al., 2011; Calders et al., 2020). This process generates a dense set of 3D coordinates, forming the point cloud. Photogrammetry, on the other hand, utilizes a series of photographs taken from different angles to reconstruct the 3D structure of the scene (Iglhaut et al., 2019). By identifying common features in multiple images and applying mathematical algorithms, photogrammetry software can calculate the 3D coordinates of points and generate a point cloud. Due to their ability to represent geometric information, ground-based point clouds have emerged as a valuable tool for storing detailed three-dimensional information about forest plots, enabling comprehensive analysis and assessment of tree structures (Newnham et al., 2015).

One of the primary goals in forest inventories is the computation of tree metrics, such as total Tree Height (TH), Diameter at Breast Height (DBH) or tree location, which provide crucial insights into forest structure, dynamics, and ecosystem services (Van Laar, 2007; Pascu, 2019). Accurate and efficient computation of these metrics from ground-based point clouds is becoming a key step to automatically retrieve inventory information to support for effective forest management, biodiversity monitoring, carbon estimation, and ecological research (Liang et al., 2016).

In recent years, numerous tree metrics algorithms have been developed utilising ground-based point clouds (Liang et al., 2018; Ravaglia et al., 2019; Wang et al., 2021; Sadeghian et al., 2022). These algorithms leverage advanced data processing techniques, statistical analysis, and geometric calculations to extract meaningful information about individual trees within the point cloud data. Each algorithm adopts a unique approach and methodology, offering distinct advantages and limitations in terms of accuracy, efficiency, and adaptability to different forest types and plot characteristics. However, as some authors have pointed out (Liang et al., 2018; Krisanski et al., 2021; Montoya et al., 2021) the lack of practical, publicly available software that implements these algorithms is currently a bottleneck that limits their use by the user community.

Current non-commercial software implementations specifically designed to compute tree metrics at plot level from ground-based point clouds include: *CompuTree* (Piboule et al., 2013), *3DForest* (Trochta et al., 2017), *TreeLS* (de Conto et al., 2017), *DendroCloud* (Mokros & Koreň, 2019) *TreeTool* (Montoya et al., 2021), *FSCT* (Krisanski et al., 2021) and *FORTLS* (Molina-Valero et al., 2022). In addition, some proprietary commercial options are also available, like *LiDAR360* (GreenValley International, 2013), *AID-FOREST* (López Serrano et al., 2022), or *OPALS* (Pfeifer et al., 2014), that was not initially designed for ground-based point clouds, but nowadays offers enough capabilities to do so. Although able to provide tree metrics in a fairly automatic manner, *TreeTool* and *FSCT* are available solely as *Python* (Van Rossum & Drake, 1995) libraries, which reduces the potential number of users from the general public that may employ them. Similarly, *TreeLS* and *FORTLS*, are only available as *R* packages (R Core Team, 2023). Conversely, *DendroCloud*, *3DForest*, *CompuTree*, *LiDAR360*, *AID-FOREST* and *OPALS* are available as standalone programs, which eliminates the burden associated with programmatic access (installing requirements, versioning, scripting, etc.). These offer a step-by-step approach to perform the analysis of the point cloud, which has the benefit of a more controlled run. This allows the users to check if things are not going as expected at earlier stages. However, it leads to a situation where a new learning curve on how to use the software tools appears, which may translate into an obstacle for non-expert users and deter them from using ground-based point clouds.

Here we introduce a new software, *3DFin: 3D Forest Inventory*, that has been developed to advance the automatization of forest inventories. Thanks to its simplified interface, it allows users, by simply selecting a point cloud and pressing a button, to directly obtain tree metrics (diameter at different heights including DBH, TH and tree location) and several complementary computations (normalized height of the points, distance from any point in the cloud to closest tree axis, quality-of-measure indicators) of the forest plot. Moreover, *3DFin* has been integrated into the larger and widely used computer programs *CloudCompare* and *QGIS*, to simplify its integration into the users' workflow. We first describe the developed algorithm and its implementation into *3DFin* software, then we evaluate its performance by processing public data with *3DFin* and we end by discussing its strengths and limitations compared with other available software.

2. Algorithm

3DFin's underlying algorithm leverages state-of-the-art point cloud processing techniques to accurately detect and locate the trees in ground-based point clouds from forest plots, and also calculate essential parameters, such as diameters along the stem -including specifically the DBH-, and TH. Its application to the point clouds is highly parametrizable using *3DFin*'s graphical user interface (GUI). The algorithm behind the software is an updated version of that presented in Cabo et al. (2018) and includes some of the extensions developed in Prendes et al. (2021). The algorithm is mainly based on rules, although it uses clustering in some stages. The algorithm can be divided in four main steps:

1. Height-normalization of the point cloud.
2. Identification of stems within user-provided horizontal stripe.
3. Tree individualization based on point-to-stems distances.
4. Computation of stem diameters at different section heights.

2.1 Height-normalization of the point cloud

The first step of the algorithm is to normalize the heights of the input point cloud. This is depicted in Figure 1. The height-normalization is achieved by generating a Digital Terrain Model (DTM). From there, the normalized heights for each point in the cloud are obtained as the difference between their (z) value and the elevation of the DTM in their vertical projection.

Figure 1: Figure caption

To generate the DTM, a Cloth-Simulation Filter (CSF) as described in Zhang et al. (2016) is applied to the point cloud. The DTM is stored as a collection of 3D points (the nodes of the 'cloth mesh'), and the vertical projections are performed using *k*d-tree k-neighbour queries (Bentley, 1975) and weighted averages of the (z) values of the DTM points. For each point (p_i) in the original cloud, the three nearest DTM points are queried. Then, a weighted average of their (z) value based on the distance to p_i is computed, and the normalized height value (z_0) of p_i is computed as the difference between its original (z) value and the weighted average (z) value of the three DTM points. Optionally, a denoising step may be added prior to the height-normalization when running *3DFin* to prevent the influence of noise and artifacts below the ground. This is achieved by voxelating (Cabo et al., 2014) the point cloud using a relatively large voxel size (i.e., 0.15 m), then clustering by Euclidean distance the resulting voxels using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm (Ester et al., 1996) and finally filtering clusters smaller than a certain cluster size. This process is illustrated in Figure 2.

Figure 2: Figure caption.

2.2 Identification of stems within a horizontal stripe

In the second step, a horizontal stripe, defined as a subset of the normalized point cloud delimited by a lower height $Z_{h(low)}$ and an upper height $Z_{h(high)}$, is defined. This horizontal stripe represents a region in the 3D-space where it is expected to mostly encounter stems (Cabo et al., 2018). The points within the stripe are voxelated (using now a smaller voxel size of 0.02-0.06 m) and their verticality (Hackel et al., 2016) is computed, based on fixed-radius neighbourhoods. Then, the voxels are filtered based on their verticality value: it is reasonable to assume that the structure of points that is associated to a scanned stem would score a high verticality value, and that any other structure has a lower value. Finally, the remaining points (the ones with high values) are clustered by Euclidean distance using the DBSCAN algorithm, in a similar fashion as the clustering process detailed in Section 3. These two filters -eliminating points with low verticality values and removing clusters smaller than a certain cluster size- can be conceptualized as akin to 'limbing the trunks' and they are repeated iteratively, to ensure that the stems are isolated appropriately within the horizontal stripe. This step is illustrated in Figure 3.

