




# Approach for graph-based individual branch modelling of meadow orchard trees with 3D point clouds

Jonas Straub<sup>1</sup>  · David Reiser<sup>1</sup> · Nils Lüling<sup>1</sup> · Alexander Stana<sup>1</sup> · Hans W. Griepentrog<sup>1</sup>

Accepted: 28 September 2022 / Published online: 8 October 2022  
© The Author(s) 2022

## Abstract

The cultivation of meadow orchards provides an ecological benefit for biodiversity, which is significantly higher than in intensively cultivated orchards. However, the maintenance of meadow orchards is not economically profitable. The use of automation for pruning would reduce labour costs and avoid accidents. The goal of this research was, using photogrammetric point clouds, to automatically calculate tree models, without additional human input, as basis to estimate pruning points for meadow orchard trees. Pruning estimates require a knowledge of the major tree structure, containing the branch position, the growth direction and their topological connection. Therefore, nine apple trees were captured photogrammetrically as 3D point clouds using an RGB camera. To extract the tree models, the point clouds got filtered with a random forest algorithm, the trunk was extracted and the resulting point clouds were divided into numerous K-means clusters. The cluster centres were used to create skeleton models using methods of graph theory. For evaluation, the nodes and edges of the calculated and the manually created reference tree models were compared. The calculated models achieved a producer's accuracy of 73.67% and a user's accuracy of 74.30% of the compared edges. These models now contain the geometric and topological structure of the trees and an assignment of their point clouds, from which further information, such as branch thickness, can be derived on a branch-specific basis. This is necessary information for the calculation of pruning areas and for the actual pruning planning, needed for the automation of tree pruning.

**Keywords** 3D point cloud · Photogrammetry · Tree modelling · Tree pruning

## Introduction

Orchard meadows are an essential part of a small-scale cultural landscape, especially in the south of Germany. In addition, orchard meadows provide valuable habitats for many living organisms. They form an important retreat for many animal and plant species and play a

---

✉ Jonas Straub  
jonas.straub@uni-hohenheim.de

<sup>1</sup> Institute of Agricultural Engineering, University of Hohenheim, Garbenstr. 9, D-70599 Stuttgart, Germany

significant role in biodiversity (Schuboth & Krummhaar, 2019). However, orchard meadows are not currently profitable from an economic point of view, due to the high labour input combined with low profit from the yields. As a result, orchard meadows are usually poorly maintained, thus their future is threatened (Zehnder & Wagner, 2008). In the state of Baden-Württemberg, more than 80% of the meadow orchard trees are currently only irregularly pruned, or are not pruned at all (MLR, 2015). A substantial decline of 17% in the number of meadow orchard trees was recorded within ten years (Borngräber et al., 2020). For the preservation of the trees, regular pruning is necessary. Due to the fact that many meadow orchards are no longer adequately maintained, the knowledge of correct pruning is increasingly lost among many owners (Zehnder & Wagner, 2008). However, the use of external experts who can perform professional pruning is expensive. The use of professional contractors could therefore be supported by automation, thus reducing costs and preserving the cultural landscape. In order to be able to implement automated pruning, the branch structure has to be detected and analysed.

There are already several projects dealing with the semantic interpretation of orchard trees in research. Most of them are using different types of 3D sensor data. For example, Sanz et al. (2018), Tsoulias et al. (2020) and Méndez et al. (2016) used 2D-LiDAR, Alvites et al. (2021) and Sun et al. (2021) used a 3D-LiDAR sensor in the form of a terrestrial laser scanner while Tabb and Medeiros (2017) used stereo vision. In addition, ToF (Time of Flight) cameras are often used in related applications which was shown in a review by He and Schupp (2018). These sensors have their advantages and disadvantages in terms of real-time capability, resolution, noise behaviour, cost, etc. A terrestrial laser scanner provides both a high point density and a high geometric accuracy of the point clouds. In contrast, conventional scanners have a low real-time capability and a relatively high price. In comparison, 2D LiDAR sensors, which are usually much cheaper, deliver a much lower point density and accuracy. In addition, the sensor platform must provide at all times accurate position and orientation information for a registered 3D point cloud. Stereo and ToF cameras can already capture larger 3D point clouds in their image area from one viewpoint, but the accuracy and spatial resolution are often below that of a terrestrial laser scanner.

Based on the sensor data, there are different approaches in research for the analysis of fruit trees. In addition to the sensor technology, such algorithms can be subdivided by the analysis method used and the purpose of the analysis. Sanz et al. (2018) used the volume calculated by a voxel representation to establish a connection with the leaf-area of the trees and for calculating tree heights. Tsoulias et al. (2020) used a LiDAR point cloud for the detection of apples in a plantation orchard. This was used for yield estimation, and for calculating fruit positions, which is essential for many robotic tasks in orchards. Something similar was implemented by Kang and Chen (2020), where apples, as well as the branch structure, were recognised via semantic segmentation with a neural network. For the segmentation itself, only RGB information was used, but the resulting masks can be supplemented by the depth channel when using a depth camera, which then also results in a segmented 3D point cloud. Alvites et al. (2021) on the other hand, calculated a quantification of timber assortments in a Mediterranean mixed forest. In forestry, the conditions are similar to meadow orchards, unordered and unpredictable. However, they are still very different, as the trees have completely different growth forms and the environment is different. Nevertheless, in both cases, the algorithms must have greater robustness to special cases. The basic procedure starts with the initial processing of the point cloud and the removal of uninteresting objects (timber-leaf discrimination) and then proceeds with the detection of the tree trunk and the remaining tree structure using many cylinders. However, the focus there has been on the determination of the quantity of the timber and not

on the connectivity of the individual branches. Another option is a graph-based topological structure of a tree, which has been used by Arikapudi et al. (2015). This was used for the representation of manually digitized trees, but not for the automatic generation of such a model. A distinction was made between the trunk, main branches, and sub-branches to build a model as a hierarchical graph based on this classification. A classification was used in their paper which distinguished instead between trunk, major and minor branches. However, several studies, such as Méndez et al. (2016) and Tabb and Medeiros (2017), used a graph-based representation of an orchard tree for automatic modelling. In these studies, the focus was on the analysis of orchard trees in plantation cultivations that have different preconditions, which means that the structure of these trees does not reach the same complexity as in an orchard meadow and it may therefore be difficult to capture a large meadow orchard tree this way. Another graph-based approach, but with a different computational approach, was from Sun et al. (2021). They used a Laplacian-based contraction method for the generation of skeleton point clouds of cotton plants. Like in the method proposed in this work, they also used a minimum spanning tree to cut circles in their graph. However, the structure of the plants is clearly different from that of the meadow orchard trees.

