Assigning apples to individual trees in dense orchards using 3D colour point clouds

Mouad Zine-El-Abidine, Helin Dutagaci, Gilles Galopin, David Rousseau

Assigning Apples to Individual Trees in Dense Orchards using 3D Colour Point Clouds

Mouad Zine-El-Abidine[a], Helin Dutagaci[b], Gilles Galopin[c], David Rousseau[a,c,*]

[a] *Laboratoire Angevin de Recherche en Ingénierie des Systèmes (LARIS), Université d'Angers, France*
[b] *Eskisehir Osmangazi University, Department of Electrical-Electronics Engineering, Eskisehir, Turkey*
[c] *INRAe, UMR1345 Institut de Recherche en Horticulture et Semences, Angers, France*

Abstract

We propose a 3D colour point cloud processing pipeline to count apples on individual apple trees in trellis structured orchards. Fruit counting at the tree level requires separating trees, which is challenging in dense orchards. We employ point clouds acquired from the leaf-off orchard in winter period, where the branch structure is visible, to delineate tree crowns. We localise apples in point clouds acquired in harvest period. Alignment of the two point clouds enables mapping apple locations to the delineated winter cloud and assigning each apple to its bearing tree. Our apple assignment method achieves an accuracy rate higher than 95%. In addition to presenting a first proof of feasibility, we also provide suggestions for further improvement on our apple assignment pipeline.

Keywords: Fruit detection, Apple detection, Apple trees, Tree trunk detection, Point Cloud, Semantic segmentation, Phenotyping

Nomenclature

$(\Delta_x, \Delta_y, \Delta_z)$ Edge lengths of the voxels for voxelization of a point cloud

$(\hat{x}, \hat{y}, \hat{z})$ Coordinates of a 3D point $\hat{p}$

(A, B, C, D) Parameters of the trellis-plane

*Corresponding author
Email address: david.rousseau@univ-angers.fr (David Rousseau)

(I_s, J_s)	Location of the s^{th} detected peak in IG
(N_x, N_y, N_z)	Size of B and S
(x, y, z)	Coordinates of a 3D point p
$(x_{r,1}, y_{r,1}, z_{r,1})$	Coordinates of the point $p_{r,1}$
$(x_{r,2}, y_{r,2}, z_{r,2})$	Coordinates of the point $p_{r,2}$
Γ	Set of semantic labels
γ_i	Predicted semantic label of the i^{th} point p_i
γ_i^{GT}	Ground truth semantic label of the i^{th} point p_i
$\hat{L}_j = (0, \hat{y}_j, 0)$	Location of verified trunk j
$\hat{L}_s = (0, \hat{y}_s^{ct}, 0)$	Location of candidate trunk s
$\hat{p}$	A 3D point in PC_w^{TP}
$\hat{p}_e$	e^{th} end-point of C_c
$\hat{p}_{q,j}$	Intersection point of q^{th} trellis-line and j^{th} trunk
$\hat{p}_{s,bottom}$	Bottom point of SK_s along the Z-axis
$\hat{p}_{s,top}$	Top point of SK_s along the Z-axis
$\hat{y}_s^{ct}$	y coordinate of the s^{th} candidate trunk location in $\mathcal{CT}$
$\hat{z}_q$	Height of tl_q
$\mathcal{A}^{\mathcal{GT}}$	Set of ground truth apples
$\mathcal{A}$	Set of detected apples
$\mathcal{CC}$	Set of connected components of S_{trees}
$\mathcal{CT}$	Set of candidate trunk locations
$\mathcal{IG}_{i,j}$	Set of points in PC_{ts} projected to the ground in the grid (i, j)
$\mathcal{L}_{HL}$	Set of 3D horizontal lines detected through applying Hough Transform to IH
$\mathcal{L}_{TL}$	Set of trellis-lines
$\mathcal{T}$	Set of located trees in the scene
τ_a	Predicted tree identity of the a^{th} apple in $\mathcal{A}$
τ_c	Predicted tree identity of the c^{th} connected component C_c
τ_g	Ground truth tree identity of the g^{th} apple in $\mathcal{A}^{GT}$
τ_i	Predicted tree identity of the i^{th} point p_i
τ_i^{GT}	Ground truth tree identity of the i^{th} point p_i
$\{(C_f, \tau_f)\}$	Connected components already assigned to a tree
$\{C_{c,d}\}$	Set of connected components of C_c after cutpoints are removed
ACC	Accuracy of apple assignment to trees
B	Binary 3D volumetric form of PC_w^C
B_s	Binary 3D volumetric form of PC_s^{CT}
B_{trees}	Binary 3D volumetric form of PC_w^{trees}
C_c	c^{th} connected component in $\mathcal{CC}$
CA	Class Accuracy
CP	Connecting path between adjacent trees
$d(p, hl_r)$	Distance between point p and the line hl_r
d_R^{cc}	The minimum distance of the ColorChecker tripod stick to the tree row
d_s^{SP}	Length of main axis SP_s
d_T^{cc}	The distance of the ColorChecker tripod stick to a designated tree
$F1$	F1 score
FN	Number of false negatives
FP	Number of false positives
hl_r	r^{th} horizontal line in $\mathcal{L}_{HL}$

IG	Histogram of points in PC_{ts} projected to the ground
IH	Binary image resulting from projecting S to the YZ-plane
IoU	Intersection over Union
l_e	Line fitted to the points around p_e
L_j	Location of j^{th} tree in the scene corresponding to (x_j, y_j, z_j) coordinates of the base of the tree
$ls_{j,j+1}$	The line defined by the points $\hat{p}_{q,j}$ and $\hat{p}_{q,j+1}$
N_e	Number of end-points of C_c
N_F	Number of connected components already assigned to a tree
N_P	Number of detected peaks in IG and number of canditate trunk locations in $\mathcal{CT}$
N_W	Number of points in PC_w^C
N_{apples}	Number of apples in $\mathcal{A}$
N_{apples}^{GT}	Number of apples in $\mathcal{A}^{\mathcal{GT}}$
N_{comp}	Number of connected components in $\mathcal{CC}$
N_{comp}^c	Number of connected components of C_c after cutpoints are removed
N_c	Number of trees spanned by a connected component C_c
N_{HL}	Number of horizontal lines in $\mathcal{L}_{HL}$
N_{TL}	Number of trellis-lines in $\mathcal{L}_{TL}$
n_{TP}	Unit normal of the trellis-plane
N_{trees}	Number of verified trees in the scene
N_{trees}^{GT}	Number of ground truth trees
p	A 3D point
p_a^{α}	Position of the a^{th} apple in $\mathcal{A}$
$p_g^{\alpha,GT}$	Position of the g^{th} ground truth apple in $\mathcal{A}^{\mathcal{GT}}$
p_i	i^{th} point in PC_w^C
$p_{r,1}$	A point on hl_r
$p_{r,2}$	A point on hl_r
PC	A 3D colour point cloud
PC_h	Reconstructed harvest point cloud before calibration
PC_h^C	Calibrated harvest point cloud
PC_s^{CT}	Subset of PC_w^{TP}; set of points within a cylindrical region around the s^{th} candidate trunk location
PC_w	Reconstructed winter point cloud before calibration
PC_w^C	Calibrated winter point cloud
PC_w^{TP}	Winter point cloud aligned to the trellis-plane
$PC_{j,j+1}^q$	Cylindrical region around the q^{th} trellis-line between the points $\hat{p}_{q,j}$ and $\hat{p}_{q,j+1}$
PC_{tr}	Subset of PC_w^C; set of points within 1cm distance to the horizontal lines on the trellis-plane
PC_{ts}	Subset of PC_w^{TP}; set of points within 5cm distance to the trellis-plane
PC_{wh}^C	Winter point cloud aligned to PC_h^C
PC_w^{trees}	Subset of PC_w^{TP}; set of points with trellis wires and support pole removed
Pr	Precision
R	Rotation matrix to transform PC_w^C to PC_w^{TP}
R^{wh}	Rotation matrix to align PC_w^C to PC_h^C
Re	Recall
S	Skeleton of B
S_s	Skeleton of B_s
S_{trees}	Skeleton of B_{trees}
SK_s	Set of points on the skeleton of s^{th} trunk candidate

SP_j	Set of points on the main axis of j^{th} verified trunk
SP_s	Set of points on the main axis of s^{th} trunk candidate
T^{wh}	Translation vector to align PC_w^C to PC_h^C
T_j	j^{th} tree in $\mathcal{T}$
t_j	Identity of j^{th} tree in $\mathcal{T}$
tl_q	q^{th} trellis-line in $\mathcal{L}_{TL}$
TN	Number of true negatives
TP	Number of true positives
TP_C	Number of true positive apples correctly assigned to respective trees
u_X	X-axis of the reference frame aligned to the trellis-plane
u_Y	Y-axis of the reference frame aligned to the trellis-plane
u_Z	Z-axis of the reference frame aligned to the trellis-plane
x_{max}	Maximum of the x coordinates of the points in a point cloud
x_{min}	Minimum of the x coordinates of the points in a point cloud
y_{max}	Maximum of the y coordinates of the points in a point cloud
y_{min}	Minimum of the y coordinates of the points in a point cloud
z_{max}	Maximum of the z coordinates of the points in a point cloud
z_{min}	Minimum of the z coordinates of the points in a point cloud
HSV	Hue, Saturation, and Value components of the colour of a 3D point
RGB	Red, Green, and Blue channels of the colour of a 3D point

1. Introduction

Apple yield is an important trait for both orchard management and variety testing of apple trees. Manual fruit counting is usually conducted by sampling a fixed percentage (e.g. 5 or 10%) of trees randomly or systematically and extrapolating the counts on these trees for total yield estimation of the entire orchard (Wulfsohn et al., 2012). This sampling and extrapolation process, in addition to being time-consuming and labor-intensive, does not always produce the desired precision of yield estimation. Computer vision techniques, on the other hand, provide a faster and more accurate alternative to manual counting of fruits (Gongal et al., 2015).

While the majority of computer vision techniques for fruit counting relied on RGB (Red, Green, Blue) images, other types of data including RGB-Depth images (Gené-Mola et al., 2019b; Nguyen et al., 2016; Tao & Zhou, 2017; Tu et al., 2018; Lin et al., 2019; Fu et al., 2020), spectral images (Safren et al.,

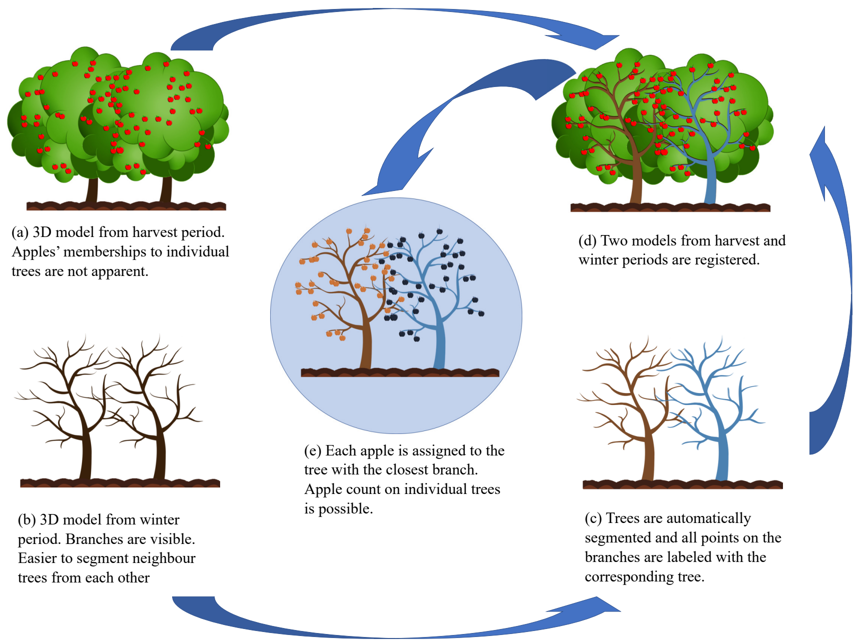


Figure 1: Apple detection algorithms usually estimate the cumulative apple count from the harvest season. Our aim is to count the number of apples on each individual tree. The main idea is to register the 3D model from the harvest period (a) with the delineated 3D model from the winter period (c) to align the branches with the detected apples (d). We assign a different label to each delineated tree as an output of the automatic tree separation algorithm we perform on the winter model (c). Finally the detected apples from the harvest model are mapped to their closest branches, and membership of each apple to an individual tree is determined (e).

2007), thermal images (Stajnko et al., 2004; Bulanon et al., 2008, 2009; Wachs et al., 2010; Gan et al., 2020) images or LiDAR (Light Detection and Ranging) data (Gené-Mola et al., 2019a) have also been used. In traditional approaches for fruit detection through such sensor information, relevant information is extracted from each data instance separately according to a manually predefined algorithm. The representative quantitative information obtained in this manner is generally referred to as a hand-crafted feature. Hand-crafted approaches can involve techniques such as colour thresholding, colour space clustering, shape analysis, blob detection, circular Hough transform, Ncut algorithm, employment of Histogram of Oriented Gradients (HOG), Local Binary Patterns (LBP) and Upright Speeded Up Robust Features (U-SURF) for separating fruits from the canopy (Wang et al., 2012; Sengupta & Lee, 2014; Sabzi et al., 2018; Tao & Zhou, 2017; Gongal et al., 2016; Nguyen et al., 2016; Roy & Isler, 2016; Bargoti & Underwood, 2017; Samiei et al., 2020; Sun et al., 2019; Gong et al., 2013; Lu et al., 2018; He et al., 2020; Kelman & Linker, 2014; Linker, 2018; Wu et al., 2019). Recently, deep learning methods have become commonplace for fruit detection and counting (Apolo-Apolo et al., 2020; Bargoti & Underwood, 2017; Bresilla et al., 2019; Chen et al., 2017; Häni et al., 2018; Häni et al., 2020; Tian et al., 2019; Gené-Mola et al., 2019b; Liu et al., 2018; Tu et al., 2018; Williams et al., 2019; Fu et al., 2020; Gan et al., 2020; Xiong et al., 2020). Deep neural networks are employed to learn predictors from a set of training data through optimising the parameters of feature extraction and localisation of fruits simultaneously. After prediction, further processing, such as circular Hough transform and watershed transform (Bargoti & Underwood, 2017) for verification and filtering of multiple counts through 3D (3-Dimensional) reconstruction (Gené-Mola et al., 2020; Gongal et al., 2016; Häni et al., 2020) can be applied to extract the final fruit count.

The main objective of most fruit counting methods is to estimate the total number of observable fruits in the sensed data (Gené-Mola et al., 2020; Gongal et al., 2016; Häni et al., 2018; Häni et al., 2020; Liu et al., 2018; Bargoti & Underwood, 2017; Bargoti & Underwood, 2017; Bresilla et al., 2019; Chen et al., 2017; Fu et al., 2020). The fruits are not mapped to their bearing trees; i.e. the number of fruits on each tree is not computed. Examples to applications that will benefit from fruit counting on individual trees are precise yield mapping at tree scale, management of individual trees to maximise uniformity within the orchard, and individual tree-based analysis in variety testing experiments.

Estimation of fruit count on each tree requires separating individual trees and identifying which tree each detected fruit belongs to (*tree membership of the fruit*). Individual tree delineation is the process of separating individual trees, including trunk detection and crown boundary delineation; i.e. identifying the trunk and branches belonging to a single tree (Zhen et al., 2016). Delineation of trees in dense orchards or forests is a challenging task due to interlacing and touching branches of adjacent trees, particularly when there is high variation among the trees in terms of crown size and shape (Zhen et al., 2016). Occlusion caused by dense leaf cover during harvest period further complicates the delineation of trees. Using leaf-off data collected during winter can alleviate the occlusion and facilitate the capture of trunk and branch geometry (Brandtberg et al., 2003; Lu et al., 2014).

The architectural structure that determines the connectivity of the branches to a particular tree trunk becomes ambiguous in 2D images, even during winter period. 2D projection causes loss of shape and connectivity information of the branches of neighbouring trees. Processing 3D point clouds is more adequate for our application since 3D data enables a detailed analysis of the geometric structure of trees and localisation of branches and fruits in the 3D world.

