Project Number: 774571
Start Date of Project: 2017/11/01
Duration: 48 months

Type of document 4.2 – V1.0

3D Tree Models

Dissemination level	PU
Submission Date	2020-04-03
Work Package	WP4
Task	T4:3
Type	Report
Version	1.0
Author	Sebastian Lamprecht, Steve Ahlswede
Approved by	Andrea Gasparri + PMC

Executive Summary

The given document comprises a detailed description of PANTHEON's task *T4.3 – Tree Geometry Reconstruction*, which focuses on the generation of three-dimensional (3D) hazelnut tree models, as well as on the detection of suckers. In particular, modeling pipelines have been developed and implemented to create 3D vector models of hazelnut trees and to mark suckers.

The following sub-tasks have been identified and addressed:

1 Supervised classification of laser scans in general.

2 3D tree models.

  2.1 Identification of the hazelnut branches.

  2.2 Filtering of branch nodes.

  2.3 3D vector modeling of the branching structure.

3 Sucker detection.

  3.1 Sucker Detection Using High Resolution point clouds

  3.2 Sucker Detection Using Low Resolution point clouds

4 Point cloud volume estimation.

The algorithms have been bench-marked with the accuracy measures defined in deliverable *D2.1 – Requirements, Specifications and Benchmarks.*

Table of Contents

List of Figures

List of Tables

Abbreviations and Acronyms

2D / 3D	two-dimensional / three-dimensional
UGV	Unmanned Ground Vehicle
GPS	Global Positioning System
NIR	Near InfraRed
RGB	Red-Green-Blue
LiDAR	Light Detection And Ranging
LAI	Leaf Area Index
NDVI	Normalized Difference Vegetation Index
ExG	Excess Green index
NDRE	Normalized Difference Red Edge index
DBH	Diameter at Breast Height
ROI	Regions Of Interest
Level of Detail	LoD
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error

1 Introduction

The given document comprises a detailed description of task *T4.3 – Tree Geometry Reconstruction addressing* PANTHEON's Objective 2.1. This objective is accomplished through the use of data acquired from mobile ground-based LiDAR sensors to reconstruct the geometry of the tree and extract synthetic indicators or attributes describing the structure of the hazelnut trees and the presence of suckers. This objective involves the definition of a synthetic formalism to describe tree structure and suckers, as well as the adaptation of existing algorithms for tree reconstruction to the specific case of hazelnut trees.

A detailed tree geometry reconstruction is required for suckers' detection and pruning planning. The necessary data is acquired using a laser scanner *Faro Focus S70* and a *Velodyne VLP-16* mounted on a ground rover (see Section 2). The LiDAR point clouds are homogenized and pre-classified (ground vs. vegetation as defined in deliverable *D4.1 – Multispectral LiDAR Point Clouds*). Additionally, the point clouds are co-aligned by using the initial GPS position and by identifying structures suitable to identify the displacement (like bare ground or branches). For tree geometry reconstruction at the required level of detail (LoD 4-5), various automatic algorithms have been proposed [1]. To focus on the branching structure rather than on quantitative structure models (as done in [2] or [3]), the principal ideas of the *VecTree* approach [4], the *SkelTre* algorithm [5] and [6] are used to develop an algorithm extracting the skeleton of the multi-trunked hazelnut trees as a vectorized three-dimensional graph. These models implicitly provide information on the branching structure and branching density.

Since the geometry reconstruction is suitable for the localization of individual branches in the field and is, in combination with leaf-on point clouds, an indicator for both light interception and the leaf area index (LAI), these models provide relevant data for automated pruning planning (Objective 3.1). The suckers have shown to be best identified using image classification and object recognition approaches like [7], rather than extracting the explicit tree geometry.

Next to the pure extraction of the tree geometry and suckers, methods to estimate the volumes are addressed to fulfill the needs of tasks *T5.1 – Sucker's management protocol* and T*5.2 – Pruning protocol*.

PANTHEON's study area is composed of three fields selected within the Azienda Agricola Vignola, a farm located in the municipality of Caprarola, in the province of Viterbo. In the fields, the cultivar Nocchione are planted as multi stemmed bushes in a regular 4.5 m x 3.0 m layout.

2 Hardware

In PANTHEON, data to address task *T4.3 – Tree Geometry Reconstruction* is typically collected with the laser scanner *Faro Focus S70* [8] mounted on the ground robot *SHERPA HL robotic platform R-A* (Fig. **1**). The laser scanner provides high resolution 3D point clouds for reality capture. If required for other tasks, like *T4.7 – Pest and Disease Detection* or *T4.8 – Fruit Detection*, the laser scans are also spectrally enriched with images of the *MicaSense RedEdge-M* five band multispectral camera and a *Sony a5100* RGB camera. Details can be found in *D3.1 – Robotic Prototypes and D4.1 – Multispectral LiDAR Point Clouds.*

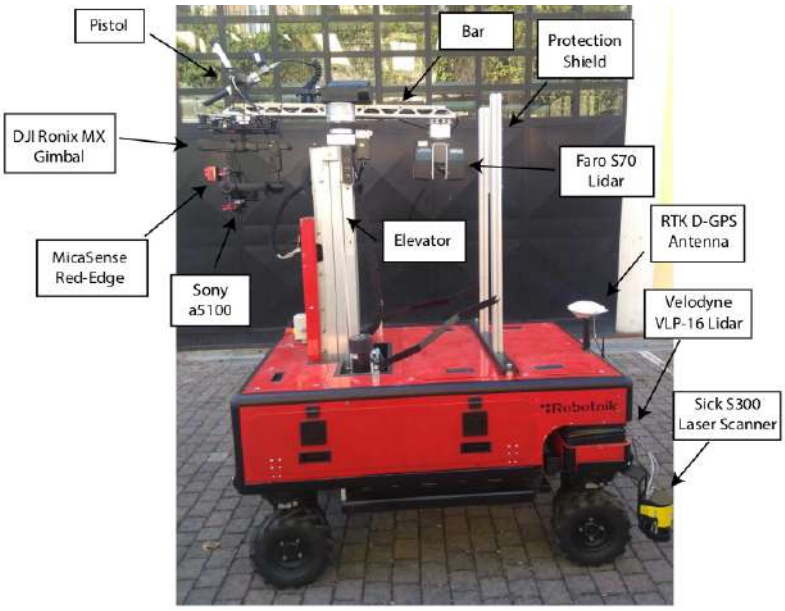

Fig. 1: The robotic platform *SHERPA HL robotic platform R-A* with sensors.

To reduce the economic impact for sucker detection, the *Velodyne VLP-16* laser scanner—originally intended for the navigation of the UGV—is also tested for the identification suckers and volume estimation. The *Velodyne* (refer deliverable *D3.1 – Robotic Prototypes* for details) is complemented with a *Genius WideCam F100* camera. The *Genius* webcam has a 120° vision and an image resolution of 1280×720. The camera focus was set manually and has been kept fixed for all the experiments. The *Genius* is located directly below the *Velodyne* and both are mounted parallel to the ground plane, as illustrated in Fig. **2**.

Fig. 2: Front view of the *SHERPA HL robotic platform R-A* with the *Velodyne VLP-16* laser scanner and the *Genius WideCam F100* webcam.

3 Laser Scan Classification

To address the sub-tasks *3D Tree Models* (Section 4) and *Suckers Detection* (Section 5), an automated classification of the laser point clouds is required. But, also future tasks—like task *T4.8 – Fruit Detection*—might benefit from a universal pipeline to classify point clouds or images. For this reason, the following procedure has been elaborated:

1. Conversion of laser scans to images.
2. Supervised image classification performed by human experts.
3. Mapping of classification layers to the laser scan or image.
4. Automated derivation of point cloud features.
5. Training of a classifier.
6. Automated classification of arability scans using the classifier and point cloud features.

The image classification performed by the human experts serve for training and validation of the classifiers. The procedure and program code can be easily adapted to related image classification tasks. In particular, the pipeline can be reused for the tasks *T4.6 – Water Stress Measurement*, *T4.7 – Pest and Disease Detection* and *T4.8 – Fruit Detection*.

3.1 Supervised Classification

The supervised image classification is performed by human experts. The classification is based on the tentative classification scheme presented in Tab. 1.

Tab. 1: Tentative classification scheme used by the human experts.

Class ID	Label	Description
1	sky	sky
2	soil	soil or ground pixels
3	grass	grass or weeds
4	branch	trunks, branches and twigs
5	leaf	leaves not associated with suckers
6	sucker	leaves of suckers
7	dead_sucker	died off suckers
8	catkin	catkins
9	fruits	hazelnuts or hazelnut clusters
99	miscellaneous	man-made objects or mixed classes

To ease manual classification, a 2D image representation is generated from each three-dimensional *Faro Focus S70* laser scan. For an optimal visualization, each feature of the laser scan—e.g. intensity, red, green, blue or distance to the scanner—can be interpreted as an image layer. Thus, the human expert can either create a true color or a false color composite of the laser scans. Finally, the human expert digitizes and labels the classes of interest as illustrated in Fig. **3**. The polygons should be spread randomly all around the image to avoid artefacts in the automated classification result due to a systematic placement of the reference polygons. The results are stored in the commonly used *.shp* format [9].

Fig. 3: Image representation of an RGB laser scan overlaid with manual classification labels. The class labels used in this example are; sky (blue), soil (orange), branch (brown), grass (olive), leaf (dark green), suckers (light green), and dead suckers (red).

A processing chain maps the *.shp*-files to the laser scans, resulting in an annotated 3D point cloud as illustrated in Fig. **4**. The manual classification labels along with the point features serve as input to train the classifiers.

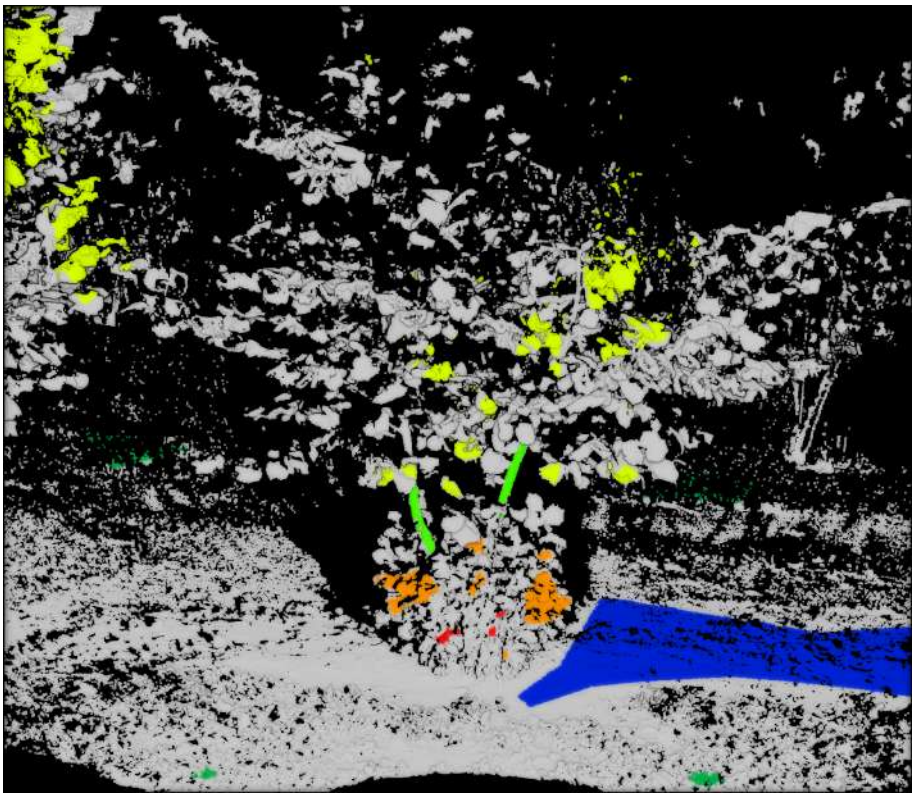

Fig. 4: Point cloud annotated with the manual classification labels. The class labels used in this example are; no data (gray), soil (blue), branch (light green), grass (dark green), leaf (yellow), suckers (orange) and dead suckers (red).

3.2 Point Cloud Features

To distinguish the classes of interest automatically, various features of the point clouds might be suitable. Similar to image classification, spectral features like RGB color might be expedient. But, also structural features, like the shape of the objects might help to distinguish classes. For this reason, multi-scale dimensionality features [10], spectral features and various complementary features are derived for each laser scan.

3.2.1 Spectral Features

In classical remote sensing, spectral features are typically used to distinguish classes [11]. Spectral features can be obtained from the RGB bands of the *Sony* camera or the multi-spectral bands of the *MicaSense* camera. Due to the proximity sensing character of the task, the spectral response of the trees will always be affected by shadows. This is expected to have a negative effect on the classification results.

On the other hand, the *Faro Focus S70* laser scanner measures at a wavelength of 1550 nm. Due to the high amount of energy emitted by the laser beam in a water absorption band [8], a negligible effect of the solar irradiation on the signal can be assumed. Thus, no shadows should be visible in the intensity image captured by the laser scanner. Given this, the intensity of a point can serve as a spectral reference for the spectral bands.

3.2.2 Multi-scale Dimensionality Features

Using the approach of [10], multi-scale dimensionality features are extracted. For a given set of core points and for each scale, the Eigenvalues [12] of the neighboring points within a Neighborhood ball are derived. The ratio of the two largest Eigenvalues gives information on the aspect—1D (linear), 2D (planar) or 3D (volumetric)—of the object at the given scale. The change of the Eigenvalues over the scales gives information on the local structure of the object.

Since the extraction of neighborhood balls and the derivation of Eigenvalues are computationally intense tasks, the features are derived for core points only. By identifying its closest core point, the features are transferred to each point of the scene.

As suggested by [10], a point filter providing a homogeneous spatial density of core points was used. In detail, the filtering algorithm of [13] with radius 2 cm is applied, ensuring a minimum point distance in a range of 2 cm up to 4 cm.

3.2.3 Knowledge-Driven Features

Next to the spectral features and the multi-scale dimensionality features, knowledge-driven features are extracted.

In particular, spectral indices might be calculated to highlight specific characteristics. When applicable, the normalized difference vegetation index (NDVI) [14], normalized difference red edge index (NDRE) [15] and the excess green index (ExG) [16]—as defined by Equations 1, 2 and 3—are derived by default.

$$NDVI = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red}} \tag{1}$$

$$NDRE = \frac{\text{RedEdge} - \text{Red}}{\text{RedEdge} + \text{Red}} \tag{2}$$

$$ExG = 2 \cdot \text{Green} - \text{Red} - \text{Blue} \tag{3}$$

The NDVI highlights vegetation in general, while the ExG highlights young light greenish leaves. As an indicator for the plant's health, NDRE can be used to identify harmed leaves.

The height of points above ground and the distance to the laser scanner might be used as supplementary features. However, these need to be used with care, as they may correlate with a systematic sampling pattern unintentionally introduced by the human expert.

3.3 Training of Classifiers

To generate the results of the given deliverable—for the sake of the simplicity of the approach—random forest classifiers [17] with 100 decision trees have been trained. To take into account the spatial variability of the features at different scales, image convolution filters were applied before passing the features to the classifier. In particular, average blurring kernels of the shapes 5x5, a 11x11, 21x21, and 41x41 [18]. For future applications, alternative machine learning techniques, like convolutional neural networks [19] or similar, might be used to tune the classification accuracy.

For each manually classified laser scan, the *.shp* class labels are mapped to the point cloud by intersecting the image pixels with the polygons. Finally, a classifier predicts its target classes based on the given features and kernel filters. The feature importances provide information on the usefulness of a specific feature to distinguish the target classes.

4 3D Vector Models

4.1 Introduction

The reconstruction of the geometry of trees is a major task which provides information on the biomass and the productivity of the hazelnut trees. In particular, the branching structure of the young trees, as well as information on the biomass, is needed as an input for task *T5.2 – Pruning Management Protocol.* Thus, along

with the timber volume of the branches, an approximation of the volume of the canopy should also be estimated.