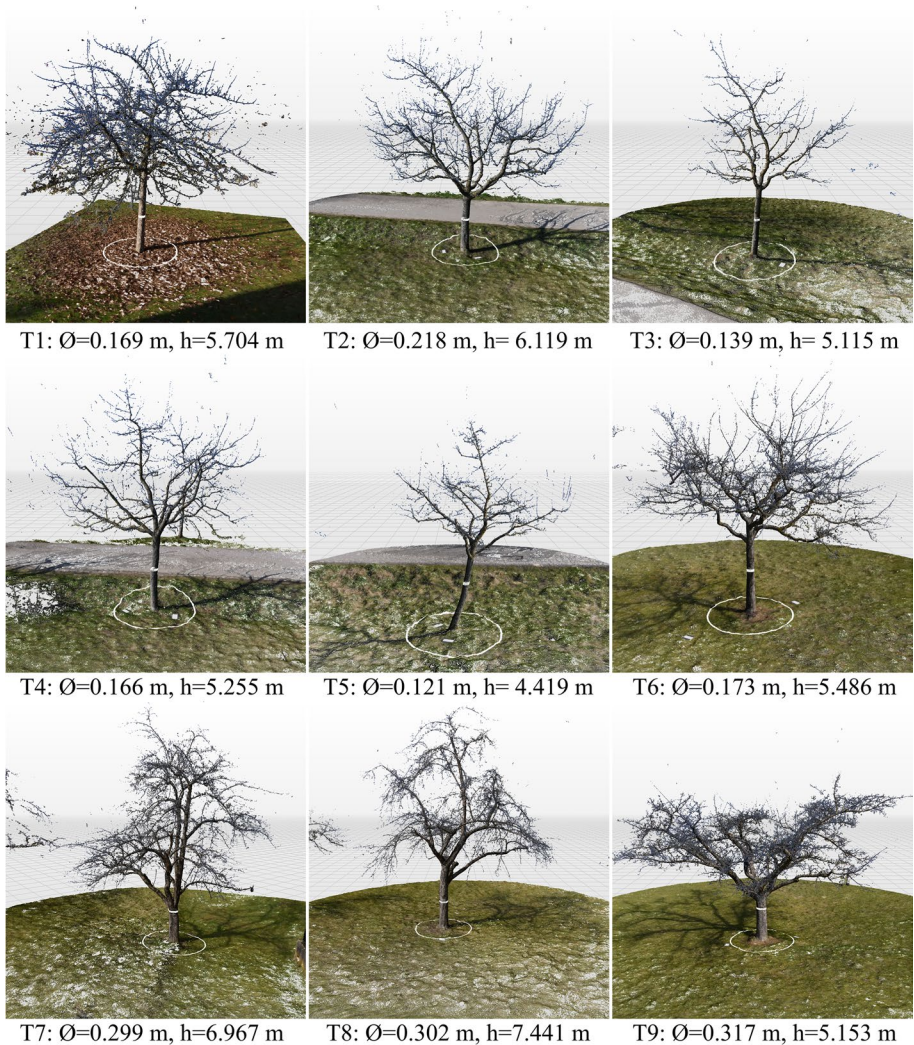
No research is known to the authors so far, which has implemented the creation of a 3D model outgoing from a photogrammetric point cloud using a graph representation for trees in meadow orchards. Therefore, in this work, an RGB camera using SfM (Structure from Motion) and MVS (Multi View Stereo) was used to generate photogrammetric point clouds. These are widely used evaluation methods that are used in different variants in most photogrammetric pipelines. SfM was used to determine the camera positions and camera geometry, while MVS was responsible for the actual generation of the dense point cloud. Even though a terrestrial laser scanner is certainly also a good choice here, an RGB camera offers a great cost advantage in addition to high resolution. The approach presented in this research enables additionally the simultaneous assignment of the points of the point cloud, which is a great advantage for the proposed application. Furthermore, the process is fully automated starting from a 3D point cloud.

The study objective was to create a geometrically and topologically accurate skeleton tree model of all relevant branch structures starting from a 3D point cloud. In addition, it is a requirement for the 3D model to be hierarchical to evaluate the branch directions. Also, the point cloud should be accessible by using the tree model. This means that all points of the relevant tree crown structure have to be assigned to a specific branch of the model which enables further local evaluation. In this way it is possible to carry out analyses considering geometric and topological features, the overlapping of different branches or the branch curvature. The fulfilment of these criteria is considered necessary in order to be able to make appropriate pruning recommendations in the future.

## Materials and methods

### Data acquisition and software

Nine different apple trees were recorded as photogrammetric 3D point clouds (see Fig. 1). The trees had trunk diameters between 0.121 m and 0.317 m, measured at 1 m, and a height between 4.419 m and 7.441 m (maximum Z-dimension in the segmented point clouds). The different trees were divided into two categories, based on their trunk diameter, medium sized trees (T1–T6 see Fig. 1), with a trunk diameter between 0.121 m and



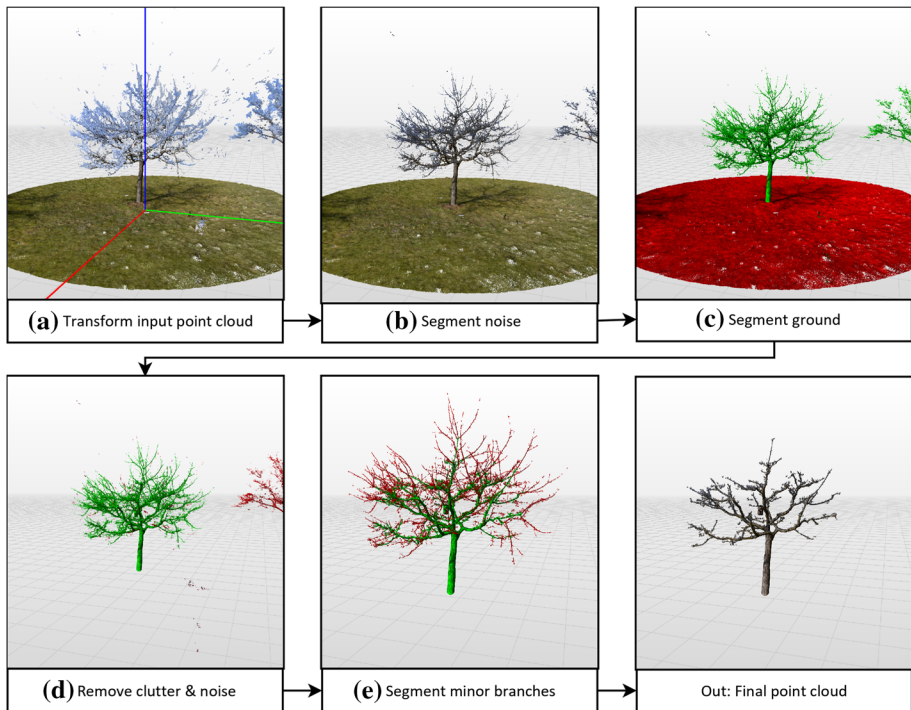
**Fig. 1** Filtered 3D point clouds of the apple trees T1–T9. The scale is shown by the white circle (radius = 1 m) and the mark on the trunk (height = 1 m).  $\text{Ø}$  is the manually measured diameter at 1m height (white mark),  $h$  is the height