Furthermore, acquiring 3D information of the trees in the orchards facilitates a number of applications in precision agriculture, robotic agriculture, and phenotyping. These applications include robotic crop harvesting (Barnea et al., 2016; Ge et al., 2020; Lin et al., 2019; Williams et al., 2019), automated pruning (Medeiros et al., 2017; He & Schupp, 2018), monitoring pruning operations (Méndez et al., 2016), and 3D visualisation tools to guide the agronomists (Yandún Narváez et al., 2016). Accurate measurements of morphological traits such as canopy volume, branch dimensions and leaf area from already available 3D models are essential for phenotyping experiments and productivity assessment (Rosell & Sanz, 2012; Coupel-Ledru et al., 2019; Tabb & Medeiros, 2017).

Computer vision techniques aiding management of fruit orchards range from complete processing pipelines to algorithms performing single tasks such as tree localisation (Tabb & Medeiros, 2017; Colmenero-Martinez et al., 2018; Medeiros et al., 2017; Zhang et al., 2017; Zeng et al., 2020; Nielsen et al., 2012; Underwood et al., 2015; Zhong et al., 2016; Bargoti et al., 2015). A vision system was developed by (Tabb & Medeiros, 2017) to reconstruct 3D fruit trees and identify branch structure and traits for automatic pruning. In (Colmenero-Martinez et al., 2018) an automatic trunk-detection system using an infrared sensor was introduced. Medeiros et al. (Medeiros et al., 2017) employed a laser sensor to model dormant fruit trees and identify primary branches for automatic pruning. In (Zhang et al., 2017) Regions-Convolutional Neural Network (R-CNN) was applied on depth images for detection of branches of apple trees and localisation of shaking points to guide a harvesting machine. Zeng et al. (Zeng et al., 2020) developed an algorithm to segment trellis wires, support poles, and tree trunks in sparse LiDAR point clouds acquired from trellis-structured apple orchards. In order to optimise the mechanisation of fruitlet and blossom thinning, Nielsen et al. (Nielsen et al., 2012) used LiDAR and stereo vision together for obtaining 3D models of orchard rows of trees. They fitted mixtures of Gaussians to the point cloud to cluster the trees into Gaussian shaped cylinders. In (Underwood et al., 2015), LiDAR data was used for individual tree separation through a hidden semi-Markov model. Their objective was to develop a pipeline for building detailed orchard maps and an algorithm to match subsequent LiDAR tree scans to the prior database, enabling correct data association for precision agricultural applications. In (Zhong et al., 2016), a procedure for segmenting canopy to individual trees was proposed. The procedure involved octree construction, clustering, trunk detection and Ncut segmentation. 3D data was obtained with terrestrial laser scanning (TLS) and mobile laser scanning (MLS). In (Bargoti et al., 2015), a tree trunk detection pipeline was proposed for identifying individual trees in a trellis structured apple orchard, using ground-based LiDAR and image data. Hough transformation was performed on 3D point cloud to search for trunk candidates. These candidates were projected into the camera images, where pixel-wise classification was used to update their likelihood of being a tree trunk. Detection was achieved by using a hidden semi-Markov model to leverage from the contextual information provided by the repetitive structure of the orchard.

The objective of this work is to delineate apple trees in a trellis structured orchard and count the number of apples on each individual tree (Fig. 1). To the best of our knowledge, this problem was not addressed before in previous works dealing with apple detection and counting. Our strategy is to reconstruct 3D models of the same set of trees twice a year, once during the winter period and once during the harvest period. We perform delineation of individual trees on the leaf-off model from winter, which we refer to as *winter point cloud*. We detect tree trunks and identify the branches connected to them using winter point cloud. We employ the 3D model from the harvest period, which we call *harvest point cloud*, to localise apples. We determine the tree-membership of each apple in the harvest point cloud by mapping their locations onto the winter point cloud, where individual trees are separated. This approach of registering data from two different time instances for fruit counting is another novelty we introduce to the field. We also propose the use of a known calibration object to facilitate the registration of two point clouds and to recover the true metric sizes of the important structures in the scenes.

The main contributions of this study are:

- Addressing the problem of apple counting on individual trees from 3D colour point clouds.
- As a way to map detected apples to individual trees, alignment of harvest point cloud to the winter point cloud, where individual trees are automatically delineated.
- A complete pipeline for detecting and removing trellis wires and support poles, detecting tree trunks and delineating crowns of individual trees in winter point clouds.
- The use of a calibration object for correct scaling and alignment of point clouds acquired in different time instances.

2. Materials and Methods

We developed a point cloud processing pipeline (Fig. 2) in order to locate and count apples on individual trees. We use a colour camera for capturing images of target trees in the orchard from multiple views during both winter and harvest periods (Fig. 2 (a)). These images are processed by a structure from motion algorithm to reconstruct winter and point clouds. The two point clouds are prepared for initial alignment which we refer to as *calibration of point clouds* (Fig. 2 (b)). A novelty of our pipeline is the use of a ColorChecker during acquisition. The ColorChecker serves both as a reference for removal of irrelevant background information and as a calibration tool. Calibration of the point cloud, in our case, involves 1) re-scaling the point cloud to the correct metric scale, 2) orienting the point cloud to a canonical reference frame, 3) extraction of region of interest, and 4) re-centering the point cloud to a predetermined position. The estimated scale allows us to impose metric parameters on the pipeline such as range of separation between trees, separation between trellis wires, diameter of trellis wires, diameter of tree trunks, the expected pole diameter and height, etc. The orientation and re-centering facilitate trellis wire removal, tree trunk detection, and delineation of tree crowns (Fig. 2 (d)). The calibration of both harvest and winter point clouds is also crucial for their correct registration (Fig. 2 (c)). We employ a colour-based apple detection algorithm to locate the apples in the harvest point cloud (Fig. 2 (e)). Finally, we map the detected apples onto the winter cloud via distance calculation to assign them to their bearing trees. We give detailed explanations of each module of our pipeline in the following subsections.

2.1. Experimental Field

The experiments were conducted in a dense apple orchard, dedicated to variety testing at INRAe-Angers (latitude: 47.48226°N, longitude: 0.6152°E) in France. The orchard was composed of 4 years old apple trees organised in I-trellis structure with support poles. Our target trees were arranged in a row, where each tree was a mutant, being tested to be established as a new apple variety. The spacing between trees was 1m in average and the height of the trees ranged from 1 to 3m. The variation of the crown shape among the trees was high.

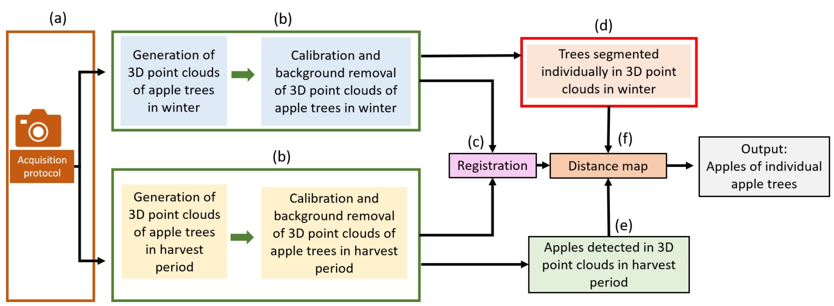


Figure 2: Pipeline proposed to assign apples to individual trees. (a) Image acquisition of apple trees in winter and harvest period. (b) Calibration of 3D models and extraction of region of interest. (c) Registration of calibrated models from winter and harvest period. (d) Separation of individual trees in winter point cloud. (e) Apple detection from harvest point cloud. (f) Distance map to assign apples to individual segmented trees.

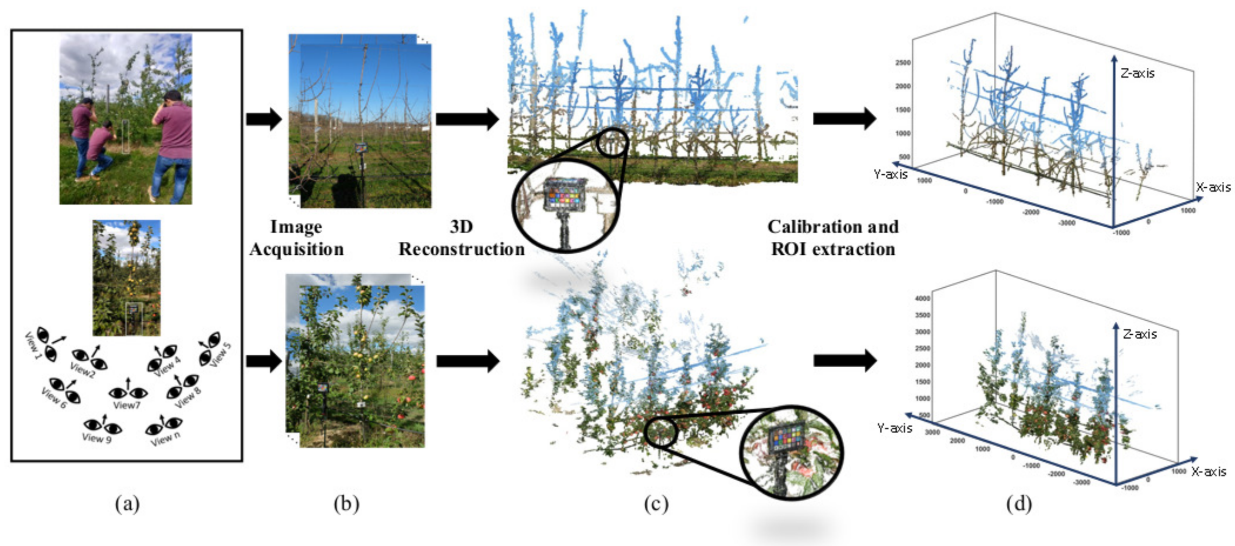


Figure 3: Data acquisition and point cloud calibration modules corresponding to (a) and (b) in Fig. 2. (a) Multi-view image acquisition. (b) Apple orchard images acquired in winter and harvest periods. (c) 3D colour point cloud reconstructions (PC_w and PC_h) of orchard scenes with zoom on the ColorChecker. (d) 3D colour point clouds after calibration and extraction of region of interest (PC_w^C and PC_h^C). See Supplementary Material A for details of the calibration process.

Table 1: Number of trees in the scenes and number of images acquired in winter and harvest periods.

	# trees	# images (winter)	# images (harvest)
Scene 1	5	236	364
Scene 2	5	189	382
Scene 3	5	221	380
Scene 4	4	183	374
Scene 5	5	206	380
Scene 6	4	199	376
Scene 7	4	227	376

2.2. Data acquisition and 3D reconstruction

Fig. 3 illustrates the data acquisition and point cloud calibration processes of our pipeline, corresponding to the modules (a) and (b) in Fig. 2. We obtained 3D colour point clouds of seven scenes from the orchard through a multi-view reconstruction process. A *scene*, in our study, refers to part of an orchard row; i.e. a set of adjacent trees in the same row. Each scene contained 4 to 5 apple trees in our experiments, although our algorithm is capable of processing an entire orchard row. The number of trees in each scene is given in Table 1.

A 3D colour point cloud (or a 3D RGB point cloud) PC is a set of 3D points, where each point is represented by its coordinates (x, y, z) and its colour (R, G, B). Here, (R, G, B) refers to the values of red, green and blue channels.

We captured multiple RGB images of size 3000×4000 pixels, of a scene with a colour camera (Fujifilm X20, Fujifilm Corporation, Tokyo, Japan) in both winter and harvest periods to reconstruct the point clouds. We acquired images from only one side of the orchard row; although it is possible to follow the procedure proposed in (Roy et al., 2018) to reconstruct and register two sides of a row. Table 1 lists the number of images used for 3D reconstruction of the scenes from winter and harvest periods. In this study, we captured multiple images from the scene manually, choosing the viewpoints and viewing angles (i.e. camera positions and orientations) to get visual information covering the scene from top to bottom and from various sides of the trees. It is important to guarantee that there is enough overlap between pairs of images for a successful 3D reconstruction. This process can be automated in a more systematic manner with path planning, using a drone (Scher et al., 2019) or a land robot equipped with multiple cameras (Tabb & Medeiros, 2017).

The multi-view images were used to reconstruct 3D colour point clouds of the scenes through VisualSFM (Wu, 2013; Wu et al., 2011) and PMVS/CMVS tool (Furukawa et al., 2010; Furukawa & Ponce, 2010). VisualSFM is a freely available software (Wu, 2013; Wu et al., 2011) that performs Structure from Motion (SfM) to estimate unknown camera locations and orientations. It provides a sparse point cloud of the scene through keypoint matching and triangulation. In order to obtain a dense point cloud, we used PMVS/CMVS tool, another freely-available software (Furukawa et al., 2010; Furukawa & Ponce, 2010). This tool takes as input the images and the camera parameters computed by VisualSFM and provides a dense reconstruction of the scene through multi-view stereo. For introductory and in-depth information on the techniques of SfM and multi-view stereo, we refer the reader to the textbook of Hartley and Zisserman (Hartley & Zisserman, 2004).

Before capturing the images of each scene, we installed a calibration object (ColorChecker Passport Photo 2, X-rite, Great Lakes, Midwestern US) mounted on a tripod stick at a known position. We placed the tripod stick in front of the trees facing the camera, such that the ColorChecker pattern is almost parallel to the tree row Fig. 2 (b). When the ColorChecker stick was installed, we manually measured two distances with a tape measure: d_R^{cc}: the minimum distance of the tripod stick to the tree row, and d_T^{cc}: the distance to a designated target tree. These values are necessary for the calibration process of the point clouds.

The reconstructed harvest point cloud and winter point cloud of a scene are referred to as PC_h and PC_w respectively. Point clouds of a sample scene are given in Fig. 3 (c) with the ColorChecker objects zoomed in.

2.3. Calibration and Extraction of Region of Interest

The calibration of the point clouds from harvest and winter periods provides an initial alignment, which is fundamental for the success of the registration of the two point clouds. Having the point cloud with the accurate scale also enables us to fix parameters, such as trunk diameter, tree height, separation between trees, according to the range of expected metric sizes of the structures in the scene.

The ColorChecker is usually employed as a colour reference to obtain accurate colours from images under varying lighting conditions (Marrero Fernández et al., 2019). In this work, we do not use the ColorChecker for this purpose. Instead, we use it as a distinct reference pattern to geometrically calibrate the raw point clouds. We developed an algorithm for automatic detection of the ColorChecker, together with the tripod stick it is mounted on, from 3D colour point clouds. The description of this algorithm can be found in Supplementary Material A. The 3D locations of the centers of the colour patches of the ColorChecker chart are used to guide the calibration of the point cloud.

The geometric calibration process takes as input the harvest and winter point clouds (PC_h and PC_w) and produces the calibrated point clouds (PC_h^C and PC_w^C), as shown in Fig. 3 (d). The details of the calibration process are given in Supplementary Material A. In summary, the calibration process consists of 1) estimation of the true scale and re-scaling the point cloud; 2) re-defining a canonical reference frame and rotating the point cloud to this new frame; 3) extraction of region of interest, which corresponds to the set of trees just behind the ColorChecker; and 4) moving the origin of the reference frame to the base of the designated tree. The canonical reference frame is defined such that Y-axis is parallel to the tree row and Z-axis is orthogonal to the ground.

2.4. Separation of Individual Trees

In this section, we describe the procedure to separate the trees from each other in the winter scenes. This procedure involves localisation of target tree trunks, finding the points on the tree trunks, detecting and removing trellis wires, the water pipe, and the support poles. After the trees are localised and irrelevant points are removed, the tree membership of all the remaining points are determined.

Let the number of points in the calibrated winter point cloud $PC_w^C = \{p_1, p_2, ..., p_{N_W}\}$ be N_W. We aim to map each point p_i to a semantic label γ_i, $i = 1, 2, ..., N_W$ where $\gamma_i \in \Gamma$. Γ is the set of four semantic labels: $\Gamma = \{$ *"Tree trunk"*, *"Branch"*, *"Trellis wire+Water pipe"*, *"Support pole"*$\}$. The process of automatically labeling the points in the cloud with one of these four classes is called *semantic segmentation* of the scene. The rationale for a semantic segmentation stage is to remove irrelevant structures and to eliminate the connectivity between adjacent trees caused by trellis wires and the water pipe.

