D5.5. Report on the computer vision algorithms and their applications

Fabrice Besnard, Jonathan Legrand, Katia Mirande, Christophe Godin, Franck Hétroy-Wheeler, David Colliaux, Timothée Wintz, Peter Hanappe

ROMI
ROBOTICS FOR MICROFARMS

Deliverable			D5.5
Deliverable title: Report on the computer vision algorithms and their applications			
Task	**T5.1 Sensors selection and data acquisition rate.** **T5.2 Data management infrastructure.** **T5.3 3D+t plant segmentation**		
Task Leader	**CNRS**	Planned Date	**31.07.2022**
		Effective Date	**31.07.2022**

	Written by	Reviewed and approved by	Authorized by
Name / Surname	**F. Besnard (CNRS) with contributions from J. Legrand (CNRS) ; K. Mirande, C. Godin & F. Hétroy-Wheeler (Inria) ; D. Colliaux, T. Wintz & P. Hanappe (Sony CSL)**	**P. Hanappe (Sony)**	**F. Besnard (CNRS)** **P. Hanappe (Sony)**

Executive Summary

1.1 Overview and description of the report

This deliverable is made of this unique report document. **It provides an overview of the different computer vision algorithms used in WP5 for the advanced sensing of plants and crops in three dimensions**. It presents the different contexts in which these different algorithms have been designed and used, it points to the connections between them, compares their strength and weaknesses and the possibility to combine them. It opens to future research directions the ROMI paved the way for.

1.2 Partners involved

Leader: **CNRS**
Participants: CNRS, Sony CSL, INRIA

1.3 Relation with other work packages and tasks

Relation to WP5 tasks: T5.1 Sensors selection and data acquisition rate ; T5.3 3D+t plant segmentation

Relation to other ROMI work packages: WP6 plant modelling

1.4 WebLinks to videos, flyers …

1.5 Dissemination / IPR policy (since the beginning of the project)

Articles in peer-reviewed journals:

- Chaudhury A., and C. Godin, 2020 Skeletonization of Plant Point Cloud Data Using Stochastic Optimization Framework. Front Plant Sci 11: 773. https://doi.org/10.3389/fpls.2020.00773

- Mirande K., Charlaix J., Tisserand M., Besnard F., Godin C., Hétroy-Wheeler F. A Graph-Based Approach for Simultaneous Semantic and Instance Segmentation of Plant 3D Point Clouds (*in prep.*)

Workshops, conferences :

- CVPPP 2018: *Timothée Wintz, David Colliaux, Peter Hanappe.* Automated extraction of phyllotactic traits from Arabidopsis thaliana.

- CVPPP 2020: *A. Chaudhury, F. Boudon, and C. Godin.* 3D plant phenotyping: All you need is labelled point cloud data. In CVPPP ECCV, pages 244–260. Springer, 2020. 1, 2

- CVPPA in ICCV2021: Chaudhury A, Hanappe P., Azaïs R., Godin C., Colliaux D. Transferring PointNet++ Segmentation from Virtual to Real Plants

- 3DV 2021: Pan, H., Hétroy-Wheeler, F., Charlaix, J., & Colliaux, D. (2021, December). Multi-scale space-time registration of growing plants. In *2021 International Conference on 3D Vision (3DV)* (pp. 310-319). IEEE.

- IPPS7 2022 (7th International Plant Phenotyping Symposium): Poster. Besnard Fabrice, Ait Taleb Nabil, Legrand Jonathan, Charlaix Julie, Macé Aurèle, Wintz Timothée, Lalhou Aliénor, Tisserand Marie, Razebassia Olivier, Vernoux Teva, Colliaux David, Godin Christophe and Hanappe Peter. Phenotyping robotics for micro-labs: a low-cost indoor open-source platform for precise phenotyping of plant shoots in 3D.

2 Main body

Introduction

The project name “Robotics for Microfarms” (Romi) highlights our ambition to innovate robotics solutions for plant culture management and phenotyping. This effectively materialises into the three available robots by the end of the project: the Rover, the Cablebot and and the Plant Imager. However, this focus on robotics should not hide an important aspect of our work: developing effective computer vision algorithms. All three robots contain one or several RGB cameras as sensors to image plants at different scales, from a single plant to an entire crop bed. To provide relevant information to farmers and researchers, the images taken by these robots must then be automatically interpreted to extract the exact desired data. For example, the images can give feedback to the robot behaviour (navigation of the rover or path of the weeding tool, movement of the cablebot along the cable, path of the camera arm in the plant Imager): such active vision and adaptive learning has been explored in WP4 and will not be treated here. Another obvious exploitation of the images is to analyse the plants themselves. Two-dimensional images (2D) images enable the rover to discriminate crops from weeds or crops from soil for weeding, and they provide enough data to follow growth of some crops with the Farmer’s Dashboard. However, precise information about the plant architecture or details of particular plant parts need more resolution and this often requires collecting data in three dimensions (3D). In WP5 of the romi project, we focused on developing advanced sensing of plant architecture in 3D, at the scale of single plants or even infra-plant scale, focusing on shoot parts or even organs. Besides, we also worked on coupling space and time by analysing plant 3D architecture over time. All these 3D computer vision algorithms are meant to be used downstream of our precise phenotyping robot, the Romi 3D Plant Imager. This report will review all the algorithms we developed in this context. Many of them are now integrated in our open-source library Plant-3d-Vision and can be applied without efforts in elaborated phenotyping analysis workflows. Whenever it is possible, we will link the described algorithms to their source code and/or integrated implementation in our phenotyping analysis tools. We will also showcase some of the results we obtained, to illustrate the potential applications of our tools.

After a synoptic view of the main tools developed in the course of the project, we will first detail how we generate 3D point clouds from 2D Images. Then we will present three different methods addressing the particularly challenging task of plant segmentation into botanically relevant entities such as stem, leaves, fruit, or flowers, a scale of great importance for both famers and plant scientists. Also, the work achieved to follow the growth of plants in 3D (3D+time) will be presented. Finally we will present how these different tools could be applied in plant phenotyping in the near future.

I. Overview of romi 3D computer vision algorithms: a modular toolbox for plant 3D precision phenotyping

During the course of the Romi project, several computer algorithms have been developed with the aim of phenotyping plant shoots in 3D with high precision (see Figure 1).

Three major steps can be identified, in relation with three types of data: generating a point cloud from 2D RGB images, segmenting a point cloud, and trait phenotyping from a segmented (or clustered) point cloud (e.g. counting, measuring). All the algorithms can be used on their own with the correct input data or combined and linked in integrated pipelines. This modularity increases the genericity of our work because all these open source algorithms are not restricted to a particular phenotyping robot or a particular trait phenotyping. Indeed, RGB images around a plant can be obtained from different procedures (including manually), point clouds are a very generic data type in computer vision and can be generated directly or indirectly from a wide array of sensors, segmenting a plant into different subpart is a compulsory step to obtain many valuable phenotyping traits. In the context of Romi, we often used the measure of phyllotaxis[1] as a proof-of-concept for trait measurement. Beyond the value of phyllotaxis for fundamental research, the methodology developed to analyse this trait can be reused for many other applications, because it captures the precise position and orientation of each different organ (maily fruits here) along a branch.

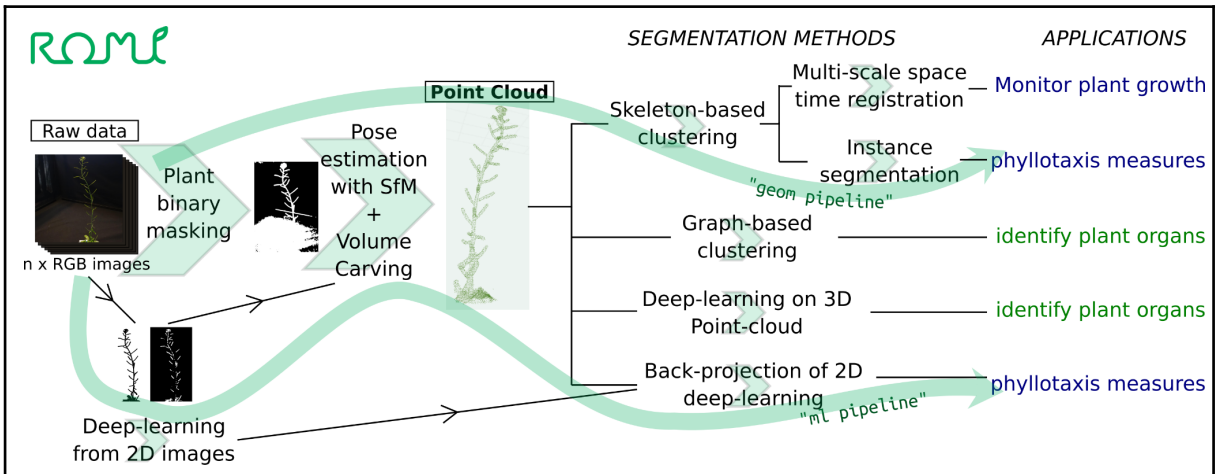

Figure 1: **Overview of the different computer vision methods developed in the romi project for plant precision phenotyping**. See text for details. Colour code in *APPLICATIONS*: blue quantitative, green qualitative phenotyping. Green wavy arrows indicate two pre-built pipelines available in the Plant-3d-Vision library.