To reconstruct the geometry of a tree, it needs to be scanned during 'leaf-off' conditions, since leaves would shadow the inner branches of the tree, leading to a significant lag of information on the branching structure. These restrictions limit the time window to scan the trees to the winter. Unfortunately, in winter the trees develop catkins (see Fig. **5**), making the extraction of the tree geometry more difficult than originally expected.

Fig. 5: Young hazelnut tree under leaf-off conditions with catkins.

4.2 Reference Measurements

For practical reasons, the result of a tree geometry reconstruction cannot be validated with ground truth data. To evaluate the topology of the tree models, the three-dimensional location and orientation of the branches and forks of the tree would have to be measured along with the branch diameter in the field. Even if the required accuracy can be achieved, the measurements need to be aligned with the laser scans. Any disturbance, like inaccurate alignment or wind would negatively affect the validation measures.

An alternative approach to gain information on the topology would be to do a manual tree geometry reconstruction using the point-clouds of the laser scans. Although the topology of the tree can be easily extracted by a human expert, the diameters of the branches are hard to measure. Thus, a validation of the timber volume would not be possible.

Next to these spatially discrete measurements, alternative in situ reference measurements, like diameter at breast height (DBH) or the total timber volume—might gain information on the accuracy of the tree models. The DBH [20]—which is measured for each tree in PANTHEON by default—is hard to compare with the tree models without spatial information, due to the multi-trunk layout of the hazelnut trees. The measurement of the timber volume would probably require invasive methods, which is not feasible in PANTHEON.

Because of the lag in feasible reference data, the decision was made to generate synthetic hazelnut trees (see Section 4.2.2). Such synthetic trees can be used to generate synthetic point clouds, which can be suitable to tentatively test and validate the algorithms for tree geometry reconstruction. Although the geometry of the synthetic trees is only an approximation of real-world trees and the laser scans are not affected by

disturbances (like catkins or wind), this approach allows for a direct comparison of the original and modeled tree. In particular, a fair comparison of the timber volume, the canopy volume, the topology and the branch diameters can be achieved.

4.3 Methods

To model the geometry of a tree, the following processing pipeline has been elaborated:

1. Scan Pre-Processing (according to D4.1 – *Multispectral LiDAR Point Clouds*)
2. Branch Classification
3. Point-Cloud Filtering
4. 3D Hazelnut Tree Graph Modeling

4.3.1 Branch Classification

The terrestrial processing chain presented in D4.1 – *Multispectral LiDAR Point Clouds* foresees a coarse pre-classification and noise reduction of the laser scans. But, to identify branches and mask out disturbing structures—like leaves, suckers or catkins—a branch classifier is trained. Fig. 6 illustrates the manual classification of a laser scan for training or validation of branch classification.

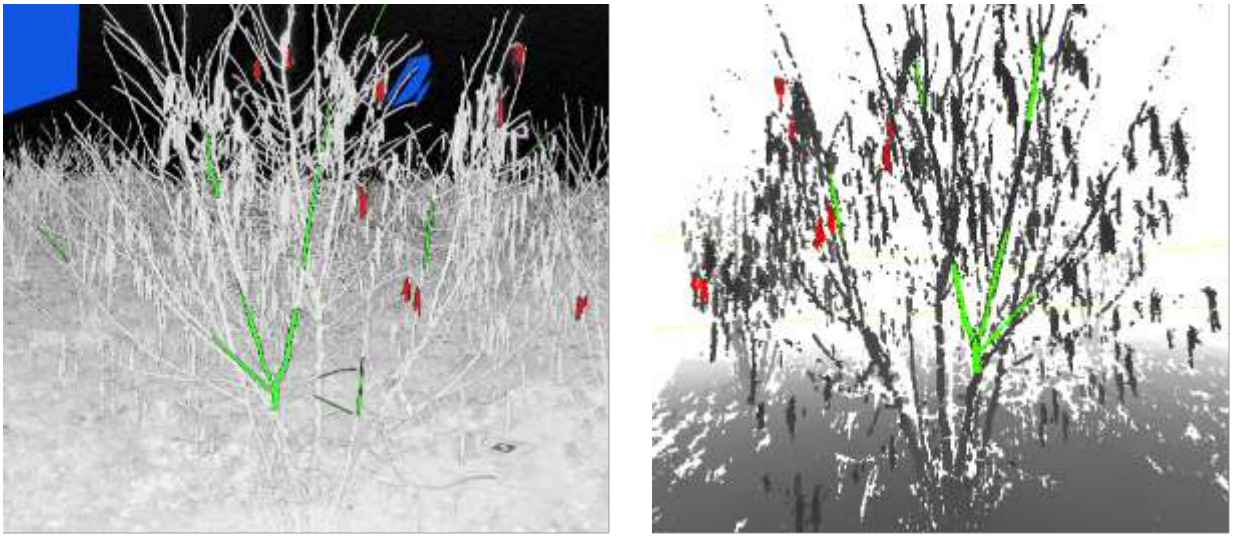

Fig. 6: Intensity image of a young hazelnut tree recorded by a laser scanner with overlaid manual classification (left). Corresponding 3D point cloud representation of the same tree (right). The class labels sky (blue), branch (green) and catkin (red) are visualized

Aside from the branch classification the scans need to be filtered for remaining (undetected) obstacles and noise. For this reason the DBSCAN [21] algorithm is applied two times using the parameters $minPts = 10$ and $\epsilon = 0.015$, and $minPts = 40$ and $\epsilon = 0.05$, respectively.

4.3.2 Point Cloud Filtering

To extract the skeleton of a hazelnut tree, the point cloud $P \subset \mathbb{R}^3$ is filtered and smoothed. The algorithm used for this task is a byproduct of D4.1 – *Multispectral LiDAR Point Clouds*, since it was expected to be useful to align point-clouds. The general idea of the filtering procedure is to identify representative points describing the structure of the trees. In particular, the center points of the branches shall be identified, to receive an initial prediction for the location and diameter of the branches.

In detail, the algorithm applies a duplicate point filter [13] with radius R ensuring a minimum point distance of at least R and at most $2 \cdot R$. After thinning the point cloud, the coordinates are smoothed by averaging the coordinates of all neighboring points within radius $2 \cdot R$. Points with less than three neighbors are not shifted to avoid the collapse at the tips of twigs. As illustrated in Fig. 7, by repeating this smoothing procedure n times, the point coordinates tend to converge to the center of the surrounding points, if the number of points and their spatial distribution are sufficient. In particular, the shift δ of the coordinates is an indicator for the radius of the branch.

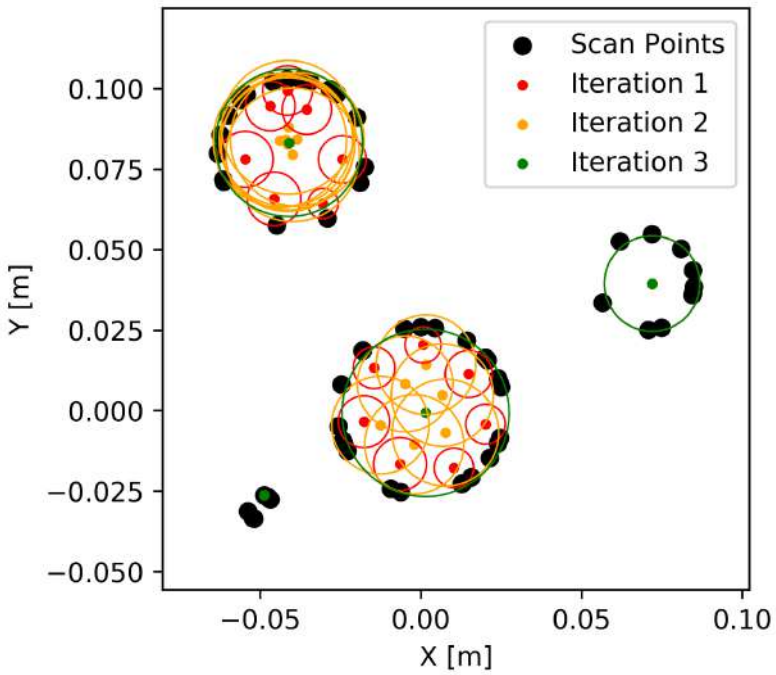

Fig. 7: Results of the filtering algorithm using a synthetic two-dimensional point cloud representing four neighbored branches of differing diameters. The circles drawn around the point centers represent the estimated cross section with radius $\boldsymbol{\delta}$ of the branch

To facilitate the convergence of coordinates on the center of the branch intersections, the averaging function uses the squared coordinate shift of the previous iteration as weight. In addition, the averaging procedure can be controlled by providing a rigidity parameter τ. A value of $\tau = 0$ means, that the coordinates are fully averaged, while a value $\tau = 1$ fully retains the original coordinates.

Since the smoothing procedure is designed to collapse point structures to a common location, it provides point duplicates. Thus, finally a duplicate point filter [13] with radius $1.5 \cdot R$ is applied.

It needs to be noted, that this filtering technique also provides the coarse structure of the canopy, under leaf-on conditions. This is in alignment with the needs of task *T5.2 – Pruning Management Protocol* as defined in *D2.1 – Requirements, Specifications and Benchmarks* to calculate the variation of the LAI during the season. To generate the results for the given deliverable, the parameters $R = 0.025$, $n = 10$ and $\tau = 0.1$ were used.

4.3.3 3D Hazelnut Tree Graph Modeling

To represent the structure of a hazelnut tree, its trunks, branches and twigs are interpreted as a three-dimensional graph G of nodes P and edges $E(G)$. Each node $p \in P$ is characterized by its location (p_x, p_y, p_z) in 3D space and its radius p_r. An edge or segment $e = (p, q) \in E(G)$ can also be interpreted as a 3D vector linking the nodes p and q. The center of the segment is defined as $e_c = p + 0.5 \cdot \overrightarrow{q - p}$. Along with the radii of both nodes, the edge is understood as a truncated cone defining the surface e_S of its branch segment. In

this concept, the bark of a real-world branch is approximated by series of truncated cones. Forks are the result of nodes connected by several edges as illustrated in see Fig. 8.

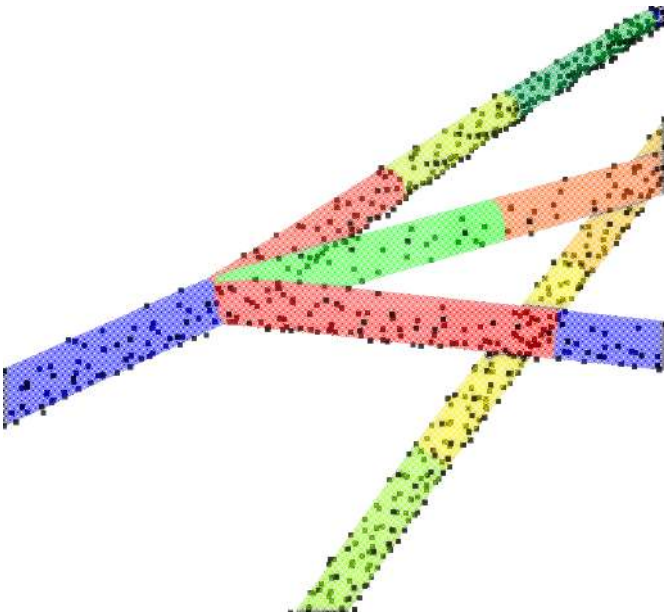

Fig. 8: Truncated cone model approximating the surface of branches captured by a synthetic laser scanner. The segments are labeled in differing colors.

To generate the 3D graph G of a hazelnut tree, the original point cloud $P \subset \mathbb{R}^3$ and its filtered point cloud $F \subset P$ of Section 4.3.2 are used as inputs. To generate the graph, the following procedure is applied:

1. 3D Graph initialization
2. Segment Features
3. Graph Optimization
4. Tree Cleaning

4.3.3.1 3D Graph initialization

Each point p of the filtered point cloud F is interpreted as a node of the graph G. For each node $p \in F$, its spatially closest neighboring node $q \in F$ is identified. Finally, each node is linked to all its neighboring nodes within radius $\omega \cdot ||q - p||$. The origin o of the graph or hazelnut tree is defined as the median of the x- and y-coordinates, as well as the minimum of the z-coordinates of the filtered point cloud.

4.3.3.2 Segment Features

To gain information on the suitability of a specific segment to approximate the surface of the tree, several features are extracted. These features are intended to be used as weights for later the graph optimization procedure.

For each edge $e = (p, q) \in E(G)$, all neighboring points $B \subset P$ of the original point cloud with distance to the truncated cone surface less than Δ_{max} are selected. The average radius of the segment e_r is set to be the median distance $P_{50\%}(\{||e_S - b|| : b \in B\})$ of the selected points to the edges surface. Edges with radius $e_r > \Delta_{max}$ are omitted.

In addition, the length of the segment $e_{length} = ||q - p||$, the height above ground $e_{height} = e_{c_z} - o_z$, the distance to the origin $e_{\Delta o} = ||e_c - o||$ and the number of points supporting the segment $e_n = |B|$ are

calculated. Next to these attributes, additional features describing the connectivity of the segment are derived. The estimated radius $e_\delta = (p_\delta + q_\delta)/2$ is drawn from the filtered point-cloud.

With the idea of the MLESAC algorithm [22], a consensus set $C = \{b \in B: ||e_S - b|| < 1/3 \cdot \Delta_{max}\}$ is derived, which shall serve as a quality measure of the suitability of the segment to approximate the surrounding points in B. Based on the consensus set, the attribute $e_{msac} = \sum_{b \in C} ||e_S - b|| + |B \setminus C|$ is derived, which is zero, if the surface approximates the points in B perfectly.

Inspired by the robustness criterion of [5], the 5% percentile of the distances between the center of the edge e_c and all points in B is calculated. The resulting robustness criterion $e_{robustness} = P_{5\%}(\{||e_c - b||: b \in B\})$ is expected to have values close to $||p - q||/2$, if the connection between the nodes p and q is badly supported by the points of the point cloud. This is typically the case, if nodes between two parallel branches are interconnected by the segment. Otherwise, values close to zero are expected.

4.3.3.3 Graph Optimization

To reconstruct the structure of the tree, the minimum spanning tree of G is generated using the Kruskal's algorithm [23]. For the given deliverable the weight of an edge $e \in E(G)$ is set to $|e_\delta - e_r| \cdot 1/e_n \cdot e_{msac} \cdot e^2_{robustness} \cdot e^2_{\Delta o}$, with the following ideas:

- $|e_\delta - e_r|$: prefers segments with a radius similar to the estimated radius from tree filtering
- $1/e_n$: prefers segments approximating a large number of points
- e_{msac}: the segment orientation shall be supported by many points
- $e^2_{robustness}$: penalizes segments linking parallel branches
- $e^2_{\Delta o}$: prefers a short path to the root of the tree, to avoid branches which link to surrounding trees

After creating the minimum spanning tree, for each node $p \in F$, its radius p_r is set to the median radius of all its edge radii.

4.3.3.4 Tree Cleaning

The graph optimization procedure cannot guarantee the branch segments not to overlap. To ensure no overlap, all neighbors of node $u \in F$ located within any segment $e = (p, q) \in E(G)$ are linked to p and q. Finally, again a minimum spanning tree is created, using $2 \cdot e_{length}$ as weight. This procedure guarantees the removal of overlapping segments while preferring short connections. Since the graph generation can lead to a lot of pseudo-forks, all nodes connected to just a single fork are also removed from the graph. Finally, the edge features and node radii are recalculated, to take into account the updated skeleton.

4.4 Remarks

Due to the multi-trunk layout of a typical hazelnut tree only an undirected graph is generated. By adding an artificial root node located in the underground and linking it to all nodes close to the ground, a directed graph can be generated on demand.

The 3D tree graph modeling has been implemented in *Python* using the core libraries *networkx* [24] and *Pyoints* [13]. To generate the results for the given deliverable, the parameters $\omega = 2.5$, $\Delta_{max} = 0.01$ [m] and $r_{max} = 0.01$ [m] were used.