0.218 m and large trees (T7–T9 see Fig. 1), with a trunk diameter between 0.299 m and 0.317 m. The tree T9 had a significantly different growth form than the other large trees and was therefore a special case and was used to test the limits of the algorithm. The data was recorded under clear sky and direct sunlight. The first tree T1 was recorded in fall, and was already utilised in Straub et al. (2021). The remaining trees were then recorded in early spring, to extend the dataset.

The images were captured with a 21 MPix APS-C DSLR Camera (D7500, Nikon Corporation, Tokyo, Japan) using a 10.5 mm fisheye lens (AF DX Fisheye-Nikkor 10.5mm f/2.8G ED, Nikon Corporation, Tokyo, Japan). The image acquisition was made in two

circles, at different heights (between 1.30 m to 1.90 m above ground) around the tree to detect all the relevant structures. The recording distance was kept constant for each tree and was chosen so that the tree could be photographed in full-size. The pictures were then processed with the Agisoft Metashape software (Metashape Professional, Agisoft LLC, St. Petersburg, Russia, 2021) using the automatic calibration of the camera. The photogrammetric software generated a dense 3D point cloud of the trees from the images (see Fig. 2a). The first tree was scaled by manually measuring the circumference of the tree trunk at 0.5 m height. The point clouds of the other trees were scaled via the use of several simple scale bars as reference object. On these are two coded targets at a known distance, which can be automatically recognised with Agisoft Metashape.

The statistical filtering and the feature calculations (see Table 1.) used for the segmentation were carried out with CloudCompare (CloudCompare v2.12, [GPL software], 2021). The calculation of features using different search radii (40, 80, 150 mm), also called multi-scale features, allowed for a better classification of points with locally similar features (Niemeyer et al., 2014). The model extraction was done in Python (2018) using the libraries scikit-learn (Pedregosa et al., 2011) for segmentation, SciPy (Virtanen et al., 2020) and NetworkX (Hagberg et al., 2008) for network and graph algorithms and NumPy (Harris et al., 2020) for general computation. For visualization of the point clouds the point processing toolkit software (Here pptk, HERE Europe B.V, Eindhoven, The Netherlands, 2018) was used.



**Fig. 2** Overview of the pre-processing sequence. The input is the photogrammetric point cloud shown in its cartesian co-ordinate system from which the segmented major branches tree point cloud is generated in steps a–e

**Table 1** List of features used for the random forest classification

Geometric feature type	Type	Search radius [mm]
Eigenvalue/-vector based	PCA1	40, 80, 150
	PCA2	
	Linearity	
	Planarity	
	Sphericity	
	Omnivariance	
	Surface variation	
	First order moment	
Density based	Number of neighbours	40, 80, 150
Surface and normal vector based	Normal change rate	40, 80, 150
	Verticality	40, 80, 150
	Roughness	40, 80, 150
Other	Cylinder distance	–

## Pre-processing

The pre-processing was used to reduce the input data to the main tree structure. It can be divided into five steps a–e (see Fig. 2a–e). In the first step (a), the point cloud was transformed into the appropriate coordinate system. For this purpose, the ground was detected by a random sample consensus (RANSAC) plane detection (Fischler & Bolles, 1981) and the point cloud was then rotated and translated in such a way that the ground plane was positioned horizontally and had an average height of zero. Then, using the camera positions from the SfM algorithm, the point cloud was centred. This resulted in a position of the base of the trunk close to  $X, Y, Z=0$  m (see Fig. 2a). This was then used to specify the tree region within the point cloud via the distance to the origin and simplify further processing of the data.

The second step (b) was then to segment the point cloud using a random forest classifier (Breiman, 2001), to filter out noise caused by the fine structure of the branches, which were photographed against the sky and differ strongly in their colour values from the real branch points (see Fig. 2b). The CIELAB colour values of each point were used as features, as proposed in Riehle et al. (2020).

Next (c) the point cloud was segmented using a simple decision tree into the classes ‘ground’ and ‘tree’, using the distance from the ground plane and the verticality (CloudCompare v2.12, [GPL software], 2021). While the distance to the ground plane allows a rough subdivision, it is still inaccurate at the transition to the trunk. Here, the additional consideration of the verticality of the tree trunk allows a precise segmentation (see Fig. 2c).

In the fourth step (d), the 3D point clouds were then subsampled to have a consistent point density and to reduce the amount of data. For sub-sampling, a minimum distance of 5 mm between points was chosen. In addition, a statistical outlier removal (SOR) filter (CloudCompare v2.12, [GPL software], 2021) was applied and disconnected components of the point cloud were removed. The resulting point cloud finally only shows components of the tree branches as seen in Fig. 2d.

As the interest of this research lay in the main branch structure of the tree, small branches and tree shoots were disregarded. Therefore, the fifth step (e) in pre-processing

was the segmentation of the tree structure into major and minor branches (see Fig. 2e). Again, a random forest (Breiman, 2001) algorithm, with the parameters set to 200 trees, a depth of 30 and a minimum sample split of 10, was applied. Even though the same parameters were chosen for the random forest, the feature set used is completely different this time. Before the feature calculation, the trees were scaled to a standard size. For this purpose, the crown was separated from the trunk and an ellipsoid was estimated into the crown points via a principal component analysis (PCA). The volumes of the ellipsoids were put into proportion with their mean value, which then resulted in the scaling factor. The point clouds were then only scaled for the feature calculation, but the unscaled point clouds were still used for the remaining process. For the density feature, it was considered that the point cloud has a different density due to the scaling, which was later compensated for by the ellipsoid volume. At this stage, only geometric features were included (Table 1) as no large derivations in colour between branches of different sizes were detected. In addition, this made the method more independent of the data acquisition method. Besides the cylinder distance feature, all features were available as standard functions calculated in CloudCompare. More information about their calculation can be found in the corresponding documentation (CloudCompare v2.12, [GPL software], 2021) and the related paper (Hackel et al., 2016). These features have been found to be capable of the application carried out.