The different algorithms pictured in Figure 1 and summarised below will be described in the present document:

- The generation of 3D point clouds from series of 2D RGB images around the plant (section II)

[1] See other Romi documents for details (e.g. D5.3, p. 4). Phyllotaxis is the geometrical pattern created by the distribution of plant organs along branches (leaves, branches, flowers or fruits). It can be characterised by the measure of divergence angles and internode length between the insertion points (ie. node) of successive organs along an axis.

- Various **segmentation methods** that have been developed in relation with different application perspectives:
    - A skeleton-based method to measure phyllotaxis (section III). This algorithm is computationally very efficient. However, it does not provide advanced semantic labels but only an organ instance segmentation: there are only a stem label and "organs" labels (where all organ types are mixed). This method is incorporated in the "**geometric pipeline**" of the Plant-3d-Vision library.
    - A deep-learning segmentation using 2D images to measure phyllotaxis (section V.A). Although more resource intensive, this method provides true biological semantics learned from the supervised training phase. This method is currently incorporated in the "**machine-learning pipeline**" of the Plant-3d-Vision library.
    - A graph-based clustering of the point cloud to identify plant organs (section IV). This method bypasses the need for a skeleton and provides a richer semantics than the skeleton-based genometric approach (although not as rich as deep-learning), especially for organs with very contrasted shapes (e.g. flat versus rod organs). Prior botanical knowledge can be incorporated to correct biological aberrations in the plant architecture. Although this has not been tested yet, this method could also be used in the future to measure phyllotaxis.
    - Another deep-learning approach using directly 3D point clouds (section V.B). Computing this deep-learning method is also resource intensive, but it has a strong potential to detect a wide range of labels. Also, since it works directly from point clouds, it could be applied to data coming from other phenotyping devices (without 2D RGB images). Although this has not been tested yet, this method could also be used in the future to measure phyllotaxis.
    - A skeleton-based method to register point clouds in space and time (section VI). This algorithm does not provide semantic labels with biological meaning. However, the segmentation of the plant is used to precisely follow its deformation and growth through time series.
- Algorithms to compute divergence angles and internode length from segmented point clouds. This method has been described once with the geometric pipeline, but it could be branched in theory at the end of any previous segmentation methods described above (section III)

II. Producing 3D point clouds, the entry point of all downstream analysis

A. Why focusing on 3D point clouds ?

3D data is incomparable to 2D data when precise phenotyping is required, especially when the observed sample has large and variable depth, or occlusion of internal structures. Plants have evolved particular shapes and architectures that make 3D vision often essential (Li et al., 2021). Indeed, flat and sometimes large leaves (a shape optimal for photosynthesis) generate a lot of occlusions. Besides, as sessile organisms exploring their environment in all directions, plants display pervasive

radial symmetry and regular distribution of organs around their axis. This often prevents finding a single side or view point that would capture most of the desired information.

Hence, working with 2D RGB cameras requires acquiring several images all around the plant and to reconstruct *a posteriori* a 3D object. This 3D object could be a mesh or a point cloud. However, point clouds are also an output format of other technologies, such as Light Detection and Ranging (Lidar), Structured Light and Time-of-Flight (ToF), or binocular stereo vision (Paulus, 2019). As a consequence, many computer vision algorithms have been developed to take point clouds as input data for further downstream analysis.

In the Romi project, we identified point clouds as a standard interoperable format for 3D analysis. We developed a program to efficiently reconstruct 3D point clouds from a series of 2D images, corresponding to our phenotyping set-up (the Plant Imager). Then, all our segmentation algorithms start from point clouds as entry points.

B. The Romi way: Generating scaled 3D point clouds from series of 2D images (mixing SfM and volume carving algorithms)

*Work performed by: Tim Wintz, David Colliaux and P. Hanpape. This work was published in **CVPPP 2018** (Wintz et al., 2018).*

The processing pipeline consists of multiple steps: pose estimation, removal of image distortion, 2D binary segmentation, computation of the visual hull, and the computation of a point cloud. In practice, all these steps have been defined as “Tasks” in our Plant-3d-Vision library. Hence, they can be used as functional units and combined in analysis pipelines, as we did in our two main pre-built pipelines (geometric and machine-learning). In the following, we also provide the name of the corresponding Task implemented in Plant-3d-Vision (hereafter coined as **P3dV Task**), which allows automatization of the reconstruction pipeline.

Pose estimation (P3dV Task = Colmap)
To map points in space to their location in each individual picture, one has to know the precise intrinsic camera models as well as the exact position of cameras in space. However, it can be challenging to perfectly calibrate a camera arm from motor positions. This is especially true in the Plant Imager where the arm holding the camera moves thanks to off-the-shelf stepper motors and their absolute position cannot be taken for granted. Instead of investing in a tedious calibration procedure of our acquisition system, we obtain camera poses using a structure-from-motion (SfM) algorithm. We used the open source Colmap (Schönberger and Frahm, 2016; Schönberger et al., 2016). Structure-from-motion is a well known algorithm that allows us to automatically obtain camera poses from a set of RGB pictures, up to a scaling of the world. The scaling coefficient is then obtained either approximately using the translation parameter of the camera arm, or more precisely using a custom calibration procedure (see RP2 and RP3 for WP5 as well as in our online documentation). SfM relies on feature matching and does not work well in textureless pictures. Therefore, we optimised the addition of texture elements in the scene to provide enough matching for Colmap without perturbing the subsequent steps of plant reconstruction (see RP3 for WP5). From this SfM pipeline, we extract camera poses (camera extrinsics), the camera’s fundamental matrix (or camera intrinsics), and distortion coefficients that are used to undistort the images. From these parameters, a projection model is constructed where any point in space can be projected onto all of the pictures.

Correction of image distortions (P3dV Task = Undistorted)
Using the camera intrinsics computed above by Colmap, we correct possible deformations of the

images created by lens optical aberrations. We integrated an existing tool from the open source opencv library (Bradski, 2000).

2D binary segmentation (P3dV Task = Masks)

The aim of this step is to separate the plant from the rest in each image at the pixel level. In the context of the indoor Plant Imager, we made the task easier by choosing a black background with dimmable light to enhance the contrast of the plant. But this task could become quite challenging outdoors. Therefore, to provide more flexibility to different imaging contexts, we implemented two methods in Plant-3d-Vision: "Excess Green Index" and "Linear SVM".

Briefly, the "Excess Green Index" method assumes the plant is green and the background is not. We compute the normalised RGB values x ∈ r,g,b for each pixel (i,j):

$$xij = Xij/(Rij + Gij + Bij)$$

where X∈R,G,B is the red, green or blue image.

Then, the green excess index is computed as:

$$ExG = 2g - r - b$$

In the "Linear SVM" method, a linear combination of R, G and B is used to compute the index S for pixel (i,j), using a weight vector w for the three colours w=(w0, w1, w2):

$$Sij = w0.Rij + w1.Gij + w2.Bij$$

A simple vector, like w=(0,1,0) may be used for example to give more weight to the green.

The advantage of the "Excess Green Index" method is to be parameter-free. However, we observed that it was less robust than the linear SVM, even when this latter was configured with a simple weight parameter vector like w=(0,1,0) (as exemplified in our documentation). In addition, weights could be learned from the training of a SVM, provided that ground truth binary masks are available. This property could help this step be more robust to varying imaging conditions, like uncontrolled light or inhomogeneous background. So far, we run all our analysis with the default w=(0,1,0) with satisfactory results.

It should be noted that the masking step is very sensitive to two other parameters. First, the threshold is the value above which the final index will convert the corresponding pixel into white: too high values can miss parts of the plants while low high values bring false positive pixels. Second, pixels of the initial image are dilated to avoid missing very thin parts of the plants. This step is performed using an existing tool of the scikit-image open library (Van der Walt et al., 2014). Adjusting correctly these parameters is crucial because all the subsequent steps rely on the correct segmentation of the plant on the 2D images.

Space carving (P3dV Task = Voxels)

The visual hull is a classical tool of 3D reconstruction from 2D images (Laurentini, 1994). For a given set of 2D views of an object, it is defined as the set of all 3D points whose projection in all views is inside the object. This volume contains all object points, and is in general a good approximation of the object shape if the number of views is sufficient. So far, all the plants of interest imaged during the ROMI project are tree-like objects: the visual hull is expected to be very close to the true shape of the object for a circular set of views around it. To compute the visual hull, we use a classical "Volume carving" approach (Kutulakos and Seitz, 1999) similar to the one in (Potmesil, 1987). This algorithm

computes an octree in which each node is either labelled as inside, outside, or in between the visual hull. The nodes in the tree are split until either their size is smaller than the requested precision or the bounding region is either completely inside or outside the visual hull.

Algorithm 1 Space carving algorithm

function SPACECARVING($V, \Omega, B_1, \pi_1, B_2, \pi_2, \ldots, B_N, \pi_N$)
▷ V is a set of voxels, each having a value and a center. ▷ B_i is a mask indicating background $B_i(x) == 1$ iff pixel x is classified as background. ▷ π_i is the projection from world coordinates to pixel coordinates. ▷ $\Omega = [0, w] \times [0, h]$ is the image domain.
  for $v \in V$ **do** v.value $\leftarrow$ unseen
  end for
  for $i \in 1, \cdots, N$ **do**
    for $v \in V$ **do**
      if $\pi_i(v) \in \Omega$ **then**
        if $B_1[\pi_i(v)] == 1$ **then** v.value $\leftarrow$ background.
        else if v.value $==$ unseen **then** v.value $\leftarrow$ seen.
        end if
      end if
    end for
  end for
end function

Point cloud with the level set method (P3dV Task = PointCloud)

To compute a point cloud from this volume, a level-set method is used (Sethian, 1998), as described in algorithm 2. The signed distance function needed for this algorithm is computed using a fast marching algorithm implemented in SciPy (Virtanen et al., 2020). This yields a set of points located on the surface of the interface between the inside and the outside of the visual hull.