In conjunction with semantic segmentation, we also detect trees in the scene and locate their trunks. Let the set of verified trees in the scene be denoted as $\mathcal{T}$. Each tree T_j in $\mathcal{T}$ is represented by its tree identity $t_j \in \{1, 2, ..., N_{trees}\}$ and its location L_j, for $j = 1, 2, ..., N_{trees}$. The location of a tree corresponds to the coordinates of its base $L_j = (x_j, y_j, z_j)$, $j = 1, 2, ..., N_{trees}$ measured in the canonical reference frame.

After removing the irrelevant structures (trellis wires, water pipe and support pole) we delineate the trees in the winter point cloud. The final output of the tree separation algorithm is the assignment of each trunk and branch point in the calibrated winter point cloud PC_w^C to one of the trees in the set $\mathcal{T}$.

2.4.1. Detection of trellis wires and tree trunks

The procedure for detecting points on trellis wires is based-on estimation of the *trellis-plane* and the *trellis-lines* along the trellis wires and operating on the points close to these estimates. Candidate trunk locations are detected along the trellis-plane based on point density. The points in a cylindrical region along each candidate location is separately skeletonised. The skeleton and the points surrounding it are examined to verify tree trunk position and to detect the presence of a supporting pole. 3D points belonging to the trunk of each individual tree and support pole are identified and labeled. Regions between tree trunks along the initial line estimates are re-examined through 3D line fitting to increase the precision of the detection and removal of the points that belong to the trellis wires. The steps of the procedure are shown in Fig. 4 and detailed below:

Step 1: Voxelization The calibrated winter point cloud PC_w^C is converted to binary volumetric form, where a voxel takes the value 1 if the voxel is occupied by the points in PC_w^C. Specifically, we fit a regular 3D grid to the bounding box defined by the minimum and maximum coordinate values (x_{min}, x_{max}), (y_{min}, y_{max}), (z_{min}, z_{max}) of the points in PC_w^C. Each cell, i.e. voxel, of the grid has edge lengths of $\Delta_x = \Delta_y = \Delta_z = 5mm$. On this grid, we define a 3D array B of size $N_x \times N_y \times N_z$, where

$$
\begin{aligned}
N_x &= \lfloor \frac{x_{max} - x_{min}}{\Delta_x} \rfloor + 1; \\
N_y &= \lfloor \frac{y_{max} - y_{min}}{\Delta_y} \rfloor + 1; \\
N_z &= \lfloor \frac{z_{max} - z_{min}}{\Delta_z} \rfloor + 1.
\end{aligned} \tag{1}
$$

Here $\lfloor \cdot \rfloor$ is the floor function. The 3D volumetric form of the point cloud corresponds to the binary function B computed as

$$
B(k,l,m) = \begin{cases} 1, & \text{if } \exists p = (x,y,z) \in PC_w^C : \\ & \lfloor \frac{x - x_{min}}{\Delta_x} \rfloor = k \ \& \ \lfloor \frac{y - y_{min}}{\Delta_y} \rfloor = l \ \& \ \lfloor \frac{z - z_{min}}{\Delta_z} \rfloor = m \\ 0, & \text{otherwise,} \end{cases} \tag{2}
$$

for $k = 0, ..., N_x - 1$, $l = 0, ..., N_y - 1$, and $m = 0, ..., N_z - 1$. In Fig. 4 (Step 1), the volumetric model of a sample scene is visualised. In the figure only the voxels with value "1" are shown.

Step 2: Skeletonisation We extract the skeleton of the volumetric model B using medial axis thinning algorithm given in (Lee et al., 1994). Formally, the skeleton of a 3D object is the set of the centers of all inscribed maximal spheres where these spheres touch the object boundary at one than more point (Lee

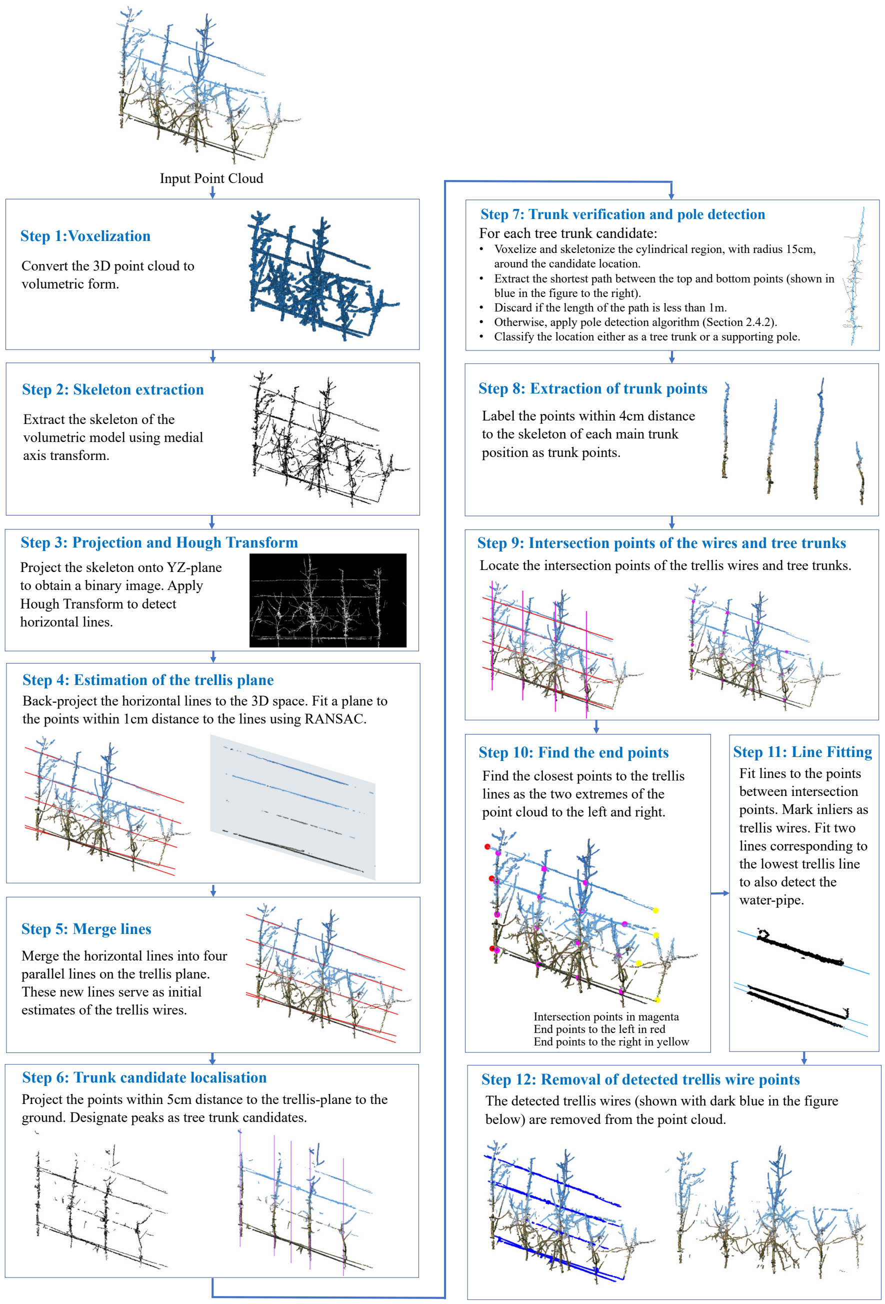


Figure 4: Block diagram for detection and removal of trellis wires and the water-pipe.

et al., 1994). The skeletonisation process produces another binary 3D grid S of size $N_x \times N_y \times N_z$, where the structures in B are pruned to curves with thickness of one voxel. In Fig. 4 (Step 2), the skeleton of a sample scene is shown.

Step 3: Projection and Hough Transform The skeleton defined in the binary 3D grid S is projected to the YZ-plane (parallel to the tree row) as a binary image, IH of size $N_y \times N_z$:

$$IH(l,m) = \begin{cases} 1, & \text{if } \sum_{k=0}^{N_x-1} S(k,l,m) > 0 \\ 0, & \text{otherwise}, \end{cases} \tag{3}$$

for $l = 0, ..., N_y - 1$, and $m = 0, ..., N_z - 1$.

In Fig. 4 (Step 3), the projected binary image of a sample scene is shown. We apply 2D Hough Transform (Duda & Hart, 1972) to IH to extract main horizontal lines in the binary image. The peaks greater than 20% of the maximum value in the Hough parameter space, and with angle with the horizontal axis less than 10° are selected as the main horizontal lines. These horizontal lines correspond to candidates for the trellis-lines in the scene.

Step 4: Estimation of the trellis-plane

The detected horizontal lines are back-projected to the 3D space of the point cloud PC_w^C, as shown with red lines in Fig. 4 (Step 4). Let the set of these horizontal 3D lines be $\mathcal{L}_{HL} = \{hl_1, hl_2, ..., hl_{N_{HL}}\}$, where N_{HL} is the number of horizontal lines. Each 3D line is defined by a pair of points on it, as $hl_r = (p_{r,1}, p_{r,2})$, with $p_{r,1} = (x_{r,1}, y_{r,1}, z_{r,1})$ and $p_{r,2} = (x_{r,2}, y_{r,2}, z_{r,2})$. We retrieve the points in PC_w^C with distance 1cm to these lines, and form the subset:

$$PC_{tr} = \{p = (x,y,z) \in PC_w^C : \min_{r=1,..,N_{HL}} d(p, hl_r) < 1cm\}. \tag{4}$$

The distance $d(p, hl_r)$ between a point p and the line hl_r is calculated as:

$$d(p, hl_r) = \frac{\|(p - p_{r,1}) \times (p - p_{r,2})\|}{\|p_{r,2} - p_{r,1}\|}, \tag{5}$$

where $\times$ is the cross product operation, and $\|\cdot\|$ is the Euclidean norm. We fit a plane to the points in PC_{tr} using M-estimator SAmple Consensus (MSAC) algorithm given in (Torr & Zisserman, 2000), which is a variant of RANdom SAmple Consensus (RANSAC) algorithm. Maximum distance for a point to be an inlier is set to be 0.5cm. The output of the algorithm is a plane model (A, B, C, D), where the parameters define the plane equation $Ax+By+Cz+D = 0$. The unit vector $n_{TP} = (A, B, C)$ corresponds to the normal of the plane. We refer to this plane as the *trellis-plane* on which trellis wires and tree trunks are located. Fig. 4 (Step 4) shows the trellis-plane fitted to the points in PC_{tr} for a sample scene.

The trellis-plane plays an important role in the following steps. We rotate the calibrated winter point cloud PC_w^C to a new reference frame such that the new YZ plane coincides with the trellis-plane and Y-axis is parallel to the trellis-lines. The new Y-axis is computed as the average of the direction vectors of the horizontal lines in $\mathcal{L}_{HL}$:

$$u_Y = \frac{\sum_{r=1}^{N_{HL}} (p_{r,2} - p_{r,1})}{\| \sum_{r=1}^{N_{HL}} (p_{r,2} - p_{r,1})\|} \tag{6}$$

The new Z-axis is orthogonal to the normal of the trellis-plane and the average direction of the trellis-lines:

$$u_Z = u_Y \times n_{TP}, \tag{7}$$

and the new X-axis is

$$u_X = u_Y \times u_Z \tag{8}$$

We transform each point $p = (x, y, z)$ in the calibrated winter cloud PC_w^C using the rotation matrix R defined in Eq. (9), and obtain a point cloud of the same size, PC_w^{TP}. We refer to this point cloud as the winter point cloud aligned to the trellis-plane.

$$PC_w^{TP} = \{\hat{p} = (\hat{x}, \hat{y}, \hat{z}) = pR : p \in PC_w^C\}; \quad R = \begin{bmatrix} u_X \\ u_Y \\ u_Z \end{bmatrix} \tag{9}$$

The origin of the new reference frame remains at the base of the target tree (see Supplementary Material A). With the transformation, the trellis-plane coincides with the $\hat{x} = 0$ plane in the new reference frame. This ensures that the $\hat{x}$ coordinate of each tree trunk is close to 0. Notice that this transformation is applied only to the winter point cloud. Once the semantic segmentation of the winter cloud is achieved and the trees are delineated, the points are transformed back to their original positions using $p = \hat{p}R^{-1}$.

Step 5: Merge lines The detected horizontal lines are on the trellis-plane; hence, they are located on the $\hat{x} = 0$ plane in the new reference frame. Their average direction is parallel to the Y-axis. Hence, we represent each line $hl_r \in \mathcal{L}_{HL}$ with the direction vector $(0, 1, 0)$ and a point on the line $(0, 0, \hat{z}_r)$. The value $\hat{z_r}$ indicates the height of a horizontal line on the trellis-plane and is calculated as:

$$\hat{p}_{r,1} = (\hat{x}_{r,1}, \hat{y}_{r,1}, \hat{z}_{r,1}) = p_{r,1}R; \quad \hat{p}_{r,2} = (\hat{x}_{r,2}, \hat{y}_{r,2}, \hat{z}_{r,2}) = p_{r,2}R; \tag{10}$$

$$\hat{z}_r = \frac{\hat{z}_{r_1} + \hat{z}_{r_2}}{2} \tag{11}$$

We merge the lines into parallel lines on the trellis-plane, each separated by at least 30cm to create the set of trellis-lines $\mathcal{L}_{TL} = \{tl_1, ..., tl_{N_{TL}}\}$. Each line is represented with the direction vector $(0, 1, 0)$ and a point on the line $(0, 0, \hat{z}_q)$, with $q = 1, ..., N_{TL}$. We use the following procedure to cluster the horizontal lines in $\mathcal{L}_{HL}$ into trellis-lines in $\mathcal{L}_{TL}$: We first sort the horizontal lines with ascending height. We start from the bottom line on the trellis-plane, and initialise $\hat{z}_1$ to the height of the first horizontal line. If the distance between the closest horizontal line is less than 30cm, we add the line to the group and update $\hat{z}_1$ to the average height of the group. Otherwise, we create a new group and proceed to the next line. In our experiments, the horizontal lines were grouped into 4 lines for all the winter scenes. Fig. 4 (Step 5) shows the resulting trellis-lines in red colour for a sample winter scene. In the rest of the paper we fix $N_{TL} = 4$. The four height values $\{\hat{z}_1, \hat{z}_2, \hat{z}_3, \hat{z}_4\}$ will be used to specify the locations of the trellis-lines.

Step 6: Trunk candidate localisation To localise candidate tree trunks along the trellis-plane we limit the search space within 5cm distance to the trellis-plane. We extract a subset of points PC_{ts} from PC_w^{TP}:

$$PC_{ts} = \{\hat{p} = (\hat{x}, \hat{y}, \hat{z}) \in PC_w^{TP} : |\hat{x}| < 5cm\} \tag{12}$$

Fig. 4 (Step 6) shows PC_{ts} of a sample scene. We define a regular 2D grid, IG on the $z = 0$ plane, which is parallel to the ground. Each cell of the grid has edge length $\hat{\Delta}_x = \hat{\Delta}_y = 1cm$. We compute the number of points in PC_{ts} falling into each cell:

$$\mathcal{IG}_{i,j} = \{\hat{p} = (\hat{x}, \hat{y}, \hat{z}) \in PC_{ts} : \lfloor \frac{\hat{x} - \hat{x}_{min}}{\hat{\Delta}_x} \rfloor = i \quad \& \quad \lfloor \frac{\hat{y} - \hat{y}_{min}}{\hat{\Delta}_y} \rfloor = j\}; \tag{13}$$

$$IG(i, j) = |\mathcal{IG}_{i,j}|, \tag{14}$$

where $\hat{x}_{min}$ and $\hat{y}_{min}$ are the minimum of the $\hat{x}$ and $\hat{y}$ coordinates of the points in PC_{ts}, and $|\mathcal{X}|$ denotes the number of elements in the set $\mathcal{X}$.