Figure 3: Figure caption.

2.3 Stem extraction and tree height measurement

Once the bases of the stems have been identified in the horizontal stripe, they are isolated and enumerated, and then, 'initial' stem axes are computed. These initial axes are straight, but not necessarily vertical representations of the main direction of the points of each stem within the stripe. The axes are assimilated as the direction of the first principal component of the (x, y, z) coordinates of each stem, as in Figure 4, and are henceforth considered as stem axes. This allows to label points along the complete point cloud based on their distance to those axes, thus assigning each point to a tree.

Figure 4: Figure caption.

During this third step of the algorithm, the TH is computed as well. For this, and for each tree, points are voxelated and clustered with the DBSCAN algorithm as in Cabo et al. (2018). Any small cluster is then discarded, and from the remaining voxels that belong to the main cluster (the one that encloses the tree), a radius of voxels around the tree axis is subset and the highest voxel among these is selected. The (z) value of this voxel will be then considered as the tree height. This process inherently excludes from the estimation of the TH the points that are far from the tree, which could belong to other trees, and any noise above the tree. This is illustrated in Figure 5.

Figure 5: Figure caption.

It is important to highlight here that the "tree individualization" performed during this step does not aim to correctly separate tree crowns, which is another task that researchers have shown interest in (Windrim & Bryson, 2020; Chen et al., 2021; Carpenter et al., 2022; Wang & Bryson, 2023). The purpose of this intermediary step is to enable the efficient extraction of the stems. Thus, for each stem that is processed, only the points that are close enough to it are saved into memory, avoiding unnecessary overhead computations.

2.4 Computation of stem diameter at different section heights

In this fourth and final step, the stem diameter is measured at different heights around the tree axes. A general overview of this process is depicted in Figure 6.

Figure 6: Figure caption.

Once every tree is 'individualized', i.e., every point in the cloud is linked to one of the axes, the algorithm extracts their whole stems. To do so, points far from any axis (i.e., 1.5 m) are discarded temporarily, thus keeping only the points close to the axis. These are candidates to belong to the tree stem, and the iterative limbing process described in 4 is applied again, but this time to the whole stems. This ensures that branches are removed before measuring them. This is depicted in Figure 7.

Figure 7: Figure caption.

Once the whole stems are identified and "limbed", circles are fitted to them at several section heights. Those heights are evenly separated along the stems and the distances between section heights can be defined by the user. The circle fittings are computed for every tree and section by least-squares minimizations. This is performed using the (x, y) locations of the points in horizontal slices at the specified heights, as initially described in Cabo et al. (2018) and improved in Prendes et al. (2021). By refining the initial selection of stem points through prefiltering, 3DFin effectively addresses some of the common sources of error encountered in previous circle fitting methods (Koren et al. 2017), such as the presence of understory and branches. Additionally, the robustness of the circle fittings and diameter calculations is checked in four steps. To accomplish this, the number of points inside the fitted circle, the percentage of occupied sectors within the circle, the radius of the circle and the vertical deviation from the tree axis and other sections are analysed.

First, a complementary inner circle is placed inside the fitted circle, the latter referred henceforth to as 'outer circle' for clarity. The centre of the inner circle has the same (x, y) coordinates than the centre of the outer circle, but its radius is a proportion of the latter. The inner circle is used to explore how points are distributed in the section, based on the idea that the points are expected to be outside the inner circle, as the point cloud should only represent the surface of the stems. Depending on the technology used to obtain the point cloud, some noise might be expected, so a small number of points inside the inner circle might not necessarily mean that the outer circle is wrongly fitted. However, if there are too many points inside the inner circle (i.e. more than what could be expected due to noise), then, it probably has been fitted wrongly. Second, the section is divided into several sectors to check if there are points within them (so that they are occupied). If there are not enough occupied sectors, the section fails the test, as the points within itself may potentially have an abnormal, non-desirable structure, or the diameter of the fitted circle may not be reliable. Third, it is checked whether the diameter of the fitted circle lies within a specific range to discard anomalies. I.e., if there is *a priori* information about the distribution of the diameters within the plot, it could be reasonable to discard computed diameters outside the range of that distribution. These first three checks are illustrated in Figure 8.

Figure 8: Figure caption.

Finally, the circles fitted along the stems are expected to follow an approximately linear sequence which, however, does not necessarily have to be completely straight, nor vertical. To assess that, an indicator value based on assumed locally coherent inclinations is generated. To derive the indicator value, the tilt angle of each section, compared with all other sections, is computed, looking for local outlier inclinations. An important property of this approach as compared to a simpler approach (i.e., just checking for deviations from a straight standing cylinder) is that it also suitable for leaning stems, a common feature in forests. Figure 9 illustrates this last step.

Figure 9: Figure caption.

An important feature of these checks is that, regardless of the result, the computed diameters are output by *3DFin*. This allows the users to decide, based on further visual inspection, if the diameters adjust well to the stem.

3. Software Architecture and Implementation

3DFin implements the custom algorithm described in Section 2. The program has been written using several popular *Python* libraries to efficiently process the point clouds and compute the tree parameters, including *numpy* (Harris et al., 2020), *scipy* (Virtanen et al., 2020) and *scikit-learn* (Pedregosa et al., 2011). The repository and source code of *3DFin* can be found in https://github.com/3DFin/3DFin. The repository also contains an online copy of the documentation of the software, as well as a link to a tutorial on how to use *3DFin*.

3DFin has been bundled into a user-friendly, free and open-access program equipped with a GUI. This facilitates intuitive interaction with the software and makes *3DFin* more accessible to a wide range of users, including those without a strong technical background. The GUI is divided in three main tabs: Basic, Advanced and Expert. The tabs offer the users options to modify how the data is processed. These options include how the data is input and output, and how the algorithm is applied to the data through several parameters. *3DFin* comes with a set of predefined values, which aims to reduce the expertise required to run the program successfully. These predefined values have been chosen by the developers based on trial and error. The appearance of *3DFin*'s GUI is depicted in Figure 10.

Figure 10: Figure caption.

3DFin is available on *Windows* and *Linux* machines as a plugin in *CloudCompare* via the *CloudCompare PythonRuntime* (Montaigu, 2024). The latest alpha-version of *CloudCompare* (version 2.13.1, March 2024) including the *3DFin* plugin can be downloaded from the official site https://www.danielgm.net/cc/release/. 3DFin is also downloadable on *Windows* as a standalone program from https://github.com/3DFin/3DFin/releases. Additionally, *3DFin* and its dependencies may be installed and launched on any OS (Windows, Linux and macOS) as a *Python* package, available in *PyPI*. A script entry point is also installed by *pip* in *Python* installation's bin | script directory. This enables launching *3DFin*'s GUI from the command line, which avoids the need to write *Python* code to execute *3DFin* if it is installed via this method. Finally, a plugin in *QGIS* is also available at https://github.com/3DFin/3DFin-QGIS.

A console is used by *3DFin* to prompt details about the run when a point cloud is being processed. This can be the built-in console of *CloudCompare*, the default system console in the standalone and *Python* versions, or the built-in console of *QGIS*.