Algorithm 2 From voxel grid to point cloud

function POINTCLOUD(V)
  $D \leftarrow$ SIGNEDDISTANCE$(V == 1)$
  $\nabla D \leftarrow$ GRADIENT(D)
  $points \leftarrow []$
  $normals \leftarrow []$
  for $v \in V : |D[v]| < \sqrt{2}/2$ **do**
▷ Loop on all boundary points.
    $n \leftarrow \nabla D[v] / \|\nabla D[v]\|$
    $v_1 \leftarrow v + D[v] * n$
    APPEND$(points, v_1)$
    APPEND$(normals, n)$
  end for
  return $points, normals$
end function

Results

Using this technique, we are now able to reconstruct point clouds of a variety of plants imaged in the Plant Imager (see D7.4). The resolution is relatively high, since typical thin structures of *Arabidopsis thaliana* (hereafter shortened as *Arabidopsis*) plants like floral peduncle (average diameter = 0.5 µm) can be reconstructed. We implemented an advanced integrated computational workflow, since the five steps described above can be automatically computed and piped using a single command line of the plant-3d-vision library, with simple configuration files allowing the users to tune the main parameters as needed. Usability is also reinforced by distributable docker images, dispensing end users with cumbersome installations and ensuring reproducible results. Parallel computing of several steps (Colmap, Voxels, PointCloud) optimises the performances and computation time. Current limitations concern the maximum plant height and volume, as well as possible occlusions. We are addressing these issues with the new improved version of the Romi Plant Imager: a custom robotic arm now holds two Pi Cameras, allowing in theory a wider z-range for imaging while limiting occlusions.

In the following sections, we will describe the different downstream applications we developed using these high quality point clouds as input data.

III. Using Skeletons to segment plant's part and measure phyllotaxis

A skeleton is a thin structure obtained from an object that encodes the topology and basic geometry of the object. As a compact representation of the original object, the skeleton implies less computational overhead than working with the original point cloud data. This makes skeletons very popular in computer vision (Cornea et al., 2007). Many applications of plant 3D phenotyping require skeleton structure of the input point cloud data as a prior for further processing: plant organ segmentation (Ziamtsov and Navlakha, 2019), robotic branch pruning (Chattopadhyay et al., 2016), automated growth analysis (Chaudhury et al., 2019), etc. In this section, we will present the algorithms we developed to compute skeletons from point clouds, how this skeleton is used for plant segmentation and how we derive phyllotaxis measures. As before, In the following, the names of the corresponding Plant-3d-Vision Task (**P3dV Task =**) are indicated.

A. Skeletonization Algorithms and relation to trait measurement

*Work performed by: Tim Wintz, David Colliaux and P. Hanappe. This work was published in **CVPPP 2018** (Wintz et al., 2018).*

Building the skeleton (P3dV Tasks = TriangleMesh + Curve Skeleton)

The construction of the skeleton is a crucial step to be able to detect the main stem and the fruits. There are multiple possible algorithms for the skeletonization of a point cloud, we tested the following (ordered by their date of publication):

1. A method using a ***kNN graph*** and a ***clustering algorithm***, specifically adapted for trees (OpenAlea implementation) (Xu et al., 2007). We also tested the improvements on this algorithm as proposed in (Chaudhury et al., 2019), described below.
2. ***Laplacian based contraction*** method as described in Cao et al. (2010).
3. ***Mean curvature flow*** skeleton as described in Tagliasacchi et al. (2012) and implemented in open software CGAL (The CGAL Project, 2022). Note that this method requires a triangulated surface mesh as input, so it must be preceded by a conversion of the point cloud to a surface mesh. We tested two implementations of the Poisson surface reconstruction algorithm

(Kazhdan et al., 2006) to compute this surface mesh from a point cloud with normals, one in CGAL and the other in the open software Open3D (Zhou et al., 2018).

4. ***L1-medial skeleton*** as described in (Huang et al., 2013).

We also combined some of these methods (for example L1-medial skeleton algorithm applied on the output of the Laplacian based contraction). Although none of these methods give perfect results on all the data we tested, we found the mean curvature flow skeleton to be the most robust. In the figure below, we show the result of this algorithm as well as of the combination of 2 and 4.

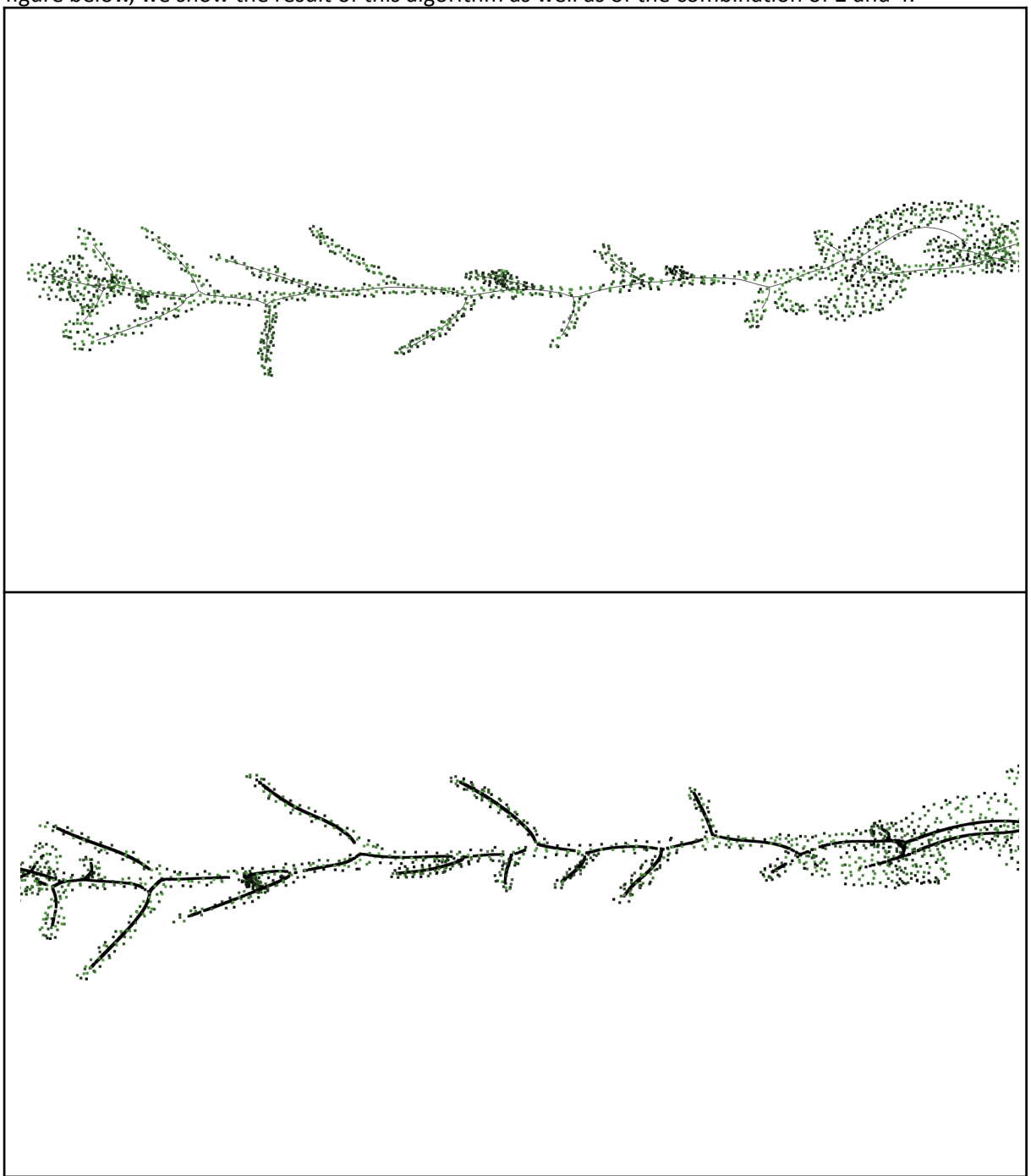

Figure 2: **Example skeletons computed from the point cloud of an *Arabidopsis* main inflorescence stem**. Skeleton obtained with the Mean flow curvature method (top) and a combination of Laplacian based contraction and L1 medial skeleton (Bottom). Leaves at the base of the stem (on the right side here) perturb the skeleton in both cases. The lower skeleton is interrupted at several

sites, which can cause several issues in downstream segmentation of organs.

The output of this step is a curve skeleton.

Segmentation (P3dV Task = TreeGraph)
Curve skeletons are used for many 3D objects, e.g. animals point clouds. In the case of plant architecture, they must be converted into a **rooted tree graph** to provide a meaningful interpretation leading to plant segmentation. A graph is just defined by a set of vertices connected by edges, a tree graph is a very restricted type of graph: it is acyclic, it contains only one connected component, and there exists a unique vertex called the root that has no parent. This root will be the base of the stem.