IG is the histogram of the points in PC_{ts} projected to the ground. The points on the tree trunks form the densest regions in the histogram correspond to the peaks of IG. The locations of the peaks are detected via non-maximum suppression (Gonzalez & Woods, 2006) as $\{(I_1, J_1), (I_2, J_2), ..., (I_{N_P}, J_{N_P})\}$, where N_P is the number of detected peaks. The set of candidate trunk locations in the 3D space are then defined as $\mathcal{CT} = \{(0, \hat{y}_1^{ct}, 0), (0, \hat{y}_2^{ct}, 0), ..., (0, \hat{y}_{N_P}^{ct}, 0)\}$; with $\hat{y}_1 < \hat{y}_1 < ... < \hat{y}_{N_P}^{ct}$. Recall that the trunks intersect with the trellis-plane. $\hat{y}_s^{ct}$ for $s = 1, ..., N_P$ is calculated as:

$$\hat{y}_s^{ct} = J_s \hat{\Delta}_y + \hat{y}_{min}. \tag{15}$$

Fig. 4 (Step 6) shows the locations of the candidate trunks as vertical purple lines passing through $(0, \hat{y}_s^{ct}, 0)$.

Step 7: Trunk verification

Not all the peaks detected in the previous step correspond to tree trunks. In this step, we examine the points at each candidate trunk location to verify whether it is a tree trunk, a support pole, or neither. We construct the set of trees $\mathcal{T}$ using the verified trunks. Each tree T_j in $\mathcal{T}$ is represented by its tree identity $t_j \in \{1, 2, ..., N_{trees}\}$ and the location of its base $\hat{L}_j = (\hat{x}_j, \hat{y}_j, \hat{z}_j)$, for $j = 1, 2, ..., T_{N_{trees}}$. The procedure for constructing the set of detected trees is given in Algorithm 1, and explained below:

Algorithm 1: Tree trunk verification

Data: PC_w^{TP}: The winter point cloud aligned to the trellis-plane;

$(0, \hat{y}_s^{ct}, 0)$: Candidate tree trunk locations for $s = 1, ..., N_P$

Result: $\mathcal{T} = \{T_1, ..., T_{N_{trees}}\}$: Set of detected trees;

N_{trees}: Number of detected trees;

t_j: Tree identity of $T_j \in \mathcal{T}$;

$\hat{L}_j = (\hat{x}_j, \hat{y}_j, \hat{z}_j)$: Location of $T_j \in \mathcal{T}$;

SP_j: Set of points on the main axis of T_j

Initialise $\mathcal{T} = \emptyset$; $N_{trees} = 0$; $j = 0$;

for $s \leftarrow 1$ *to* N_P **do**

  Extract the point set PC_s^{CT} using Eq. (16);

  Convert PC_s^{CT} to binary volumetric form B_s through voxelization;

  Compute the skeleton S_s of B_s using medial axis thinning (Lee et al., 1994);

  Obtain SK_s by retrieving the 3D points on the skeleton S_s ;

  Find the top and bottom points in SK_s with the largest and smallest z-coordinates and designate them as $\hat{p}_{s,top}$ and $\hat{p}_{s,bottom}$;

  Extract the shortest path between $\hat{p}_{s,top}$ and $\hat{p}_{s,bottom}$ using Breadth-first search (Cormen et al., 2009) ;

  Collect the points on the shortest path to form the main axis SP_s;

  Calculate the length d_s^{SP} of SP_s;

  if $d_s^{SP} > 1m$ **then**

    Run Support Pole Detection Algorithm on s (Section 2.4.2) ;

    if *s is not a Support Pole* **then**

      $j \leftarrow j + 1$; $N_{trees} \leftarrow N_{trees} + 1$; $t_j = j$;

      $\hat{L}_j = (0, \hat{y}_s^{ct}, 0)$;

      $SP_j = SP_s$;

      $T_j = (t_j, \hat{L}_j, SP_j)$;

      $\mathcal{T} \leftarrow \mathcal{T} \cup T_j$

We first initialise the set of trees as $\mathcal{T} = \emptyset$ and the number of tree trunks as $N_{trees} = 0$. For each candidate trunk indexed with s, we define a cylindrical region, with radius 15 cm, centered at the candidate trunk location $(0, \hat{y}_s^{ct}, 0)$, along the trellis-plane. We extract the points inside this region from PC_w^{TP}:

$$PC_s^{CT} = \{\hat{p} = (\hat{x}, \hat{y}, \hat{z}) \in PC_w^{TP} : \sqrt{\hat{x}^2 + (\hat{y}^2 - \hat{y}_s^{ct})^2} < 15cm\} \tag{16}$$

The point cloud PC_s^{CT} is converted to binary volumetric form B_s with voxel size $\hat{\Delta}_x = \hat{\Delta}_y = \hat{\Delta}_z = 5mm$. Then, the skeleton S_s is extracted from B_s using medial axis thinning algorithm given in (Lee et al., 1994). The points on the skeleton are retrieved from the point cloud PC_s^{CT}, and denoted as SK_s.

The two top and bottom points of the set SK_s along the Z-axis $\hat{p}_{s,top}$ and $\hat{p}_{s,bottom}$ are retrieved. The points on the shortest path between these two points is computed using the Breadth-first search algorithm described in (Cormen et al., 2009). We refer to the set of the points on the shortest path as the *main axis* of the s^{th} trunk, and denote it as SP_s. Fig. 4 (Step 7) shows the skeleton with black dots and the points on the shortest path with blue dots for a candidate trunk location.

If the length of the shortest path d_s^{SP} is less than 1m, then the candidate trunk location is discarded. Otherwise, it is passed to the support pole detection procedure described in Section 2.4.2. If it is not identified as a support pole, then we update $N_{trees} \leftarrow N_{trees} + 1$, and insert the verified trunk into $\mathcal{T}$. We also store the main axis of the verified trunk. See Algorithm 1 for the formation of the set $\mathcal{T}$.

Step 8: Extraction of trunk points The previous step gives the attributes of each tree $T_j = (t_j, \hat{L}_j, SP_j) \in \mathcal{T}$. The main axis of the j^{th} detected tree is represented by the set of points SP_j. We label a point $\hat{p}_i$ in the point cloud PC_w^{TP} as *"Tree trunk"* if its distance to the main axis of one of the trees is less than 3cm. Specifically:

$$\gamma_i = \textit{"Tree trunk"} \quad \text{if} \quad \min_j \min_{\hat{p} \in SP_j} \|\hat{p} - \hat{p}_i\|^2 < 3cm \tag{17}$$

Fig. 4 (Step 8) shows the points semantically labeled as *"Tree trunk"* in a winter scene.

Step 9: Locating the intersection points of trellis wires and tree trunks In Step 4, the set of trellis-lines $\mathcal{L}_{TL} = \{tl_1, tl_2, tl_3, tl_4\}$ is determined. Recall that each line is represented with the direction vector $(0, 1, 0)$ and a point on the line $(0, 0, \hat{z}_q)$, with $\hat{z}_1 < \hat{z}_2 < \hat{z}_3 < \hat{z}_4$. Now, having located the trunks at $\hat{L}_j = (0, \hat{y}_j, 0)$ with $\hat{y}_1 < \hat{y}_2 < ... < \hat{y}_{N_{trees}}$, we find the points where the trellis wires intersect with the trunk locations. For a trellis-line with index q and a trunk location with index j, we find the point $\hat{p}_{q,j} \in PC_w^{TP}$ closest to the location $(0, \hat{y}_j, \hat{z}_q)$. Fig. 4 (Step 9) shows the trellis-lines, located tree trunks and the intersection points $\hat{p}_{q,j}$ for a winter scene.

Step 10: Finding the end points of trellis wires The end-points corresponding to the trellis wires in the scene are determined by finding the closest points to the trellis-lines at the two extremes of the point cloud along the Y-axis. Specifically, for a trellis-line with index q, we locate two points $\hat{p}_{q,0} \in PC_w^{TP}$ and $\hat{p}_{q,N_{trees}+1} \in PC_w^{TP}$, which are closest to the locations $(0, \hat{y}_{min}, z_q)$ and $(0, \hat{y}_{max}, z_q)$, respectively. Here, $\hat{y}_{min}$ and $\hat{y}_{max}$ are the minimum and maximum Y-coordinates of the points in PC_w^{TP}. Fig. 4 (Step 10) shows the end points for a winter scene with red and yellow dots.

Step 11: Line fitting to find the points on trellis wires and the water-pipe The region between each adjacent intersecting points of trellis-lines and the trunks are examined for a precise determination of the points on the trellis wires and the water-pipe. For each pair of intersecting points $\hat{p}_{q,j}$ and $\hat{p}_{q,j+1}$, $q = 1, ..., 4 \quad j = 0, 1, ..., N_{trees}$ we extract the points:

$$\begin{aligned} PC_{j,j+1}^q = &\{\hat{p} = (\hat{x}, \hat{y}, \hat{z}) \in PC_w^{TP} : \\ &(\hat{y}_j + 4cm < \hat{y} < \hat{y}_{j+1} - 4cm) \ \& \ (d(\hat{p}, ls_{j,j+1}) < 10cm)\} \end{aligned} \tag{18}$$

where $ls_{j,j+1}$ is the line defined by the points $\hat{p}_{q,j}$ and $\hat{p}_{q,j+1}$, and $d(\hat{p}, ls_{j,j+1})$ is the distance between point $\hat{p}$ and line $ls_{j,j+1}$. This region corresponds to a cylinder of radius 10cm with axis $ls_{j,j+1}$. We set an offset value of 4cm from the trunk locations not to include trunk points to the search region for trellis wire points.

Using MSAC algorithm given in (Torr & Zisserman, 2000), we fit two lines to the points in $PC^0_{j,j+1}$ corresponding to the regions along the lowest trellis-line (one for the trellis wire and one for the water-pipe). One line is fitted to the points for the rest of the regions $PC^q_{j,j+1}$ with $q = 2, 3, 4$. For a point to be an inlier, the maximum distance to the fitted line is set to be 7cm for $q = 0$ and 4cm for $q = 2, 3, 4$. Fig. 4 (Step 11) shows points in two regions along the trellis wires in blue and the lines fitted to them in black.

If a point $\hat{p}_i \in PC^{TP}_w$ is an inlier of one of the fitted lines we set its semantic label as $\gamma_i =$ *"Trellis wire + Water pipe"*.

Step 12: Removal of detected trellis wire points

The detected trellis wire points and the points on the support pole, if there is any, are removed from the point cloud to form the set:

$$
\begin{aligned}
PC^{trees}_w = &\{\hat{p}_i \in PC^{TP}_w : \\
&(\gamma_i \neq \textit{"Trellis wire + Water pipe"}) \ \& \ (\gamma_i \neq \textit{Support pole"})\}
\end{aligned}
\tag{19}
$$

The procedure for retrieving points on the support pole is given in Section 2.4.2. The point cloud PC^{trees}_w is supposed to include only the points on the trees. In Fig. 4 (Step 12) the points labeled as *"Trellis wire+Water pipe"* are shown in dark blue. Also the resulting PC^{trees}_w is given for a sample winter scene.

2.4.2. Detection of Support Poles

During the procedure for trellis wire detection and localisation of tree trunks, we examine each trunk candidate to determine whether it corresponds to a support pole or an actual tree trunk. We consider the points in a vertical cylindrical region of radius 15cm centered at the candidate trunk location. We partition the points into horizontal slices of height 2cm. We project the points in each slice onto the XY-plane (the ground plane) and fit a circle of radius 4.5cm (the actual radius of a support pole in the orchard) to the projected points, and estimate the center. The centers of the slices form the axis of the candidate support pole and the new cylindrical region. We count the points in the cylindrical shell with inner and outer radii, $4.5 - 0.5$ and $4.5 + 0.5$cm, and with height 2.3m (the actual height of a support pole). If the ratio of this number to the total number of points in the initial cylindrical region is higher than 0.8, then we declare that the structure corresponds to a support pole. We label the points in the cylindrical shell as pole points.

2.4.3. Identifying tree membership of points (Tree Separation)

This module of our pipeline is responsible for delineating the trees in PC_w^{trees}, which is the point cloud with trellis wires, the water-pipe and the support pole removed. The output of the delineation process is the assignment of each point in PC_w^{trees} to one of the trees $T_j = (t_j, \hat{L}_j, SP_j) \in \mathcal{T}$.

The main steps of the tree separation process is given in Fig. 5. We convert PC_w^{trees} to binary volumetric form and apply skeletonisation to conduct a connectivity analysis. We delineate adjacent trees if they are touching and we assign isolated connected components to one of the two nearest trees through a set of rules. The details of the steps are as follows:

Step 1: Voxelization The point cloud PC_w^{trees} is converted to binary volumetric form B_{trees} with voxel size $\hat{\Delta}_x = \hat{\Delta}_y = \hat{\Delta}_z = 5mm$.

Step 2: Skeletonisation The skeleton S_{trees} is extracted from B_{trees} using medial axis thinning algorithm given in (Lee et al., 1994).

Step 3: Extraction of connected components Connected components of the skeleton S_{trees} are extracted using flood fill algorithm (Torbert, 2016). We denote the set of connected components as $\mathcal{CC} = \{C_1, C_2, ..., C_{N_{comp}}\}$, where C_c is the c^{th} connected component and N_{comp} is the number of connected components. Fig. 5 shows each connected component in a sample S_{trees} in a different colour.

Step 4: Labeling connected components We compute the minimum distance of each connected component C_c to all trunk locations $\hat{L}_j = (0, \hat{y}_j, 0)$. If this distance is below 30cm, then we assign C_c to t_j. Fig. 5 (Step 4) gives the trunk locations as lines in different colours and the connected components coloured according to the assigned tree for a sample S_{trees}.

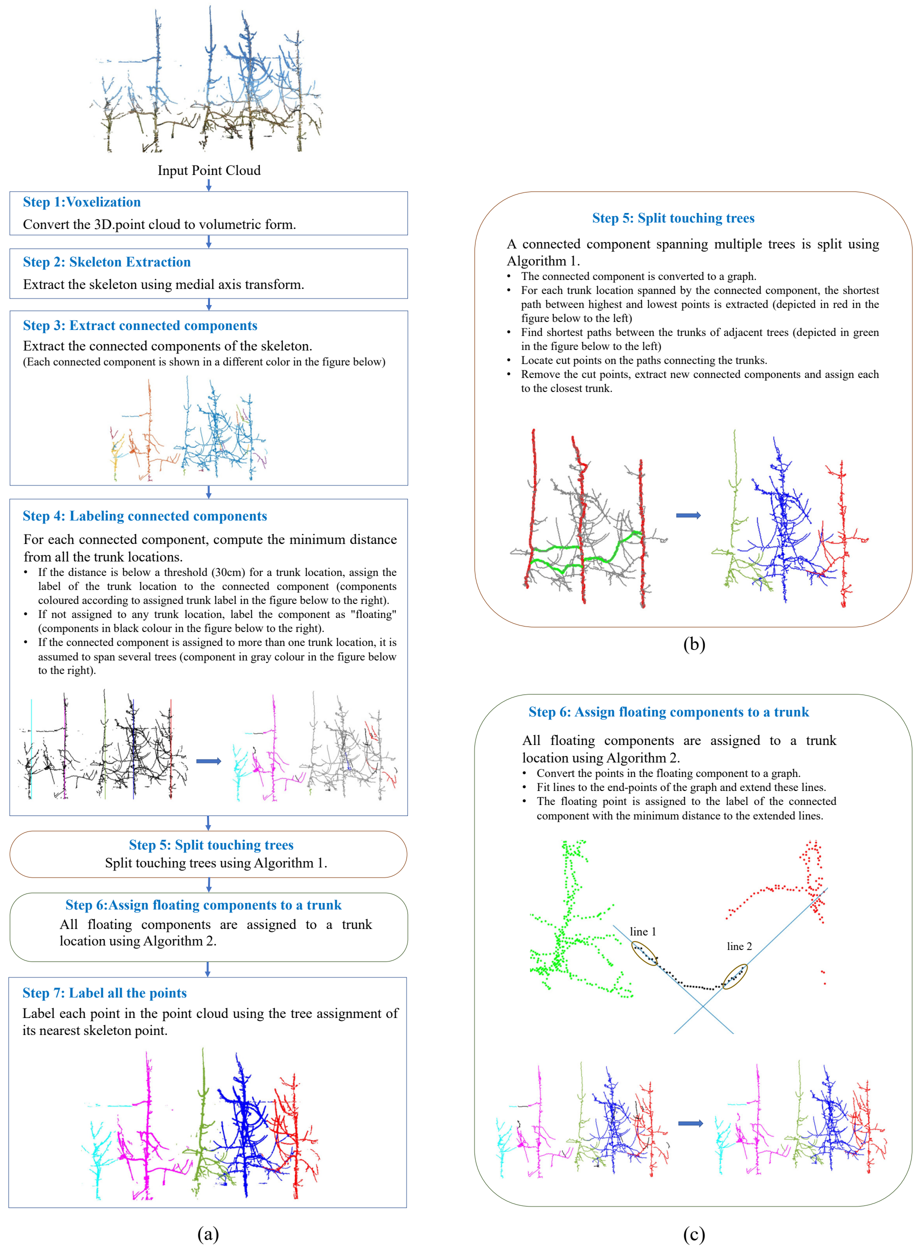


Figure 5: (a) Block diagram for separating individual trees. (b) Illustration of Step 5 for splitting touching trees, (c) Illustration of Step 6 for labeling floating components.