When validating the model, a few things have to be considered. Besides the generalisation of the model to untrained data (points in the point cloud), the generalisation to untrained trees has to be given as well. A two-stage split was made, first a split into training and validation trees and then within the training trees a second split into 90% training and 10% validation points. Therefore, some of the trees were used exclusively for the validation. This was decided because the selection of separate validation trees has already allocated a relatively large portion of the data for validation. The amount of training data should therefore not be reduced any further. Furthermore, the validation trees are also more relevant for the generalisation of the model. It is also important to note that trees of medium and large size occur in both the training and validation data, so that the generalisation is as accurate as possible. The distribution of the trees is shown in Table 2. T9 thereby was treated separately, which resulted in a 1/3–2/3 training to validation ratio for large trees.

Following the segmentation, a morphological closing operator (Soille, 2004) was used, which first applies a dilation and then an erosion to the two classes. With dilation, a point that has a neighbour in the major branch class within a radius of 30 mm is also assigned to this class, with erosion being the inverse. This helps to close gaps in the segmentation. The number of dilations and erosions depends on the tree's average point density [feature *Number of neighbors* ( $r = 150$  mm)]. This is because smaller trees with a lower point density are more vulnerable to segmentation errors and therefore have more noise. The segmentation performed in the fifth step in combination with the subsequent closing can be seen as an example in Fig. 2e. The segmented point cloud then represents the final result of the pre-processing.

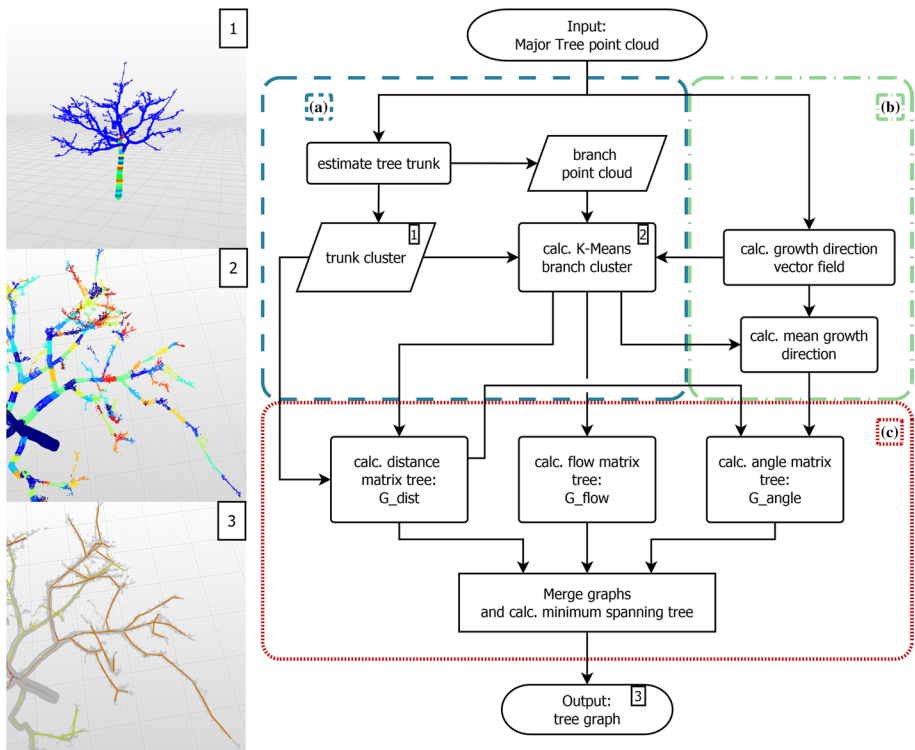
**Table 2** Splitting the trees into training and test data

Tree group	Training	Validation
Medium	T1, T4, T5, T6	T2, T3
Large	T7	T8, T9

## Approach for graph-based tree reconstruction

The first step of the algorithm (see Fig. 3a) to estimate the tree model was detecting the tree trunk. This is straightforward because a cylinder fitting for the trunk was calculated previously for *Cylinder distance*. Points within a distance of 0.05 m were assumed to be trunk points. These were verified for connectivity to ensure that all points of the trunk were connected. After that, the points were divided vertically into different clusters, with each cluster having an extent in the Z-direction of half the cylinder diameter (as shown in Fig. 3a). A cylinder was again estimated for the points of these clusters, adding further points, lying on its surface, to the cluster. This makes it possible to recognise the trunk even if there are larger variations within the trunk diameter.

Next, using the centre of the lower tree cluster as the starting point, a graph was constructed in which all points at a distance of fewer than 0.1 m are linked. Based on this graph, the shortest link to the starting point was calculated for each point using a Dijkstra algorithm (Dijkstra, 1959). Thereby a growth direction vector (see Fig. 3b) and the graph distance were determined for each point of the tree using the respective predecessor. This contains information about the direction of the shortest path in the branch structure. Afterwards, the XYZ co-ordinates, the graph distance to the root, and the three components of the growth direction vector were used



**Fig. 3** Flow chart of the algorithm to create the skeleton tree model. **a** Shows the clustering of the point cloud, **b** shows the usage of the growth direction and **c** shows the creation of graph representation of the tree with the adjacency matrices  $G_{dist}$ ,  $G_{flow}$  and  $G_{angle}$  of the unfiltered tree graph. 1, 2 and 3 show an example of the respective elements



to cluster the point cloud into small branch segments using a K-Means algorithm (Hartigan, 1975). These clusters were then further modified by splitting clusters that extend, for example, on parallel branches and at intersections if possible. Clusters that contain too few points were dissolved and the points were assigned to the surrounding clusters or not assigned if there were no other clusters nearby. The total number of these clusters was determined by the voxel volume of the point cloud. The centres of the resulting clusters (see Fig. 3c) served as branch nodes for the construction of a graph representing a virtual 3D tree model. The centres of the tree trunk clusters were referred to as trunk nodes. In this case, the total number of clusters was defined as 25 clusters per 100 voxels, at a voxel size of 0.16 m.