In the context of the Romi project, we adapted our algorithms to our phenotyping scenario: measuring the phyllotaxis of the main inflorescence stem of *Arabidopsis*. First, this plant has a particular geometry with very few leaves (only some narrow curled ones at the base) and mostly cylindrical structures (the stem, branches, fruits and their pedicels). The typical morphology of *Arabidopsis* fruits (small straight rods) thus creates simple branching patterns of only on order (or descendant) all along the stem. Second, plants are imaged in their natural up-right position, where the main stems are monotonically going upwards: the base of the stem at the lowest position and the tip at the highest.
Based on this observation, we derived the following algorithm to segment the curve skeleton. We used the Python package Networkx (Hagberg et al., 2008) to perform all the graph-related operations. First, the skeleton is converted into a graph and the root is chosen as the node with the lowest position on the vertical z-axis. Then, using Dijkstra's method (Dijkstra, 1959), the main stem is computed as the shortest path to the point furthest from the root and all other nodes are ordered relative to their root-to-shoot position along the main stem. The final structure of the tree is obtained by computing a minimum spanning tree (MST), with Edmond's algorithm (Chu, 1965): the organ vertices are the weakly connected components of the MST in which we have removed the stem.

Measuring phyllotaxis (P3dV Task = AnglesAndInternodes)
Each organ connects to a stem at the node. Measuring phyllotaxis implies measuring between each pair of successive organs along the stem the divergence angle (measured in the plane orthogonal to the main stem) and the internode length. We designed a method to estimate these two parameters even in bent stems, which often occurs because of gravity or oriented growth (because of tropisms or mechanical constraints). This robustness makes the use of tutors dispensable if the stem is thick enough to hold itself. Therefore, we introduce the following method of angle estimation.
We denote the set of points of the *i*th organ from the bottom of the plant by Fi , for $1 \leq i \leq N_f$, where N_f is the number of organs. Let us assume that the stem points lie on a one-dimensional submanifold of $\mathbb{R}^3$, contained inside an euclidean plane $\mathcal{P}$. For any point p of the stem, let us denote the unit upwards tangent vector at p by tp. Both $\mathcal{P}$ and t_p can be approximated using a principal component analysis (PCA). Let $\mathcal{P}'$and t_p' be such approximations of $\mathcal{P}$ and t_p respectively. Let us fix an arbitrary orientation for $\mathcal{P}'$. We then define the following frame at p for any point p of the stem: let u_p be the orthogonal projection of t_p' onto $\mathcal{P}'$. Let us complete (u_p) into a direct orthonormal basis (u_p, v_p) of $\mathcal{P}'$. Finally, let us complete (u_p, v_p) into a direct orthonormal basis (u_p ,v_p ,w_p) of $\mathbb{R}^3$. For i ∈ {1,⋯ ,N_f}, let us define the following projection:

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{1}{N_i} \sum_{j=1}^{N_i} \begin{pmatrix} v_p x_j - p \\ w_p y_j - q \end{pmatrix},$$

where {(x_j , y_j), $1 \leq j \leq N_i$} = F_i and (p,q) is the branching point between the organ F_i and the stem. We then define α_i , the angle for the *i*th organ, as the angle between (1,0) and (x, y):

$$\alpha i := \ arctan2(y, \ x)$$

Internode length can simply be computed as the distance in the graph between successive branching nodes in the graph along the stem. Other quantities such as organ length and angle between organ and stem can also be easily computed from the analysis, but are not included in the evaluation.

All together, we coined this skeleton-based segmentation from point cloud the "***geometric pipeline***" (which include all steps from the raw images and Point Cloud generation). This pipeline (hereafter called the "geom pipeline") corresponds to an improved version of our previous work presented at CVPPP 2018 (Wintz et al., 2018).

Result & evaluation of the geometric pipeline
Work performed by F. Besnard.

This evaluation was detailed in the progress report RP2. It was made from twelve real *Arabidopsis* main inflorescences, for which a manual measurement of phyllotaxis was also performed as ground truth. Here, we summarise the main results:

- The point cloud is visually accurate and overlay precisely the plant contour on each images ;
- The mesh is also visually accurate but is very sensitive to the dilation factor: the mesh is absent from the too thin parts of the point cloud ;
- The skeleton is perfectable. It generally captures the main stem but contains some errors: cycles (often associated with leaves curling back on the stem), zigzag deformations at branching sites. The root is sometimes confused with the tip of the lowermost organs. The skeleton is very sensitive to the bounding box: when too large, other elements (e.g. other parts on the plant that are on the image) than the main stem can be reconstructed and the skeleton can be deviated to such structures.
- The segmentation is the most problematic part. Under- or over-segmentation often occurs (37% errors on average), with a variable frequency depending on the organ type. Leaves associated with branches are causing several errors. This can be explained by the fact that they create a more complex branching pattern that was not modelled in the tree graph. The tiny tip of the inflorescence where flowers are densely packed also concentrates many errors, which may indicate the resolution limit of our phenotyping ;
- When they are not perturbed by segmentation errors, the automated measures faithfully reproduce the natural variation of the phyllotaxis but their precision could be improved compared to the current manual standard.

Table 1: The segmentation results of the geometric pipeline.

type of organ pair	Organ pairs in ground truth measurements		**correct segmentation** (the two organs of the pair are correctly identified)	
	number	percent	number	percent of expected (**success**)
branch to branch	35	11 %	13	58 %
branch to flower	12	4 %	5	37 %
fruit to fruit	263	85 %	178	68 %
TOTAL	310	100 %	196	63%

As a conclusion, our evaluation highlighted the point cloud quality, which validates the upstream part of this 3D reconstruction and analysis (section II above). However, it suggested several areas for achieving a better segmentation: either improving the skeleton or radically changing the method to take into account the type of organs (leaves, fruit, flowers). In the next section, we will first describe the first solution to compute better skeletons.

This evaluation also pointed to the need for automated evaluation tools: this time-consuming task cannot be repeated at each incremental improvement of the pipeline or to efficiently explore new algorithms. This motivated the development of several metrics, dedicated scripts and new algorithms (e.g. sm-dtw, see D6.5), as well as the creation of virtual testing environment with synthetic ground truths (Virtual plants and the Virtual Plant Imager -see RP2, synthetic phyllotaxis data and error simulator, see D6.5)

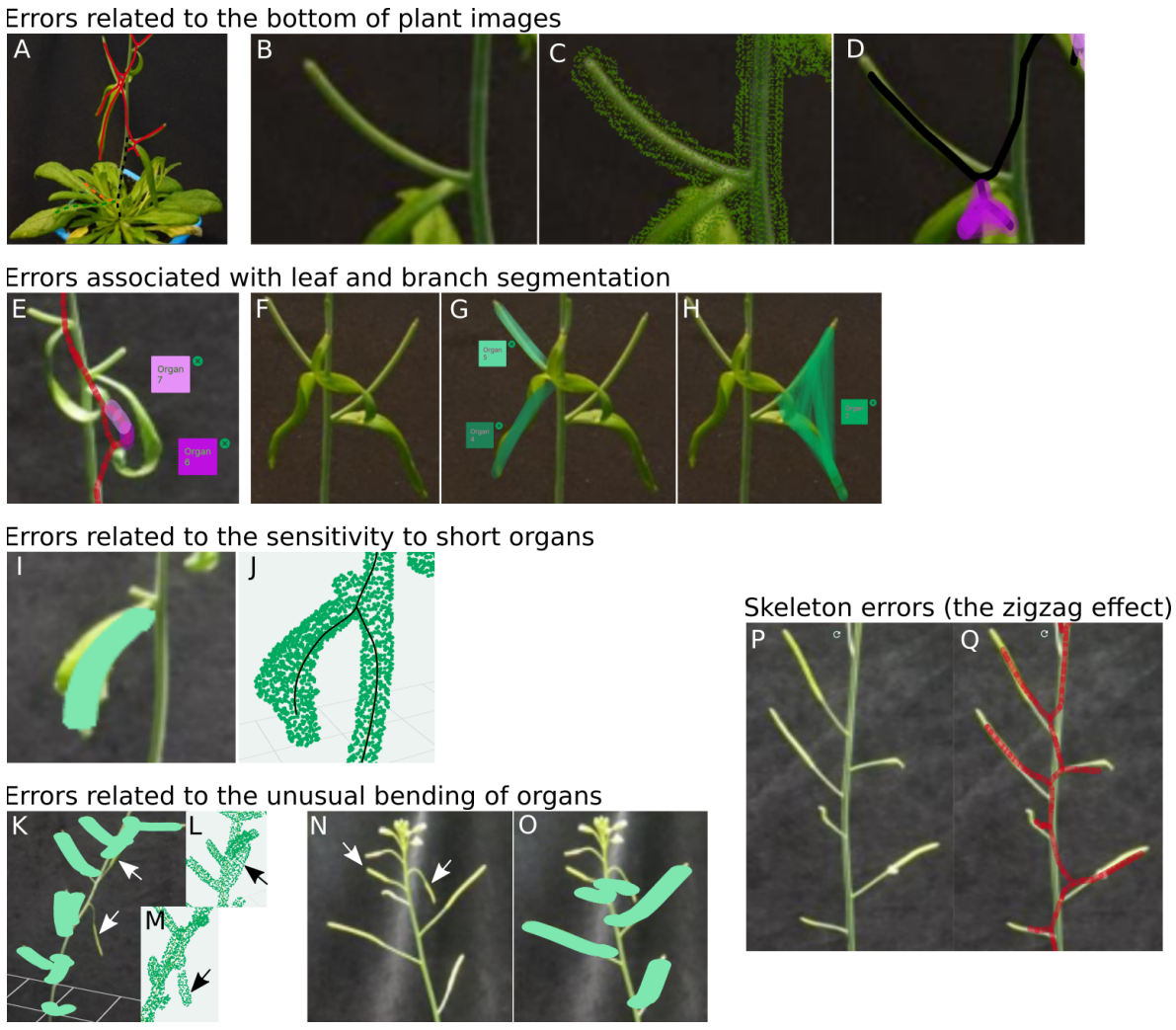

Figure 3: **Gallery of typical errors recurrently found in the results of the geometric pipeline.** The phenotyping task consists in the segmentation of an *Arabidopsis* inflorescence. (A-C) Errors related to the detection of the base of the plant: (A) the base of the inflorescence (dashed lines: black=stem, orange=branch, green=leaf) is not segmented and confounded with the rosette (red line: automated skeleton). (B-C) This basal branch (B) is detected (C, enclosed in the point cloud) but interpreted as the base of the inflorescence main stem (D, black line=automated skeleton,