It needs to be noted, that the edge weights used for this deliverable are tentative. Different weights might lead to better results without having to change the algorithm. By manually labeling some of the branch

segments as “correct” or “incorrect” by a human expert, a classifier could be trained to improve the graph optimization identifying more appropriate weights.

4.4.1 Volume estimation

Based on the aligned and laser point clouds of a tree, the filtered point cloud (see Section 4.3.2) can be used to estimate the canopy volume. In particular, the convex hull or an alpha-hull similar to [25] can be derived to estimate the canopy volume. An alternative would be to use the method presented in Section 5. An advantage of these kind of volume estimation methods is that these approaches can be applied even on trees under leave-on conditions without having to model the 3D tree skeleton. In case, the 3D tree model has been generated, and in addition to the canopy volume, also the timber volume can be estimated. The timber volume is derived as the sum of the truncated cone volumes across all branch segments.

4.4.2 Generation of Synthetic Trees

To validate the algorithms developed for 3D tree modeling, synthetic trees with synthetic laser scans are generated. To generate the synthetic trees a growth cycle is simulated to generate to overall structure. The final 3D shape is a result of simulating gravitational drag.

4.4.2.1 Growth Cycle

A tree is generated, by repeatedly performing the following growth cycle. Each repetition represents a growing season.

1. Seeding
   New randomly distributed suckers are created.
2. Growing
   The outer branches of the tree grow by a random length in a range of 8 cm and 12 cm. If a branch segment exceeds a length of 30 cm, it is split into two segments of equal length. By splitting a slight random bending of the branch is applied. Finally, the diameter of all branches is increased by 4%. The growing procedure is repeated eight times.
3. Budding
   At the end of the season a random number of new buds are generated at the outer branches of the tree. These buds define new forks of the tree and consequently the orientation of the new branches.

Since a random generation of forks and bent branches leads to a dissatisfying result, based on the gravitational force f [N], as well as the bending strength σ_b [N/m³] and compressive strength σ_e [N/m³] of the timber, the gravitational distortion of the tree is approximated.

Synthetic trees of two age classes were generated. Young trees were represented by a five year cycle, adult trees are represented by a eight year cycle. Examples of resulting tree models are illustrated in Fig. **9**.

4.4.2.2 Gravitational Drag

For a given branch segment with orientation vector $\vec{v}$ and average radius R (in meters), the longitudinal compression c [m] is calculated by Equation 4, using the cross sectional area $A = \pi \cdot R$[m²], and the bending force in direction of gravity $b = A \cdot \sigma_b/4$ [N]. The vertical displacement is approximated by Equation 5 using the compressive force in direction of gravity $e = A \cdot \sigma_e/4$ [N].

$$c = f/e \cdot \overrightarrow{v_z} \tag{4}$$

$$z = -f/b \cdot \sqrt{(\overrightarrow{v_x}^2 + \overrightarrow{v_y}^2)} \quad (5)$$

Since the vector changes its orientation after applying Equations 4 and 5, the gravitational distortion is repeated until convergence.

Based on the approximate timber density $\rho = 610$ [kg/m³] [26], the gravitational drag on each branch—resulting from the subsequent branch segments—is calculated. Beginning from the root to the leaves, the orientation of each branch segment is realigned by applying the gravitational effects of Equations 4 and 5. The resulting tree shapes of the synthetic young and adult hazelnut trees can be found in Figure 9.

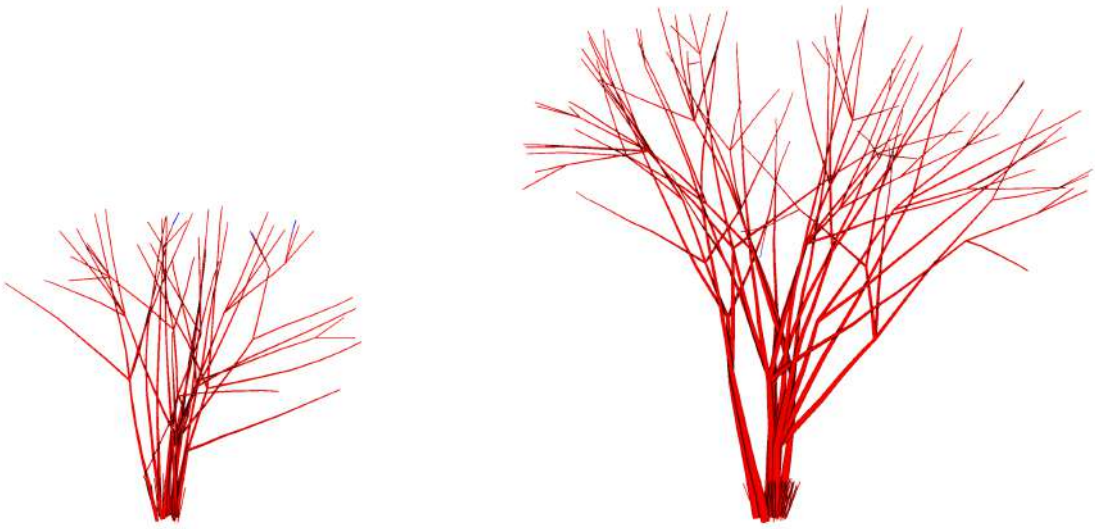

Fig. 9: 3D visualization of a synthetic young tree (left) and a synthetic adult tree (right).

4.4.2.3 Synthetic Laser Scans

For each synthetic tree, a synthetic point cloud is generated. Four virtual laser scanners are placed around the tree at 1.7 m above ground, according to the scanning layout defined in *D4.1 – Terrestrial Remote Sensing pipeline*, assuming a planting pattern of 4 m x 5 m. To generate the point cloud, the scanner is configured similar to the *Faro Focus S70* with the *quality=1/32* setting. Each laser beam is ray-traced to gain the intersection points with the tree surface using the python library *Pyoints* [13]. Finally, normally distributed noise with standard deviation 2.5 mm—similar to the distance accuracy of the *Faro Focus S70* specified by the manufacturer [8]—is added to the laser beam. The resulting single scan point cloud is illustrated in Fig. 10. The point clouds of all four scans are fused to an individual point cloud as illustrated in Fig. **11**.

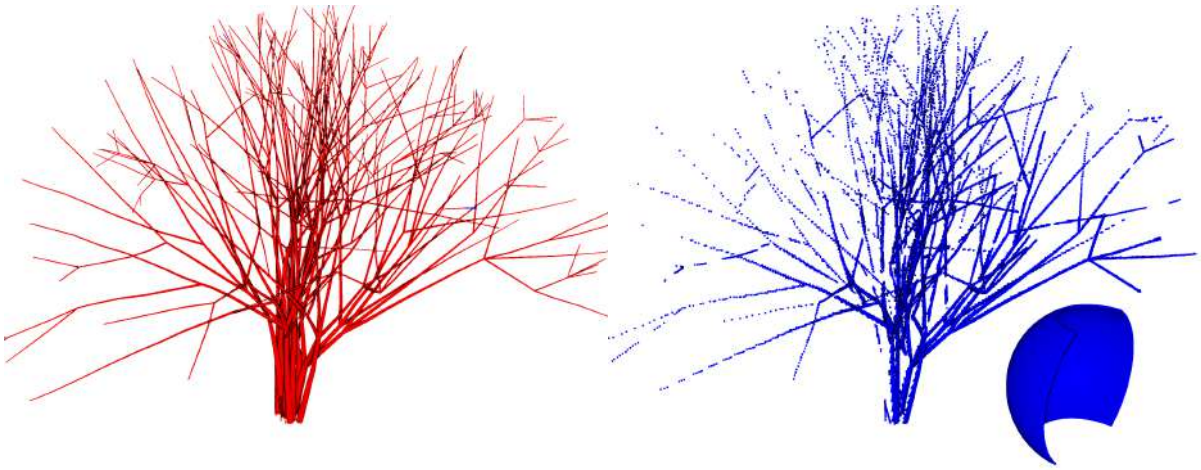

Fig. 10: Synthetic adult hazelnut tree (left) with the result of a single synthetic laser scan (right). The partial sphere corresponds to the virtual scanner position.

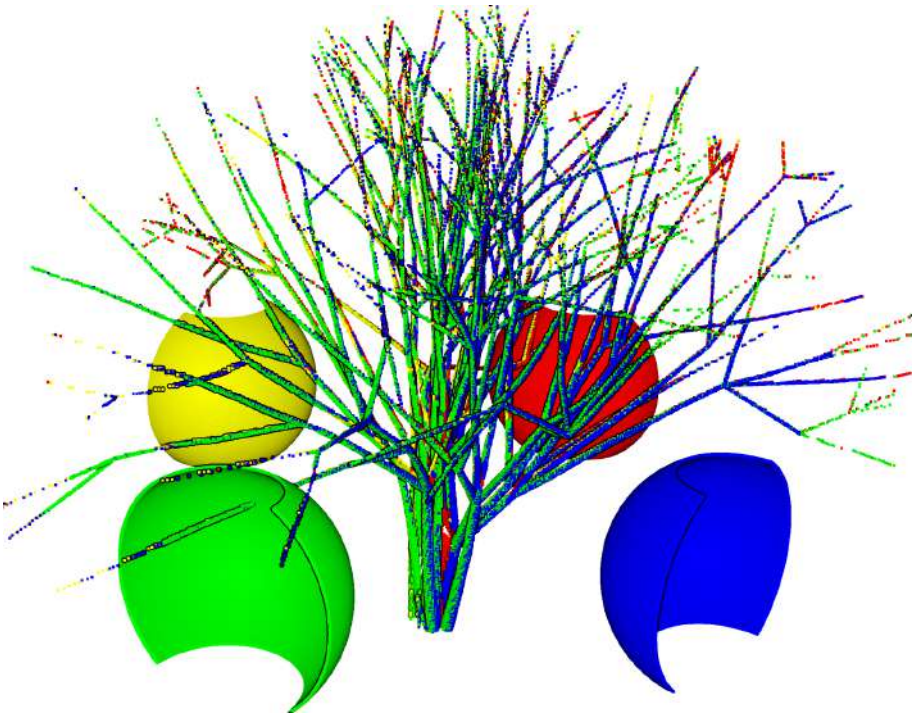

Fig. 11: Point-cloud of a synthetic hazelnut tree generated by four virtual laser scanners (partial spheres). Each point of the tree is labeled according to the color of its scanner.

4.5 Benchmarks

In deliverable *D2.1 – Requirements, Specifications and Benchmarks* the following benchmarks for the tree geometry model have been defined*:*

- **Validation measure A2:**
  Algorithm capable of detecting branches with a minimum diameter of 2 cm.
- **Validation measure A3:**
  3D model representing at least 80% of the input point cloud for bare-leaf plants.

Next to these benchmarks, also the accuracy of the volume estimation is evaluated.

4.5.1 Validation measure A2

To address V*alidation Measure A2*, the synthetic 3D trees are used, since they provide well known branch diameters. To evaluate whether branches of a minimum diameter of 2 cm can be identified by the modeling procedure; for each node of a modeled tree it is evaluated whether it is located within any segment of the original tree. If so, the original segment is considered to be detected and the radius at the location of the model node is calculated. In consequence, a modeled tree fulfills validation measure A2, if the minimum diameter of the detection is smaller or equal 2cm.

4.5.2 Validation Measure A3

To address V*alidation Measure A3*; for each tree the distance of the original point cloud—excluding other objects, like catkins, soil, UGV and grass—to the 3D tree model is calculated. A point is assumed to be represented by the point cloud if its absolute residual is less than 10mm. This threshold is in alignment with the expected scan alignment accuracy. In deliverable D4.1, an alignment error of up to 3mm was observed. Thus, even with an optimal model, a significant proportion of model errors are caused by the misalignment. It needs to be noted, that this validation measure is applicable even for real-world trees without having information on the original tree geometry.

4.5.3 Diameter and Volume

The relative error of the convex hull volume, as well as the timber volume, are calculated for all 3D tree models of the synthetic trees. In addition to the volumes, the mean absolute percentage error (MAPE) [27] of the branch diameters are derived.

4.5.4 Topology

To gain information on the topology of the tree models, the synthetic trees are evaluated as their geometry is well known.

We define the **correctness** of a tree model as the fraction of the modeled segment centers located within any of the segments of the original tree.

We define the **completeness** of a tree model as the fraction of the original segment centers located within any of the segments of the modeled tree.

We define the **forking accuracy** of a tree model as the fraction of original segment centers connected to a fork located within any of the segments of the modeled tree.

4.6 Experimental Setup and Data

The tree geometry reconstruction is based on 3D point clouds provided by the *Faro Focus S70* laser scanner. To gain information on the 3D structure in terms of branch topology, each tree is scanned from four surrounding positions. For each tree, the surrounding scans are aligned and merged to an individual point-cloud per date.

As a result of D4.1 – *Terrestrial Remote Sensing pipeline*, the scanner parameters *Distance=near*, *Resolution=1/8* and *Quality=2x* were used.

The results presented in this document are based on the following two laser scanning campaigns.

- **Campaign 2020/01/10**

The campaign of January 2020 took place in fields 16 and 18. The purpose of this campaign was to gain information on the tree geometry of young and adult trees. Due to GPS failure, only the scans of field 16 were geo-referenced.

- **Campaign 2019/12/11**

The RGB laser scans of December 2019 were mostly collected for sucker detection (see Section 5). But, since a manual classification of the scans was performed either way, these scans are also used to train the branch classifier.

Five scans from Campaign 2020/01/10 and four scans from *Campaign 2019/12/11* serve as training data for the branch classifier. Two scans from *Campaign 2020/01/10*, and two scans from *Campaign 2019/12/11* were utilized for validation. In addition, the full processing chain for branch detection and 3D tree modeling are applied on two adult trees and three young trees from *Campaign 2020/01/10.* The point clouds of the trees are the result of a manual alignment of four scans for each of the two age classes. As a consequence, the overlap of the scans for the outer trees is lower than the actual data collection procedure foresees.

To generate the results of the real-world application, edges with $e_{robustness} < 0.06$ are removed from the 3D tree model, since they obviously result from connections to remaining undetected catkins. Since this can lead to unconnected sub-graphs, all sub-graphs with less than two nodes are removed.

4.7 Results and Discussion

4.7.1 Branch Classification

Fig. **12** illustrates the results of the branch classifier. The linear structures of branches are well identified for all scales. In particular the catkins can be separated, but also leaves are mostly identified.

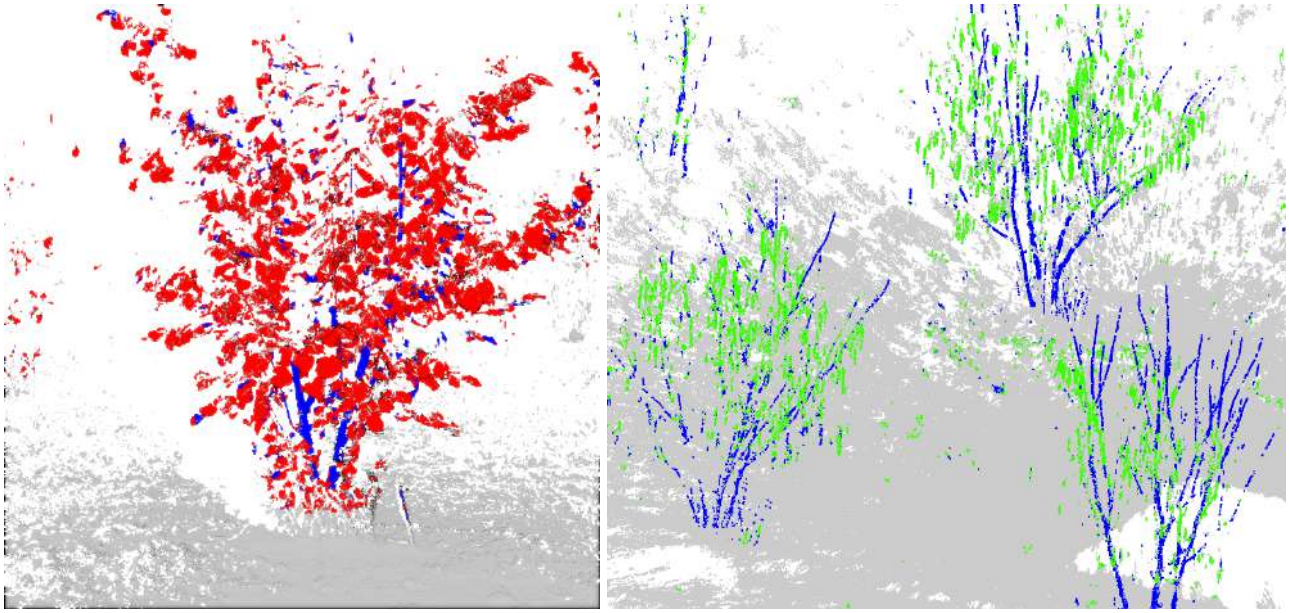

Fig. 12: Results of the branch classification for an RGB scan under leaf-on conditions (left) and under bare-leaf conditions (right). Ground is labeled gray, branches blue, leaves red and catkins green.