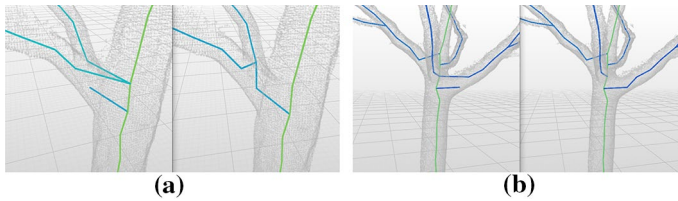
All nodes were connected to one graph by building several graphs alongside each other, starting from each of the individual trunk nodes. Distances were calculated between the trunk nodes and the surrounding branch nodes. Then, the distances to the surrounding branch nodes were calculated from each branch node. From these, a combined adjacency matrix with the distances as weights was created to calculate the minimum distance of the respective cluster points. All connections with a distance > 30 mm were removed. Based on these connections, the minimum distance between points within each cluster was calculated and used as weights for the adjacency matrices which were built for each trunk node. The respective trunk node was used as the highest parent node in this group of adjacency matrices. This stack of matrices can simply be combined into a single graph representing the whole tree, as long as no main branch specific features were calculated. This would be given for example if a previous edge is needed in the calculation of the following one. Then the trunk nodes were connected in the Z-direction to complete the graph. After joining the stack of matrices, the result was the adjacency matrix  $G_{\text{dist}}$ , representing the graph (see Fig. 3c). Based on its connections,  $G_{\text{flow}}$  and  $G_{\text{angle}}$  were calculated using the branch cluster growth direction (mean growth direction in one K-Means cluster), as well as the branch cluster itself. The  $G_{\text{flow}}$  graph resulted from the set of close points between two clusters, and their distance, indicating if there is a strong connectivity. On the other hand, the  $G_{\text{angle}}$  resulted from the angle between the vectors of the growth direction of the two nodes. For the calculation of  $G_{\text{all}}$  the values of these three graphs were scaled by their standard deviation  $\sigma$ , and are added with the following weights (see Eq. 1).

$$G_{\text{all}} = 2 \times \frac{G_{\text{dist}}}{\sigma_{G_{\text{dist}}}} + 4 \times \frac{G_{\text{flow}}}{\sigma_{G_{\text{flow}}}} + \frac{G_{\text{angle}}}{\sigma_{G_{\text{angle}}}} \quad (1)$$

At this point  $G_{\text{all}}$  still contained many false connections. These were removed using a minimum spanning tree using Kruskal's algorithm (Kruskal, 1956). This has the advantage over using the shortest path (Dijkstra, 1959) algorithm, which was used in Straub et al. (2021), because it minimises the total value of all nodes in the graph, which in some cases, as shown in Fig. 4a and b, can lead to a better connection. The result was the final tree graph, which represented the whole tree (Fig. 3–3). Each node of this graph represented a cluster with associated points of the tree point cloud. These points can be assigned to different sub-branches using the graph, which enabled branch-specific segmentation of the point cloud in addition to the skeleton.

## Evaluation methods

To evaluate the quality of the generated tree graphs, reference graphs were created by hand for the recorded trees. These reference graphs were almost identical in structure to the graphs



**Fig. 4** Two examples **a**, **b** illustrating the difference between the Dijkstra algorithm (left) and minimum spanning tree (right) in the reconstructed tree graph

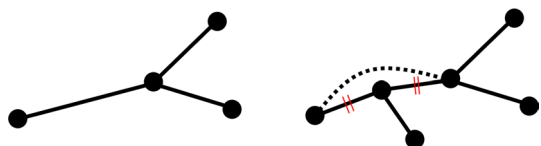
in the generated models, with the difference that nodes only exist at points where the branch forks. To compare the two models all graph nodes with only two connections were removed in the computed one. This ensured that there were also only branch fork points in the graph. First, the corresponding nodes in the reference graph and the generated graph were matched. This involves searching for potential pairs within a radius of 0.15 m. These were then compared with the reference model together with their direct neighbours. As a measure of similarity, the graph edit distance (GED) (Abu-Aisheh et al., 2015) was calculated between these graphs. This describes, in the comparison of two graphs, the number of necessary changes of the one graph so that it is equivalent to the second one. For example, a graph in which all neighbours are mapped to their corresponding graph has a GED of zero. The GED was then weighted and normalised by the distance of the connected segments. The potential partner with the lowest GED was then chosen as the correspondent. If two reference nodes choose the same node as correspondent, the one with the lower GED will get it, while the second one will get the second best one. Through these correspondents, the equivalent graphs could then be created. Now the edges between the nodes can be compared directly. Missing nodes have a very large effect on the result, even if only one node is false. To reduce this effect, missing nodes in the reference graph are “skipped” (see Fig. 5). However, these cases cannot always be detected, so this effect is still present.

## Results and discussion

### Pre-processing

The segmentation of the noise caused mainly by the background (sky) worked effectively with the chosen method. As shown in Fig. 2b, this was a simple and effective way to remove this kind of noise. The illumination conditions of the point clouds differed only slightly, but since the points were taken from different viewing directions, a degree of robustness could be observed. The segmentation of the ground points (in Table 3) achieved an overall accuracy (OA) of 99.78%. This meant that virtually all points from the tree could be separated without difficulty. The results decreased slightly if there were other objects besides the tree in the point

**Fig. 5** Comparison between the edges of a generated graph (left) and a reference graph (right), skipping a node for better comparability

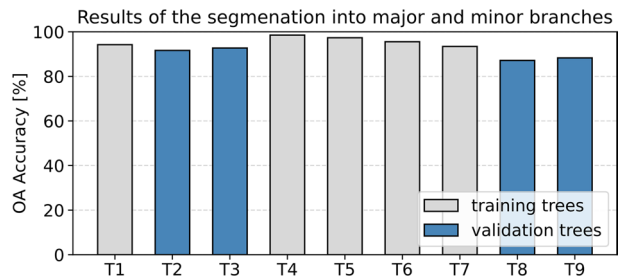


**Table 3** Results for segmenting the ground points and removing the clutter from the tree points

	T1	T2	T3	T4	T5	T6	T7	T8	T9	All
Ground [%]	99.97	99.87	99.95	99.11	99.97	99.98	98.89	99.97	99.97	99.78
Clutter [%]	97.86	95.76	98.46	98.24	97.89	97.85	97.36	97.22	98.98	97.79

“All” is thereby the mean value weighted by the number of points

**Fig. 6** Results of the segmentation into the classes major and minor branches for validation data of the training trees (grey) and of the validation trees (blue). Trees T1–T6 are medium trees, trees T7, T8 and T9 are large trees, with T9 having an unusual tree structure compared to the other (Color figure online)



cloud, as in T7 (see Fig. 1). However, it must be considered that the results are very specific to this use case. Therefore, this cannot be transferred to other applications in all cases. For the segmentation of other objects, with residual noise and other artefacts (clutter), which do not belong to the target trees, the OA was 97.79%. The number of clutter points varies strongly depending on the presence of other objects in the point cloud. However, almost all objects were reliably removed. If errors occur, as in T2, these were mainly in the area of the minor branches, which were subsequently segmented and removed as well.