3.1 Inputs and Outputs

3DFin's main input is a ground-based point cloud from a forest plot. It can come from terrestrial photogrammetry, TLS, MLS, a combination of those, and/or a combination of those with data gathered from aerial platforms (unmanned aerial vehicles -UAV- and/or airborne photogrammetry or laser scanning -ALS-). In the standalone and *Python* versions of *3DFin* as well as in the *QGIS* plugin, the input point cloud must be a *LAS / LAZ* file. *LAS* is a standardized file format used for storing and exchanging point cloud data, while *LAZ* is a compressed version of the former. *LAS* versions 1.2, 1.3 and 1.4 are accepted by the software. The input file may contain extra fields (*LAS* standard or not). On the other hand, the *CloudCompare* plugin can process any point cloud format compatible with *CloudCompare*.

The main outputs of the program are point clouds and tabular data. Several *LAS* files are output by the standalone and the *Python* package versions of the program to store the point clouds. In the *CloudCompare* plugin and in the *QGIS* plugin, several in-memory entities are produced in the current running instance. The tabular data, which contain the numeric results of the computations, are output as a single *XLSX* file or as several *TXT* files.

The output point clouds include a point cloud with the detected stems in the horizontal stripe, a point cloud containing the computed tree axes, a point cloud containing the THs, a point cloud where all the original points are kept, but that is enriched with additional scalar fields (distance to closest tree axis, normalized height, tree ID), a point cloud containing the computed diameters, a point cloud containing tree locators, and a DTM. The output point clouds produced by 3DFin are illustrated in Figure 11.

The numeric outputs include a T × 4 table where T is the number of trees, that contains the computed DBH (in m), the computed TH (in m) and (x, y) coordinates of each tree; a 1 × S table, where S is the number of section heights, that contains the heights at which stem diameters have been computed; and several tables displaying information about these diameters. These latter tables are of T × S dimensions, and they contain the computed diameters, their location, and the quality indicators described in Section 2.4. Figure 12 shows a schematic view of the numeric outputs of *3DFin*.

Figure 11: Figure caption.

Figure 12: Figure caption.

4. Performance

To evaluate the efficacy and robustness of *3DFin* for deriving essential tree metrics from ground-based point cloud data, a comprehensive testing suite across diverse data has been employed. Here we evaluate the program's performance, assessing its accuracy, adaptability, and potential limitations. By subjecting the algorithm to a range of scenarios involving distinct environmental settings and tree species compositions, we aim to provide a comprehensive understanding of its capabilities and ascertain *3DFin*'s utility as a user-friendly yet versatile tool for accurate tree attribute estimation.

This testing has been carried out on an OMEN by HP Laptop 17-ck1xxx with the following specifications: 12th Gen Intel®Core™ i9-12900H, 2500 Mhz, 14 Cores x64-based processor with a NVIDIA GeForce RTX 3080 Ti Laptop GPU, with 32 GB of RAM and 8 x 2 GHz processing units, Windows®11 Home operating system version 10.0.22621 64-bit. *CloudCompare* has been used for the visualization of the point clouds, the processing of the data and the extraction of the tree metrics has been done in *3DFin v0.3.2*, and the analysis of the results has been carried out in the statistical computing software *R*.

4.1 Dataset

For testing 3DFin, ground-based point clouds from forest plots measured during the SilviLaser conference in Vienna 2021 (Hollaus & Chen; 2023) were used. During the Silvilaser 2021 conference, over 100 point clouds were acquired in the Vienna Woods (Vienna, Austria) and made public (Hollaus & Chen; 2023). From these, ten point clouds from four 25 m radius circular plots (A1, A2, C1 and D1; Table 1) were selected. These forest plots were scanned using three different ground-based technologies: TLS, MLS and photogrammetry. The TLS points clouds used were acquired with a Riegl VZ-400i device, and the MLS point clouds were captured with a GeoSlam ZEB Horizon RT handheld device. The Riegl VZ-400i has a range of up to 800 m, a field of view (FOV) of 100º x 360 º and captures up to 500,000 points per second with a relative accuracy of up to 3 mm. The MLS device has a range of 100 m, a field of view (FOV) of 360º x 270 º and it can capture 300,000 points per second with a relative accuracy of up to 6 mm. Additionally, for plots A1 and A2, photogrammetric point clouds captured from a multi-camera setup were used as well to test the capability of *3DFin* to process this particular type of data. Photos of the forest sites where the four plots were set are shown in Figure 13.

Figure 13: Figure caption.

The selected plots encompass a wide range of forest conditions, such as different tree species, forest structures or age classes. More specifically, A1 is a dense, mixed forest plot with abundant deadwood, where Norway spruce (*Picea abies*) is the predominant species (97 trees), although there are some red beech (*Fagus sylvatica*) (7), fir (*Abies alba*) (4), pine (*Pinus* spp.) (2), and European larch (*Larix decidua*) (1) trees. The DBH of the trees ranges from ~10 cm to ~40 cm. A2 is a dense, coniferous forest plot with lower species diversity, with the vast majority of individuals being spruce and larch trees (102), in addition to 1 pine tree. The DBH of the trees ranges from ~14 to ~42 cm. C1 is a natural regeneration, mixed forest plot that features trees from five different species, both broadleaf and coniferous. It includes red beech (26), spruce (23), black alder (*Alnus glutinosa*) (10), fir (6), and ash (*Fraxinus excelsior*) (1) trees, and the DBH of the trees varies from ~10 cm to ~77 cm. Lastly, D1 is a multi-layer, mixed forest plot consisting of mostly fir (11) and spruce (7) trees, and 4 broadleaf trees (2 oaks, 2 red beech trees). The DBH ranges from ~20 to ~74 cm in plot D1. Table 1 presents a summary of the characteristics of each plot.

Table 1: Table caption.

Finally, one important consideration about these data is that they are already public, and they had been acquired transparently. This allows easier replicability of the results presented here from any interested user. The dataset is freely available at https://researchdata.tuwien.ac.at/records/kndye-egv02 (Hollaus & Chen; 2023). It consists of one metadata file, the ground-based point clouds, ALS data, and corresponding DTMs derived from the ALS data. The point clouds are in *LAZ* 1.4 format and the files containing the TLS point clouds are SL21BM_TER_046, SL21BM_TER_047, SL21BM_TER_050 and SL21BM_TER_052. The MLS point clouds are stored in files SL21BM_TER_001, SL21BM_TER_002, SL21BM_TER_005 and SL21BM_TER_007 and the photogrammetric point clouds are SL21BM_TER_102 and SL21BM_TER_103. The field-based reference measures of the trees have been kindly made available by the authors of the dataset and include the position of the trees using (x, y) coordinates, the DBH, the tree species and a dead-alive indicator.

4.2 Performance Metrics

To comprehensively assess the performance of *3DFin*, a set of standard metrics and benchmarks were employed. Specifically, to validate the tree mapping, completeness and correctness were calculated. Completeness of the tree mapping is a measure of how many of the reference trees were detected by the algorithm, and correctness measures how many of the trees detected by the algorithm were actual reference trees. To match the trees, the (x, y) coordinates of the reference trees provided by the authors of the dataset were compared to those detected by *3DFin*. The latter are available in the *XLSX* file output by the program. The chosen metrics are simple, yet powerful measurements commonly employed within the forestry community: examples of their use can be found, among others, in Cabo et al. (2018), Liang et al. (2018), Prendes et al. (2021), Krisanski et al. (2021) and Montoya et al. (2021). Completeness and correctness were computed using the following formulas:

$$Completeness = \frac{n_{mat}}{n_{ref}} * 100,$$

$$Correctness = \frac{n_{mat}}{n_{det}} * 100,$$

where n_{ref} is the number of reference trees, n_{det} is the number of trees detected by the algorithm and n_{mat} is the number of matched trees; that is, reference trees that were detected by the algorithm. It might happen that the algorithm misses some tree(s) or that it mistakes some other objects as a tree. In these two situations, both completeness and correctness would be lower than 100 %.