Algorithm 2: Separation of a connected component into multiple trees

Data: C_c: Connected component spanning multiple trees;

$\{T_j\}$: Trees spanned by C_c; $j = j_1^c, ..., j_{N_c}^c$;

$\{SP_j\}$: Main axes of trees;

$\{\hat{p}_{j,top}\}$: Top points of the main axes

Result: $\{C_{c,d}\}$: Detached connected components each assigned to a tree; $d = 1, ..., N_{comp}^c$

for $j \leftarrow j_1^c$ *to* $j_{N_c}^c - 1$ **do**

  $CP \leftarrow \emptyset$;

  Extract the shortest path CP between the points $\hat{p}_{j,top}$ and $\hat{p}_{j+1,top}$;

  while $CP \neq \emptyset$ **do**

    $CP \leftarrow (CP \setminus SP_j) \setminus SP_{j+1}$;

    Select the global extremum of the z-coordinate in CP as the cut-point;

    Remove the cut-point from C_c ;

    Extract the shortest path CP between the points $\hat{p}_{j,top}$ and $\hat{p}_{j+1,top}$;

Apply connected components to C_c to obtain $\{C_{c,d}\}$;

Assign each connected component $C_{c,d}$ to the closest tree trunk;

After this procedure a connected component might be assigned to 1) only one tree, 2) to multiple trees, or 3) none of the trees. If the connected component is assigned to multiple trees, it is assumed to be spanning several trees that are touching each other. We label the connected components not assigned to any tree as "floating". The floating components are shown in black colour in Fig. 5 (Step 4).

Step 5: Splitting touching trees For a connected component C_c spanning N_c trees $\{T_j\}$, $j = j_1^c, ..., j_{N_c}^c$, we run Algorithm 2. Before running the algorithm, we update SP_j, *main axis* of the j^{th} trunk, together with the points $\hat{p}_{j,top}$ and $\hat{p}_{j,bottom}$. Recall that $\hat{p}_{j,top}$ and $\hat{p}_{j,bottom}$ are the top and bottom points of the skeleton of the j^{th} tree trunk and SP_j is the shortest path connecting them.

Fig. 5 (b) shows a connected component spanning three trees. The main axes of them are plotted in red colour, on the left.

Algorithm 2 takes as input the set of trees identities $\{T_j\}$, $j = j_1^c, ..., j_{N_c}^c$ spanned by the connected component C_c. For each adjacent tree pair T_j, T_{j+1}, the shortest path between their top points $\hat{p}_{j,top}$ and $\hat{p}_{j+1,top}$ is extracted. We call this path CP, the connecting path, which contains the touching point of branches from trees T_j and T_{j+1}. Each such path is searched for a cut-point to separate the connected adjacent trees. The cut-point is removed from the component C_c to break the connectivity at that point. The process is repeated and CP is updated until there remains no connected path between $\hat{p}_{j,top}$ and $\hat{p}_{j+1,top}$. Fig. 5 (b) depicts the connecting paths CP between adjacent trees with green dots.

After all connecting paths are extracted and the cut-points are found and removed, detached connected components $\{C_{c,d}\}$; $d = 1, ..., N_{comp}^c$ of C_c are extracted. Then each connected component is assigned to the tree identity of the closest tree trunk. Fig. 5 (b) shows the detached connected components each coloured according to its tree identity.

It is challenging to determine the point where branches from two trees touch each other. Many architectural and morphological rules concerning apple tree branches can be incorporated. However, here, we use a simple heuristic based on the assumption that the point that changes direction along the z-axis (upwards or downwards) corresponds to a meeting point along the path. We select the global extremum of the z-coordinate as the cut-point of the connecting path.

Step 6: Assigning floating components to a tree The tree membership of a floating component C_c is determined using Algorithm 3. Before running Algorithm 3, we identify the set $\mathbb{C} = \{(C_1, \tau_1), ..., (C_{N_F}, \tau_{N_F})\}$ of connected components already assigned to a tree. Here $\tau_f \in \{t_1, ..t_{N_{trees}}\}$ is the tree identity of the component C_f. We determine the two closest components in $\mathbb{C}$ to the floating component C_c. If the distance to one connected component is more than 3 times than the distance to the other component, we assign the points in C_c to the tree identity of the closest component. Otherwise, we locate the end-points in C_c, fit lines to these end-points and extend these lines, as shown in Fig. 5 (c). The minimum distance of the two closest connected components to these lines are calculated. The floating component is then assigned to the tree identity of the connected component with the minimum distance to the extended lines. Algorithm 3 gives the details of the process.

Step 7: Labeling all points with tree identities After Steps 5 and 6, all connected components in S_{trees} are assigned to a tree label $\tau_c \in \{t_1, ..t_{N_{trees}}\}$. Recall that the connected components are extracted from the skeleton S_{trees} of the point cloud PC_w^{trees}. For each point $\hat{p} \in PC_w^{trees}$, we locate the closest component of S_{trees} and assign the tree identity of the component to the point $\hat{p}$. Fig. 5 (Step 7) shows the points of a sample PC_w^{trees} coloured according to their tree identities.

Recall that PC_w^{trees} is a subset of PC_w^{TP}, which is the winter point cloud aligned to the trellis-plane. To find the tree identities of the points in the calibrated winter cloud PC_w^C, we first apply $p = \hat{p}R^{-1}$ to each point $\hat{p} \in PC_w^{trees}$ with tree identity $\tau \in \{t_1, ..., t_{N_{trees}}\}$. Then, we retrieve the closest point $p_i \in PC_w^C$ to p and set $\tau_i = \tau$.

Algorithm 3: Assignment of a floating branch to a neighbouring tree.

Data: C_c: Floating connected component;

$\mathbb{C} = \{(C_f, \tau_f)\}$: Connected components already assigned to a tree $(f = 1, 2, ..., N_F)$

Result: $\tau_c \in \{t_1, ..t_{N_{trees}}\}$: Tree identity of C_c

for $f \leftarrow 1$ *to* N_F **do**
    Calculate the minimum distance d_f between the points in C_c and the points in C_f;

$d^{F1} \leftarrow$ Minimum of d_f; $d^{F2} \leftarrow$ Next minimum of d_f;

$C^{F1} \leftarrow$ Component with d^{F1} ; $C^{F2} \leftarrow$ Component with d^{F2};

$\tau^{F1} \leftarrow$ Tree identity of C^{F1} ; $\tau^{F2} \leftarrow$ Tree identity of C^{F2};

if $\frac{d^{F2}}{d^{F1}} > 3$ **then**
    $\tau_c \leftarrow \tau^{F1}$;

else
    Extract the end-points p_e of C_c, $e = 1, 2, ..., N_e$;
    for $e \leftarrow 1$ *to* N_e **do**
        Extract K nearest neighbours of p_e with $K = 10$;
        Fit a line l_e to the neighbours;
        $d^{eF1} \leftarrow$ Minimum distance of the points in C^{F1} to the line l_e;
        $d^{eF2} \leftarrow$ Minimum distance of the points in C^{F2} to the line l_e;
    if $min\{d^{eN1}\} < min\{d^{eN2}\}$ **then**
        $\tau_c \leftarrow \tau^{F1}$
    else
        $\tau_c \leftarrow \tau^{F2}$;

2.5. Apple detection

To detect apples, we applied simple colour thresholding to the calibrated 3D colour point cloud of the harvest scene PC_h^C. First, the RGB colours of points are converted to HSV (Hue, Saturation, Value) representation. The points in the hue range [0.15-0.2] are assumed to correspond to green/yellow apple points. The red apple points are assumed to be in the hue range [0-0.05] and [0.95-1]. The points with hue values in these ranges are retrieved and converted to volumetric form. The connected components of the volumetric form and their bounding boxes are extracted. The centers of these bounding boxes are mapped to the 3D space of PC_h^C and are considered to be the locations of detected apples. We denote the set of detected apples in a harvest scene as $\mathcal{A} = \{p_1^\alpha, ..., p_{N_{apples}}^\alpha\}$, where p_a^α is the location of a detected apple.

Although our apple detection approach is primitive, it provides recall rates in the range of 74% to 90% (see Section 3.2). This level of detection success is sufficient to demonstrate the effectiveness of our approach for assigning retrieved apples to their respective trees.

2.6. Assigning apples to individual trees

The main objective of this work is to automatically assign detected apples to their respective trees; i.e. to determine the tree identity $\tau_a \in \{t_1, ..., t_{N_{trees}}\}$ of each detected apple $p_a^\alpha \in \mathcal{A}$. To this end, we align calibrated winter cloud PC_w^C and summer cloud PC_h^C and assign apple p_a^α detected from PC_h^C to the tree identity of the closest branch point in the aligned winter cloud.

Since both point clouds were transformed, through calibration, to a common reference frame with the origin at the base of a reference tree (see Supplementary Material A for details), they are already initially aligned. We apply the standard Iterative Closest Point (ICP) algorithm (Chen & Medioni, 1992) to improve the alignment. Point to point metric is used to minimise the alignment error. ICP returns the transformation parameters; a rotation matrix R^{wh} and a translation vector T^{wh} that align the points in PC_w^C to the points in PC_h^C:

$$PC_{wh}^C = \{p_i' = p_i R^{wh} + T^{wh} : (p_i \in PC_w^C) \;\&\; \tau_i \in \{t_1, ..., t_{N_{trees}}\}\}. \qquad (20)$$

Once the transformed winter point cloud PC_{wh}^C is obtained, the closest branch point in PC_{wh}^C labeled with a tree identity to the apple location p_a^α is retrieved:

$$i^* = \arg \min_{p_i' \in PC_{wh}^C} \|p_i' - p_a^\alpha\|; \qquad (21)$$

and the tree identity of apple a is set as

$$\tau_a = \tau_{i^*}. \tag{22}$$

2.7. Ground truth and evaluation metrics

To provide ground truth for evaluation of our semantic segmentation scheme, we manually labeled each point $p_i \in PC_w^C$ with one of the semantic labels $\gamma_i^{GT} \in$ {*"Tree trunk"*, *"Branch"*, *"Trellis wire+Water pipe"*, *"Support pole"*}. We used CloudCompare (2.11, GPL software, 2020) to label the point cloud. Fig. 6-(a) shows a sample winter scene with points coloured according to their manually annotated ground truth labels.

We evaluated the performance of the semantic segmentation module described in Section 2.4.1 using Recall (Re), Precision (Pr), F1 score ($F1$), Intersection over Union (IoU), and Class Accuracy (CA), defined as

$$Re = \frac{TP}{TP + FN} \tag{23}$$

$$Pr = \frac{TP}{TP + FP} \tag{24}$$

$$F1 = 2 \times \frac{Pr \times Re}{Pr + Re} \tag{25}$$

$$IoU = \frac{TP}{TP + FN + FP} \tag{26}$$

$$CA = \frac{TP + TN}{TP + TN + FP + FN}, \tag{27}$$

where TP, TN, FP and FN, correspond to the number of True Positives, True Negatives, False Positives, and False Negatives, respectively. These cases for the *"Tree trunk"* are determined as follows:

$$\text{Case}_i = \begin{cases} \text{True Positive} & \text{if } \gamma_i = \gamma_i^{GT} = \textit{"Tree trunk"} \\ \text{True Negative} & \text{if } (\gamma_i \neq \textit{"Tree trunk"}) \& (\gamma_i^{GT} \neq \textit{"Tree trunk"}) \\ \text{False Positive} & \text{if } (\gamma_i = \textit{"Tree trunk"}) \& (\gamma_i^{GT} \neq \textit{"Tree trunk"}) \\ \text{False Negative} & \text{if } (\gamma_i \neq \textit{"Tree trunk"}) \& (\gamma_i^{GT} = \textit{"Tree trunk"}), \end{cases} \tag{28}$$

where γ_i^{GT} is the ground truth label of point p_i and γ_i is the label predicted by our automatic semantic segmentation scheme. The cases for *"Support pole"* and *"Trellis wire+Water pipe"* are obtained in a similar manner.

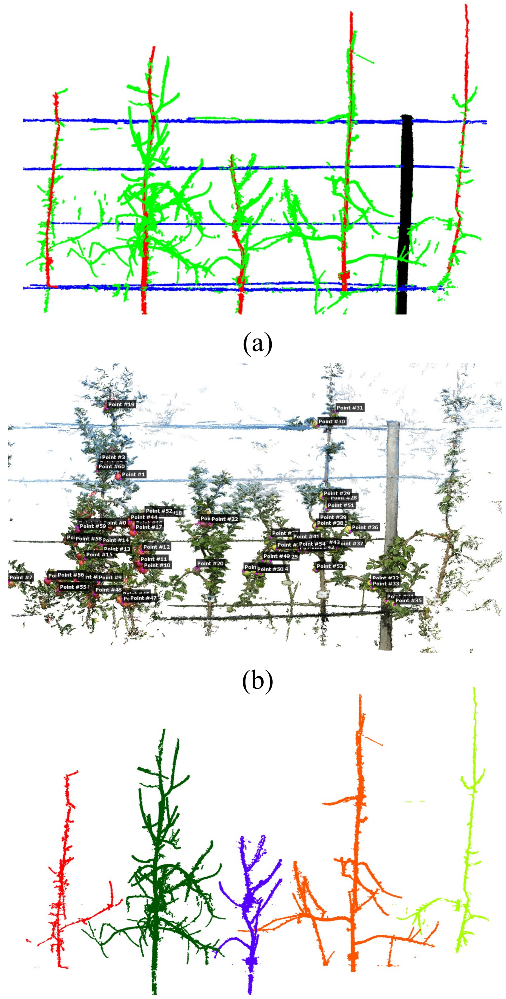
In order to assess the performance of the colour-based apple detection approach, we manually marked the apple positions in the harvest point clouds and obtained the set of points $\mathcal{A}^{GT} = \{p_g^{\alpha,GT}\}; g = 1, ..., N_{apples}^{GT}$. In Fig. 6-(b), a harvest point cloud with ground truth apple positions is shown. For evaluation, we used Recall (Re) and Precision (Pr) metrics, defined in Eq. (23) and (24). Here, the True Positives correspond to the cases where a ground truth apple is correctly localised. The False Positives are wrong detections returned by the algorithm. The False Negatives correspond to the ground truth apple locations missed by the algorithm. A detection $p_a^\alpha \in \mathcal{A}$ is considered a True Positive if there is a ground truth apple $p_g^{\alpha,GT} \in \mathcal{A}^{GT}$ such that $\|p_a^\alpha - p^{\alpha,GT}\| < 10cm$ and there is no other detected apples closer to $p^{\alpha,GT}$. We pair the indices (a, g) to indicate that $p_a^\alpha \in \mathcal{A}$ corresponds to $p_g^{\alpha,GT} \in \mathcal{A}^{GT}$. The number of False Positives and False Negatives are then calculated as:

$$FP = N_{apples} - TP \tag{29}$$

$$FN = N_{apples}^{GT} - TP \tag{30}$$

where TP is the number of True Positives, N_{apples} is the number of detected apples in $\mathcal{A}$ and N_{apples}^{GT} is the number of ground truth apples in $\mathcal{A}^{GT}$.