purple=organ).(E-F) Leaf and branches are causing several errors. (E) curled leaves crossing the main stem generate inaccurate leaf segmentation and skeleton loops (dark and light purple=organs; red=automated skeleton). The (leaf-branch) pairs in F are either segmented as two equivalent organs (G, inadapted for phyllotaxis measurement) or as a single fused organ (H, inaccurate botanical semantics, potential precision errors for phyllotaxis). (I-J) Branches pruned too short (I) are detected in the point cloud (J, green dots) but not identified by the skeleton (black line). (K) Fruits bending too much downwards or upwards (white arrows) are not detected as organs (green highlight), although they are in the point cloud (L,M, dark arrows). (P-Q) zigzag of the automated skeleton (Q, red line) at each fruit insertion region. Snapshots taken from the Plant-3d-Explorer.

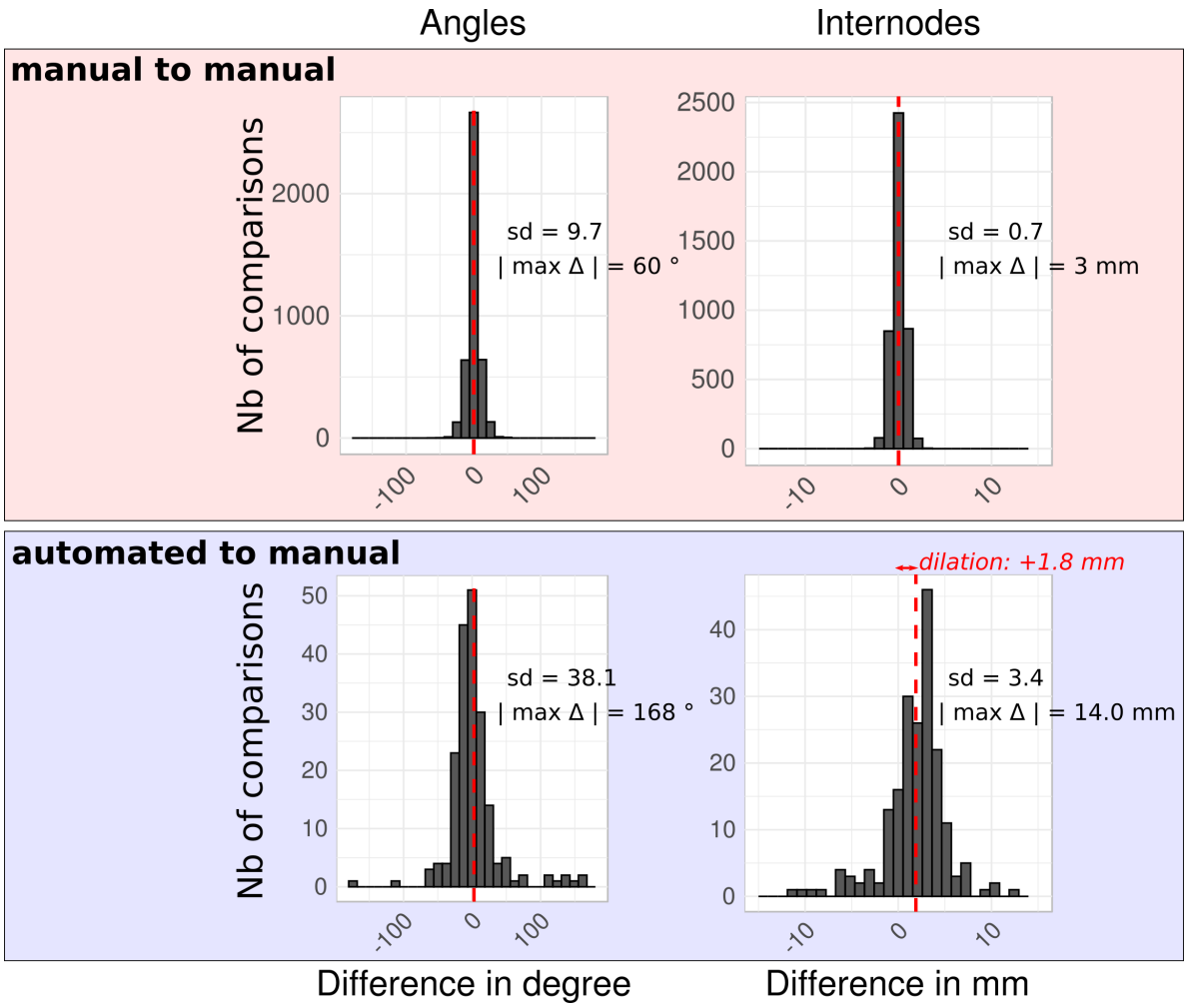

Figure 4: **assessment of geometric pipeline precision**. The measures are divergence angles and internode lengths measured on the main inflorescence of *A. thaliana* plants (left and right plots, respectively). Pipeline version: v0.4. See text for method details. Top row (red shade): the distribution of pairwise differences between several independent manual measures of the same biological sample (3 different plants, total of 126 measures replicated up to 10 times): the spread (sd) quantifies the unbiased (centred to zero) uncertainty inherent to the manual technique. Bottom row (blue shade): the distribution of differences between automated and manual measures in the evaluation database for intervals without segmentation errors (n=196, see table 1). Larger spread (sd) for both angles and internodes indicate that automated measures are less

precise than manual measures. A dilation is also generated for lengths: the source has been identified and can now be corrected (see external calibration in the progress report of WP5). Vertical red dashed line: average value (arithmetic mean) of the distribution.

B. Improving the realism of skeletons

This work was published:

Chaudhury A., and C. Godin, 2020 Skeletonization of Plant Point Cloud Data Using Stochastic Optimization Framework. Front Plant Sci 11: 773. https://doi.org/10.3389/fpls.2020.00773

Ideally, the skeleton computed above should follow the exact centerline of the point cloud. Hence, within a local neighbourhood, the distance from each skeleton point to the enclosing shape boundaries should be the same. The first evaluations of our pipeline revealed obvious defects of the skeleton, including missing organs, oversegmentation, loops or zigzagging of the main stem. In **zigzag**, the skeleton clearly departs from the centerline of the point cloud. In the case of phyllotaxis, these defects directly impact the results of measure (erroneous phyllotactic sequence ; reduced precision in angle or internode estimation). Existing approaches all generate the type of defects we observed, pointing to a lack of realism and biological relevance. This problem is of generic importance given the widespread use of **skeleton extraction from 3D plant point clouds in a myriad of phenotyping studies**.

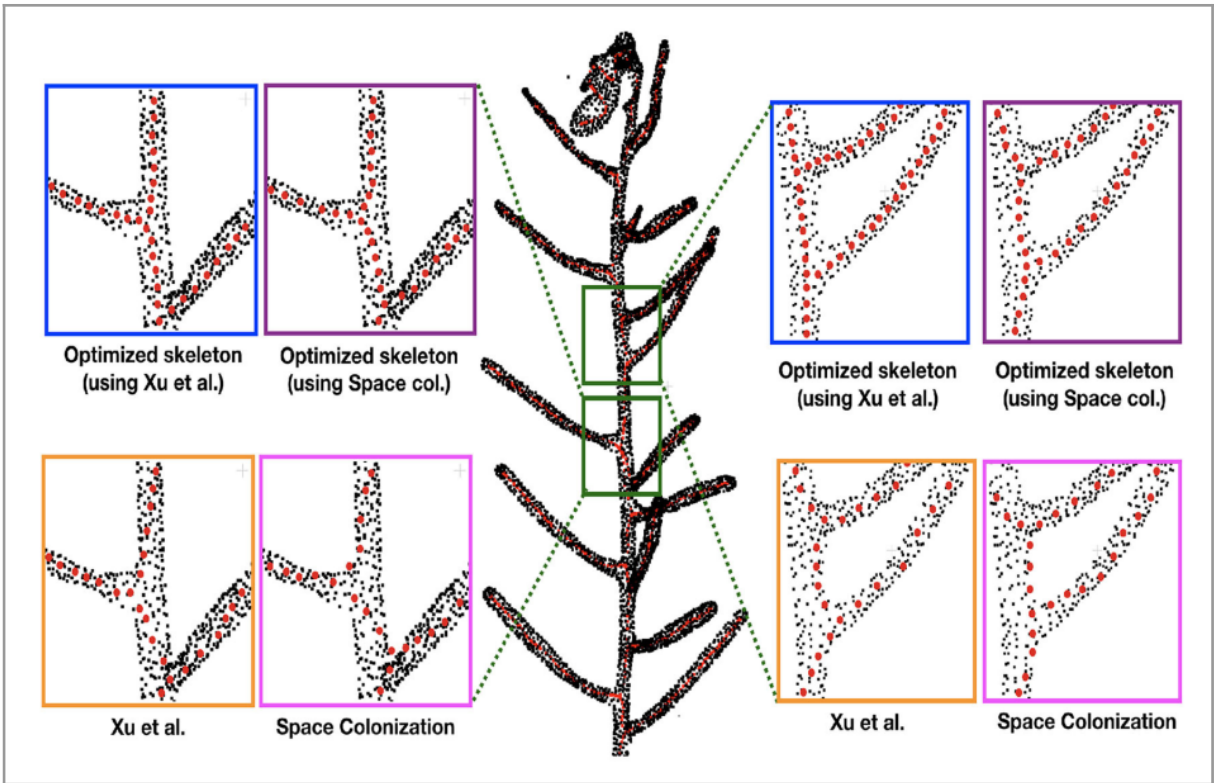

Figure 5: **Optimization of skeleton realism from the point cloud of an *Arabidopsis* inflorescence**. Bottom rectangles: Initial skeleton (red dot) is computed by two different algorithms (Xu et al., or Space colonization) and two different regions are zoomed for better visualization. Upper rectangles: correction provided by optimization algorithm: zigzag at branching is reduced and skeleton outside the point cloud no longer exists. Adapted from Chaudhury et al., 2020 (Fig.3).