Crucial for further processing was the segmentation into major and minor branches. Here (see Fig. 6), an OA of 88.96% for the validation trees was achieved, which was worse than the OA of the training trees (94.96%). This suggests a slight overfitting of the model to those trees. However, this was not unexpected under the given variety and the results are reasonable. There are still more minor differences between the different tree types. For example, in Fig. 6, T7, T8, and T9, representing the large or non-standard trees, performed slightly worse than the medium-sized trees. But this was also not unexpected, since there were significantly fewer training data available for these types of trees. A larger amount of training data would help, as the large trees were only trained with one tree. Overall, a good generalisation of the model for medium to large trees was achieved. The model was not investigated for trees with a trunk diameter smaller than 0.12 m. Small trees with a trunk diameter of less than 0.1 m are not the focus of this work.

### Approach for graph-based tree reconstruction

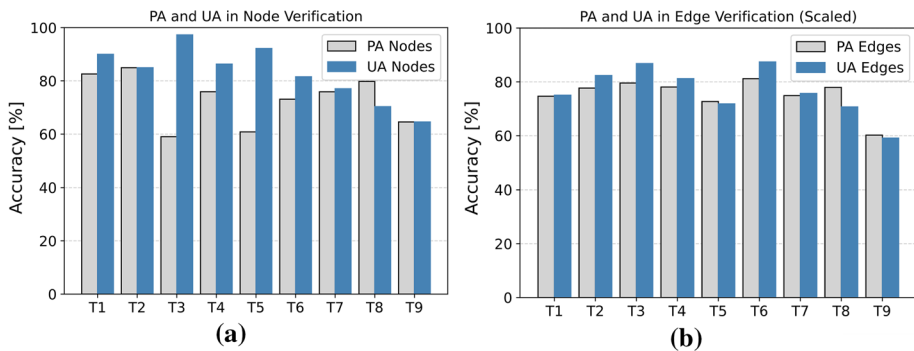
The result of the node verification, averaged over all trees (see Table 4), achieved a correct matching with the reference graph, here referred to as producer’s accuracy (PA), of 72.37% and a correct proportion of the calculated nodes, referred to as user’s accuracy (UA), of 78.24%. After weighting by the edge length, the PA was 73.67%, and the UA was 74.30% for the edge verification. The results of the individual trees are shown in Fig. 7. The trees of medium size T1–T6 (see Figs. 8, 9) showed promising results for edge verification (see Fig. 7b) and range from approx. 75–87% for both PA and UA. The only exception was T5

with results for PA of 72.68% and UA of 71.81%. For node verification, the situation was similar for the medium trees. However, T5 and T3 were noticeable, since there was a considerable variation between the PA and the UA in the node verification (see Fig. 7a). This, combined with the still good results in edge verification, suggests that relatively small and short branches were not correctly detected. T3 and T5 were the two smallest trees in the data set, with accordingly more of the finer structures. The results were still sufficient, but indicated a limitation of the algorithm and/or the sensor technology with regard to smaller trees.

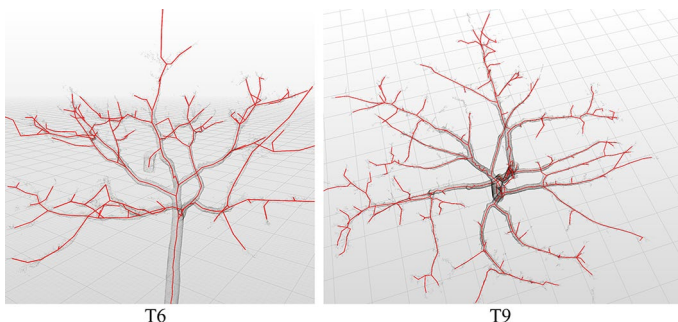
The large trees T7 and T8 (see Figs. 7, 9) performed slightly worse on average, both for node and edge verification. The results were between 70% and 78% for both UA and PA. T9, which was a large tree with a different and unusual structure, performed worst overall with a PA of 60.21% and a UA of 59.11%. This one had a complex structure with many small branches and all main branches start more or less at a similar trunk height. This can quickly introduce errors in the reconstruction. However, it should be noted, that errors are

**Table 4** Results of the node and edge verification (edges get scaled by length) for all trees as PA and UA

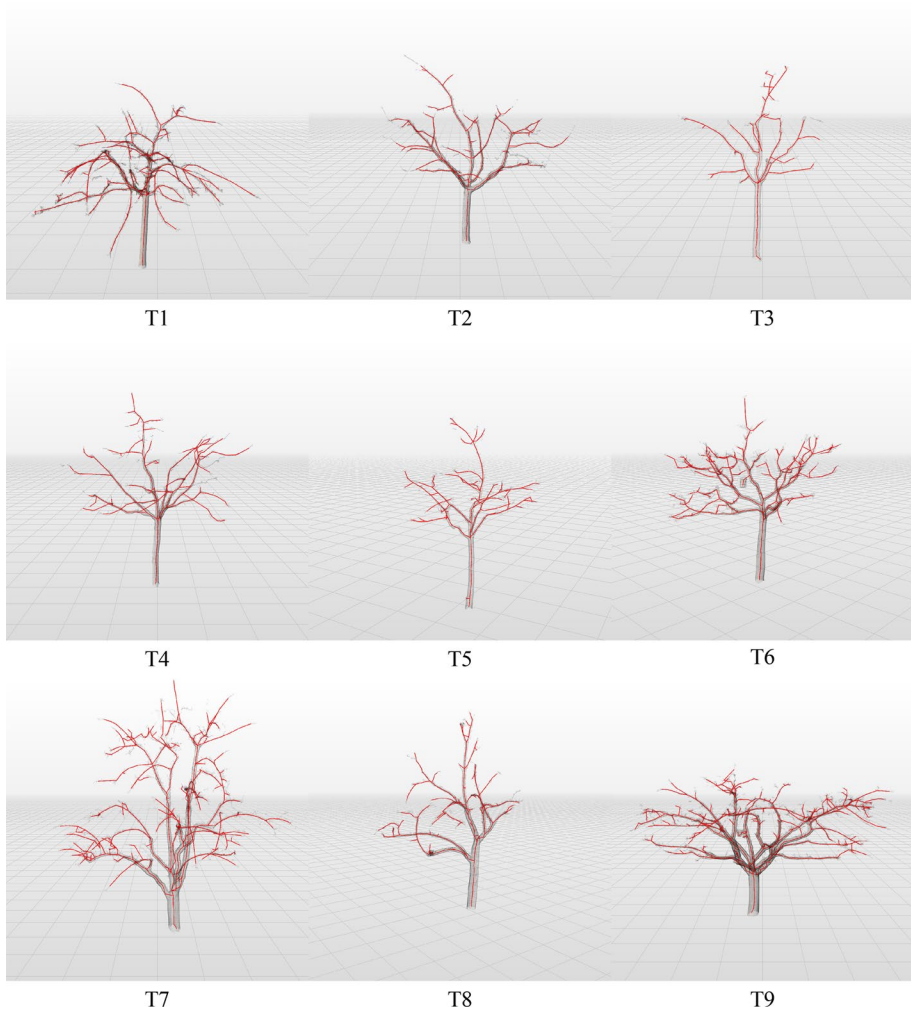
	PA [%]	UA [%]
Node verification	72.37	78.24
Edge verification (scaled)	73.67	74.30