Fig. **13** illustrates the importance of each feature according to the Random-Forest classifier for branch detection. The highest feature importances are achieved for the intensity and distance. In particular, the distance gives valuable information about the structure of the trees, while the intensity is suited to distinguish vegetation from other materials. The Eigenvalues (p1 and p2) achieve reasonable feature importances over all scales. Since these describe the local structure of the surface elements, these are particularly useful to distinguish planar leaves from linear branches and noisy catkins.

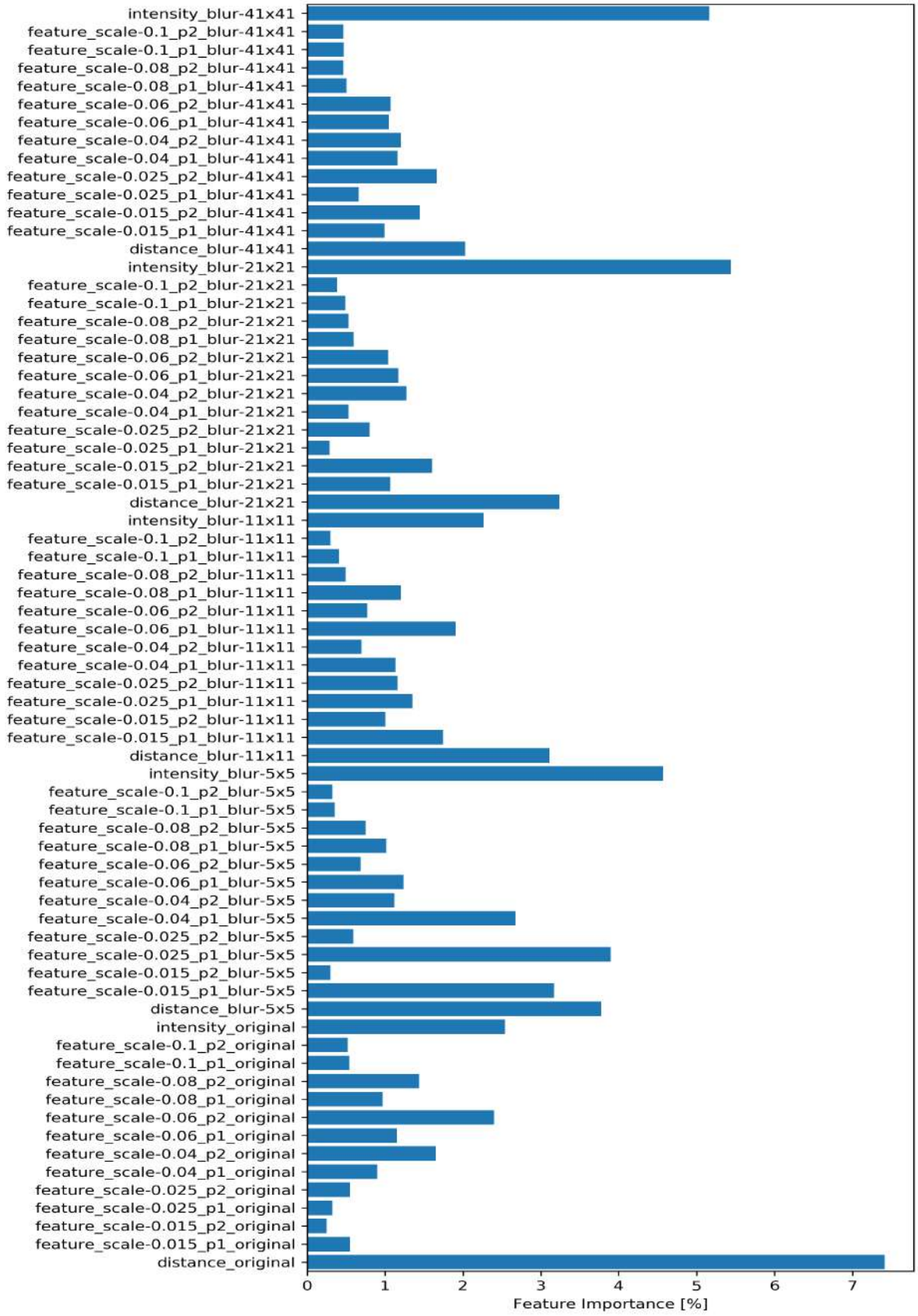

Fig. 13: Feature importances identified by the Random-Forest classifier for branch detection.

The branch classifier estimates an out of the bag (OOB) score of 99.7%. Tab. **2** provides the classification accuracy metrics of the validation dataset focusing on the vegetation—identified by the pre-classification—only. Similar to the out of the bag score, the validation data set provides a total accuracy of 99.7%. An average precision of above 95% is achieved. Only the recall of the branches is negatively by affected by the huge number of miscellaneous objects.

Tab. 2: Confusion matrix for branch classifier.

		Reference			
		Branch	**Catkin**	**Other**	**Precision [%]**
Predicted	**Branch**	51550	1043	1378	95.5
	Catkin	1922	19216	4	90.9
	Other	4381	1	3196294	100
	Recall [%]	89.1	94.8	100	99.7
			95.0		95.4

4.7.2 Synthetic 3D Models

The synthetic hazelnut trees were characterized by an average timber volume of 0.026 ± 0.011 m^3 for the young trees and 0.18 ± 0.069 m^3 for the adult trees.

Tab. **3** and Tab. **4** summarize the results of the accuracy assessment for the synthetic young trees, receptively adult trees.

Branches with minimum diameters of about 10 mm (range [10.00, 10.03]) were detected independently from the age class. These values are in the magnitude of the minimum diameters the modeled trees provide. Due to the chosen method to generate a graph from filtered point clouds, *Validation Measure A2* is fulfilled free of statistical doubt for the synthetic trees.

About 90% (range [79.8, 94.6]) of the residuals were smaller than 10mm for both the young and adult trees, indicating that the point clouds are well represented by the 3D tree models. Thus, *Validation Measure A2* is fulfilled for the synthetic trees.

As the radius estimation for twigs is dominated by the noise of the laser scan, the predicted branch diameters of the young trees are overestimated significantly by about 24% (range [16.4, 34.4]). Thus, the timber volume estimation is also highly over-predicted for the young trees by on average 60% (range [33.3, 90.1]). This effect is minor for adult trees, resulting in an underestimation of the timber volume by about -15% (range [-33.9, 2.3]) due to missing segments. In contrast, the estimation of the convex hull volume is robust for both age classes. The slight under-prediction of about -4 to -5% (range [-7.8, -2.3]) is mostly caused by missing segments at the leaves of the graph.

Tab. 3: Accuracy measures of the synthetic young hazelnut trees. Displayed are the mean values and standard deviations of all 100 synthetic trees.

Validation Measures	Accuracy Metric	Mean accuracy (standard deviation)
Validation Measure A2	minimum radius detected [mm]	10.0 (0.01)
Validation Measure A3	fraction of residuals smaller 10mm [%]	88.9 (2.12)
Topology	correctness [%]	84.2 (2.54)
	completeness [%]	83.0 (2.62)
	forking accuracy [%]	80.7 (6.06)
Volume	convex hull volume error [%]	-3.8 (0.98)
	timber volume error [%]	57.1 (12.13)
Diameter	MAPE of branch diameters [%]	23.9 (3.26)

Tab. 4: Accuracy measures for the synthetic adult hazelnut trees. Displayed are the mean values and standard deviations of all 100 synthetic trees.

Validation Measures	Accuracy Metric	Mean accuracy (standard deviation)
Validation Measure A2	minimum radius detected [mm]	10.0 (0.01)
Validation Measure A3	fraction of residuals smaller 10mm [%]	91.0 (1.52)
Topology	correctness [%]	88.5 (1.64)
	completeness [%]	79.3 (2.68)
	forking accuracy [%]	79.5 (5.02)
Volume	convex hull volume error [%]	-4.7 (1.0)
	timber volume error [%]	-14.9 (7.6)
Diameter	MAPE of branch diameters [%]	18.9 (1.9)

Fig. **14** illustrates, that the 3D modeling method is able to reconstruct the geometry of a hazelnut tree in general. This finding is supported by the high correctness and completeness values for both age classes. However, Fig. **15** illustrates, that the method is not fully able to link the forks and trunks correctly.

The effect of wrongly linked fork segments is highlighted by the low forking accuracy of about 80% for both the young and adult trees. By improving the edge weights, this effect is expected to be reduced. But, due to the filtering procedure, not every fork center is accurately represented by a node. Thus, the forks will always be modeled with a lower accuracy than the rest of the model.

The effect of inaccurate linkages in the trunk section of the trees is caused by the high amount of trunks, branches and suckers at the bottom of the trees. This leads to significant shadowing effects resulting in a lag of data for the modeling procedure, but also to a low separability of the branches. Just by improving the methods, without increasing the number of laser scans, this effect is not expected to vanish.

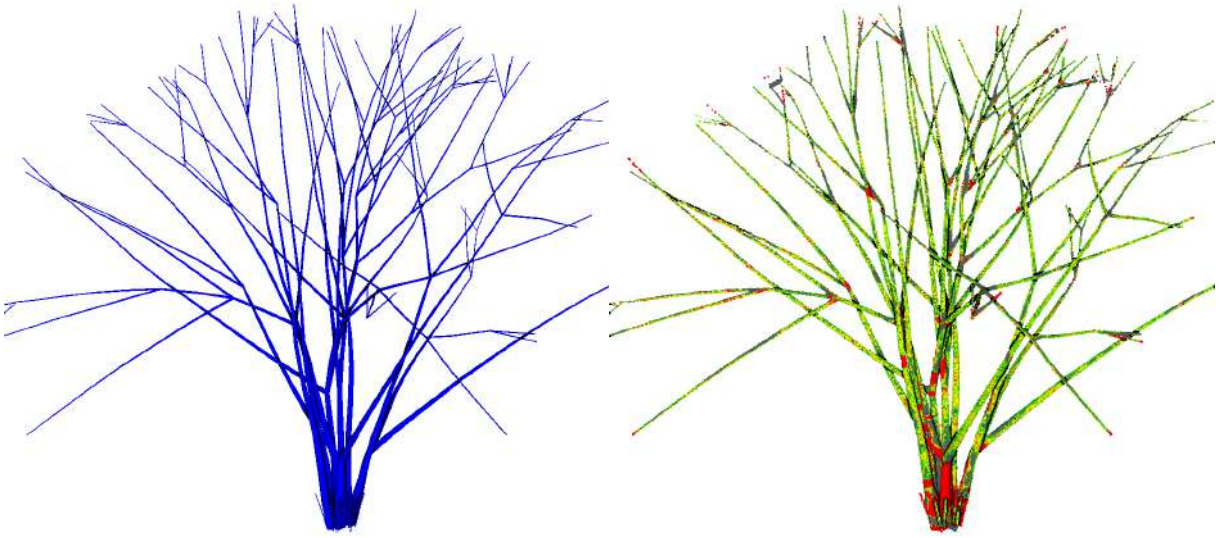

Fig. 14: Original tree geometry of a synthetic adult tree (left) and its model generated by the tree geometry reconstruction (right). The colors of the points of the original point-cloud represent the model residuals ranging from -10 mm (blue) over 0 mm (green) to 10 mm (red).

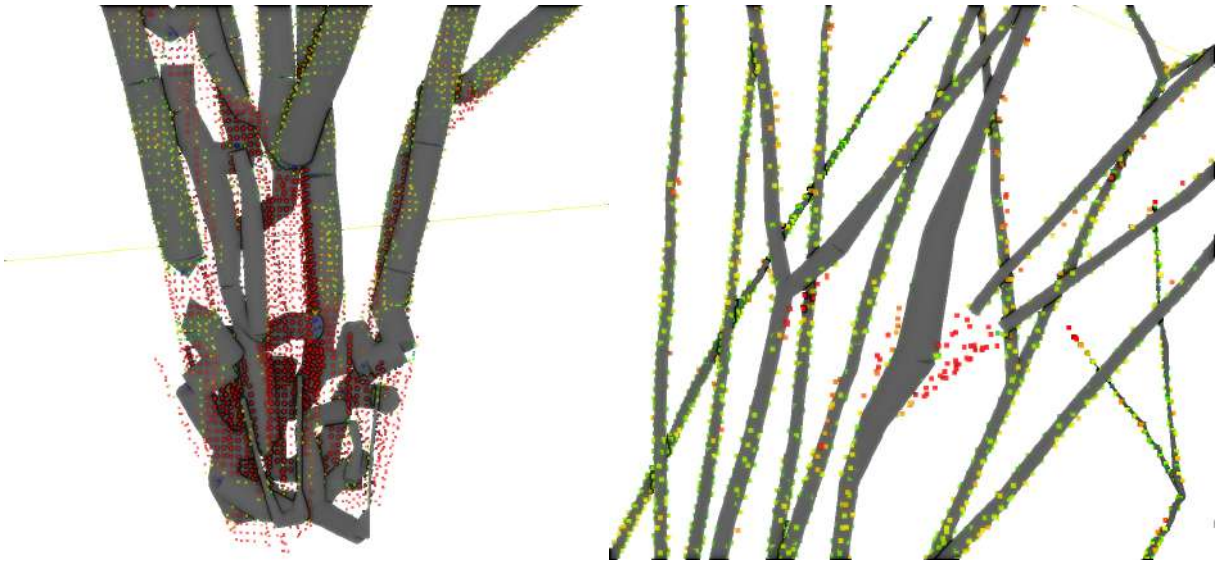

Fig. 15: Detail view of the tree geometry in the trunk section of a synthetic adult tree (left) and a detailed view of several forks (right). The colors of the points of the original point-cloud represent the model residuals ranging from -10 mm (blue) over 0 mm (green) to 10 mm (red).

4.7.3 Real-World 3D Models

Fig. 16 illustrates the result of the branch classification algorithm for an adult and two young real-world hazelnut trees from *Campaign 2020/01/10*. All trees are heavily affected by catkins. Due to the high amount of catkins, in some regions the branching structure is not captured by the laser scans due to shadowing

effects. Most of the visible branches could be identified, but not all catkins could be removed from the scans, resulting in a noisy result. Since the branch classification algorithm has been trained on datasets of the young trees only, branches of the adult trees are partially misclassified.