To **improve existing state-of-the-art skeleton structures**, we generated a stochastic framework which is supported by the biological structure of the original plant. Compared to the previous curve skeleton generated in the previous section, we introduced the notion of **β-splines** to form a **curve tree** defined as a finite set of curves joined in a tree topology with a certain level of smoothness. In the next phase, **we force the discrete points in the curve tree to move toward the original point cloud** by treating each point in the curve tree as a centre of Gaussian, and points in the input cloud data as observations from the Gaussians. The task is to find the correct locations of the Gaussian centroids by maximizing a likelihood. The **optimization technique** is iterative and is based on the **Expectation Maximization (EM) algorithm**. The E-step estimates which Gaussian the observed point cloud was sampled from, and the M-step maximises the negative log-likelihood that the observed points were sampled from the Gaussian Mixture Model (GMM) with respect to the model parameters. To test our method, we first used three real world datasets with very different scales: two containing point clouds of tree plants (cherry and apple trees (Tabb and Medeiros, 2018) and [2]) and the real *Arabidopsis* inflorescence stems acquired with our romi Plant Imager (Fig. 5). We also quantified our result with these different datasets by generating ground truth of junction points of the branches and we demonstrated the robustness of this approach over the state-of-the-art

In parallel to this work dedicated to improving skeletonization, we also developed two skeleton-free approaches to address plant segmentation.

IV. Deriving both semantic and instance segmentation from a geometric and topological analysis of point clouds (spectral clustering and quotient graph)

This work is about to be submitted as the following article:
A Graph-Based Approach for Simultaneous Semantic and Instance Segmentation of Plant 3D Point Clouds. *Authors: Katia Mirande, Charlaix Julie, Tisserand Marie, Besnard Fabrice, Christophe Godin, and Franck Hétroy-Wheeler*
(Paper in preparation, link**: https://seafile.unistra.fr/f/b53015cc596a443eabc8/***)*

In the first geometric method described above, each organ of the plant is detected based on the local geometry of the point cloud. This approach has several caveats. First, the semantic of the segmentation is reduced to only two labels: main stem versus organs. Hence, the diversity of different organs like fruits or leaves cannot be captured. Second, the consistency of the global structure of the plant is not checked, compared to the real natural architecture corresponding to a particular plant species.
As an alternative, we developed a more elaborated, two-graph approach for the automatic, fast and accurate segmentation of a plant into each of its organs with structural guarantees. In addition, to avoid local optimization only adapted to the peculiar architecture of *Arabidopsis* inflorescences, we used another reference plant: *Chenopodium album* (hereafter coined *Chenopodium*) and we also tested our method on tomato seedlings (see below). We selected *Chenopodium* at the origin of the romi project because it is a common weed for market gardeners. Interestingly, its development strongly differs from *Arabidopsis*: it does not make a rosette but produces an elevated stem directly after germination and has simple leaves with large limbs, especially in young seedlings.
The segmentation starts from any point cloud in entry (such as the ones generated in section II) and proceeds through three successive steps (Figure 6):

- a neighbourhood graph of the points to compute local geometric features. This allows distinguishing between linear organs (main stem, branches, petioles) and two-dimensional

[2] (2010). PlantScan3D. Available online at: https://gforge.inria.fr/frs/?group_id= 2512 (accessed January 2020).

ones (limbs, apices).

- a quotient graph to connect each organ to its neighbours. This allows both refining the labelling of the organs and checking the overall consistency of the segmentation.
- a refinement loop allowing correcting segmentation defects.

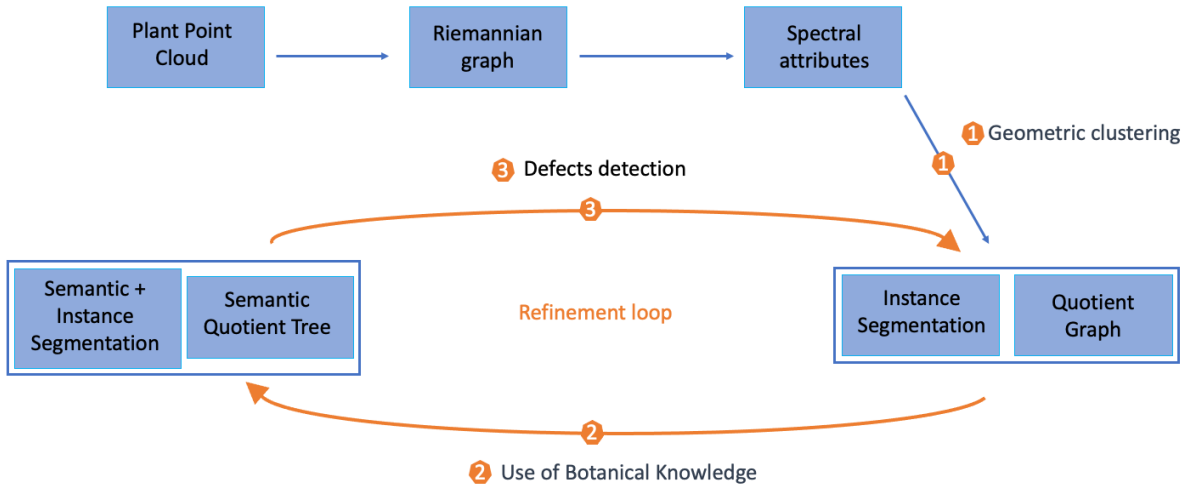

Figure 6: General principle of the proposed method. See text for details

From the 3D point cloud we first reconstruct a graph, we can be defined as a set of a set of vertices and edges. We then use spectral clustering to compute geometric attributes, because it is a well-defined algebraic graph clustering method which does not infer any particular geometry to the data and it is less sensitive to noise and neighbourhood selection than other clustering methods. When it comes to plants, spectral approaches have already given promising results in the literature, and can handle large data with uneven density (Hétroy-Wheeler et al., 2016). Hence, we compute the Laplacian of the point cloud graph and finally, following Hétroy-Wheeler *et al.* 2016, we compute the eigen-decomposition of the Laplacian. We tried to generate two different graphs: a classical Riemannian graph (representing 3D points as vertices and connecting them by edges to their nearest neighbours) and a surface mesh (based on triangles primitives covering the point cloud surface). The discrete Laplacian of the surface mesh was approximated using cotangent weights (Sorkine, 2005). Both graphs give similar results. However, we show that the gradient of the vertex values corresponding to the second eigen-vector (known as the Fiedler vector) provides a number of key features characterising the different organs. Indeed, the Fiedler eigenvector basically gives a value to every node according to the longest axis of the graph. In the case of plants, it thus ordinates the points along the main stem as it is the plant's longest organ. Besides, we discovered a more interesting property: abrupt changes in the norms and directions of the Fiedler gradient are locally associated with botanical structures. We thus designed simple methods based on K-means clustering to automatically detect apices and limbs with the norm of the fiedler gradient and to discriminate linear organs like main stem, branches and petioles with the direction of this vector.

Tree quotient graph: building a global tree architecture

However, this first spectral analysis and clustering cannot segment all plant ontological elements (leaf, internode, fruits) as different instances. We next build a quotient graph, where nodes represent connected components, and edges their adjacency relations inferred from the adjacency relation of their vertex components. We additionally define a root node on the quotient graph (either automatically defined as the lowest component of the plant or manually selected by the user). This produces a much **smaller graph** which reflects the structure of the riemannian graph, at a

macroscopic scale (Sanders and Schulz, 2011). We refined the tree quotient graph structure by merging neighbouring components that share similar directions for the gradient of the Fiedler vector. To do this end, we associated an energy to each edge of the connected components (or nodes) of the quotient graph based on the mean of their gradient direction. If the edge-associated energy exceeds a heuristic threshold, corresponding nodes are fused iteratively. This quotient graph provides a first rough representation of the plant structure at the organ level since the initial point cloud is segmented into instances corresponding to either a linear organ (main stem, branch, petiole) or a two-dimensional organ (limb, apex) of the plant (figure 7a).