The end result of our apple assignment pipeline is the tree identity of each detected apple, indicating which tree it belongs to. In order to evaluate assign-



(a)

(b)

(c)

Figure 6: Ground truth. (a) Manually labeled point cloud for assessment of trellis wire, tree trunk and support pole detection, (b) Harvest point cloud with ground truth apple locations, (c) Point cloud manually segmented to individual trees.

ment performance, we provided the correct tree identities of the ground truth apples via manual inspection; i.e. we determined $\tau_g \in \{1, .., N_{trees}\}$ for each $p_g^{\alpha,GT} \in \mathcal{A}^{GT}$. We computed the accuracy of the apple assignment (ACC) as the ratio of the number of correctly assigned true positives TP_C to the total number of true positives TP in the scene:

$$ACC = \frac{TP_C}{TP} \tag{31}$$

A detection $p_a^\alpha \in \mathcal{A}$ is considered to be a correctly assigned true positive if its tree identity τ_a, determined by Eq. (21) and (22), is equal to the tree identity τ_g of its matched ground truth apple $p_g^{\alpha,GT} \in \mathcal{A}^{GT}$.

Recall that we assigned each apple $p_a^\alpha \in \mathcal{A}$ to the tree identity τ_{i^*} of the closest branch point p_{i^*} in the aligned winter cloud through Eq. (21) and (22). In order to decouple the apple assignment errors due to branch deformation between winter and summer trees and errors due to our automatic tree separation method, we performed the apple assignment procedure on two types of data:

1. Manually Separated: We manually separated the winter point clouds into individual trees and provided the ground truth tree identities $\tau_i^{GT} \in \{1, ..., N_{trees}^{GT}\}$ of the trunk and branch points in the winter cloud. We used CloudCompare (2.11, GPL software, 2020) for annotation. One example is shown in Fig. 6-(c).
2. Automatically Separated: We used the tree identities $\tau_i \in \{1, ..., N_{trees}\}$ of the trunk and branch points in the winter cloud predicted by our automatic tree separation procedure.

3. Results

We first report the results of the semantic segmentation method, which detects the trellis wires, tree trunks and support poles. Then, we provide the performance of the apple detection method and the assignment procedure of apples to individual trees in the scene.

3.1. Evaluation of detection of trellis wires, tree trunks and support poles

In Fig. 7, we give visual results of our semantic segmentation method for two winter scenes. The visual results for all the seven scenes can be found in

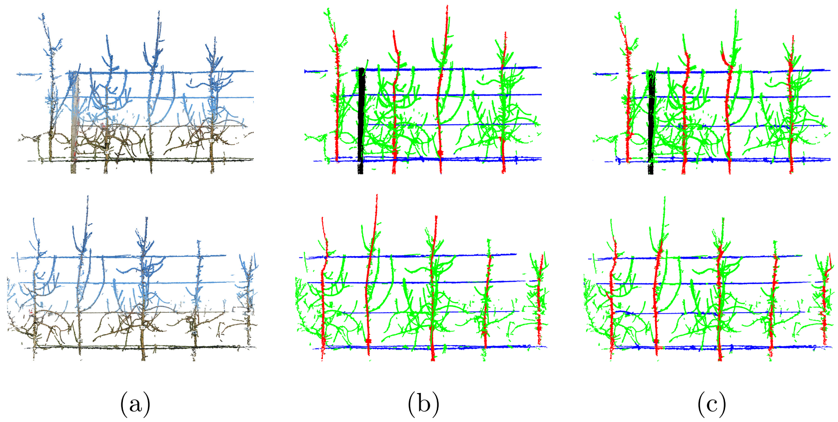
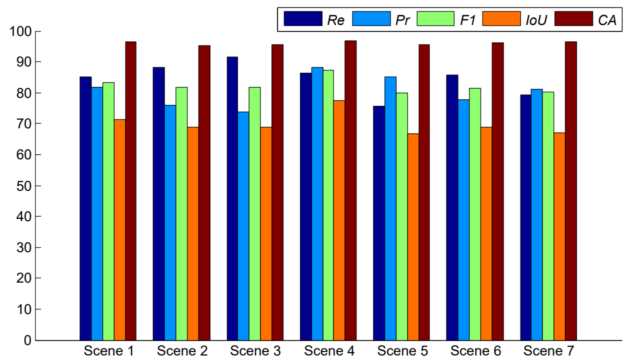


Figure 7: (a) Calibrated point clouds, (b) Manually generated Ground Truth (cyan:trellis wires, red: tree trunks, black: support poles), (c) Semantic labels obtained by our method for automatic detection of trellis wires, tree trunks, and support poles

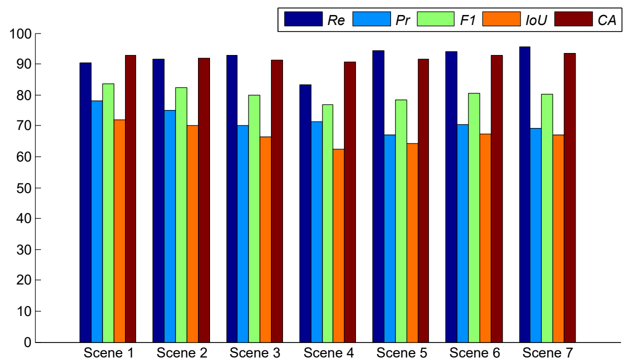
Supplementary Material B. We can observe that all the trees in the scenes of the apple orchard, the trees were correctly localised. The number of detected tree trunks and the actual number of trees were equal for all seven scenes; $N_{trees} = N_{trees}^{GT}$.

Table 2 provides quantitative evaluation of our semantic segmentation method. In Fig. 8, the results are given as bar graphs. The recall and precision values for the trellis wires are satisfactory. All the support poles in the scenes were correctly identified and segmented with over 90% success. The recall rate for the trunks is over 90% for all but one scene, meaning that most of the trunk points are retrieved. The precision rates are satisfactory for our purposes. The less than perfect precision is due to the fact that branching points close to the tree trunks are also classified as trunks by our method.

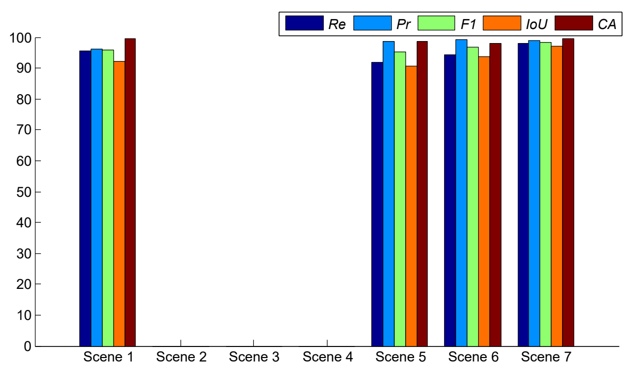
It should be recalled that our aim is not to provide a perfect segmentation, but rather 1) to detect and remove the trellis wires to break connectivity between adjacent trees, 2) to locate the tree trunks correctly to be able to separate individual trees, and 3) to remove the support poles. For the purposes of our application, these aims were achieved with this level of automatic point labeling of the scene.



(a) Trellis wires



(b) Tree trunks



(c) Support Poles

Figure 8: Performance of the method for detection of trellis wires, tree trunks and support poles in terms of Recall (Re), Precision (Pr), F1 score ($F1$), Intersection over Union (IoU), and Class Accuracy (CA) (in %).

Table 2: Performance of the method for detection of trellis wires, tree trunks and support poles in terms of Recall (Re), Precision (Pr), F1 score ($F1$), Intersection over Union (IoU), and Class Accuracy (CA). NP is for non-present.

	Trellis wires				
	% Re	% Pr	% $F1$	% IoU	% CA
Scene 1	84.98	81.61	83.26	71.32	96.42
Scene 2	88.16	76.01	81.63	68.96	95.19
Scene 3	91.48	73.65	81.61	68.93	95.52
Scene 4	86.48	88.20	87.33	77.51	96.88
Scene 5	75.47	85.21	80.04	66.73	95.64
Scene 6	85.82	77.75	81.59	68.90	96.06
Scene 7	79.24	81.16	80.19	66.93	96.48
	Tree trunks				
	% Re	% Pr	% $F1$	% IoU	% CA
Scene 1	90.26	77.97	83.67	71.92	92.83
Scene 2	91.49	74.89	82.36	70.01	91.77
Scene 3	92.77	70.03	79.81	66.40	91.31
Scene 4	83.23	71.23	76.76	62.29	90.55
Scene 5	94.25	67.03	78.34	64.40	91.56
Scene 6	94.02	70.39	80.51	67.37	92.78
Scene 7	95.47	69.19	80.24	66.99	93.27
	Support poles				
	% Re	% Pr	% $F1$	% IoU	% CA
Scene 1	95.50	96.24	95.87	92.07	99.44
Scene 2	NP	NP	NP	NP	NP
Scene 3	NP	NP	NP	NP	NP
Scene 4	NP	NP	NP	NP	NP
Scene 5	91.83	98.74	95.16	90.77	98.65
Scene 6	94.44	99.26	96.79	93.78	98.15
Scene 7	97.94	98.96	98.45	96.94	99.64

3.2. Evaluation of apple detection and assignment to individual trees

The precision and recall values obtained with colour-based apple detection are given in Table 3 and, also presented in Fig. 9 as a bar graph. Despite the simplicity of the detection approach, we achieved over 90.75% recall; i.e. most of the apples in the ground truth were retrieved. The false negatives occurred since we did not post-process the connected components for resolving clusters of apples. The over-detection (precision 65,37%) can be explained by the sensitivity of the colour-based algorithm and the lack of shape-based apple verification. Fig. 11 (a) and (b) visually illustrate the performance of our apple detection method on two sample scenes.

Table 3: Apple detection performance in terms of Recall (Re) and Precision (Pr).

3D scenes	% Re	% Pr
Scene 1	74.50	61.29
Scene 2	87.34	62.16
Scene 3	88.54	58.21
Scene 4	90.00	48.64
Scene 5	90.62	58.58
Scene 6	77.41	65.62
Scene 7	80.85	66.66

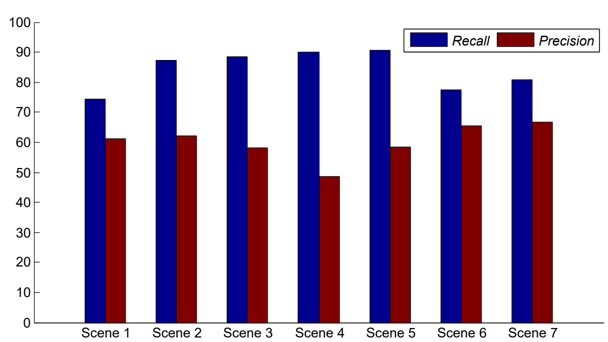


Figure 9: Apple detection performance in terms of Recall (Re) and Precision (Pr) (in %).

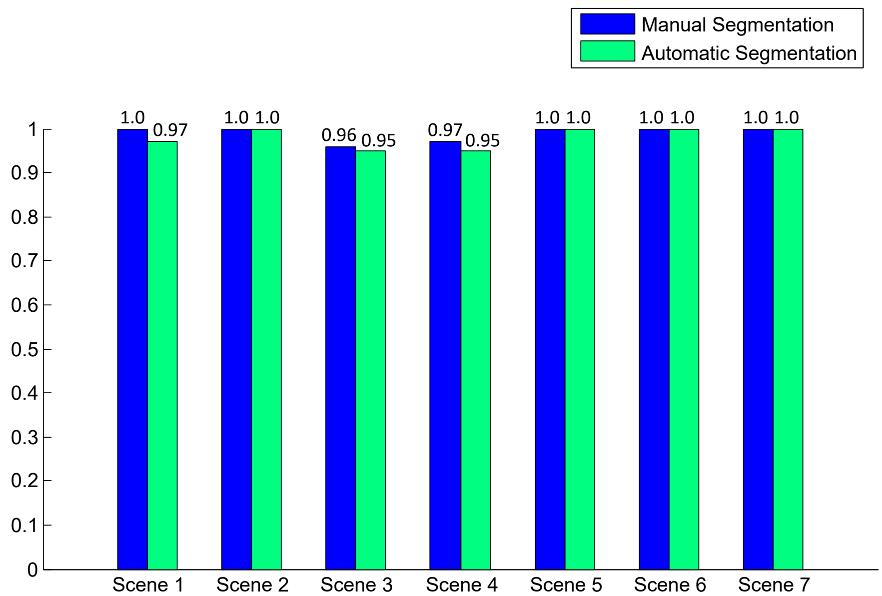


Figure 10: Accuracy of assigning apples to the correct apple trees in 3D models.

Our main task is to correctly assign the detected apples to the individual trees they belong to. As we have stated earlier, we performed the assignment procedure to two types of data: 1) The winter point clouds which are manually segmented to individual trees, and 2) The winter point clouds where the trees are segmented using our automatic tree separation method. Fig. 10 shows the assignment accuracy (ACC) on both type of data. The performance is high for both cases (100% on four scenes). With automatic tree separation, a performance drop of less than 3% is observed, demonstrating that our automatic pipeline was able to detach individual trees and correctly assign the detected apples.

Fig. 11 (c) and (d) show the registration result of winter and harvest point clouds for two sample scenes. Each separated tree in the winter clouds is shown in a different colour. In Fig. 11 (e) and (f), the detected apples are shown with the colour of their corresponding tree labels.

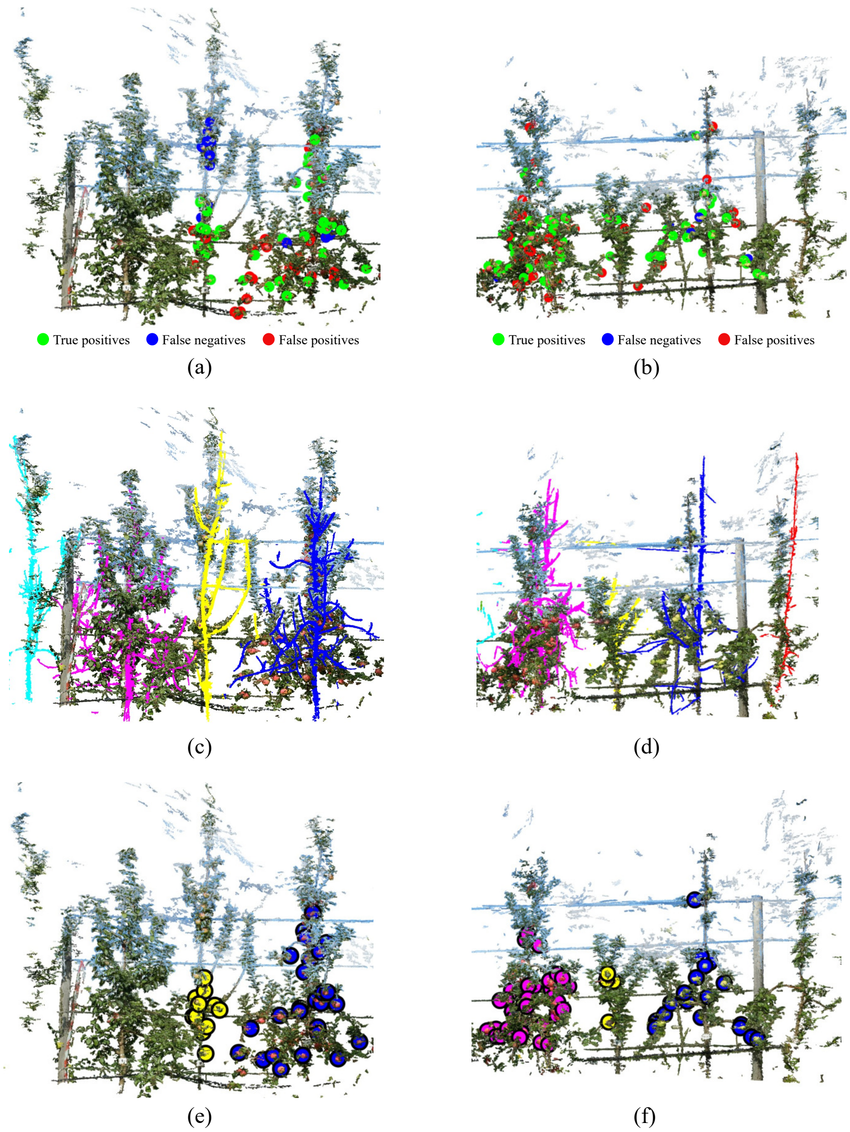


Figure 11: (a), (b) True positives, false negatives, and false positives obtained with colour-based apple detection method for two sample scenes. (c), (d) Registration of harvest and winter clouds for the two scenes. Each separated tree is shown with a different colour. (e), (f) Assignment of true positives to their corresponding trees for the two scenes.