Additionally, the accuracy of the extracted DBHs was evaluated by comparing them to the field-based reference measures. The Root Mean Squared Error (RMSE) and the bias were calculated to quantify the algorithm's accuracy. The RMSE gives an idea of how much error has been incorporated, in average, into the estimates. It is computed using the following formula:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

where n is the number of observations or data points, y_i represents the observed or actual value for the *i*-th data point (the true DBH of the reference tree) and $\hat{y}_i$ represents the predicted or estimated value for the *i*-th data point (the computed DBH value for the detected tree).

Bias, on the other hand, refers to the systematic error or deviation of the estimator (in this case, *3DFin*'s algorithm) from the true value of a population parameter (in this case, the true DBH). A remark of the bias is that it is signed, where positive values indicate that there has been an overestimation of the population parameter, and negative values indicate that there has been an underestimation. It can be computed using the following formula:

$$Bias = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)$$

To put the computed RMSE and bias into perspective, two complementary metrics were computed. To express the RMSE as a percentage of the reference DBH, the following formula was used:

$$RMSE\ (\%) = \frac{RMSE}{\bar{y}} * 100,$$

where $\bar{y}$ is the mean of the reference DBH. A similar analysis has been performed with the bias, using:

$$Bias\ (\%) = \frac{Bias}{\bar{y}} * 100.$$

Finally, the time taken by the algorithm to process the point clouds and generate tree metrics was recorded. This parameter is crucial for real-world applications where efficiency is a concern. To provide an estimation of the time needed to obtain the tree metrics using *3DFin*, every point cloud was processed three times using the software, and the mean time computed and rounded to the nearest integer. The processing time is automatically estimated by *3DFin* and reported (written) in the console when a point cloud is processed.

4.3 *3DFin* Settings

After visual inspection of the point clouds, it was clear that some trees were not referenced by the field operators that originally measured the trees. These include some partially captured, large trees situated at the border of plots A1, C1 and D1, and many thin, inconspicuous trees that are disseminated throughout the plots. Figure 14 and Figure 15 show examples of these.

Figure 14: Figure caption.

Figure 15: Figure caption.

To mimic the criterion followed by the field operators (not including such trees), all partially captured, large trees were identified and removed during the analysis, and the thin trees were identified thanks to the setting of *3DFin* parameters. Plot A1 was processed using default parameters except for "Expert > Computing Sections > Minimum expected diameter", which was set to 0.1 m. Plot A2 was processed using default parameters. Plot C1 was processed using default parameters except for "Expert > Computing Sections > Minimum expected diameter", which was set to 0.09 m. Lastly, Plot D1 was processed with default parameters except for "Basic > Stripe upper limit", which was set to 7.2 m, "Basic > Stripe lower limit", that was set to 4.2 m and "Expert > Computing Sections > Minimum expected diameter", which was set to 0.2 m. The unreferenced trees were discarded before computing the metrics. Four (4) trees, clearly visible in the point clouds, but unreferenced in the field data, were manually removed from plot A1, zero (0) were manually removed from plot A2, two (2) trees were manually removed in plot C1, and five (5) trees were manually removed in plot D1. However, the correctness obtained before removing those trees was computed too. The accuracy of the DBH retrieved by *3DFin* was calculated employing the DBH output in the XLSX file and the in-situ measures. All ten validation point clouds were processed using the standalone version of *3DFin*.

4.4 Results

Table 2 displays the assessment results for the TLS data, Table 3 shows the results for MLS data and Table 4 displays the results from the assessment of the photogrammetric data. In terms of tree mapping, *3DFin* achieved a completeness near or equal to 100% across all plots and technologies, and the correctness after removal of unreferenced trees was near 100 % across the three technologies as well. Moreover, in terms of DBH metrics, *3DFin* yielded an average RMSE of under 2 cm in the TLS (Table 2) and MLS data, performing the best with MLS (averaging a RMSE of 1.66 cm, Table 3). Conversely, a higher average DBH RMSE of 0.0396 meters was reported in the photogrammetric data (Table 4). Regarding bias on the estimation of DBH, *3DFin* achieved values under 1 cm across all technologies. A positive average bias of 0.97 centimetres was reported in the TLS data (Table 2), and a negative average bias of -0.83 cm was yielded in the MLS data. (Table 3) A minimal average bias of 0.41 centimetres was extracted from the processing of the photogrammetric data (Table 4).

Table 2: Table caption.

Table 3: Table caption.

Table 4: Table caption.

The processing times for TLS data are presented in Table 5, the processing times for MLS data are shown in Table 6, and Table 7 displays the times required to process the photogrammetric data. Additional characteristics of the point clouds relevant to this measurement (number of points per plot, number of reference trees in the plot and plot area) are also given. The processing time is notably variable across the plots and technologies. In the case of TLS data, *3DFin* required the highest time (approximately 5 minutes) to process Plot A1, which had the largest number of trees (112), and the fastest processing time (over 2 minutes) was obtained in Plot D1, which had the lowest number of trees (22) (Table 5). For the MLS data, where all plots had similar number of points (around 33-38 million), the plot that took the highest time to process (slightly less than 7 minutes) was Plot A2, whereas the lowest time (over 4 minutes) was obtained in Plot D1 (Table 6). Lastly, the processing time ranged from slightly less than 3 minutes to over 3 minutes in the photogrammetric data (Table 7).

Table 5: Table caption.

Table 6: Table caption.

Table 7: Table caption.

5. Discussion

5.1 User-friendliness

First and foremost, *3DFin* can be seamlessly integrated as a plugin within the popular point cloud processing software, *CloudCompare*, which is downloaded 200,000 - 300,000 times annually according to its official site. This integration simplifies the user experience by embedding the tool directly within an environment that users are already familiar with, reducing the learning curve associated with adopting a new software. In addition, *3DFin* can be used as a standalone program, offering independence from specific point cloud processing platforms and providing users with the flexibility to execute tree metric computations in isolation. Furthermore, for those users who prefer to work within the *Python* ecosystem, *3DFin* is available as a *Python* package, allowing for seamless integration with *Python*-based data analysis pipelines and facilitating automation and scripting of tree metric calculations. Lastly, recognizing the significance of Geographic Information Systems (GIS) in forestry and environmental research, *3DFin* has also been implemented as a plugin within *QGIS*, offering geospatial professionals the ability to incorporate tree metrics directly into their GIS workflows. This multifaceted approach to implementation ensures that *3DFin* is accessible and adaptable to the preferences and requirements of a broad user base, promoting its widespread utility in the analysis of ground-based point cloud data for tree-related research and applications.