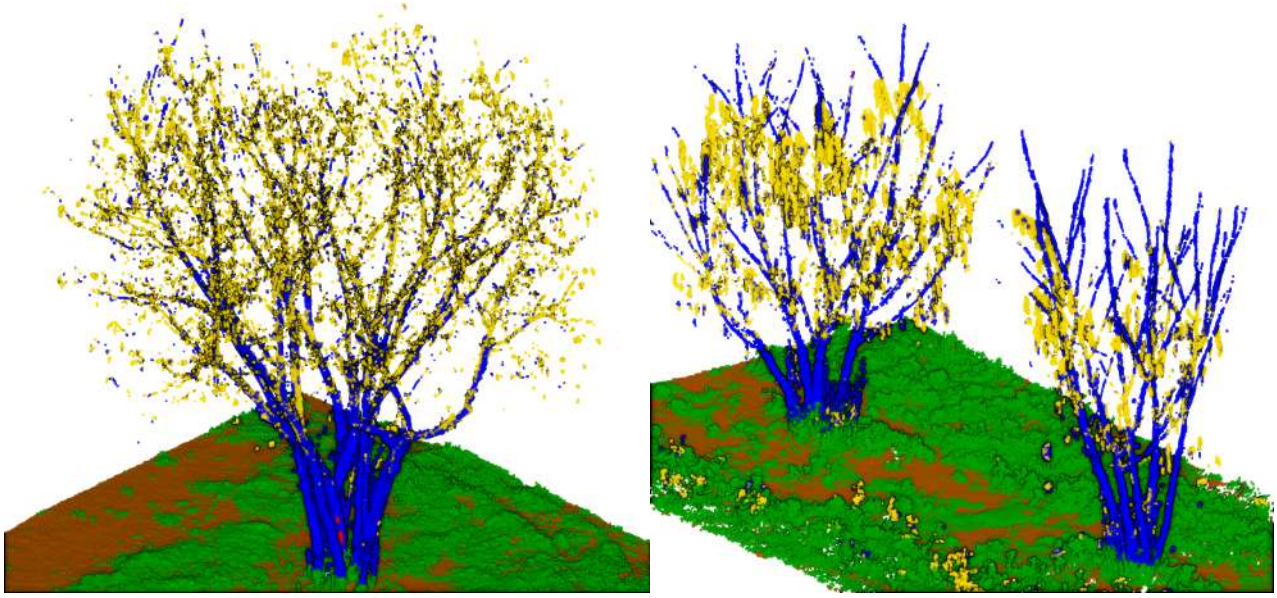

Fig. 16: Result of the branch classification algorithm for an adult real-world hazelnut-tree (left) and two young real-world trees (right). In particular the young trees are heavily affected by catkins. Ground is colored brown, low vegetation green, identified branches blue, and catkins yellow. Noise has already been removed from the scan.

Fig. 17 and Fig. 18 illustrate the modeled 3D vector representation of the given trees. Although the general structure of the trees could be extracted, not all branches could be connected due to the shadowing effects. In addition, some artifacts can be seen in the data due to remaining catkins. This can also be seen in the distribution of model residuals as summarized in Tab. **5**. Half of the residuals range from about -2.5 mm to 4 mm for the young trees, and from -3.5 mm to 5 mm for the adult trees. The fraction of residuals smaller than 10mm ranges from 77.6% to 85.5% for the young trees, and from 67.6% to 81.9% for the adult trees. Based on these values *Validation Measure A3* is not fully achieved for each tree.

Tab. 5: Residual analysis of the real-world 3D tree models. Trees indicated with a star are outer trees with limited scan overlap.

Age class	Tree ID	25% to 75% residual range [mm]	Fraction of absolute residuals smaller 10mm [%]
Young	Tree 1*	[-2.23, 4.18]	84.6
	Tree 2	[-2.31, 3.66]	83.9
	Tree 3	[-2.75, 5.38]	77.6
	Tree 4*	[-2.00, 3.06]	85.5
Adult	Tree 5*	[-4.38, 8.67]	67.6
	Tree 6	[-3.14, 4.78]	81.9
	Tree 7*	[-3.50, 4.87]	81.2

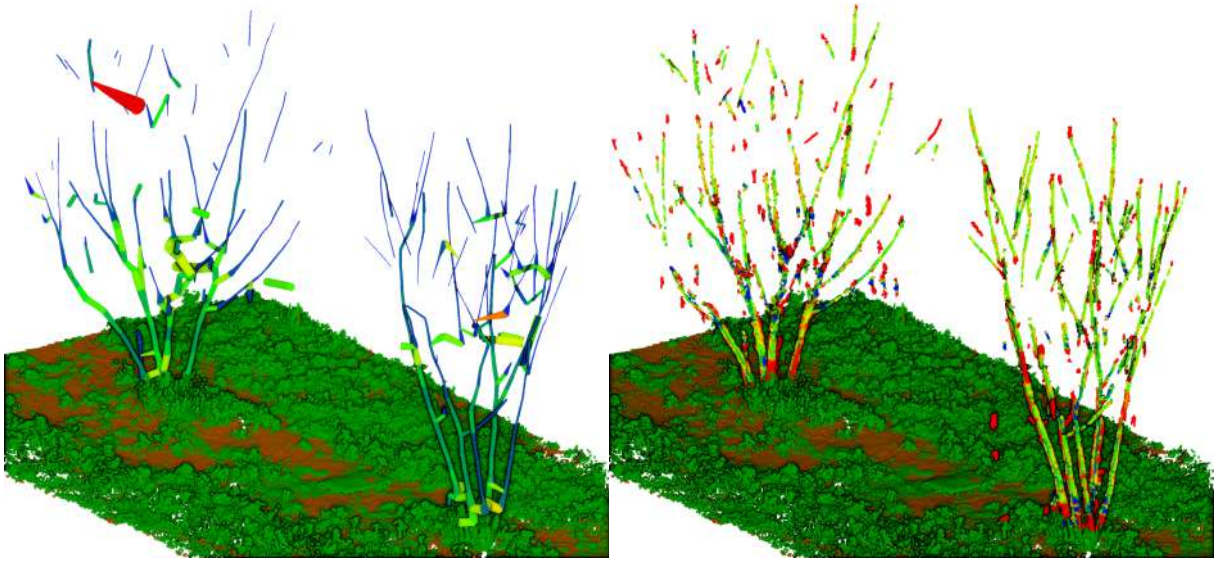

Fig. 17: 3D vector models of two young hazelnut trees (left) and corresponding visualization of the point residuals (right). The skeleton color corresponds to the predicted average branch diameter. The residuals are scaled from -1cm (blue) to 0cm (green) to 1cm (red).

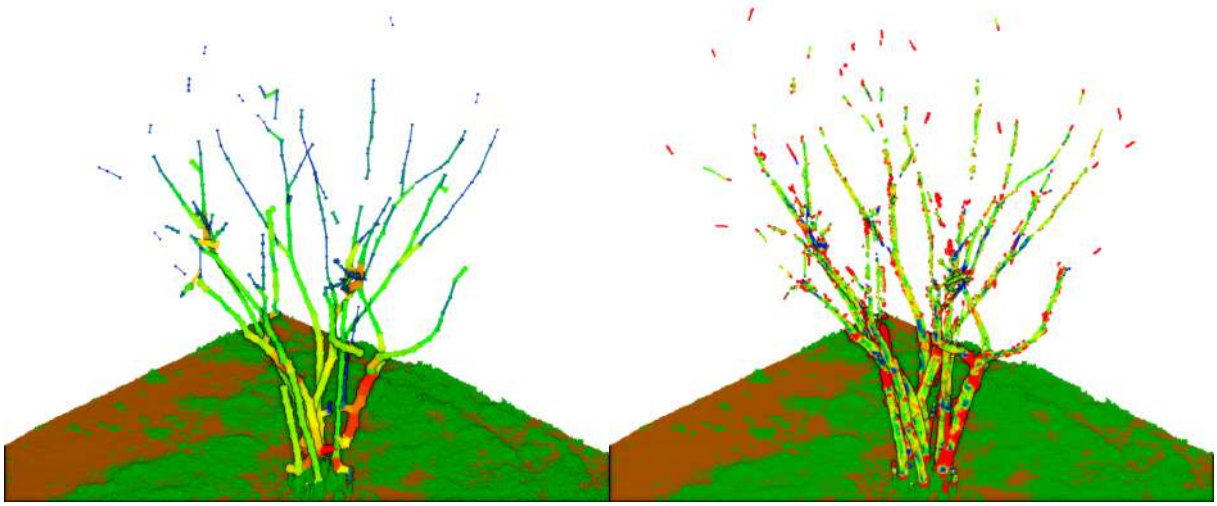

Fig. 18: 3D vector models of an adult hazelnut trees (left) and corresponding visualization of the point residuals (right). The skeleton color corresponds to the predicted average branch diameter. The residuals are scaled from -1cm (blue) to 0cm (green) to 1cm (red).

Due to the problems with filtering the branches, it is planned to increase the amount of reference data to further improve the branch classification accuracy. Additionally, a manual labeling of the forks is planned to improve the graph weights. After the final definition and implementation of the data repository (task T3.3) these improvements in branch calssification are expected to be easily achievable.

5 Sucker Detection

As they are a major input in PANTHEON for task *T5.1 – Sucker's management protocol*, suckers need to be identified and located. Several features of the suckers might be extracted to make a management decision.

To address different aspects of the sucker detection, two different methods for sucker detection were developed.

The first method (Section 5.3) provides high detail information on the geometry of the suckers. It takes advantage of the capabilities of a high-resolution laser scanner to get detailed information on the shape and volume of the suckers. It might also be used as a reference to calibrate the sucker volume estimation of the second method.

The second alternative method (Section 5.4) has been developed to identify suckers instantly in the field using a low-resolution LiDAR or stereo cameras. It analyses a coarse point cloud, which is suitable to receive the location of the suckers and approximate their volumes. The method is mainly intended to be used as a variant of computer vision to enable a ground robot to apply herbicides for sucker control autonomously.

5.1 Pre-Analysis

To get an impression about the suitability of different spectral bands to identify suckers, *MicaSence RedEdge-M* images were taken. After an alignment of the spectral bands, the spectral information is inspected to distinguish trunks from other types of surfaces.

Compared to adult leaves, young suckers appear light green (Fig. 19). Fig. 19 highlights that the spectral response of suckers—compared to adult leaves—differs significantly for the green band (560 nm) and the Red-Edge band (717 nm). As a consequence, the *ExG* greenness index and the *NDRE* are able to highlight suckers. Using the NDVI, the rest of the plant can also be distinguished from ground or other materials. In

addition to these spectral features, the local 3D structure of the suckers seems to be characteristic of other parts of the plant, like leaves or branches.

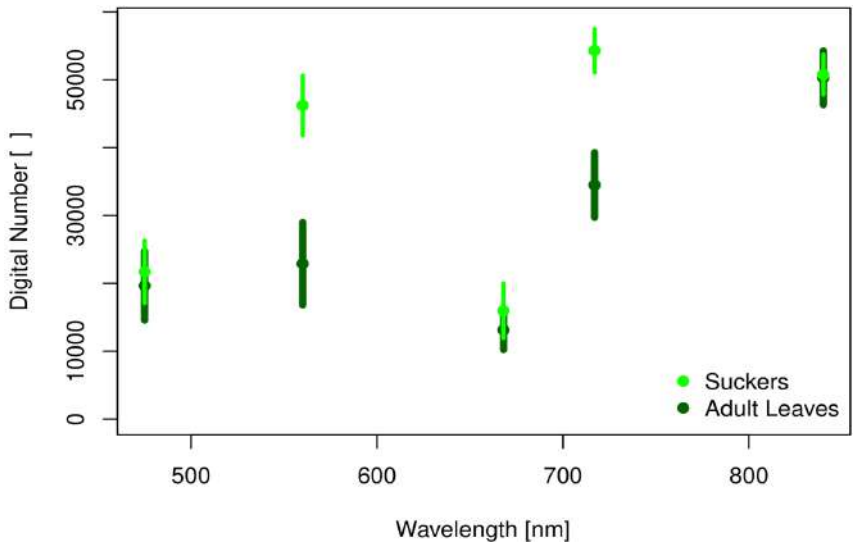

Fig. 19: Spectral response of leaves associated with suckers in comparison to adult leaves.

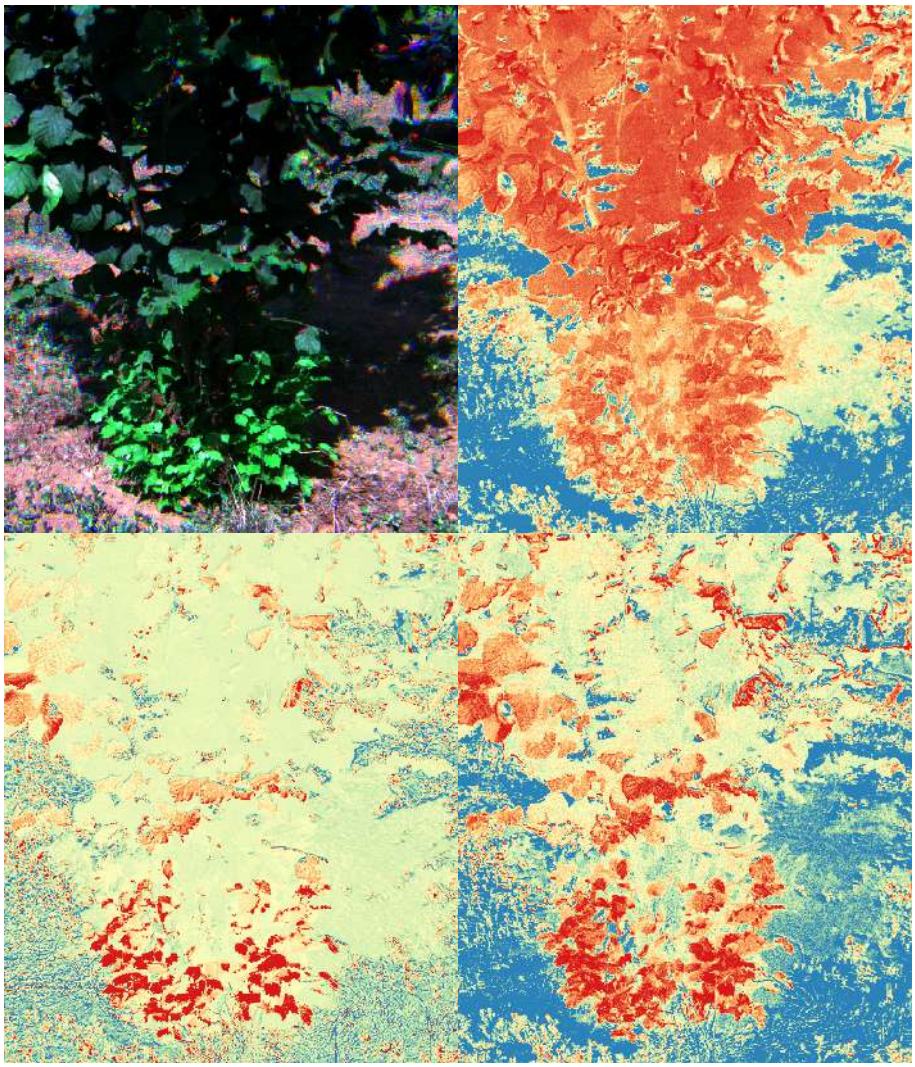

Fig. 20: True color composite (top-left), NDVI (top-right), ExG (bottom-left) and NDRE (bottom-right) of hazelnut tree.

As Fig. 20 shows, the hazelnut trees develop dozens of tight neighborhood suckers. In consequence, a spatial separation of individual suckers is hard to achieve. However, by identifying points associated with suckers, various features can be extracted, which are suitable to make pruning decisions. Thus, the algorithms developed for this deliverable label the 3D laser point clouds to be able to extract a sub-point cloud associated with suckers for each tree.

Point cloud features suitable to make a pruning decision for a specific tree might be:

- Number of points associated with suckers
- Volume of the suckers
- Vertical extent of the points associated with suckers
- Area of the points associated with suckers

Based on such features, along with a sample of pruning decisions of human experts, a classifier can be trained to make pruning decisions automatically.