3.3. *Processing time*

The average processing times, together with the standard deviations, for the steps of the pipeline are provided in Table 4. Given a 3D point cloud reconstructed through SFM, the total time for processing the point cloud is approximately 3 minutes. We also give the characteristics of the machine we used to process the data in Table 5.

Table 4: Average processing times of the steps of the pipeline in seconds.

Step	**Processing time**
Calibration and ROI extraction	16.53 ± 2.02 s
Detection of tree trunks, trellis wires and support poles	50.20 ± 2.42 s
Tree separation	18.41 ± 2.56 s
Apple detection and assignment	118.40 ± 3.78 s
Total	203.54 ± 10.78s

Table 5: Machine characteristics.

RAM	**Processor**	**GPU membership**
64 GB	Intel® Xeon® Silver 4114 CPU @2.20 GHz and 2.19 GHz (2 processors)	Quadro P4000 GPU of 8 GB

4. Discussion

The full pipeline presented and tested in this manuscript achieves great performance for assigning apples to individual trees in dense orchards. The main strategy is aligning summer and winter point clouds. The sub-steps of the pipeline, for which we chose standard approaches for implementation, are open to improvement for further performance increase.

Images were acquired manually with a standard camera. This is a rather time consuming process for producing hundreds of images per tree. The speed of acquisition can be increased and the amount of images can be optimised by a drone with a camera or a land robot with multiple cameras and automatic navigation via GPS localisation (Mogili & Deepak, 2018). The object of reference for calibration and registration of the summer and winter point cloud was chosen to be the X-Rite ColorChecker, since it is a standard tool in the computer vision community. In principle, any reference object with a distinctive geometric pattern could serve the same purpose.

The deformation we observed with our data (young trees of four years old) becomes even more pronounced for older trees. Registration of winter and summer calibrated point clouds could be performed efficiently with non-rigid registration while dealing with older trees, where the deformation during summer could be larger due to increased fruit load. Non-rigid registration is widely used in medical imaging when data from two different modalities, such as MRI and X-Ray images, should be registered. Non-rigid deformation between the image sets are commonly observed due to movement of the patient or artifacts of the imaging systems. The literature on non-rigid registration of medical images can thus be revisited for our plant imaging problem (Holden, 2007). To avoid having a too large exploration space for this non-rigid registration, one could also use botanical and physical knowledge on the development of trees. The size and weight of the fruits is important because it can cause arching of the branches, therefore, a deformation of the architecture. Another factor that alters the architecture is the secondary growth of the branches. Expert knowledge on such processes can be used to constrain the deformation space and fix the hyperparameters of the non-rigid registration algorithms.

We based the evaluation of the registration of winter and harvest point clouds on the rate of correct assignment of the apples to their corresponding trees. A thorough evaluation of the matching error is possible through computing distances between corresponding keypoints in the two point clouds. However, such a procedure necessitates manually establishing ground truth correspondences between well-located keypoints. It will be a worthy endeavor to measure the registration error and decouple errors due to changing structures in the trees and errors due to the limitations of the matching method. The decoupling can be done by locating keypoints both on the fixed structures (e.g. on the trellis structure) and on the trees (e.g. branching locations).

Another issue is the applicability of our method to other orchards and other fruit trees. The registration method we proposed relies on the initial alignment of two point clouds, each of which are calibrated separately. The calibration procedure operates on three requirements: 1) There should be a reference pattern (in our case, the ColorChecker) in place to recover the scale and to establish upwards and leftwards directions; 2) The trees should be organised in a row to establish the -y- direction; and 3) The relative position of a designated tree to the reference pattern should be known to locate the origin. As long as these requirements are met, our calibration procedure will be applicable to fruit orchards organised as rows.

The parameters of our method were fixed for all the scenes we processed in our experiments. We avoided fine tuning the parameters required by some standard procedures such as voxelization and Hough line extraction via trusting the added robustness of our further processing steps. For example, for the grid size for voxelization, the choices of both 5mm and 10mm work effectively for extraction of a representative skeleton, although the latter gives a coarser skeleton. The grid size should not be too small compared to the point resolution or to cause computational overload; and it should not be too large to cause merging unconnected structures into one voxel. The dependency on parameters in Hough transform on extracting candidate lines is controlled by the line merging step to extract the expected four trellis lines. These parameters will not require adjustment for application to apple orchards with a similar trellis-wire organisation.

For other parameters, such as the inlier distance to detect the trellis-plane (1cm – to cover the thickness of a trellis wire), trunk candidate search distance (5cm – to cover the thickness of a trunk), and the minimum distance between successive trellis wires (30cm), we used the actual metric quantities of these structures, again avoiding fine-tuning. For application to other orchards, these parameters can be adjusted according to geometric priors such as trellis-wire thickness, distance between trellis-wires and expected trunk thickness.

The point clouds used in our experiments spanned four to five trees reconstructed using manually acquired images. The extension of the method to process the whole tree row in an automatic manner is possible through installation of multiple low-cost reference patterns along the row. Such patterns can also be installed per tree, together with a QR code as a tree identifier. Using multiple patterns will help reduce the error in the directions of the fixed frame through multiple estimations. They can also be used to reset the reconstruction process of SFM at predetermined locations to avoid the accumulative drift through providing multiple point clouds covering overlapping regions along the tree row. These point clouds can be processed separately, and if necessary, can be calibrated and registered to obtain the entire row model. Our future work includes installation of one board with printed ColorChecker structure and a QR code at each tree and the analysis of the performance of suggested solutions.

In this work, we used connectivity analysis and simple heuristics to disconnect touching trees. Alternatively, the identification of each tree unit can be achieved using the architectural criteria specific to each tree. They are linked to the basic architectural models defined for each taxon (Hallé, 2004). They are supplemented by the growth conditions specific to each tree and are assessed by the diameter, length, age and branching angles of the branches but also by the location of inflorescences and fruits.

The apple detection algorithm chosen in this work was extremely simple and it will be necessary to revisit the huge literature on apple detection to improve the performance, specially on groups of apples or to reduce the amount of false positives. State-of-the-art methods employing deep learning architectures, such as (Roy et al., 2019; Häni et al., 2018, 2020) have been highly successful. It is possible, through the projection parameters estimated by the multi-view reconstruction process, to combine image-based fruit detection methods with 3D point reconstruction of the scene. Indeed, Dong et al. (2020) proposed a method where apple detection is performed on 2D images and detected apples in images are used to segment the apples in the 3D reconstruction. A similar method that establishes correspondences between 3D points and the pixels in 2D images and classifies each 3D point as apple/non-apple can be integrated into our scheme.

One of the common issues in image-based fruit detection methods is the occlusion of fruits caused by leaves and other fruits (Gongal et al., 2015). Imaging trees from various viewpoints greatly increases the possibility that occluded fruits will be visible in more than one image. However, association of the same fruits occurring in different images is necessary for 2D image-based methods to avoid double-counting. The 3D reconstruction pipeline we used in this work greatly alleviates the occlusion problem by utilising multi-view reconstruction and inherently registering multiple sightings of a single fruit. A further research question can be formulated as the systematic investigation of the severity of occlusion with employing the ground truth count of harvested apples and the analysis of the impact of the imaging systems and acquisition protocols on capturing heavily occluded fruits.

Our pipeline enables the assignment of apples to the trees that bear them. This makes it possible to assess the production and the quality of the fruiting body in variety testing applications and also in the agronomic management of orchards. We know that fruiting is the expression of primary and secondary growth followed by a flowering process with the formation of inflorescences and flowers. One, two or three years old axes that are part of the overall architecture of the tree carry these inflorescences. In this biological process, Laury et al. (Lauri et al., 1996, 2010) showed the importance of the age of branches, their position in the architecture and secondary growth on the fruit load of the tree. Our pipeline opens the way to acquire data at different developmental stages, analyse the architecture of individual trees, track primary and secondary growth, determine their axes of different ages. The location of the fruits and the identification of the characteristics of the axes that carry them, supplemented by a temporal monitoring of the architectural development could make it possible to obtain information to manage and improve the agronomic management of fruit trees.

5. Conclusion

In this article, we presented, for the first time to the best of our knowledge, a pipeline to assign detected apples to their corresponding apple trees in 3D colour point clouds. The pipeline was able to detect and filter out trellis wires and support poles. It successfully located trunk locations in the scene and retrieved trunk points with more than 90% recall rate. The detected apples were assigned to their corresponding trees with more than 95% accuracy.

This first proof of feasibility has shown the possibility and benefit of registration of 3D models of orchard scenes obtained in two different seasons. A direction for further development could be more frequent acquisition and reconstruction during the year, for instance during flowering period to link flower density to apple yield on individual trees. As another application, the configuration of the fruits in the harvest period can be used to guide the pruning process in early spring. These perspectives are now open with the pipeline proposed in this study.

6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

7. Acknowledgements

This work is supported by the EU H2020 INVITE project under grant agreement 817970.

8. Appendix A. Supplementary material

- **Supplementary Material A:** Calibration of 3D RGB Point Clouds of Apple Orchard Scenes

- **Supplementary Material B:** Visual results for detection of trellis wires, tree trunks and support poles

References

Apolo-Apolo, O. E., Pérez-Ruiz, M., Martínez-Guanter, J., & Valente, J. (2020). A cloud-based environment for generating yield estimation maps from apple orchards using UAV imagery and a deep learning technique. *Frontiers in Plant Science*, *11*, 1086. URL: https://www.frontiersin.org/article/10.3389/fpls.2020.01086. doi:10.3389/fpls.2020.01086.

Bargoti, S., & Underwood, J. (2017). Deep fruit detection in orchards. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 3626–3633).

Bargoti, S., & Underwood, J. P. (2017). Image segmentation for fruit detection and yield estimation in apple orchards. *Journal of Field Robotics*, *34*, 1039–1060. doi:10.1002/rob.21699.

Bargoti, S., Underwood, J. P., Nieto, J. I., & Sukkarieh, S. (2015). A pipeline for trunk detection in trellis structured apple orchards. *Journal of Field Robotics*, *32*, 1075–1094.

Barnea, E., Mairon, R., & Ben-Shahar, O. (2016). Special issue: Robotic agriculture. *Biosystems Engineering*, *146*, 57–70. doi:10.1016/j.biosystemseng.2016.01.013.

Brandtberg, T., Warner, T. A., Landenberger, R. E., & McGraw, J. B. (2003). Detection and analysis of individual leaf-off tree crowns in small footprint, high sampling density lidar data from the eastern deciduous forest in north america. *Remote Sensing of Environment*, *85*, 290 – 303. URL: http://www.sciencedirect.com/science/article/pii/S0034425703000087. doi:https://doi.org/10.1016/S0034-4257(03)00008-7.

Bresilla, K., Perulli, G. D., Boini, A., Morandi, B., Corelli Grappadelli, L., & Manfrini, L. (2019). Single-shot convolution neural networks for real-time fruit detection within the tree. *Frontiers in Plant Science*, *10*, 611. URL: https://www.frontiersin.org/article/10.3389/fpls.2019.00611. doi:10.3389/fpls.2019.00611.

Bulanon, D., Burks, T., & Alchanatis, V. (2008). Study on temporal variation in citrus canopy using thermal imaging for citrus fruit detection. *Biosystems Engineering*, *101*, 161 – 171. URL: http://www.sciencedirect.com/science/article/pii/S1537511008002420. doi:https://doi.org/10.1016/j.biosystemseng.2008.08.002.

Bulanon, D., Burks, T., & Alchanatis, V. (2009). Image fusion of visible and thermal images for fruit detection. *Biosystems Engineering*, *103*, 12 – 22. URL: http://www.sciencedirect.com/science/article/pii/S1537511009000610. doi:https://doi.org/10.1016/j.biosystemseng.2009.02.009.

Chen, S. W., Shivakumar, S. S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C. J., & Kumar, V. (2017). Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters*, *2*, 781–788.

Chen, Y., & Medioni, G. G. (1992). Object modeling by registration of multiple range images. *Image and Vision Computing*, *10*, 145–155.

Colmenero-Martinez, J. T., Blanco-Roldán, G. L., Bayano-Tejero, S., Castillo-Ruiz, F. J., Sola-Guirado, R. R., & Gil-Ribes, J. A. (2018). An automatic trunk-detection system for intensive olive harvesting with trunk shaker. *Biosystems Engineering*, *172*, 92 – 101. URL: http://www.sciencedirect.com/science/article/pii/S1537511017309741. doi:https://doi.org/10.1016/j.biosystemseng.2018.06.002.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, Third Edition*. (3rd ed.). The MIT Press.

Coupel-Ledru, A., Pallas, B., Delalande, M., Boudon, F., Carrié, E., Martinez, S., Regnard, J.-L., & Costes, E. (2019). Multi-scale high-throughput phenotyping of apple architectural and functional traits in orchard reveals genotypic variability under contrasted watering regimes. *Horticulture Research*, *6*, 1–15.

Dong, W., Roy, P., & Isler, V. (2020). Semantic mapping for orchard environments by merging two-sides reconstructions of tree rows. *Journal of Field Robotics*, *37*, 97–121. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21876. doi:https://doi.org/10.1002/rob.21876. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21876.

Duda, R. O., & Hart, P. E. (1972). Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM*, *15*, 11–15. URL: https://doi.org/10.1145/361237.361242. doi:10.1145/361237.361242.

Fu, L., Majeed, Y., Zhang, X., Karkee, M., & Zhang, Q. (2020). Faster R–CNN–based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting. *Biosystems Engineering*, *197*, 245 – 256. URL: http://www.sciencedirect.com/science/article/pii/S1537511020302002. doi:https://doi.org/10.1016/j.biosystemseng.2020.07.007.

Furukawa, Y., Curless, B., Seitz, S. M., & Szeliski, R. (2010). Towards Internet-scale multi-view stereo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 1434–1441).

Furukawa, Y., & Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*, 1362–1376.

Gan, H., Lee, W. S., Alchanatis, V., & Abd-Elrahman, A. (2020). Active thermal imaging for immature citrus fruit detection. *Biosystems Engineering*, *198*, 291 – 303. URL: http://www.sciencedirect.com/science/article/pii/S1537511020302348. doi:https://doi.org/10.1016/j.biosystemseng.2020.08.015.

Ge, Y., Xiong, Y., & From, P. J. (2020). Symmetry-based 3D shape completion for fruit localisation for harvesting robots. *Biosystems Engineering*, *197*, 188 – 202. URL: http://www.sciencedirect.com/science/article/pii/S1537511020301963. doi:https://doi.org/10.1016/j.biosystemseng.2020.07.003.

Gené-Mola, J., Gregorio, E., Guevara, J., Auat, F., Sanz-Cortiella, R., Escolà, A., Llorens, J., Morros, J.-R., Ruiz-Hidalgo, J., Vilaplana, V., & Rosell-Polo, J. R. (2019a). Fruit detection in an apple orchard using a mobile terrestrial laser scanner. *Biosystems Engineering*, *187*, 171 – 184. URL: http://www.sciencedirect.com/science/article/pii/S1537511019308128. doi:https://doi.org/10.1016/j.biosystemseng.2019.08.017.

Gené-Mola, J., Sanz-Cortiella, R., Rosell-Polo, J. R., Morros, J.-R., Ruiz-Hidalgo, J., Vilaplana, V., & Gregorio, E. (2020). Fruit detection and 3D location using instance segmentation neural networks and structure-from-motion photogrammetry. *Computers and Electronics in Agriculture*, *169*, 105165. URL: http://www.sciencedirect.com/science/article/pii/S0168169919321507. doi:https://doi.org/10.1016/j.compag.2019.105165.