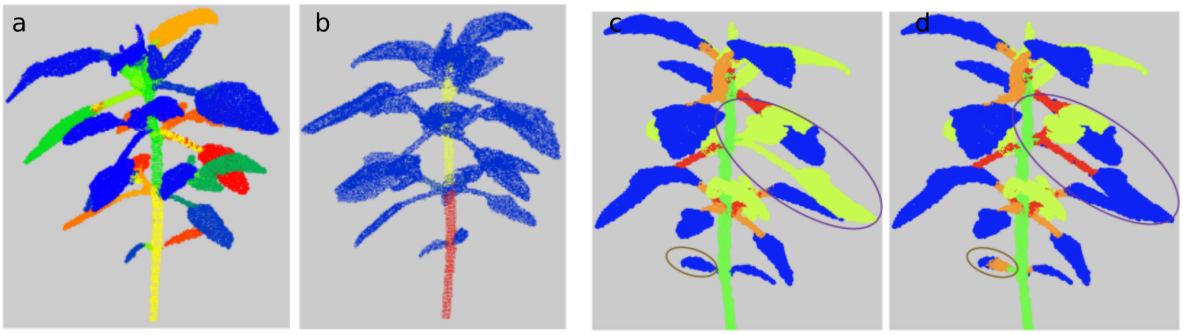

Figure 7: **From instance to a semantic segmentation and its correction (exemple in a *Chenopodium* plant)** a) instance segmentation obtained at the end of the stage 1 (see figure 6) of our algorithm. b) Determination of the main stem using the quotient graph (blue-to-red color-code corresponds to the *load* on each node, which basically represents the number of leaves supported by the corresponding node): the main stem is the path that accumulates the highest load among the shortest paths from apex nodes to the root node (that is manually defined). c and d) Correction by the refinement loop. The purple circle shows an ensemble of a leaf and an apex and its branch that were classified in the semantic class apex. The brown circle shows a cotyledon. As it is not detected differently than the limbs, the algorithm considered that the limb was directly connected to the main stem and clustered it again.

Refinement loop: correcting segmentation defects

In the resulting tree graph, there is no guarantee that the connections between neighbouring instances are botanically correct, as they are purely geometry-based. To check the plausibility of these connections, we designed a refinement loop that uses botanical knowledge about the plant on the quotient graph. Briefly, the relations between the clusters (nodes) of the quotient graph will be used to deduce a semantics related to botany (general and common structures like stem, leaf, branch, etc…) and define a rooted tree with a main axis and an apex (figure 7b). Prior knowledge about the architecture of the plant studied can include potential specificity (e.g simple versus compound leaves). The quotient graph is updated to generate a semantic quotient tree with no cycles. However, botanical inconsistencies can persist in the segmentation (e.g. or a limb node directly connected to branch node, without petiole, see figure 7c). Corresponding clusters are then re-clustered again locally and the semantic quotient tree updated again, until the botanical prior knowledge is satisfied (figure 7d).

Evaluation and testing

We tested our method on virtual Chenopodium plants (see RP3 and D6.4 for the development of the new LPy model) and an open dataset of real tomato plants, showing that our graph-based segmentation complemented by this refinement loop provides effective and fast results. We are also currently evaluating our method on real *Chenopodium* plants acquired with the Romi Plant Imager.

V. Machine-learning pipelines for segmentation and precise measurement

In the following approaches, we seeked to develop other skeleton-free segmentation methods which are no longer based on geometry but would use all possible associated features correlating with the botanical parts we want to identify, without a priori, using machine-learning techniques. The current rise of ever more powerful neural networks make it now possible to explore the wide multi-dimensional space of possible features from different input data and to learn their relevant use to perform a classification task. In our work, we considered working with two types of input data, in relation with our phenotyping workflow: either 2D RGB images (part A) or 3D point clouds (Part B).

A. Segmentation of a point cloud using deep-learning from 2D-images

Work performed by: Lahlou Alienor, Wintz Timothée, Julie Charlaix, Fabrice Besnard, Christophe Godin, and Peter Hanappe.
Article in preparation.

This approach has been implemented as an automatized workflow which can be combined using two softwares modules developed in the Romi project: the virtual-plant-imager generates virtual datasets of RGB images around a virtual plant (for training) and 3D reconstruction and segmentation is performed using the already mentioned Plant-3d-Vision. We use a unified workflow management system for all Romi modules, corresponding Tasks that can be assembled in automated pipelines are indicated below.

Virtual Plants

Virtual plants were produced using a generative model of *Arabidopsis thaliana*, using the dynamical language L-systems developed by Lindenmayer to model plant growth (Prusinkiewicz and Lindenmayer, 1990). The model used was specifically written in the L-py programming language, which is the python extension, part of OpenAlea, which implements Lsystems (Boudon et al., 2012). These virtual models are extensively described in other deliverables (D6.3 and D6.4.)

Virtual Plant Imager (Implemented task in the Virtual Plant Imager: VirtualScan)

The idea is to provide a simulated environment similar to the real Plant Imager, so we called it the virtual Plant Imager. The open source software Blender (Community, 2018) was chosen for the rendering part. The blender Eevee renderer allows for fast and realistic looking rendering of scenes and the python bpy modules allows for easy scripting. A python script using bpy and flask was written to generate and visualise the generated plant 3D models in Blender. HTTP requests allow us to easily load different plant models and backgrounds into the Blender scene, to move the loaded object, as well as change the camera position, rotation and its intrinsic parameters. The rendering of the scene into an image can also be triggered by a simple request. This enables us to take pictures of the model around the plant, as we would on the real world setup. To further increase the diversity of the simulated dataset and prevent over-fitting on the chosen plant textures, random colours are taken from a reference image and applied as the base colour of the plant material. The scene background is a 360◦ real world picture and rendering it in blender reproduces the original scene lighting. In order to increase the complexity of the images acquired with the virtual PlantImager, several sets of backgrounds were used, without limitations on the lighting environment – night, daylight, sunset – or the type of scene – indoor or outdoor. A simulated flash light is triggered with a random level of intensity to simulate different artificial lighting conditions during the acquisition of real life pictures

The ground truth plant part segmentation is acquired by rendering the plant material by material, with one material per plant part. The plant parts considered for Arabidopsis are leaf, stem, flower, fruit and peduncle. As we will see later for the 3D reconstruction, in the pinhole model we use for 3D to 2D projection, a whole line in 3D projects onto a single pixel. As a consequence, a single pixel can belong to several classes depending on the organs crossed by the line. The labelling method takes this plurality into account by generating a ground truth image for each class. The virtual Plant Imager was used to generate a training dataset of plant images with plant part segmentation for the training of neural networks.

Image segmentation using a segmentation convolutional neural network (P3dV Task = Segmentation2D)

The objective of image semantic segmentation is to convert an input image in a stack of probability maps of the same dimension as the image. If C is the number of different classes, semantic segmentation produces C output images with values between 0 and 1, giving for each pixel of the input image the probability of that pixel belonging to the corresponding class. To produce such semantic probability maps, we used a segmentation convolutional neural network (Guo et al., 2018). These segmentation networks are based on a contracting structure from an image to a low dimensional feature space, called the latent space, which encodes the content of the image. It is branched to a symmetric expanding structure that translates the information from the feature space to an image similar to the original one, but with only the content of interest reconstructed. The contracting structure is directly inspired from classification neural networks, and made of convolution layers, non-linearities and down-pooling layers. The convolution kernels allow to filter the information to emphasise the content, and the down-pooling allows to reduce the dimension of the information. The expanding structure reproduces the path in the other direction, with up-pooling layers and convolutions. The spatial information is reinjected to reconstruct the image properly, by providing the down-pooling coordinates from the contracting phase. We decided to use a neural network architecture inspired by U-Net (Ronneberger et al., 2015). However, to leverage the power of already existing labelled datasets, the contracting structure, or encoder, was replaced by a classification network trained on ImageNet, and with six classes to segment. Thanks to the great diversity in the ImageNet dataset, this classification structure has already learned to encode the semantic representation of an image in the latent space. The classification network that we used is ResNet (He et al., 2016) which is a deep neural network where inputs from previous layers are regularly re-injected into deeper layers in order to maintain the geometry and avoid vanishing gradients (Hochreiter, 1998).

Training

The network was trained with images from the virtual Plant Imager. The training set consisted of 2520 images (896x896 pixels), 18 views of 140 different virtual models. The distance and position of the camera around the plant varied to offer more diversity. To make the network robust to images and lighting conditions that are not in the dataset, we artificially augment the dataset by adding Gaussian noise ($\sigma = 0.01$) and random rotations. The images were normalised to correspond to the mean and variance of the training set of ImageNet, on which was initially trained the semantic structure (mean: (0.485, 0.456, 0.406) standard-deviation: (0.229, 0.224, 0.225)). The dataset was split into 3 sets: a training set to train the network (70% of the dataset), a validation set on which the network is not trained and is used to evaluate and compare the different networks (7% of the dataset), and a test set used at the very end on the selected architecture (23% of the dataset). To evaluate the segmentation

of the images, a combination of metrics were used. As it is a multi-class problem, a sigmoid is applied to each output in order to contain the numerical range of the predictions. First cross-entropy was used, it is a per-pixel metric. It represents the uncertainty of the prediction compared to the ground-truth. If the class of a pixel has a very low predicted probability, it will highly penalise the loss. If the probability is close to one, the contribution to the loss is close to zero. The notion of entropy represents this discrepancy between the ground truth distribution and the predicted distribution. It is naturally translated at the mathematical level with the negative of the logarithm:

$$L = -\sum_{i}^{n}\sum_{k}^{C} y_{i,gt}^{k} \ln(y_{\mathrm{i,pred}}^{k})$$

where n is the number of pixels in the image, C the number of possible classes, $y^{k}_{gt} = 1$ if pixel i is in class k, 0 otherwise, and $y^{k}_{i,pred} = p(\text{label}(\mathcal{Y}_{i,gt}) = k)$.
The second loss we used was the Dice coefficient, which theoretically writes as:

$$s = 1 - \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{C} y_{\mathrm{i,gt}} y_{\mathrm{i,pred}}^{k}$$

This coefficient compares the number of right predictions to the number of samples, for each class. In practice, the product of the ground truth by the predictions is summed, so that only the prediction of the right class contributes to the sum.
For example for point n, the class is k, and the prediction for class k is pk, the loss for point n will be:

$$1 - \frac{2p_k}{2} = 1 - p_k$$

The total loss is computed vector-wise, and lies between 0 and 1. We used the mean of these two losses for the training, as the cross-entropy has more stable gradients, but the Dice loss is what we want to minimise, and is less sensitive to class imbalance.