5.2 Accuracy

As shown in Tables 2, 3 and 4, *3DFin* has been able to reach completeness and correctness of nearly 100 % across the four plots, which include mixed and coniferous forest that feature a variety of structural characteristics (Table 1, Figure 7), and three kinds of point clouds (TLS, MLS and photogrammetric). An exception is plot D1, where completeness remained under 96 % in the TLS point cloud. Plot D1 features only 22 trees, and one was missed by the program. This tree was detected in the MLS point cloud, though, where the completeness reached 100 % (Table 3). Regarding the DBH values extracted by *3DFin*, these can be considered very accurate, as the RMSE and the bias remained low in all plots and across all data collection technologies (Tables 2-4). It should be noted that RMSE is lower for MLS than for TLS despite the GeoSlam scanner having a lower relative accuracy than the TLS Riegl scanner. A possible explanation for this result may lie in the fact that MLS produces point clouds where stems are scanned all around, which allows the algorithm to determine more clearly which points belong to the stems. Another possible cause is that the co-registration of the TLS scans into a single point cloud can induce small deformations. These deformations might slightly affect the precision of the DBH measurements by altering the spatial relationships between points that represent the tree stems. It was expected, however, that the DBH RMSE would be highest for the photogrammetric point clouds, as they are visibly the noisiest among the three technologies. The bias on the DBH estimation remained low across all point clouds (less than 1 cm, which accounts for less than 3 %). In addition to the results presented here, the capabilities of the initial versions of the algorithm have been assessed before. Cabo et al. (2018) showed that the initial version of the algorithm was able to achieve nearly 100 % tree mapping completeness and correctness in the plots that they tested. As to DBH and TH, the RMSE of the algorithm estimations ranged from 0.8 cm to 1.3 cm and from 0.3 m to 0.7 m, respectively. Similarly, Prendes et al. (2021) obtained comparable results presenting 97 % tree mapping completeness and 100 % correctness, as well as 1.14 cm RMSE in DBH estimation and 1.52 m RMSE in TH estimation. In both studies, the point clouds were acquired via TLS devices. Although those results are not directly applicable to the current version of the algorithm described here, which has undergone improvements in the robustness and speed of the computations since they were initially published (Cabo et al., 2018; Prendes et al., 2021), they might be seen as a reinforcement of the positive results of *3DFin*.

Other authors have also reported completeness, correctness and DBH RMSE values of algorithms that compute tree metrics in ground-based point clouds. Liang et al. (2018) compared the performance of 18 algorithms that compute tree metrics in multiple-scan TLS point clouds. These point clouds were divided in easy, medium, and hard difficulty by the authors. Across the algorithms that reported completeness and completeness, the best performant produced 90.4 % completeness with 93.6 % correctness across the easy plots, 88.0 % completeness paired with 89.2 % correctness in medium plots and 66.2 % completeness coupled to 92.8 % correctness in the hard plots. Among the 14 algorithms that produced DBH measurements, the best performing algorithms reported DBH RMSE of 2 cm in the easy plots, which equated to a 5-15 % of the mean DBH value. Nevertheless, the averaged DBH RMSE obtained by the 14 algorithms in the easy plots was, approximately, 5.3 cm (24.97 %). This value increased to 6.77 cm (34.98 %) average RMSE in the medium plots and up to 10.17 cm (53.70 %) in the hard plots. Montoya et al. (2021) employed these same plots in their study and reported 2.83-3.25 cm mean DBH RMSE across all plots, paired with 82.0 % completeness and 84.0 % correctness in the easy plots, 66.0 % completeness and 86.0 % correctness in the medium plots and 52.5 % completeness and 91.0 % correctness in the hard plots. Krisanski et al. (2021) reported 7.2 cm DBH RMSE and 90.98% completeness across 49 point clouds of 12 trees each, acquired with multiple scan TLS. Table 8 shows those results and the results obtained with 3DFin for easier comparison.

Table 8: Table caption.

While the plots, trees, and point clouds employed in these previous studies may not offer direct comparability to one another, nor to the test data utilized in this investigation, the numerical outcomes regarding completeness, correctness and DBH, calculations achieved by *3DFin* are, at a minimum, on par with the top-performing results obtained in the evaluation of previous algorithms.

5.3 Processing time

For our tested dataset, the maximum processing time was under 7 min, with the lowest processing time being over 2 min. It is difficult to compare these results to the processing times achieved by other software / algorithms, as the total processing time is rarely reported. Only one of the software programs that produce tree metrics from ground-based point clouds described in Section 1 reported processing times. Krisanski et al. (2021) reported a processing time using *FSCT* of up to 60 min in TLS and MLS point clouds from forest plots of 12 trees. This was achieved using a high-end pc with Intel i9-10900K (overclocked to 4.99GHz in all cores) CPU, NVIDIA Titan RTX (24 GB RAM) GPU and 128 GB DDR4 at 3200 MHz RAM. Trochta et al. (2017) did not report computing times of the whole process of extracting tree metrics with *3DForest*; however, in a later study, Klemt et al. (2021) used *3DForest* and reported that processing a point cloud with this software and producing a table with the tree metrics took 3-4 h for unexperienced users of *3DForest*, and 1 h for experienced users. The point cloud used in this study was acquired from a 50 × 50 m forest plot using a Leica BLK 360 terrestrial laser scanner in multiple scan positions and was downsampled to keep 1 of every 5 points. The specifications of the computer used to process the point cloud were not described. Although these times are not fully comparable to the processing times obtained with *3DFin*, as the datasets are different in terms of number of trees and the processing power of the computers used in each study are different, *3DFin* provides the fastest computing times among the three tools by a large margin.

5.4 Known limitations

It is worth noting that *3DFin*, while providing powerful tree metric computation capabilities, may have certain limitations inherent to the problem that it aims to solve. The accuracy of the computed tree metrics relies on the quality and completeness of the input point cloud data. As a result, noisy or incomplete data may affect the accuracy of the results (Liu et al., 2017). This effect is noticeable in the results obtained from the photogrammetric data, were the RMSE of the computed DBH was much higher than from the LiDAR data. Moreover, processing large-scale point cloud data may require significant computational resources, including memory and processing power. It is recommended to use a system with at least 16 GB of RAM to run *3DFin* on average-sized point clouds (50 million points or lower) and at least 32 GB of RAM to process larger clouds. A final limitation is that *3DFin* is specifically designed for processing ground-based point clouds obtained through techniques such as TLS, MLS or photogrammetry, as it relies heavily on a good representation of the ground and lower parts of the stems. Thus, it is not suitable for aerial or satellite-based point cloud data alone, where the ground and stems are often underrepresented in comparison to the tree canopy.

5.5 Future Development

The development of 3DFin highlights an important evolution in utilizing terrestrial point clouds for forest inventories, aiming for increased automation and precision. Looking into the future, research and development efforts will focus on both enhancing the existing capabilities of *3DFin* and exploring new avenues to broaden its application in forest management and ecological studies. Progressing with these developments, a central tenet of the 3DFin project remains to enhance the usability and accessibility of our software. Our aim is to ensure that 3DFin is approachable and user-friendly, even for those who may not have specialized expertise in geomatics or computer science. This commitment to inclusivity is reflected in our ongoing efforts to improve 3DFin's integration with open-source platforms, such as CloudCompare and QGIS, thereby making advanced geospatial analysis and data processing capabilities more accessible to a wider range of users.