Gené-Mola, J., Vilaplana, V., Rosell-Polo, J. R., Morros, J.-R., Ruiz-Hidalgo, J., & Gregorio, E. (2019b). Multi-modal deep learning for Fuji apple detection using RGB-D cameras and their radiometric capabilities. *Computers and Electronics in Agriculture*, *162*, 689 – 698. URL: http://www.sciencedirect.com/science/article/pii/S0168169919301413. doi:https://doi.org/10.1016/j.compag.2019.05.016.

Gong, A., Yu, J., He, Y., & Qiu, Z. (2013). Citrus yield estimation based on images processed by an Android mobile phone. *Biosystems Engineering*, *115*, 162 – 170. URL: http:

//www.sciencedirect.com/science/article/pii/S1537511013000500. doi:https://doi.org/10.1016/j.biosystemseng.2013.03.009.

Gongal, A., Amatya, S., Karkee, M., Zhang, Q., & Lewis, K. (2015). Sensors and systems for fruit detection and localization: A review. *Computers and Electronics in Agriculture*, *116*, 8 – 19. URL: http://www.sciencedirect.com/science/article/pii/S0168169915001581. doi:https://doi.org/10.1016/j.compag.2015.05.021.

Gongal, A., Silwal, A., Amatya, S., Karkee, M., Zhang, Q., & Lewis, K. (2016). Apple crop-load estimation with over-the-row machine vision system. *Computers and Electronics in Agriculture*, *120*, 26 – 35. URL: http://www.sciencedirect.com/science/article/pii/S016816991500335X. doi:https://doi.org/10.1016/j.compag.2015.10.022.

Gonzalez, R. C., & Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. USA: Prentice-Hall, Inc.

Hallé, F. (2004). *Architectures de Plantes*. 109 Avenue de Lodève, 34070 Montpellier, France: JPC Editions.

Häni, N., Roy, P., & Isler, V. (2018). Apple counting using convolutional neural networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2559–2565). IEEE.

Häni, N., Roy, P., & Isler, V. (2020). A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, *37*, 263–282.

Hartley, R. I., & Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. (2nd ed.). Cambridge University Press, ISBN: 0521540518.

He, L., & Schupp, J. (2018). Sensing and automation in pruning of apple trees: A review. *Agronomy*, *8*, 211. URL: http://dx.doi.org/10.3390/agronomy8100211. doi:10.3390/agronomy8100211.

He, Z., Xiong, J., Chen, S., Li, Z., Chen, S., Zhong, Z., & Yang, Z. (2020). A method of green citrus detection based on a deep bounding box regression forest. *Biosystems Engineering*, *193*, 206 – 215. URL: http://www.sciencedirect.com/science/article/pii/S1537511020300647. doi:https://doi.org/10.1016/j.biosystemseng.2020.03.001.

Holden, M. (2007). A review of geometric transformations for nonrigid body registration. *IEEE Transactions on Medical Imaging*, *27*, 111–128.

Häni, N., Roy, P., & Isler, V. (2018). Apple counting using convolutional neural networks. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2559–2565).

Häni, N., Roy, P., & Isler, V. (2020). A comparative study of fruit detection and counting methods for yield mapping in apple orchards. *Journal of Field Robotics*, *37*, 263–282. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21902. doi:10.1002/rob.21902. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21902.

Kelman, E. E., & Linker, R. (2014). Vision-based localisation of mature apples in tree images using convexity. *Biosystems Engineering*, *118*, 174 – 185. URL: http://www.sciencedirect.com/science/article/pii/S1537511013001918. doi:https://doi.org/10.1016/j.biosystemseng.2013.11.007.

Lauri, P. E., Kelner, J. J., Trottier, C., & Costes, E. (2010). Insights into secondary growth in perennial plants: its unequal spatial and temporal dynamics in the apple (Malus domestica) is driven by architectural position and fruit load. *Annals of Botany*, *105*, 607–616. doi:10.1093/aob/mcq006.

Lauri, P.-E., Terouanne, E., & Lespinasse, J.-M. (1996). Quantitative analysis of relationships between inflorescence size, bearing-axis size and fruit-set —an apple tree case study. *Annals of Botany*, *77*, 277–286. doi:10.1006/anbo.1996.0031.

Lee, T., Kashyap, R., & Chu, C. (1994). Building skeleton models via 3-D medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, *56*, 462 – 478. URL: http://www.sciencedirect.com/science/article/pii/S104996528471042X. doi:https://doi.org/10.1006/cgip.1994.1042.

Lin, G., Tang, Y., Zou, X., Li, J., & Xiong, J. (2019). In-field citrus detection and localisation based on RGB-D image analysis. *Biosystems Engineering*, *186*, 34 – 44. URL: http://www.sciencedirect.com/science/article/pii/S1537511018312765. doi:https://doi.org/10.1016/j.biosystemseng.2019.06.019.

Linker, R. (2018). Machine learning based analysis of night-time images for yield prediction in apple orchard. *Biosystems Engineering*, *167*, 114 – 125. URL: http://www.sciencedirect.com/science/article/pii/S1537511017306785. doi:https://doi.org/10.1016/j.biosystemseng.2018.01.003.

Liu, X., Chen, S. W., Aditya, S., Sivakumar, N., Dcunha, S., Qu, C., Taylor, C. J., Das, J., & Kumar, V. (2018). Robust fruit counting: Combining deep learning, tracking, and structure from motion. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (pp. 1045–1052).

Lu, J., Lee, W. S., Gan, H., & Hu, X. (2018). Immature citrus fruit detection based on local binary pattern feature and hierarchical contour analysis. *Biosystems Engineering*, *171*, 78 – 90. URL: http://www.sciencedirect.com/science/article/pii/S1537511017301502. doi:https://doi.org/10.1016/j.biosystemseng.2018.04.009.

Lu, X., Guo, Q., Li, W., & Flanagan, J. (2014). A bottom-up approach to segment individual deciduous trees using leaf-off LiDAR point cloud data. *ISPRS Journal of Photogrammetry and Remote Sensing*, *94*, 1–12. doi:10.1016/j.isprsjprs.2014.03.014.

Marrero Fernández, P. D., Guerrero Peña, F. A., Ing Ren, T., & Leandro, J. J. (2019). Fast and robust multiple ColorChecker detection using deep convolutional neural networks. *Image and Vision Computing*, *81*, 15 – 24. URL: http://www.sciencedirect.com/science/article/pii/S0262885618301793. doi:https://doi.org/10.1016/j.imavis.2018.11.001.

Medeiros, H., Kim, D., Sun, J., Seshadri, H., Akbar, S. A., Elfiky, N. M., & Park, J. (2017). Modeling dormant fruit trees for agricultural automation. *Journal of Field Robotics*, *34*, 1203–1224. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21679. doi:10.1002/rob.21679. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21679.

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. URL: http://www.sciencedirect.com/science/article/pii/S1537511016300332. doi:https://doi.org/10.1016/j.biosystemseng.2016.04.013.

Mogili, U. R., & Deepak, B. (2018). Review on application of drone systems in precision agriculture. *Procedia Computer Science*, *133*, 502–509.

Nguyen, T. T., Vandevoorde, K., Wouters, N., Kayacan, E., De Baerdemaeker, J. G., & Saeys, W. (2016). Detection of red and bicoloured apples on tree with an RGB-D camera. *Biosystems Engineering*, *146*, 33 – 44. URL: http://www.sciencedirect.com/science/article/pii/S1537511016000088. doi:https://doi.org/10.1016/j.biosystemseng.2016.01.007. Special Issue: Advances in Robotic Agriculture for Crops.

Nielsen, M., Slaughter, D. C., Gliever, C., & Upadhyaya, S. (2012). Orchard and tree mapping and description using stereo vision and lidar. In *International Conference of Agricultural Engineering*.

Rosell, J., & Sanz, R. (2012). A review of methods and applications of the geometric characterization of tree crops in agricultural activities. *Computers and Electronics in Agriculture*, *81*, 124–141.

Roy, P., Dong, W., & Isler, V. (2018). Registering reconstructions of the two sides of fruit tree rows. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1–9).

Roy, P., & Isler, V. (2016). Vision-based apple counting and yield estimation. In *Kulić D., Nakamura Y., Khatib O., Venture G. (eds) 2016 International Symposium on Experimental Robotics. ISER 2016. Springer Proceedings in Advanced Robotics* (pp. 478–487). volume 1. doi:10.1007/978-3-319-50115-4_42.

Roy, P., Kislay, A., Plonski, P. A., Luby, J., & Isler, V. (2019). Vision-based preharvest yield mapping for apple orchards. *Computers and Electronics in Agriculture*, *164*, 104897.

Sabzi, S., Abbaspour-Gilandeh, Y., García-Mateos, G., Ruiz-Canales, A., & Molina-Martínez, J. M. (2018). Segmentation of apples in aerial images under sixteen different lighting conditions using color and texture for optimal irrigation. *Water*, *10*, 1634.

Safren, O., Alchanatis, V., Ostrovsky, V., & Levi, O. (2007). Detection of green apples in hyperspectral images of apple-tree foliage using machine vision. *Transactions of the ASABE*, *50*, 2303–2313.

Samiei, S., Rasti, P., Richard, P., Galopin, G., & Rousseau, D. (2020). Toward joint acquisition-annotation of images with egocentric devices for a lower-cost machine learning application to apple detection. *Sensors*, *20*, 4173.

Scher, C., Griffoul, E., & Cannon, C. (2019). Drone-based photogrammetry for the construction of high-resolution models of individual trees. *Trees*, *33*, 1385–1397. doi:10.1007/s00468-019-01866-x.

Sengupta, S., & Lee, W. S. (2014). Identification and determination of the number of immature green citrus fruit in a canopy under different ambient light conditions. *Biosystems Engineering*, *117*, 51 – 61. URL: http://www.sciencedirect.com/science/article/pii/S1537511013001141. doi:https://doi.org/10.1016/j.biosystemseng.2013.07.007. Image Analysis in Agriculture.

Stajnko, D., Lakota, M., & Hočevar, M. (2004). Estimation of number and diameter of apple fruits in an orchard during the growing season by thermal imaging. *Computers and Electronics in Agriculture*, *42*, 31 – 42. URL: http://www.sciencedirect.com/science/article/pii/S0168169903000863. doi:https://doi.org/10.1016/S0168-1699(03)00086-3.

Sun, S., Jiang, M., He, D., Long, Y., & Song, H. (2019). Recognition of green apples in an orchard environment by combining the GrabCut model and Ncut algorithm. *Biosystems Engineering*, *187*, 201 – 213. URL: http://www.sciencedirect.com/science/article/pii/S1537511019308207. doi:https://doi.org/10.1016/j.biosystemseng.2019.09.006.

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)* (pp. 6005–6012). doi:10.1109/IROS.2017.8206497.

Tao, Y., & Zhou, J. (2017). Automatic apple recognition based on the fusion of color and 3D feature for robotic fruit picking. *Computers and Electronics in Agriculture*, *142*, 388 – 396. URL: http://www.sciencedirect.com/science/article/pii/S0168169917302764. doi:https://doi.org/10.1016/j.compag.2017.09.019.

Tian, Y., Yang, G., Wang, Z., Wang, H., Li, E., & Liang, Z. (2019). Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Computers and Electronics in Agriculture*, *157*, 417 – 426. URL: http://www.sciencedirect.com/science/article/

pii/S016816991831528X. doi:https://doi.org/10.1016/j.compag.2019.01.012.

Torbert, S. (2016). *Applied Computer Science*. (2nd ed.). Springer Publishing Company, Incorporated.

Torr, P., & Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, *78*, 138 – 156. URL: http://www.sciencedirect.com/science/article/pii/S1077314299908329. doi:https://doi.org/10.1006/cviu.1999.0832.

Tu, S., Xue, Y., Zheng, C., Qi, Y., Wan, H., & Mao, L. (2018). Detection of passion fruits and maturity classification using Red-Green-Blue Depth images. *Biosystems Engineering*, *175*, 156 – 167. URL: http://www.sciencedirect.com/science/article/pii/S1537511017309844. doi:https://doi.org/10.1016/j.biosystemseng.2018.09.004.

Underwood, J. P., Jagbrant, G., Nieto, J. I., & Sukkarieh, S. (2015). Lidar-based tree recognition and platform localization in orchards. *Journal of Field Robotics*, *32*, 1056–1074.

Wachs, J. P., Stern, H. I., Burks, T. F., & Alchanatis, V. (2010). Low and high-level visual feature-based apple detection from multi-modal images. *Precision Agriculture*, *11*, 717–735.

Wang, Q., Nuske, S., Bergerman, M., & Singh, S. (2012). Automated crop yield estimation for apple orchards. In *Experimental Robotics: The 13th International Symposium on Experimental Robotics* (pp. 745–758). volume 88. doi:10.1007/978-3-319-00065-7_50.

Williams, H. A., Jones, M. H., Nejati, M., Seabright, M. J., Bell, J., Penhall, N. D., Barnett, J. J., Duke, M. D., Scarfe, A. J., Ahn, H. S., Lim, J., & MacDonald, B. A. (2019). Robotic kiwifruit harvesting using machine vision, convolutional neural networks, and robotic arms. *Biosystems Engineering*, *181*, 140 – 156. URL: http://www.sciencedirect.com/science/article/pii/S153751101830638X. doi:https://doi.org/10.1016/j.biosystemseng.2019.03.007.

Wu, C. (2013). Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision - 3DV 2013* (pp. 127–134).

Wu, C., Agarwal, S., Curless, B., & Seitz, S. M. (2011). Multicore bundle adjustment. In *Conference on Computer Vision and Pattern Recognition 2011* (pp. 3057–3064).

Wu, G., Zhu, Q., Huang, M., Guo, Y., & Qin, J. (2019). Automatic recognition of juicy peaches on trees based on 3D contour features and colour data. *Biosystems Engineering*, *188*, 1 – 13. URL: http://www.sciencedirect.com/science/article/pii/S1537511019308451. doi:https://doi.org/10.1016/j.biosystemseng.2019.10.002.

Wulfsohn, D.-L., Zamora, F., Tellez, C., Lagos, I., & Garcia-Finana, M. (2012). Multilevel systematic sampling to estimate total fruit number for yield forecasts. *Precision Agriculture*, *13*, 256–275. doi:10.1007/s11119-011-9245-2.

Xiong, J., Liu, Z., Chen, S., Liu, B., Zheng, Z., Zhong, Z., Yang, Z., & Peng, H. (2020). Visual detection of green mangoes by an unmanned aerial vehicle in orchards based on a deep learning method. *Biosystems Engineering*, *194*, 261 – 272. URL: http://www.sciencedirect.com/science/article/pii/S1537511020300970. doi:https://doi.org/10.1016/j.biosystemseng.2020.04.006.

Yandún Narváez, F. J., Salvo del Pedregal, J., Prieto, P. A., Torres-Torriti, M., & Auat Cheein, F. A. (2016). LiDAR and thermal images fusion for ground-based 3D characterisation of fruit trees. *Biosystems Engineering*, *151*, 479 – 494. URL: http://www.sciencedirect.com/science/article/pii/S1537511016300563. doi:https://doi.org/10.1016/j.biosystemseng.2016.10.012.

Zeng, L., Feng, J., & He, L. (2020). Semantic segmentation of sparse 3D point cloud based on geometrical features for trellis-structured apple orchard. *Biosystems Engineering*, *196*, 46 – 55. URL: http://www.sciencedirect.com/science/article/pii/S1537511020301288. doi:https://doi.org/10.1016/j.biosystemseng.2020.05.015.

Zhang, J., He, L., Karkee, M., Zhang, Q., Zhang, X., & Gao, Z. (2017). Branch detection with apple trees trained in fruiting wall architecture using stereo vision and regions-convolutional neural network(R-CNN). In *2017 ASABE Annual International Meeting*. doi:10.13031/aim.201700427.

Zhen, Z., Quackenbush, L., & Zhang, L. (2016). Trends in automatic individual tree crown detection and delineation—evolution of LiDAR data. *Remote Sensing*, *8*, 333. doi:10.3390/rs8040333.

Zhong, L., Cheng, L., Xu, H., Wu, Y., Chen, Y., & Li, M. (2016). Segmentation of individual trees from TLS and MLS data. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *10*, 774–787.