5.2 Benchmarks

The benchmarks defined in deliverable D2.1 – *Requirements, Specifications and Benchmarks* foresee the possibility that small suckers (smaller than 5cm) are harder to detect than taller suckers. Thus, **VALIDATION measure A4** for the detection of presence and absence of suckers has been defined as follows:

- Suckers longer than 5 cm: omission error below 20% and commission error below 20%
- Suckers shorter than 5 cm: omission error below 25% and commission error below 25%

5.3 Sucker Detection Using High Resolution LiDAR

5.3.1 Methods

The sucker detection for the *Faro Focus S70* is based on an automated classification of the image pixels respective to the 3D points of the laser scans. As presented in Section 3, structural and spectral features are extracted for each point of the scan. Selected RGB scans are classified manually for training and validation. Based on the manual classification and point features, two Random-Forest classifiers are trained—one using just the intensity, structural features and dimensionality features (named *Sucker Classifier*) and another also using the ExG index (named *Spectral Sucker Classifier)*—to be able to assign each point to the most probable class. The *Sucker Classifier* is trained to distinguish healthy or dead suckers against any other class, while the *Spectral Sucker Classifier* is trained to distinguish the classes “healthy sucker”, ”dead sucker” and “no sucker”.

Since the classifiers will assign each point individually to a class, some point associated with suckers will be incorrectly assigned. Nevertheless, if these incorrect assignments will be spatially randomly distributed, the volumetric structure of the suckers will be maintained. Therefore, the volume and structure of the suckers can be evaluated by inspecting the subset of the point-cloud classified as suckers.

5.3.2 Experimental Setup and Data

The sucker detection is based on 3D point clouds provided by the *Faro Focus S70* laser scanner. To gain information on the 3D structure and volume of the suckers, each tree is scanned from four surrounding positions as described in D4.1 – *Terrestrial Remote Sensing pipeline*. The results presented in this document refer to the following campaign:

- **Campaign 2019/12/11**

In December 2019, six RGB scans of young hazelnut trees were taken to introduce spectral information in the sucker detection classification. To ensure an optimal alignment of spectral images, and a lag of database integration, instead of spectrally enriched point clouds—as described in D4.1 – *Terrestrial Remote Sensing pipeline*—native RGB scans were taken. The scans were not geo-referenced, since the scans were taken with the scanner mounted on a tripod only. In particular, the dataset is intended to inspect the separability of healthy suckers from dead suckers and other materials, like soil, grass, branches or leaves.

The default scanner parameters (*Distance=near*, *Resolution=1/8* and *Quality=2x*) were used.

5.3.3 Methods for validation

To address *VALIDATION measure A*, several classification metrics were derived. To address the expected increased uncertainty for small suckers, or suckers close to the ground, the validation dataset is split vertically into a point cloud of points lower 5 cm above ground, respectively higher 5 cm. For each of the tree validation datasets, along with a confusion matrix [28] and overall accuracy, the precision and recall [29] metrics are calculated for each class. Points pre-classified as ground or noise are excluded from the accuracy assessment, since they are typically removed when applying the sucker filtering.

To ensure a high level of independence between the training and validation points, the *Sucker Classifier* and the *Spectral Sucker Classifier* have been trained with four RGB scans, while two RGB scans are utilized for validation.

It needs to be mentioned, that the datasets used for the given deliverable were characterized by suckers taller 5cm. Thus, selecting points below 5cm is a workaround. When the integration of the processing pipeline has been fully achieved, terrestrial reference measurements of the number, volume, height and diameter of suckers can be used to reevaluate *Validation measure A4*.

5.3.4 Results and Discussion

Fig. 21 and Fig. 22 illustrate the identified feature importance of the *Sucker Classifier* and *Spectral Sucker classifier*. In particular, the height above ground convolutions were important features for both classifiers. This is logical, as the suckers are expected to be located close to the ground. The dimensionality features—*feature_scale-<scale>_p1* and *feature_scale-<scale>_p2*—provide valuable information to distinguish suckers from adult leaves. Due to the regular planting layout and the resulting regular scanning pattern, the structural features of height and distance allow for a focused search for suckers, resulting in high feature importances.

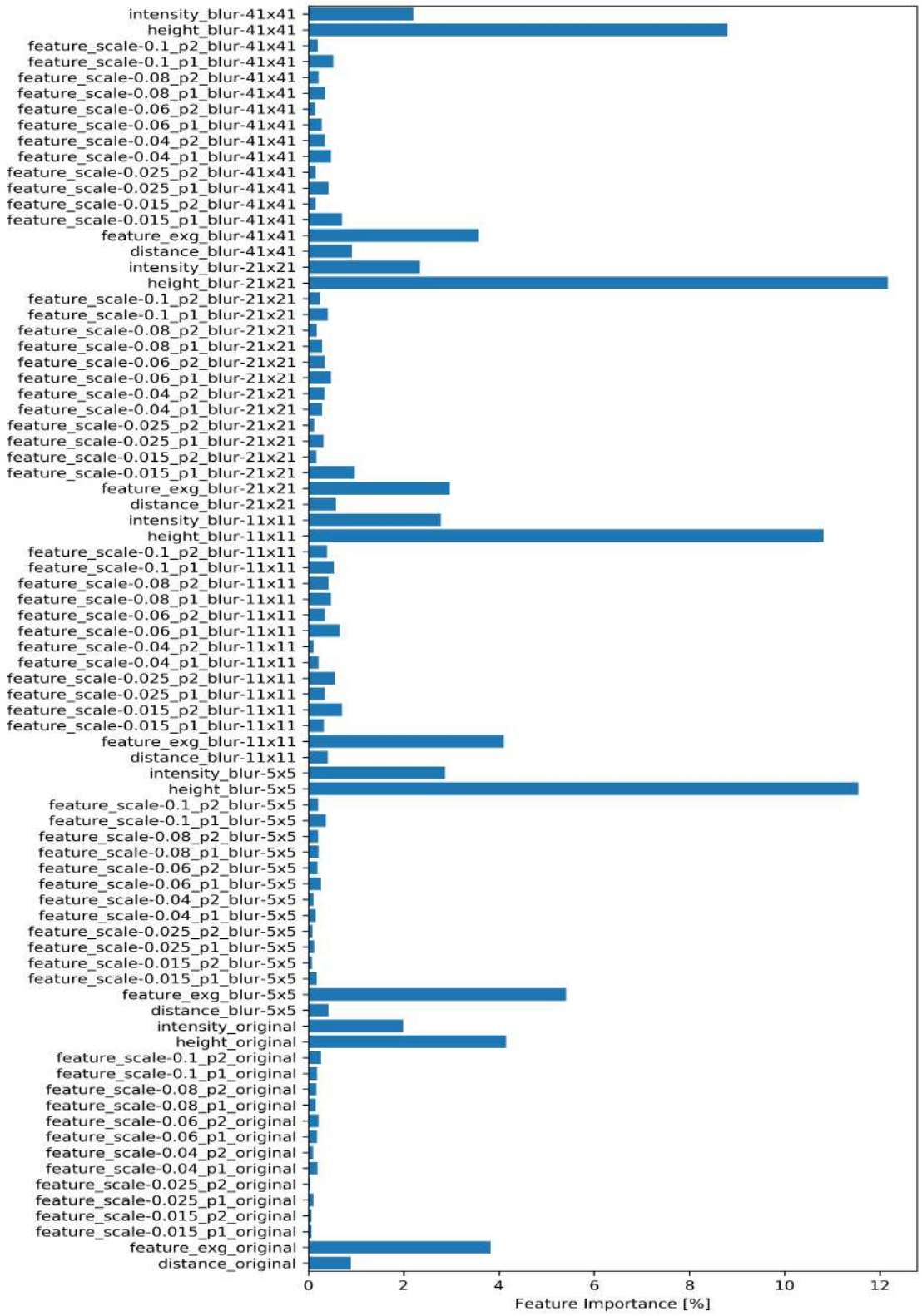

Fig. 21: Feature importance of the Random-Forest classifier for sucker classification using spectral features.

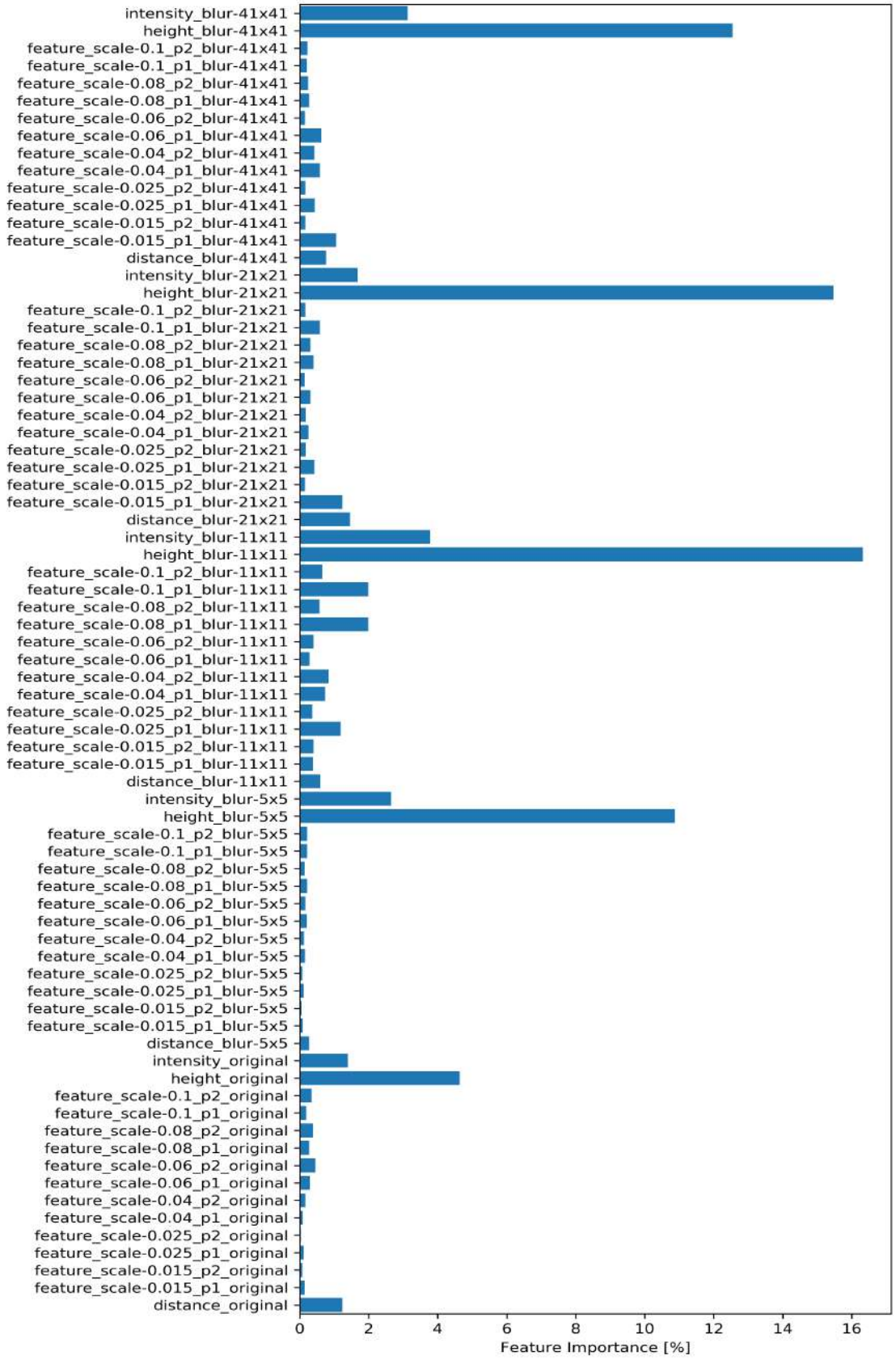

Fig. 22: Feature importance of the Random-Forest classifier for sucker classification ignoring spectral features.

Fig. 23 illustrates the result of the *Spectral Sucker Classifier* for a young hazelnut tree in field 16. The suckers can be distinguished from branches and other materials. In some cases, suckers close to the canopy cannot

be distinguished from adult leaves. The same problem occurs for some grass or herbs. It needs to be mentioned, that the spectral information is also capable of identifying drought suckers.

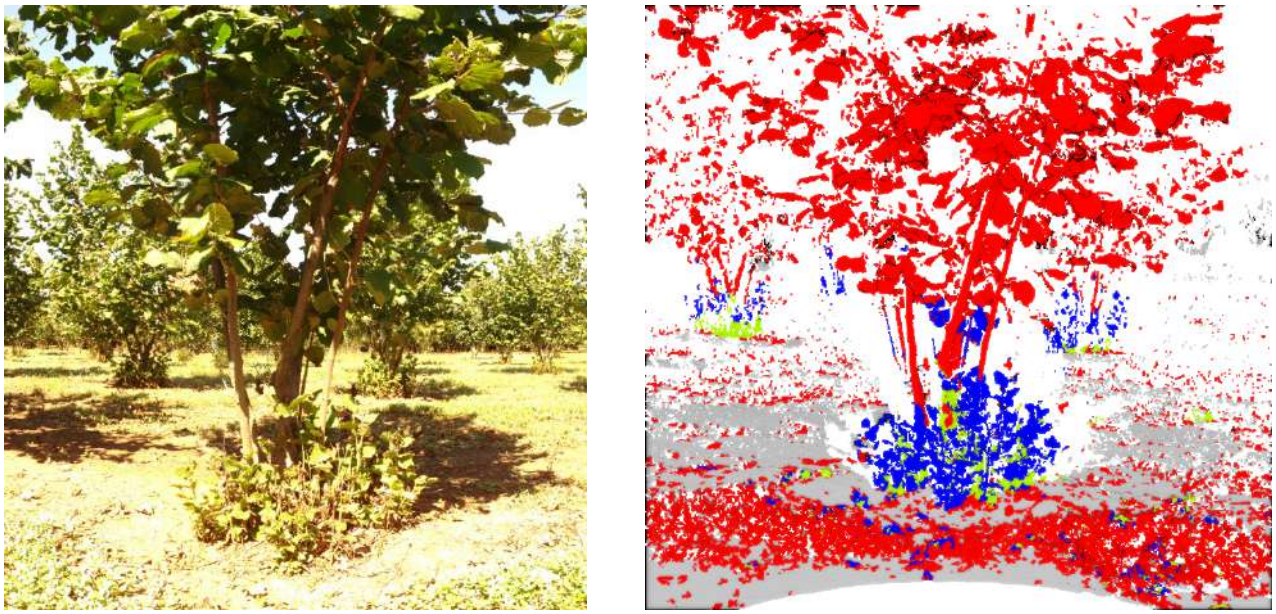

Fig. 23: 2D image representation of an RGB scan of a young hazelnut tree (left) and the 3D result of the *Spectral Sucker Classifier* (right). Healthy suckers are marked in blue, dead suckers in green, soil in gray and other classes in red**.**

The accuracy metrics for the *Spectral Sucker Classifier* are summarized in the confusion matrices of Tab. 6 and Tab. **7**. The classifier achieves a total accuracy of about 95%. With a commission error of about 4% and omission error of about 16%, the classifier is able to identify healthy tall suckers. For the small suckers, a commission of 13% and omission error of 3% is not exceeded. Based on these accuracy metrics *Validation Measure A4* is achieved at the point-level. The dead suckers are harder to detect, which leads to an omission error of about 61% and an omission error of about 14% for tall suckers. Due to missing reference data, the accuracy of the small dead suckers can't be estimated.

Tab. 6: Confusion matrix of the *Spectral Sucker Classifier* for suckers taller than 5cm**.**

		Reference				
		Healthy Sucker	**Dead Sucker**	**No Sucker**	**Precision [%]**	
Predicted	**Healthy Sucker**	4556	3	838	84.4	74.5
	Dead Sucker	138	333	377	39.3	
	No Sucker	27	52	24126	99.7	
	Recall [%]	96.5	85.8	95.2	95.3	
			92.5			

Tab. 7: Confusion matrix of the *Spectral Sucker Classifier* for suckers smaller 5 cm.