**Fig. 7** Results of node verification (a) and edge verification (b), showing the PA and the UA of the generated tree model to the reference model



**Fig. 8** Detailed view of the results of the tree model of T6—best result (left) and T9—worst result (right)



**Fig. 9** Calculated 3D-tree skeletons T1–T9 with their respective point clouds

very rapidly noticeable, especially with the edge verification, since a wrong node always affects several edges, which can, however, be correct.

For better comparability, two of the trees were also visually compared. These trees were among the best resulting trees, namely T6 and the worst resulting tree T9 and are shown in more detail in Fig. 8. As expected from the previous results, the model showed a reasonable reconstruction of the point cloud. The course of all leading branches was correct, and with a few exceptions, almost all the smaller branches were correctly reconstructed. An interesting error observed in Fig. 8 was the reconstruction of the bird-house as a short branch. Overall, the results were very good. Compared to T6, the results of T9 showed noticeably more errors. Smaller branches were occasionally not recognised, but there were also significantly more of these than in T6. A noticeable error here was that in very large leading branches, several clusters sometimes appeared against the direction of growth. This can also be observed to a lesser extent in the large trees T7 and T8 (see Fig. 9). In T9,

this led to unwanted distortions of the model, even if the correct linkage of the different branches was still given. These errors could occur when leading branches have very large diameters and were therefore incorrectly clustered by the K-Means algorithm. However, even at T9 all of the important structures of the tree were present and the model could still be considered acceptable.

The results of the approach for building the tree model were found to be sufficient to reconstruct the recorded trees. The trees of medium size gave good results. Depending on the size of the trees, they degraded slightly towards the top and bottom, but the models still provided a suitable reconstruction. Nevertheless, this is where most of the potential for further improvement lies. Possible here would be an extension of the K-Means features to include other elements such as branch thickness. This could be done in two stages using the first clusters for this calculation. This would primarily help to improve the reconstruction of trees with strong leading branches. In addition, it would be a suitable feature to link the branches in the model and would generally add useful information that could be integrated into the model for other applications. Improvements could be made for smaller trees by extending the pre-segmentation of major and minor branches. Here, further training data could be useful. However, it should be noted here that for thinner branches, the resolution of the point cloud is not sufficient at some point.

Based on a tree model, like the one extracted here, further development for the automation of the pruning of meadow orchard trees can be made. Based on this model, pruning recommendations could now be calculated using geometric and topological criteria. The model allows the observation of overlaps of different branches, a first indicator for pruning. The integration of the previously segmented minor branches could be used in order to estimate the remaining space between two branches. Otherwise, the observation of water sprouts is also feasible, which could be recognised by their typical vertical growth using the angles between the edges of the graph. Fast growth and changes or areas which were pruned could be analysed using several observations over time. By the same logic, such a multi-temporal analysis could be used for the detection of deadwood, as there is no growth of small branches in this case. Another possibility would be an application as a decision support system, in which such a graph can be used to select pruning points to supervise a human worker.

## Conclusions

The photogrammetric 3D point clouds of nine apple trees were recorded. These point clouds were pre-segmented using various methods including a random forest algorithm. This resulted in an OA of 88.96% for the sub-division into major and minor branches. From the major branch structure point clouds, 3D models in the form of skeletons were created with the method presented. A PA of 72.37% and a UA of 78.24% were achieved for the node verification of the tree graphs while, for the edge verification a PA of 73.67% and a UA of 74.30% were reached. The computed tree models are creating the possibility to analyse the crowns based on their topological as well as their geometrical structure and to use them to derive areas where pruning is necessary. The identification of such areas is crucial for the identification of pruning points and therefore for automated tree pruning. In addition, such a model is necessary to plan an autonomously performed pruning, as it allows the target point to be clearly defined and possible interfering objects to be avoided. Furthermore, there is a great potential for scientific tree analysis in terms of observing changes over multiple scans, for example in growth or in analysing performed pruning, or also for the creation of a digital twin.