One of the primary objectives in the next phase of the development of *3DFin* is to include tree volume estimation functionalities. Current capabilities such as the computation of DBH, TH, and diameters at various section heights, lay a solid foundation for estimating tree volume. In this sense, an important aspect of ongoing research will be to rigorously test and validate the capabilities of *3DFin* in estimating tree height and diameters at various heights along the stem, beyond the standard Diameter at Breast Height (DBH) measurements. Recognizing the critical role these metrics play in forest inventory and ecological research, we aim to conduct extensive field tests to compare *3DFin*'s outputs with ground-truth data across diverse forest types and conditions. This will help to refine the software’s algorithms and ensure its accuracy and reliability in capturing a full range of scenarios.

Alongside, we acknowledge the necessity of conducting a comprehensive sensitivity analysis of *3DFin*'s parameters. Given the software’s complex architecture, a step-by-step sensitivity analysis is imperative to understand the influence of each parameter on the software’s performance. This analysis will be instrumental in optimizing *3DFin*’s settings for different forest environments and operational conditions, thus enhancing the robustness and adaptability of the software. Although this analysis has not yet been conducted due to the sheer number of modifiable parameters, future research will prioritize this task. By systematically evaluating the impact of each parameter, we aim to provide users with clear guidelines and optimized presets that facilitate the effective application of *3DFin* in various forest inventory scenarios.

Another direction for the research linked to 3DFin is the development of a complementary software tool focused on the semantic segmentation of point clouds into different vegetation structures. This tool will build upon the capabilities of *3DFin*, employing advanced deep learning techniques to distinguish between various types of vegetation elements within a forested scene. The segmentation results can greatly enhance the accuracy of forest inventories, ecological studies, and habitat assessment, providing valuable insights into forest structure. This advancement, together with the volume computations, will enable more comprehensive biomass assessments and contribute to carbon stock estimation, enhancing the utility of the software in sustainable forest management and climate change research.

Conclusions

Here we present *3DFin*, a user-friendly, free, and open-source software designed for automatic 3D forest inventory using ground-based point clouds. The aim of this program is to offer a flexible and accessible set of tools for computing tree metrics, ensuring compatibility with various platforms and *software ecosystems*. *3DFin* is designed to accommodate diverse user preferences and workflows, catering to the needs of researchers and practitioners in forestry and environmental sciences. The current implementation of *3DFin* provides reliable and efficient results with minimal user input and parametrization.

3DFin marks a notable progression in the automation and accessibility of forest inventories through ground-based point clouds. This software simplifies the inventory process, maintaining a “two-click software” approach, while ensuring precision and reliability in the results. The streamlined approach offered by *3DFin* holds potential for enhancing forest management efficiency and facilitating informed decision-making. Additionally, its integration with widely used software for processing ground-based remote sensing data opens up new possibilities for forest resource assessment and monitoring.

Funding

This work was supported by the UK NERC project [NE/T001194/1]: 'Advancing 3D Fuel Mapping for Wildfire Behaviour and Risk Mitigation Modelling', the Spanish Knowledge Generation project [PID2021-126790NB-I00]: ‘Advancing carbon emission estimations from wildfires applying artificial intelligence to 3D terrestrial point clouds’, and the Spanish “Ramón y Cajal” programme [RYC2018-025797-I].

Acknowledgement

The authors wish to thank Markus Hollaus and Yi-Chen Chen for providing the reference data for the point clouds used in this paper (Silvilaser 2021 Benchmark Dataset). Additionally, we extend our gratitude to Daniel Girardeau-Montaut and Thomas Montaigu for their involvement in the development of the *CloudCompare* plugin. Finally, we would like to express our sincere gratitude to the editor F.F. and the anonymous reviewers for their insightful feedback and constructive suggestions, which significantly contributed to improving the quality of this paper.

Conflicts of Interest

The authors declare no conflict of interest. The funders had no role in the design of the study, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

Data availability statement

The data underlying this article are available in TU Wien Research Data, at https://doi.org/10.48436/afdjq-ce434.

References

Bentley J.L. 1975. Multidimensional Binary Search Trees Used for Associative Searching. In Communications of the ACM. **18**.

Cabo C. et al. 2014. An algorithm for automatic detection of pole-like street furniture objects from Mobile Laser Scanner point clouds. ISPRS J. Photogramm. Remote Sens. **87**: 47–56.

Cabo C. et al. 2018. Automatic dendrometry: Tree detection, tree height and diameter estimation using terrestrial laser scanning. Int. J. Appl. Earth Obs. Geoinf. **69**: 164–174.

Calders K. et al. 2020. Terrestrial laser scanning in forest ecology: Expanding the horizon. Remote Sens. Environ. **251**.

Carpenter J. et al. 2022. An Unsupervised Canopy-to-Root Pathing (UCRP) Tree Segmentation Algorithm for Automatic Forest Mapping. Remote Sens. **14**.

Chen Y. et al. 2021. 3D point cloud semantic segmentation toward large-scale unstructured agricultural scene classification. Comput. Electron. Agric. **190**.

de Conto T. et al. 2017. Performance of stem denoising and stem modelling algorithms on single tree point clouds from terrestrial laser scanning. Comput. Electron. Agric. **143**: 165–176.

Dassot M. et al. 2011. The use of terrestrial LiDAR technology in forest science: Application fields, benefits and challenges. Ann. For Sci. **68**: 959–974.

Ester M. et al. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*.

GeoSLAM (A FARO Technologies, Inc. Company) 2023. Geoslam ZEB Horizon RT. Available online: https://geoslam.com/solutions/zeb-horizon-rt/.

GreenValley International 2013. LIDAR360. Available online: https://www.greenvalleyintl.com//LiDAR360/.

Hackel T. et al. 2016. Contour detection in unstructured 3D point clouds. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1610–1618.

Harris C.R. et al. 2020. Array programming with NumPy. Nature. **585**: 357–362.

Hollaus, M., & Chen, Y.-C. 2023. SilviLaser 2021 Benchmark Dataset - Terrestrial Challenge (1.1) [Data set]. TU Wien. https://doi.org/10.48436/afdjq-ce434

Iglhaut J. et al. 2019. Structure from Motion Photogrammetry in Forestry: a Review. Current Forestry Reports. **5**: 155–168.

Klemmt, H.J. et al. 2021. Anwendung der Software 3D Forest auf TLS-basierte Waldmessdaten. AFZ-Der 33–37.

Koreň M. et al. 2017. Accuracy of tree diameter estimation from terrestrial laser scanning by circle-fitting methods. Int. J. Appl. Earth Obs. Geoinf. **63:** 122–128.

Krisanski S. et al. 2021. Forest structural complexity tool—an open source, fully-automated tool for measuring forest point clouds. Remote Sens. **13**.

Liang X. et al. 2016. Terrestrial laser scanning in forest inventories. ISPRS J. Photogramm. Remote Sens. **115**: 63–77.

Liang X. et al. 2018. International benchmarking of terrestrial laser scanning approaches for forest inventories. ISPRS J. Photogramm. Remote Sens. **144**: 137–179.

Liang X. et al. 2022. Close-Range Remote Sensing of Forests: The state of the art, challenges, and opportunities for systems and data acquisitions. IEEE Trans. Geosci. Remote Sens. **10**: 32–71.

Liu J. et al. 2017. Automated matching of multiple terrestrial laser scans for stem mapping without the use of artificial references. Int. J. Appl. Earth Obs. Geoinf. **56**: 13–23.

López Serrano F.R. et al. 2022. Artificial intelligence-based software (AID-FOREST) for tree detection: A new framework for fast and accurate forest inventorying using LiDAR point clouds. Int. J. Appl. Earth Obs. Geoinf. **113**.