		Reference			
		Healthy	**Dead Sucker**	**No Sucker**	**Precision [%]**
Predicted	**Healthy Sucker**	408	0	11	97.4
	Dead Sucker	35	0	25	NA
	No Sucker	24	0	1969	98.8
	Recall [%]	87.4	NA	98.2	96.2
			92.8		98.1

Tab. 8 and Tab. **9** summarize the accuracy metrics for the *Sucker Classifier*. With a commission error of about 24% and omission error of close to 0%, the classifier is able to identify tall suckers. For the young suckers, a commission of 15% and omission error of 18% are not exceeded. Based on these values, the commission error of *Validation Measure A4* is also achieved on the point-level. But these results highlight the importance of the spectral information for sucker identification. Significantly improved classification results are expected by enriching the laser scans with the multi-spectral images.

Tab. 8: Confusion matrix of the *Sucker Classifier* for suckers taller 5 cm.

		Reference		
		Sucker	**No Sucker**	**Precision [%]**
Predicted	**Sucker**	5108	1583	76.3
	No Sucker	1	23758	100.0
	Recall [%]	100.0	93.8	94.8
		96.9		96.0

Tab. 9: Confusion matrix of the *Sucker Classifier* for suckers smaller 5 cm.

		Reference		
		Sucker	**No Sucker**	**Precision [%]**
Predicted	**Sucker**	397	82	82.3
	No Sucker	70	1923	96.5
	Recall [%]	85.0	95.9	93.9
		95.9		89.7

5.4 Sucker Detection Using Low Resolution Point Clouds

5.4.1 Methods

5.4.1.1 Calibration and Data Association

To identify suckers using low resolution point clouds, 3D LiDAR and RGB camera data was collected with the UGV (refer Section 2 for details on the Hardware). In order to link visual and 3D data we carried out a preliminary extrinsic parameters calibration procedure.

We assume the intrinsic calibration parameters of the camera to be known. The first step is to obtain a set of unique correspondences between the LiDAR and the camera. As stated directly in [30], it is recommended to perform a calibration among two point clouds, since a 2D knowledge of the points in the camera frame might lead to non-negligible errors. With this in mind, two static markers have been disposed in front of the two sensors enabling for a 3D knowledge of their corners in the camera frame, and in the LiDAR frame as well through an edge detection process.

By doing so we have the set of 3D point correspondences under the form of two point clouds P_c and P_l, in the camera and in the LiDAR frames, respectively.

The transformation between the two frames is thus estimated by means of the Iterative Closest Point algorithm (see [31]) given in Equation (6, which tries to minimize the 3D Euclidean distance between the two point clouds:

$$\min_{R\in SO(3),t\in R^3} \|(RP_l + t) - P_c\|^2 \tag{6}$$

Where R represents the rotation matrix and t the translation vector mapping the 3D points in the LiDAR frame to the camera frame. By means of these calibration parameters it is possible to uniquely associate each 3D point X_l collected by the LiDAR to the camera image plane point p through the following set of equations:

$$X_c = RX_l + t \tag{7}$$

$$x' = \frac{x_c}{z_c}; \quad y' = \frac{y_c}{z_c} \tag{8}$$

$$p = K[x', y', 1]^T \tag{9}$$

5.4.1.2 Algorithm

In this section we focus on the sucker detection problem, i.e., the problem of extracting a set of Regions Of Interest (ROIs) $B = (B_0, B_1, \dots, B_i)$ (see Fig. **24** for an example of ROI) from the input images, such that they bound the suckers to inspect. Detecting suckers in an open-environment present a few major challenges: (i) the background noise (e.g. grass, or other plants [7]) and (ii) the extremely variable light conditions.

We tackle these issues by fine-tuning the state-of-the-art convolutional neural network YOLOv3 [32]. More specifically, we choose the YOLOv3-tiny architecture (depicted in Tab. **10**) as a trade-off between computational time and accuracy. It is composed of successive and convolutional layers, and it accepts input images with size 608 x 608.

Tab. 10: YOLOv3-tiny architecture.

Layer type	Filters	Filter size	Image output size
Convolutional	16 / 2	3 × 3	304 × 304
Convolutional	32 / 2	3 × 3	152 × 152
Convolutional	64 / 2	3 × 3	76 × 76
Convolutional	128 / 2	3 × 3	38 × 38
Convolutional	256 / 2	3 × 3	19 × 19
Convolutional	512 / 2	3 × 3	10 × 10
Convolutional	1024 / 2	3 × 3	10 × 10
Convolutional	256 / 2	1 × 1	10 × 10
Convolutional	512 / 2	3 × 3	10 × 10
Convolutional	256 / 2	1 × 1	10 × 10
Avgpool		Global	
Connected		1000	
Softmax			

The fine tuning is performed on a pre-trained model, trained on the ImageNet [33] dataset, with on-field gathered datasets. To achieve a reliable sucker detection, the datasets have been gathered in different weather and daylight conditions. The accuracy of the proposed neural network architecture will be discussed in the experimental section.

5.5 Experimental Setup and Data

To validate the methods young trees of the and the validation of the proposed approach have been conducted within PANTHEONs field 16, which hosts the young orchard.

5.5.1 Datasets

To test the sucker detection process, we gathered four datasets in different weather and daylight conditions: $DataSet_A$, composed of 715 images and acquired in the middle afternoon and late morning: $DataSet_B$, composed of 61 images and acquired in the early morning with cloudy sky; $DataSet_C$, composed of 181 images and acquired at midday; $DataSet_D$, composed of 213 images and acquired in the late afternoon. We point out that the heterogeneity in the size of the collected datasets depends on the daylight conditions. Indeed, the most varying light conditions happen in the middle afternoon and in the late morning, leading $DataSet_A$ to be the largest dataset, while $DataSet_B$ to be the smallest dataset since the cloudy sky does not involves variable shadowing conditions. The four datasets were recorded across four different weeks in the summer, and each one contains different maneuvers where the robot approaches a hazelnut plant with a sucker. $DataSet_A$ and $DataSet_C$ were then divided into train and test datasets, namely $Train/TestSet_A$ and $Train/TestSet_C$. Conversely, $DataSet_B$ has been entirely used for training, while $DataSet_D$ has been split into two individual test sets, namely $TestSet_{D1}$ and $TestSet_{D2}$. It is important to remark that the fast growth rate of the suckers potentially leads to sensibly change their appearance in a week. Therefore, our aim is to use $TestSet_{D1,2}$ to test how the neural network generalizes on previously unseen sucker conditions.

5.5.2 Results and Discussion

Fig. **24** shows the output of the convolutional neural network across the different test datasets. We report numerical results in Tab. **11** according to the following metrics: (i) the average Intersection over Union (IoU), (ii) the recall rate at an IoU percentage of 50% (.5R), and (iii) the recall rate at an IoU percentage of 75% (.75R). To evaluate the sucker detection process for each approaching maneuver, the datasets have been further divided into $TestSet_I$ and $TestSet_{II}$. As reported, the IoU is always kept above the 50%, also in $TestSet_I$. The same trend is also reported in the recall rates. The only exception is the $SubSet_{II}$ in $TestSet_C$ where the camera was facing the sun, leading the RoIs to contain a significant noise. Mitigating this effect will be object of future work. A feasible solution may be to gather more data with similar light conditions, supplementing the current training datasets.

Fig. 24: Examples of network predictions for $TestSet_A$, $TestSet_B$ and $TestSet_{D1,2}$.

Tab. 11: Sucker detection statistics.

Test Set	Sub Set I			Sub Set II		
	IoU	.5R	.75R	IoU	.5R	.75R
$TestSet_A$	65.4	87.3	22.1	62.4	100	22.5
$TestSet_B$	72.7	100	42.3	59.6	76.3	05.2
$TestSet_C$	51.2	55.8	43.4	64.9	95.9	84.4
$TestSet_E$	72.7	85.2	39.3	69.3	53.2	21.1

6 Sucker Volume Estimation

6.1 Introduction

Given a 3D sparse point-cloud set S representing a sucker (Fig. **25**, left column), our goal is to find a function that computes a volumetric estimation of the sucker. Notably, the set can be either collected by one or more 3D sensors (e.g. LiDAR and RGBD cameras) or generated in a post-processing phase by a photogrammetry-based 3D reconstruction software. To provide the results for the given deliverable, the volumetric estimations have been done in real-time by collecting data from 3D sensors.

To derive an effective method, we made the following assumptions on the data collection process and on the structural properties of the sucker canopy:

i. The 3D point-cloud set mainly belongs to the target sucker;

ii. The leaves are connected through minimal paths.

Assumption i) is an essential data requirement and is fulfilled by extracting 3D points inside the ROI surrounding the target sucker. In this regard, we also perform a further outlier rejection refinement step by pruning out all those points that might possibly belong to the soil terrain.

Assumption ii) is a reasonable simplification assumption since, in a real-world scenario, the different branches of a plant are individually linked to a single connected component. Indeed, this latter assumption is explicitly exploited for the sucker volumetric estimation.

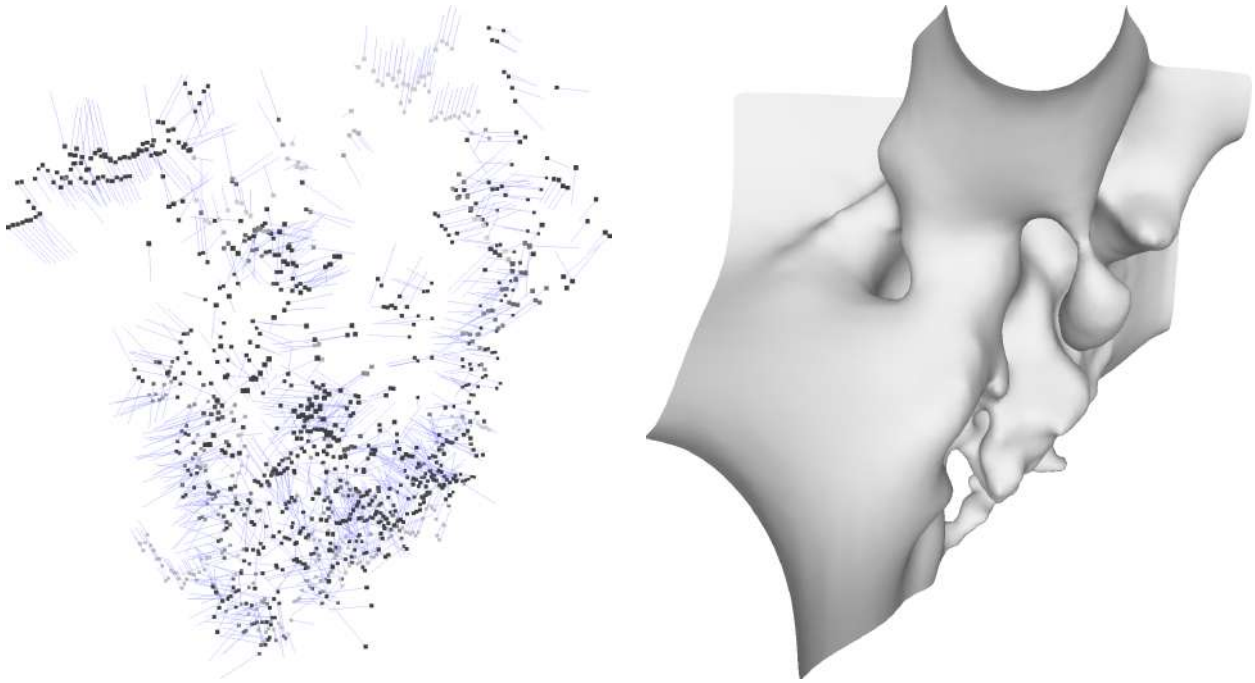

Fig. 25: The poor 3D mesh reconstruction provided by Screened Poisson reconstruction [34] (right), due to a noisy normal computation (shown in blue) and a sparse input point set (left).

6.2 Methods

In this section, we focus on the estimation of the sucker canopy volume. Notably, this process cannot be performed by solely relying on visual data. To this end, we exploit the sparse 3D point cloud dataset S obtained according to the data association described in the calibration section. We thus filter out data which lies outside of the ROIs. ROIs are determined through the detection process described in the sucker detection section.

At the end of this process, it is very likely that a non-negligible portion of the remaining 3D points does not belong to the sucker leaf surface (e.g. tree trunk or soil terrain). This small number of outliers could lead the volume estimation algorithm to yield poor results. Thus, as a preliminary step these misleading data are filtered out. We achieve this goal by means of an ExG index thresholding operation.

In our case, having a unique map between 3D and color data, we exploit the above introduced index to filter out all of those points which do not specifically belong to the sucker canopy by checking their ExG index magnitude. Despite its simplicity, this method enables pruning of most of the outliers. However, more refined strategies for outlier processing and removal will be the object of future work.

At this point, given the filtered point cloud, we start the estimation procedure by building a 2-manifold triangle mesh. Notably, if such mesh is also watertight and free of self-intersections, then its exact volume can be determined using the simple algorithm described in [35]. In practice, the volume can be determined by summing up the signed volume of all the tetrahedrons created by connecting the three vertices of each face with the origin. The main input required with this method is a 3D mesh that fulfills the topological conditions we cited above.

There are several methods to reconstruct such a 3D mesh from a point cloud. This research area has seen substantial progress in the past two decades, the reader is referred to [36] for a comprehensive overview of this topic. While most of these methods are actively used in the industry to produce well-structured triangular meshes, they usually require a dense point cloud as an input, a pre-condition which unfortunately

does not hold in our application scenario. As a matter of fact, open-air vegetation is extremely challenging to acquire using scanning devices, and the result is usually a quite sparse point cloud where state-of-the-art reconstruction methods perform poorly (see Fig. **25**). Hence, we propose a novel automated strategy to derive a volumetric approximation of the sucker canopy.

input: A set of samples $s_i \in S$
output: A new set of connected Samples

STEP 1 - AVERAGE : computing average distance α between each sample and the closest;
$\alpha \leftarrow 0$;
forall $s_i \in S$ **do**
    $s_i^C \leftarrow FindClosest(S, s_i)$;
    $\alpha \leftarrow |s_i^C - s_i|$;
end
$\alpha \leftarrow \alpha / |S|$;

STEP 2 - CLUSTERING : Cluster samples whose distance is below α;
forall $s_i \in S$ **do**
    $c_i = \{s_i\}$;
end
merged ← *true*;
while *merged* **do**
    merged ← *false*;
    forall $c_i \in C$ **do**
        $c_i \leftarrow FindClosestCluster(c_i)$;
        if $(ClusterDistance(c_i, c_i^C) < \alpha)$ then
            $c_i \leftarrow c_i \cup c_i^C$;
            $C \leftarrow C - c_i$;
        merged ← *true*;
        end
    end
end

STEP 3 - LINK : Greedy add links between partitions until a single connected component is created;
while |C| > 1 **do**
    $c_0, c_1 \leftarrow FindClosestClusterPair(C)$;
    $c_{link} \leftarrow FindLinkSamples(c_0, c_1)$;
    $c_{new} \leftarrow c_0 \cup c_1 \cup c_{link}$;
    $C \leftarrow C \cup c_{new}$;
    $C \leftarrow C - c_0$;
    $C \leftarrow C - c_1$;
end

Algorithm 1: A pseudocode illustrating the different steps of our clustering procedure.

Given a sparse point cloud set S (see Fig. **26** (b)), we first estimate a distribution factor α, as the average distance between each point $s_i \in S$ to its closest sample s_i^C (see Fig. **26** (c)). Then we connect the samples whose distance is below α, and we cluster the connected components (see Fig. **26** (d)). Intuitively, if we create a sphere with a radius of α for each sample, then every cluster will result in a connected volume. However, in a real-world scenario, the different branches of a plant are inexorably linked to a single connected component. Consequently, we connect the different branches by using a greedy strategy that

favors the connection between closest cluster. Given the closest pair of clusters c_0 and c_1, we connect them to a new cluster c_{new} by creating a new set of samples which are distributed along the shortest segment that connects c_0 with c_1. We iterate this process by repeatedly connecting the closest pair of clusters until all the samples are joined to a unique cluster. Notice that for these new samples, we defined a sphere with a smaller radius $\alpha/5$ as they represent the branches of the sucker (see Fig. **26** (e)). We point out that the value of alpha for the branch of the sucker is the result of a trial and error procedure that has been carried out in order to obtain the lowest volumetric estimation error.