**Acknowledgements** The project robotics for the maintenance of meadow orchards received funding's from the Baden-Württemberg Stiftung within the elite program for postdocs. The authors would like to thank the Baden-Württemberg Stiftung for the financial support.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Abu-Aisheh, Z., Raveaux, R., Ramel, J. Y., & Martineau, P. (2015). An exact graph edit distance algorithm for solving pattern recognition problems. *Proceedings of the International Conference on Pattern Recognition Applications and Methods* (Vol. 1, pp. 271–278). Setubal, Portugal: SCITEPRESS—Science and Technology Publications.
- Agisoft Metashape Professional (2021). (version 1.7.2) (commercial software). Retrieved June 2021, from <http://www.agisoft.com/downloads/installer/>
- Alvites, C., Santopuoli, G., Hollaus, M., Pfeifer, N., Maesano, M., Moresi, F. V., & Lasserre, B. (2021). Terrestrial laser scanning for quantifying timber assortments from standing trees in a mixed and multi-layered mediterranean forest. *Remote Sensing*, 13(21), 4265. <https://doi.org/10.3390/rs13214265>
- Arikapudi, R., Vougioukas, S., & Saracoglu, T. (2015). Orchard tree digitization for structural-geometrical modeling. In: J.V. Stafford (Ed.) *Precision Agriculture '15: Proceedings of the 10th European Conference on Precision Agriculture*. (pp. 329–336) Wageningen, The Netherlands: Wageningen Academic Publishers. [https://doi.org/10.3920/978-90-8686-814-8\\_40](https://doi.org/10.3920/978-90-8686-814-8_40)
- Borngräber, S., Krismann, A., & Schmieler, K. (2020). Ermittlung der Streuobstbestände Baden-Württembergs durch automatisierte Fernerkundungsverfahren (Determination of meadow orchard stands in Baden-Württemberg using automated remote sensing methods). *Naturschutz und Landschaftspflege Baden-Württemberg*, 81, 1–17
- Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. <https://doi.org/10.1023/A:1010933404324>
- CloudCompare (2021). (version 2.12). [GPL software]. Retrieved August 1, 2021, from <http://www.cloudcompare.org/>
- Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269–271. <https://doi.org/10.1007/BF01386390>
- Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381–395. <https://doi.org/10.1145/358669.358692>
- Hackel, T., Wegner, J. D., & Schindler, K. (2016). Fast semantic segmentation of 3d point clouds with strongly varying density. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, III–3, 177–184. <https://doi.org/10.5194/isprs-annals-III-3-177-2016>
- Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In: *7th Python in Science Conference (SciPy 2008)*. (SciPy) (pp. 11–15)
- Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., et al. (2020). Array programming with NumPy. *Nature*, 585(7825), 357–362. <https://doi.org/10.1038/s41586-020-2649-2>
- Hartigan, J. A. (1975). *Clustering algorithms*. New York, USA: Wiley.

- He, L., & Schupp, J. (2018). Sensing and automation in pruning of apple trees: A review. *Agronomy*, 8(10), 1–18. <https://doi.org/10.3390/agronomy8100211>
- Here pptk (2018). (version 0.1.1) pptk - Point Processing Toolkit, Retrieved December 01, 2020, from <https://heremaps.github.io/pptk/viewer.html>
- Kang, H., & Chen, C. (2020). Fruit detection, segmentation and 3D visualisation of environments in apple orchards. *Computers and Electronics in Agriculture*, 171(2019), 105302. <https://doi.org/10.1016/j.compag.2020.105302>
- Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1), 48–50. <https://doi.org/10.2307/2033241>
- Méndez, V., Rosell-Polo, J. R., Pascual, M., & Escolà, A. (2016). Multi-tree woody structure reconstruction from mobile terrestrial laser scanner point clouds based on a dual neighbourhood connectivity graph algorithm. *Biosystems Engineering*, 148, 34–47. <https://doi.org/10.1016/j.biosystemseng.2016.04.013>
- MLR (2015). Streuobstkonzeption Baden-Württemberg (Conception for orchards in Baden-Württemberg), Ministerium für Ländlichen Raum und Verbraucherschutz Baden-Württemberg. Retrieved December 15, 2020, from <https://mlr.badenwuerttemberg.de/fileadmin/redaktion/mmlr/intern/dateien/publikationen/Streuobstkonzeption.pdf>
- Niemeyer, J., Rottensteiner, F., & Soergel, U. (2014). Contextual classification of lidar data and building object detection in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87, 152–165. <https://doi.org/10.1016/j.isprsjprs.2013.11.001>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- Python 3.7.0 (2018). Retrieved November 26, 2021, from <https://www.python.org/downloads/release/python-370/>
- Riehle, D., Reiser, D., & Griepentrog, H. W. (2020). Robust index-based semantic plant/background segmentation for RGB-images. *Computers and Electronics in Agriculture*. <https://doi.org/10.1016/j.compag.2019.105201>
- Sanz, R., Llorens, J., Escolà, A., Arnó, J., Planas, S., Román, C., & Rosell-Polo, J. R. (2018). LIDAR and non-LIDAR-based canopy parameters to estimate the leaf area in fruit trees and vineyard. *Agricultural and Forest Meteorology*, 260–261, 229–239. <https://doi.org/10.1016/j.agrformet.2018.06.017>
- Schuboth, J., & Krummhaar, B. (2019). Untersuchungen zu den Arten der Streuobstwiesen in Sachsen-Anhalt (Studies on the species of meadow orchards in Saxony-Anhalt). *Berichte des Landesamtes für Umweltschutz Sachsen-Anhalt (Halle)*, 2, 1–408.
- Soille, P. (2004). Opening and closing. *Morphological image analysis: Principles and applications* (pp. 105–137). Berlin: Springer. [https://doi.org/10.1007/978-3-662-05088-0\\_4](https://doi.org/10.1007/978-3-662-05088-0_4)
- Straub, J., Reiser, D., & Griepentrog, H. W. (2021). Approach for modeling single branches of meadow orchard trees with 3D point clouds. In: J.V. Stafford (Ed.) *Precision Agriculture '21: Proceedings of the 13th European Conference on Precision Agriculture*. (pp. 735–741) Wageningen, The Netherlands: Wageningen Academic Publishers. [https://doi.org/10.3920/978-90-8686-916-9\\_88](https://doi.org/10.3920/978-90-8686-916-9_88)
- Sun, S., Li, C., Chee, P. W., Paterson, A. H., Meng, C., Zhang, J., et al. (2021). High resolution 3D terrestrial LiDAR for cotton plant main stalk and node detection. *Computers and Electronics in Agriculture*, 187, 106276. <https://doi.org/10.1016/j.compag.2021.106276>
- Tabb, A., & Medeiros, H. (2017). A robotic vision system to measure tree traits. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, New York, USA: IEEE, 6005–6012. <https://doi.org/10.1109/IROS.2017.8206497>
- Tsoulias, N., Paraforos, D. S., Xanthopoulos, G., & Zude-Sasse, M. (2020). Apple shape detection based on geometric and radiometric features using a LiDAR laser scanner. *Remote Sensing*, 12(15), 1–18. <https://doi.org/10.3390/RS12152481>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Jarrod Millman, K., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- Zehnder, M., & Wagner, F. (2008). Streuobstbau-Ein Auslaufmodell ohne sachgerechte Pflege (Meadow orchard cultivation—A discontinued model without the proper maintenance). *Praxisinweise aus Südwestdeutschland. Naturschutz und Landschaftsplanung*, 40(6), 165–172.