Mokros, M. and Koreň, M. 2019. DendroCloud: Free Terrestrial-based Point Cloud Processing Software for Forestry. Available online: https://gis.tuzvo.sk/dendrocloud/default.aspx.

Molina-Valero J.A. et al. 2022. Operationalizing the use of TLS in forest inventories: The R package FORTLS. Environ. Model. Softw. **150**.

Montaigu T. 2024. CloudCompare-PythonRuntime. Available online: https://tmontaigu.github.io/CloudCompare-PythonRuntime/.

Montoya O. et al. 2021. TreeTool: A tool for detecting trees and estimating their DBH using forest point clouds. SoftwareX. **16**.

Newnham G.J. et al. 2015. Terrestrial laser scanning for plot-scale forest measurement. Curr. For. Rep. **1**: 239–251.

Pascu I.S. et al. 2019. Estimating forest stand structure attributes from terrestrial laser scans. Sci Total Environ. **691**: 205–215.

Pedregosa F. et al. 2011. Scikit-learn: Machine Learning in Python. J. Mach. Learn. Res. **12**.

Pfeifer N. et al. 2014. OPALS - A framework for Airborne Laser Scanning data analysis. Comput. Environ. Urban Syst. **45**: 125–136.

Piboule A. et al. 2013. Computree: a collaborative platform for use of terrestrial lidar in dendrometry. In: *Proceedings of the International IUFRO Conference MeMoWood*, 1–4.

Prendes C. et al. 2021. An algorithm for the automatic parametrization of wood volume equations from Terrestrial Laser Scanning point clouds: application in Pinus pinaster. GIsci. Remote Sens. **58**: 1130–1150.

R Core Team. 2023. R: A language and environment for statistical computing. R Foundation for Statistical Computing. Available online: https://www.R-project.org/.

Rapidlasso GmbH. 2023. Lastools. Available online: https://lastools.github.io/.

Ravaglia J. et al. 2019. Comparison of three algorithms to estimate tree stem diameter from terrestrial laser scanner data. Forests. **10**.

Ridder, R.M. Global Forest Resources Assessment. 2010. Options and Recommendations for a Global Remote Sensing Survey of Forests. FAO Forest Resource Assessment. **141**. Available online: http://www.fao.org/3/a-ai074e.pdf.

Riegl Laser Measurement Systems GmbH. 2023. Riegl Vz-400i. Available online: http://www.riegl.com/nc/products/terrestrial-scanning/produktdetail/product/scanner/48/.

Riverbank Computing Limited. 2023. PyQt. Available online: https://www.riverbankcomputing.com/software/pyqt/.

Sadeghian H. et al. 2022. Estimating the attributes of urban trees using terrestrial photogrammetry. Environ. Monit. Assess. **194**.

The MathWorks Inc. 2023. MATLAB. Available online: https://www.mathworks.com.

Trochta J. et al. 2017. 3D Forest: An application for descriptions of three-dimensional forest structures using terrestrial LiDAR. PLoS One. **12**.

Van Laar, A. and Akça, A. 2007. Forest Mensuration; Springer. **13**: 360–383.

Virtanen P. et al. 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods. **17**: 261–272.

Wang D. et al. 2021. Individual tree extraction from terrestrial laser scanning data via graph pathing. For. Ecosyst. **8**.

Wang F. & Bryson M. 2023. Tree Segmentation and Parameter Measurement from Point Clouds Using Deep and Handcrafted Features. Remote Sens. **15**: 1086.

Windrim L. & Bryson M. 2020. Detection, segmentation, and model fitting of individual tree stems from airborne laser scanning of forests using deep learning. Remote Sens. **12**.

Wulder M.A. 2004. Encyclopedia of Forest Sciences; Elsevier. RESOURCE ASSESSMENT | GIS and Remote Sensing. 997–1001.

Zhang W. et al. 2016. An easy-to-use airborne LiDAR data filtering method based on cloth simulation. Remote Sens. **8**.

Table and Figure captions

Figure 1: The first step of the algorithm is to normalize the input point cloud. A) Original point cloud. B) Height-normalized point cloud. **Line 130**

Figure 2: Effect of denoising the point cloud before computing the DTM through Cloth-Simulation Filter. This allows to generate a more accurate DTM, which in turn improves the computation of the tree metrics by 3DFin. A) The original noisy point cloud. B) the denoised point cloud. Note how noise above the canopy is also removed. C) A faulty DTM, product of applying the CSF to the noisy point cloud. D) A correctly generated DTM produced by applying the CSF filter to the denoised point cloud. **Line 146**

Figure 3: Identification of stems within the horizontal stripe from the normalized point cloud, which is the second main step of the algorithm. A) Horizontal stripe is defined by two Z0 values. B) Verticality is computed for each point in the stripe, using fixed-radius neighbourhoods of points. C) Points with low verticality values are discarded. D) The remaining points are clustered using DBSCAN algorithm. E) Small clusters of points are discarded. B, C, D and E are repeated iteratively. F) The points that have not been discarded (those with high verticality and that remained in large clusters simultaneously) are regarded as the bases of the stems. **Line 163**

Figure 4: The third step of the algorithm, where stem axes are computed and every point in the point cloud is mapped to one of these axes. This serves as a proxy to "individualize" the trees and compute tree metrics on each of them. THs are computed at this stage. A) Computed axes for each stem. B) Mapping of points to closest axis. **Line 174**

Figure 5: TH measurement. A) Set of points that have been mapped to the same tree axis. B) Voxelization of the points. C) Clustering and filtering of the voxels, discarding small clusters. D) Voxels further than a certain threshold from the tree axis are discarded. The normalized (z) value of the highest remaining voxel is used as TH. **Line 183**

Figure 6: General overview of the fourth and last step of the algorithm, where the stems are identified and their diameter computed. To identify the stems, the limbing algorithm detailed in Section 2.2 is applied to every

tree to remove branches. To compute the sections, circles are fitted to the stems through least-squares minimization. A) Point cloud after the tree individualization described in Section 2.3. B) Computed sections for each tree. C) Detail of the computed sections. These are represented as points that form circles. **Line 198**

Figure 7: Identification of whole stems along tree axes. The limbing algorithm detailed in 2.2 is applied to every tree to remove branches. A) Points that share a common tree axis, which are used as a proxy to the whole tree. B) A verticality value is assigned to each point based on the geometrical structure of its point neighbourhood. C) Points with low verticality and small clusters of points are disregarded. B and C may be repeated iteratively. D) Points belonging to the stem. **Line 206**

Figure 8: Quality checks described above. Two sections (A and B) are used to illustrate the quality checks. Section A (top) passes all checks: there are not points inside the inner circle (top-left), a large proportion of sectors are occupied by points (13/16 in this example) (top-centre) and the diameter of the fitted circle lies within the expected boundaries (in this example 6 cm and 50 cm are used as lower and upper boundaries, respectively). Section B (bottom) does not pass any of the checks. There are several points inside the inner circle (bottom-left), only a small proportion of the sectors are occupied (2 / 16) (bottom-centre) and the diameter of the fitted circle is larger than the upper boundary (bottom-right). **Line 234**

Figure 9: Detection of outliers. Two cases are illustrated: Tree A shows stem sections (1, 2, 3… 7) of a tree with no outliers, whilst the Tree B shows the sections of a tree where there is an outlier section (section 5). The sections are represented by blue ellipses. The tilt angles (symbolized by the black arrows and red arcs) of the visualized stem sections of Tree A are all very comparable and hence the indicator would not identify an outlier here. For Tree B, the outlier section produces abnormally large / small angles. These will increase the outlier probability described above. **Line 243**

Figure 10: 3DFin's GUI, which consists of three tabs: Top left image: Basic tab. Top right image: Advanced tab. Bottom image: Expert tab. **Line 265**

Figure 11: 3DFin inputs and main point cloud outputs, illustrated with a point cloud of a single tree that has been processed with the software. A) Input, raw point cloud, B) Stems identified in the horizontal stripe.