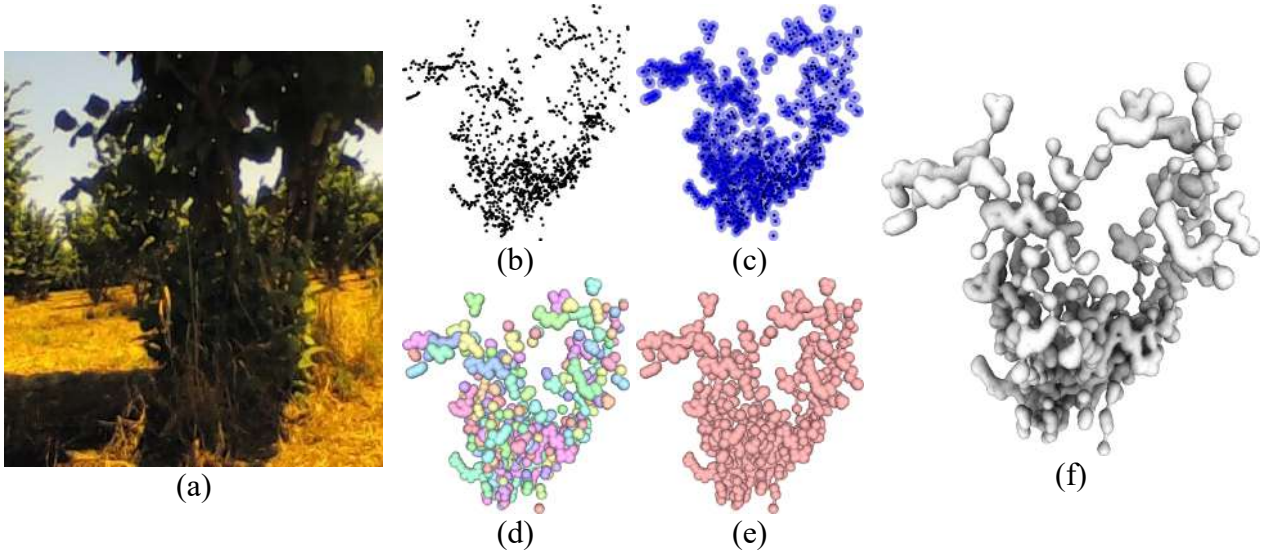

Fig. 26: The 3D reconstruction pipeline: Given a sparse set of 3D points (b) sampled through 3D scanning on the real sucker (a) we estimate the average radius α (c) and we cluster samples based on their proximity (d); then we connect the separated branches using a greedy strategy (e) and we gather the final mesh as the union of all the spheres (f).

A pseudo-code of this procedure is described by Algorithm 1. We finally create a single mesh for every sphere, and we merge them all by using the Boolean operations of [37] implemented in LibIGL [38]. At this stage, as we performed the union of a set of closed, 2-manifold, orientable spheres, we also guarantee a closed, 2-manifold, orientable mesh T as an output, matching the condition for exact volume calculation. However, the mesh resulting from the Boolean operation might have badly shaped triangles and geometrical sharp features which can affect the volume estimation. Hence, we improve the meshing by performing a couple of loop subdivision steps [39] using the implementation in the VCG library [40] (see Fig. **26** (f)). Future work will be focused on the optimization of this pipeline for the sake of real-time computation.

Finally, as we previously stated, the volume of the mesh can be computed as the sum of all the signed volumes of the tetrahedra bounded by the origin and the three vertices of a triangle. The signed volume V of the tetrahedron composed by the origin $(0,0,0)$ and the three oriented vertices of a triangle a, b, c, can be computed as:

$$V = \frac{1}{6}(a_x b_y c_z + a_y b_z c_x + a_z b_x c_y - a_x b_z c_y - a_y b_x c_z - a_z b_y c_x) \quad (10)$$

6.3 Experimental Setup

To evaluate the sucker volume estimation, a UGV campaign was performed in December 2019 (see Section 4.6).

6.4 Results and Discussion

In this section, we experimentally demonstrate the accuracy of the proposed volumetric estimation. To calculate an error metric, the first step was to obtain a reliable ground truth. In the specific case of suckers, standard 3D reconstruction approaches based on 3D dense point clouds, as shown in Fig. **25**, might suffer the 3D geometric irregularities of the sucker surface. Thus, to get an accurate estimation of the suckers real volume we used a variation of the Archimedes method. Specifically, the target sucker is suspended below the surface of the water in a container placed on an electronic scale. The volume V of the immersed object will simply be the increase in weight Δw divided by the water density ρ, that is $V = \Delta w/\rho$.

The reader is referred to [41] for a comprehensive overview of this topic. To test the accuracy of the proposed method we measured three suckers, namely $Sucker_{1,2,3}$.

Tab. 12: Sucker volumetric estimation statistics.

Sucker ID	RMSE [%]	St. Dev. [%]	Ground Truth [cm³]
$Sucker_1$	21%	5.1%	631.9
$Sucker_2$	15%	3.8%	834.3
$Sucker_3$	6%	2.5%	372.8

Notably, these preliminary results seem very promising as according to Tab. **12** as the RMSE remains below the 21% in all datasets. The entire process takes on average 10 seconds on a 2,9 GHz *Intel Core i7 Mac*, except for the Boolean union step which might take up to 2 minutes. However, the efficiency of the pipeline can be significantly improved substituting this step by extracting the implicit surface resulting from the union of spheres using Marching Cubes [39].

7 Conclusions

7.1 Completed tasks

The given report documents algorithms developed for sucker detection and tree geometry reconstruction. In particular, a generic processing pipeline has been developed and implemented to classify laser scans in general as an input for sucker detection, but also for related tasks like *T4.8 – Fruit Detection*.

Algorithms to identify branches, to filter the 3D structure of the multi-stemmed hazelnut trees, and to create 3D tree vector models have been developed and implemented by adopting existing algorithms, but also by developing new methods where reasonable. The 3D models—intended as a base input for pruning planning (Objective 3.1)—represent the geometry of a tree as a three-dimensional hierarchical graph. This representation allows for an analysis of the branching topology as well as the estimation of available timber or canopy volume. In addition, the structure has been successfully used for the web-visualization in task *T3.5 – Design and Implementation of the User Interface*.

Two complementary algorithms to identify suckers based on different hardware and different levels of detail have been developed and implemented. In particular, suckers are marked and 3D point clouds of suckers can be extracted to derive higher order information, like the suckers volume or area. The latter are required as a base input for automated sucker management as defined by task *T5.1 – Sucker's management protocol*.

The processing chains have been partially integrated in the *MongoDB* database infrastructure, to the extent possible at the date of submitting the given document. The algorithms have been tested and validated to address the validation measures defined in deliverable D2.1 – *Requirements, Specifications and Benchmarks*.

7.2 Ongoing tasks

The methods presented in this deliverable have been tested using exemplary UGV campaigns. The point-clouds were not spectrally enriched, since the processing pipelines were not fully integrated in the database layout at time of submitting the document. After the integration of the processing chains in the database concept of deliverable *D3.3 – Definition and Implementation of the Data Repository*, the spectral enrichment can be applied for all UGV campaigns with camera images available. All algorithms have been designed in a generic way, to smoothly incorporate the spectral features when available.

Training the algorithms for sucker and branch detection requires a manual classification of selected laser scans. To increase the accuracy of the methods, scans of future campaigns will also be manually classified to increase the amount of training and validation data. This is particularly necessary, since catkins hamper the accuracy of the tree geometry reconstruction. Since a manual classification is also required for Task *T4.8 – Fruit Detection*, a standardized classification procedure can be applied. Doing so is expected to lead to a significant increase in the accuracy of the presented methods due to increased amount of training data.

The validation of the 3D models has shown that obstacles can affect the tree skeleton extraction significantly. Even by identifying the catkins, significant shadowing effects remain, and a loss of available scan points associated with branches. In consequence—next to an improvement of the branch classification—the edge weights of the 3D graph optimization need to be improved to achieve more accurate results. By manually labeling the identified branch sections of 3D models as “correct” or “incorrect”, it is expected to identify more suitable edge weights semi-automatically. Thus, a constant feedback loop with the agronomists of task *T5.2 – Pruning Management Protocol* is foreseen.

7.3 Ongoing research

The algorithms developed for Task *T4.3 – Tree Geometry Reconstruction* have been implemented in a generic way. Thus, improvements by replacing e.g. a Random-Forest classifier by more advanced classifiers, like convolution neural networks, could be done where reasonable. Since Task *T4.8 – Fruit Detection* is planned to exploit the processing chains for cloud classification presented in Section 3, the sucker and branch detection are expected to benefit from future achievements made for fruit detection.

8 References

[1] X. Liang *et al.*, "Terrestrial laser scanning in forest inventories," *ISPRS J. Photogramm. Remote Sens.*, vol. 115, pp. 63–77, May 2016.

[2] K. Calders *et al.*, "Nondestructive estimates of above-ground biomass using terrestrial laser scanning," *Methods Ecol. Evol.*, vol. 6, no. 2, pp. 198–208, Nov. 2014.

[3] P. Raumonen *et al.*, "Fast Automatic Precision Tree Models from Terrestrial Laser Scanner Data," *Remote Sens.*, vol. 5, no. 2, pp. 491–520, Jan. 2013.

[4] S. Lamprecht, J. Stoffels, and T. Udelhoven, "VecTree -- Konzepte zur 3D Modellierung von Laubbäumen aus terrestrischem Lidar," *Photogramm. - Fernerkundung - Geoinf.*, vol. 2015, no. 3, pp. 241–255, Jun. 2015.

[5] A. Bucksch, R. Lindenbergh, and M. Menenti, "SkelTre," *Vis. Comput.*, vol. 26, no. 10, pp. 1283–1300, Aug. 2010.

[6] R. Li, G. Bu, and P. Wang, "An Automatic Tree Skeleton Extracting Method Based on Point Cloud of Terrestrial Laser Scanner," *Int. J. Opt.*, vol. 2017, pp. 1–11, 2017.

[7] F. Kang, H. Wang, F. J. Pierce, Q. Zhang, and S. Wang, "Sucker Detection of Grapevines for Targeted Spray Using Optical Sensors," *Trans. ASABE*, vol. 55, no. 5, pp. 2007–2014, 2012.

[8] FARO, "FARO Laser Scanner Scanner User Manual." Feb-2018.

[9] "ESRI Shapefile Technical Description." Accessed on: Mar. 31, 2020 [Online]. Available: https://support.esri.com/en/white-paper/279.

[10] N. Brodu and D. Lague, "3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology," *ISPRS J. Photogramm. Remote Sens.*, vol. 68, pp. 121–134, Mar. 2012.

[11] E. Gumus and P. Kirci, "Selection of spectral features for land cover type classification," *Expert Syst. Appl.*, vol. 102, pp. 27–35, Jul. 2018.

[12] R. H. Chan, Y. Qiu, and G. Yin, "Iterative Methods for Eigenvalues/Eigenvectors," in *Encyclopedia of Social Network Analysis and Mining*, Springer New York, 2018, pp. 1154–1160.

[13] S. Lamprecht, "Pyoints: A Python package for point cloud, voxel and raster processing," *J. Open Source Softw.*, vol. 4, no. 36, p. 990, 2019.

[14] J. W. Rouse, R. H. Haas, J. A. Schell, and D. W. Deering, "Monitoring vegetation systems in the Great Plains with ERTS." *NASA special publication*, vol. 351, p.309, 1973.

[15] E. M. Barnes *et al.*, "Coincident detection of crop water stress, nitrogen status and canopy density using ground-based multispectral data." *Proceedings of the Fifth International Conference on Precision Agriculture, Bloomington, MN, USA*, vol. 1619, 2000.

[16] D. M. Woebbecke, G. E. Meyer, K. Von Bargen, and D. A. Mortensen, "Shape Features for Identifying Young Weeds Using Image Analysis," *Trans. ASAE*, vol. 38, no. 1, pp. 271–281, 1995.

[17] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[18] F. M. Waltz and J. W. V. Miller, "Efficient algorithm for Gaussian blur using finite-state machines" in *Machine Vision Systems for Inspection and Metrology VII*, 1998, vol. 3521, pp. 334–341.

[19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] Y. Q. Li *et al.*, "Development and evaluation of models for the relationship between tree height and

diameter at breast height for Chinese-fir plantations in subtropical China," *PLoS One*, vol. 10, no. 4, Apr. 2015.

[21] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, and others, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, 1996, vol. 96, no. 34, pp. 226–231.

[22] P. H. S. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry," *Comput. Vis. Image Underst.*, vol. 78, no. 1, pp. 138–156, Apr. 2000.

[23] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proc. Am. Math. Soc.*, vol. 7, no. 1, p. 48, Jan. 1956.

[24] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," No. LA-UR-08-05495; LA-UR-08-5495. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[25] A. F. Colaço, R. G. Trevisan, J. P. Molin, J. R. Rosell-Polo, and A. Escolà, "Orange tree canopy volume estimation by manual and LiDAR-based methods," *Adv. Anim. Biosci.*, vol. 8, no. 2, pp. 477–480, Jun. 2017.

[26] "Haselnuss | Holzwurm-page, Holz mit Know How." Accessed on: Mar. 31, 2020 [Online]. Available: http://www.holzwurm-page.de/holzarten/holzart/haselnuss.htm.

[27] S. Kim and H. Kim, "A new metric of absolute percentage error for intermittent demand forecasts," *Int. J. Forecast.*, vol. 32, no. 3, pp. 669–679, Jul. 2016.

[28] R. G. Congalton, R. G. Oderwald, and R. A. Mead, "Assessing Landsat Classification Accuracy Using Discrete Multivariate Analysis Statistical Techniques," *Photogrammetric engineering and remote sensing*, vol. 49, no. 12, pp.1671-1678, 1983.

[29] T. Zhao, Y. Yang, H. Niu, D. Wang, and Y. Chen, "Comparing U-Net convolutional network with mask R-CNN in the performances of pomegranate tree canopy segmentation," in *Multispectral, Hyperspectral, and Ultraspectral Remote Sensing Technology, Techniques and Applications VII*, vol. 10780, p. 107801J, 2018.

[30] A. Dhall, K. Chelani, V. Radhakrishnan, and K. M. Krishna, "LiDAR-Camera Calibration using 3D-3D Point correspondences," *arXiv*, May 2017.

[31] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, 1992.

[32] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv*, Apr. 2018.

[33] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[34] M. Kazhdan and H. Hoppe, "Screened poisson surface reconstruction," *ACM Trans. Graph.*, vol. 32, no. 3, pp. 1–13, Jun. 2013.

[35] B. Mirtich, "Fast and Accurate Computation of Polyhedral Mass Properties," *J. Graph. Tools*, vol. 1, no. 2, pp. 31–50, Jan. 1996.

[36] M. Berger *et al.*, "A Survey of Surface Reconstruction from Point Clouds," *Comput. Graph. Forum*, vol. 36, no. 1, pp. 301–329, Mar. 2016.

[37] Q. Zhou, E. Grinspun, D. Zorin, and A. Jacobson, "Mesh arrangements for solid geometry," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 1–15, Jul. 2016.

[38] A. Jacobson *et al.*, "libigl: A simple (C++) geometry processing library." Mar. 2020.

[39] W. E. Lorensen and H. E. Cline, “Marching cubes: A high resolution 3D surface construction algorithm,” *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, Aug. 1987.

[40] P. Cignoni and F. Ganovelli, “VCG Library: The VCG Library.” Mar-2020.

[41] S. W. Hughes, “Archimedes revisited: a faster, better, cheaper method of accurately measuring the volume of small objects,” *Phys. Educ.*, vol. 40, no. 5, pp. 468–474, Sep. 2005.