C) Tree axis. D) Enriched point cloud and tree height. The enriched point cloud includes computed scalar fields (normalized height, distance to closest tree axis, tree ID). E) Computed diameters, including DBH. Sections coloured in blue pass the quality checks detailed in Section 2.4, while sections in red do not. Output point cloud containing the DTM is not illustrated here. **Line 303**

Figure 12: Schematic view of the numeric outputs produced by 3DFin. **Line 304**

Figure 13: Photos of the forest sites in Vienna Woods (Vienna, Austria) where the point clouds were captured from. Plots A1 and A2 are part of Site A (Top-left figure). Plot C1 is part of Site C (top-right figure). Plot D1 is part of Site D (bottom figure). Source: https://silvilaser2021.at/benchmark/. **Line 334**

Figure 14: Unreferenced trees in plot A1. Marked in blue, two large, partially captured trees that are not included in the reference data but are present in the point clouds. Colours have been assigned according to distance to closest tree axis, which is computed by 3DFin. **Line 407**

Figure 15: Unrefenced trees in plot C1. Marked in blue, thin, young trees that had not been included in the field-based reference dataset. Colours according to distance to closest tree axis, which is computed by 3DFin. **Line 408**

Table 2: A summary of the four 25 m radius plots employed to assess 3DFin' performance. **Line 347**

Table 2: Results of the assessment of *3DFin* on the point clouds acquired with TLS (Riegl VZ-400i). Correctness* is the correctness before removing the unreferenced trees. **Line 435**

Table 3: Results of the assessment of *3DFin* on the point clouds acquired with MLS (GeoSlam ZEB Horizon RT). Correctness* is the correctness before removing the unreferenced trees. **Line 436**

Table 4: Results of the assessment of *3DFin* on the point clouds acquired with the multi-camera setup (photogrammetry). Correctness* is the correctness before removing the unreferenced trees. **Line 437**

Table 5: Time required to process the point clouds acquired with TLS (Riegl VZ-400i) (bold), processing times, the number of points in the cloud, the number of trees present, and the area of the plot. **Line 448**

Table 6: Time required to process the point clouds acquired with MLS (GeoSlam ZEB Horizon RT) (bold), processing times, the number of points in the cloud, the number of trees present, and the area of the plot. **Line 449**

Table 7: Time required to process the point clouds acquired with the multi-camera setup (photogrammetry) (bold), processing times, the number of points in the cloud, the number of trees present and the area of the plot. **Line 450**

Table 8: Comparison of our results (3DFin, TLS; 3DFin, MLS and 3DFin, Photogrammetry) versus the results presented in Liang et al. (2018), Montoya et al. (2021) and Krisanski et al. (2021). **Line 511**

Tables

Table 1

Plot Characteristics	**Plot A1**	**Plot A2**	**Plot C1**	**Plot D1**
Nº of trees	112	103	67	22
Nº of species	6	3	5	4
Approx. age (years)	50	50	120	120
Forest type	Mixed	Coniferous	Mixed	Mixed
Other	Deadwood	Deadwood	Natural regeneration	Multi-layer
Standing Dead trees	Yes	No	No	Yes
Mean DBH (m)	0.2437	0.2702	0.3624	0.5505
Min DBH (m)	0.1085	0.1455	0.1030	0.2050
Max DBH (m)	0.4075	0.4235	0.7680	0.7400

Table 2

Riegl VZ-400i	**Plot A1**	**Plot A2**	**Plot C1**	**Plot D1**	**Average**
Completeness (%)	100	100	100	95.45	98.86
Correctness (%)	100	99.04	100	100	99.76
Correctness* (%)	96.55	99.04	97.10	80.77	93.37*
DBH RMSE (m)	0.013	0.015	0.019	0.022	0.0175
DBH RMSE (%)	5.39	5.58	5.42	4.07	5.12
DBH Bias (m)	0.007	0.008	0.012	0.013	0.0097
DBH Bias (%)	2.64	3.01	3.27	2.33	2.81

Table 3

GeoSlam ZEB Horizon RT	Plot A1	Plot A2	Plot C1	Plot D1	Average
Completeness (%)	100	100	100	100	100
Correctness (%)	100	99.04	98.5	100	99.39
Correctness* (%)	96.55	99.04	95.65	84.61	93.96*
DBH RMSE (m)	0.016	0.012	0.021	0.018	0.0166
DBH RMSE (%)	6.37	4.49	5.89	3.21	4.99
DBH Bias (m)	-0.012	-0.007	-0.007	-0.007	-0.0083
DBH Bias (%)	-4.73	-2.72	-1.95	-1.30	-2.68

Table 4

Multi-camera	Plot A1	Plot A2	Average
Completeness (%)	99.11	100	99.56
Correctness (%)	98.23	99.04	98.64
Correctness* (%)	95.69	99.04	97.37*
DBH RMSE (m)	0.034	0.045	0.0396
DBH RMSE (%)	13.93	16.67	15.3
DBH Bias (m)	0.010	-0.001	0.0041
DBH Bias (%)	3.96	-0.50	1.73

Table 5

Riegl VZ-400i	Plot A1	Plot A2	Plot C1	Plot D1
Nº of points (mil.)	59.27	48.40	72.58	42.48
Nº of trees	112	103	67	22
Area (m^2)	1409	1401	1464	1425
Processing time (s)	292	257	267	140

Table 6

GeoSlam ZEB Horizon RT	Plot A1	Plot A2	Plot C1	Plot D1
Nº of points (mil.)	35.74	37.53	33.22	35.89
Nº of trees	112	103	67	22
Area (m^2)	1426	1403	1464	1426
Processing time (s)	390	412	266	253

Table 7

Multi-camera	Plot A1	Plot A2
Nº of points (mil.)	67.70	71.04
Nº of trees	112	103
Area (m^2)	1404	1397
Processing time (s)	176	218

Table 8

Study and dataset	Completeness (%)	Correctness (%)	DBH RMSE (cm)
Liang et al. (2018), Easy plots	90.4	93.6	5.3
Liang et al. (2018), Medium plots	88	89.2	6.77
Liang et al. (2018), Hard plots	66.2	92.8	10.17
Montoya et al. (2021), Easy plots	82	84	2.83-3.25
Montoya et al. (2021), Medium plots	66	86	2.83-3.25
Montoya et al. (2021), Hard plots	52.5	91	2.83-3.25
Krisanski et al. (2021)	90.98	Not reported	7.2
3DFin, TLS	98.86	99.76	1.75
3DFin, MLS	100	99.39	1.66
3DFin, Photogrammetry	99.56	98.64	3.96