A procedure for training is available online in our documentation.

Fine-tuning to adapt to other plant species
Our network was trained on a virtual *Arabidopsis* model but aims at reconstructing real plants. Although the training performs well for real *Arabidopsis*, it transfers poorly to other species with large anatomical differences. Therefore we conceived a simple interface that allows the user to manually label a few images of interest (two or three images is enough), then run the training of the network on this small dataset. This way, the network will be able to segment similar images with correct classes. The interface uses LabelMe (Russell et al., 2007) for manual annotations, and runs the training on these images for 20 epochs. This generates a segmentation network specialised for the new species, starting from the network trained on the virtual plant models of *Arabidopsis*.

Backprojection and voting (Plant3dV task: SegmentedPointCloud)
First, a point cloud is generated using the same procedure as the geom pipeline (see section II: binary masking, volume carving and level-set techniques), but starting with the down-sampled images used for 2D segmentation by the neural network. The following task is to attribute a label to each point in this point cloud. To do so, each point is projected onto segmented images. Let $M_{i,k}$ denote the output of the segmentation algorithm for view i and class k. In the following, classes include all non-background organ classes. A threshold is applied to the probability scores, so that $M_{i,k}(x) \in \{0, 1\}$ for all x in Ω. The size of a given voxel Then, for each point, each class gets a vote from each of its projection in the masks $M_{i,k}$:

$$M_k(x) = \sum_{i \in I(x)} M_{i,k}(\pi_i(x)).$$

Then the class with the maximal number of votes is attributed to the point:

$$\text{class}(x) = \text{argmax}_k(M_k(x))$$

If a point gets no vote from any of its projections, then it is discarded as a background point. This yields a labelled point cloud of which every point is attributed a non-background class.

Instance Segmentation (Plant3dV tasks: ClusteredMesh and OrganSegmentation)
We next perform a clustering on segments labelled as fruit: this produces an instance segmentation of the fruits and their pedicels and approximates each fruit (or pedicel) by the main axis of its bounding box. The angles or internode length for successive pairs of fruits can then be estimated.

Evaluation of 2D machine-learning based pipeline ("ml pipeline")
Work performed by: Fabrice Besnard and David Colliaux

We performed the evaluation of these pipelines on both virtual and real *Arabidopis* plants. On virtual plants, we can compare the reconstructed point cloud to the ground truth. For example we checked that 20 views are enough to get the reconstruction within 2% of its best reconstruction (evaluated via the Chamfer distance on 100 views). On both virtual and real plants, we also evaluated the errors in the estimated sequence of angles and internodes. In this case, the ground truth is the set of angles/internodes given as input for the simulation of the virtual plant or the result of human measurements on real plants (details on how to evaluate phyllotaxis sequence can be found in D6.5).
We are currently integrating these tools to perform an automatic quantitative evaluation of our pipelines. In parallel, we visually inspected the 3D reconstructions, segmentations and automated measurements in the same database used for the evaluation of the geom pipeline (see above, figure 3 and table 1).

There are several advantages of the ml pipeline compared to the geom pipeline. First, it dispenses the 3D reconstruction from intermediary steps like the mesh and the skeleton computation, which could be a source of errors as seen previously (See skeleton zigzag and cycles in figure 3p,q). Second, the semantic segmentation systematically eliminates non-plant parts (tutors or linkers in figure 8d) and can resolve over-segmentation issues (impossibility to distinguish leaf from fruits in the geom pipe). However, due to strict requirements of the neural network about the input image sizes (fixed number of pixels, so image quality is downsized), the point cloud resolution in the ml pipeline is much coarser than the ground truth or than the point cloud computed with the geom pipeline. This results in some pairs of neighbouring fruits being merged and other sources of inaccuracy in the estimation of the angle sequence. In addition, local misclassification of pixels in 2D images is not rare, resulting in mislabelled points in 3D. If points are isolated, this has no consequence on the higher scale segmentation, but if several points hold the same erroneous label in a local neighbourhood, this results in over or miss-segmentation of the organs (Fig. 8e-g). We are currently testing whether some parameter tuning in the classification could mitigate those effects. However, we are also considering improving the architecture of the neural network to ensure better training and classification (e.g. filtering organs by size). Also, we could envision some post-processing steps to correct *a posteriori* the results to avoid major botanical inconsistencies, similarly the to refinement loop developed with graph-based clustering (see section IV) .

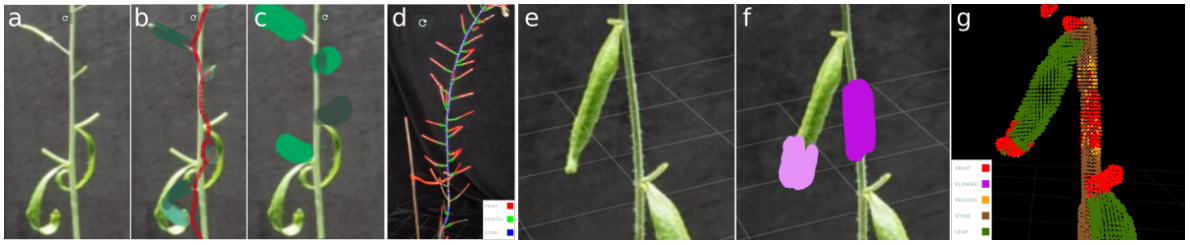

Figure 8: Qualitative evaluation of the machine-learning pipeline (comparison with the geometric pipeline).(a) input image (detail of the base of an *Arabidopsis* stem showing pruned branches and their subtending curled leaves, (b) overlay with geom pipe 3D reconstructions (skeleton in red and organs highlighted in green hues), (c) overlay with fruit segmentation generated by the ml pipe which eliminates the leaves. (d) Ml-pipe segmentation can also filter out the tutor and red linker holding this plant. (e-g) Local Segmentation errors of the ml pipe: (e) input image (detail, the plant is different than in a-c), (f) overlay with fruit segmentation showing erroneous fruit detection at the tip of a leaf and in the middle of the stem (g) Points labels show several local mislabelling in this region. Although their number is limited, this is enough to affect the fruit-level segmentation.

B. Learning directly from 3D point-cloud : Transferring PointNet++ Segmentation from Virtual to Real Plants

Work performed by: Ayan Chaudhury, Peter Hanappe, Romain Azaïs, Christophe Godin, and David Colliaux. This work was presented at the ***CVPPA workshop at the ICCV 2021*** *(Chaudhury et al., 2021).*

The aim of this work is to test whether a better machine-learning segmentation of the plant at the organ level (avoiding the errors reported above) could be achieved by learning directly from 3D point clouds rather than from 2D images. The major bottleneck of machine-learning approaches however is the training phase and the need for annotated data, whose production is expensive in terms of manual labour and time. For the case of plants, there is a scarcity of datasets to train the networks to perform organ segmentation for automated phenotyping applications.
In the previous “ml pipeline”, we generated training datasets of 2D images by using virtual plants as modelled by stochastic L-systems. Levering our expertise on virtual plants, we study whether L-system models also have the potential to train deep networks to perform well in organ segmentation on real data. We also investigate what are the features that are critical for knowledge transfer from synthetic data to real data without retraining on real data (a.k.a. fine tuning in transfer learning literature). Recently, only a couple of previous approaches investigated deep learning based segmentation of 3D point clouds of plants (Ghahremani et al., 2021; Turgut et al., 2022). In addition to the use of virtual plants for training, a key originality of our method is segmentation of the plant in one shot, without subdividing the point cloud into small patches as previously proposed by Turgut et al.

Generation of the dataset
Synthetic data
We followed the L-system modelling strategy as proposed in (Chaudhury et al., 2020) using their publicly available code. In this work, we considered the *Arabidopsis* plant for our experiments.
We developed an L-system model of *Arabidopsis* where we assigned a unique label to each organ of the plant (we used 4 labels: fruit, pedicel, stem, and leaf) in the rules of the procedural model. The scale of the plant and the parameters of the model are designed on the basis of observations from real plants. Typical examples of the parameters include length of fruit, bending of stem, number of

organs, radius of each branch, etc. We also introduced stochasticity in the parameters, so that every execution of the model can produce a different variety of the plant under consideration. This is achieved by sampling the parameter value from a Gaussian distribution with a mean of average value in real plants and a standard deviation of possible variabilities. We focus on specific geometric traits of plant growth which we believe are crucial for mimicking the real data, e.g. shape, size, length, curvature and orientation of the organs. For example, bending of the stem is an important parameter, which we model as a spline curve with controllable stochastic elasticity factor. This allows us to model a large variety of bendings that can render realistic data. Since the real plants have large variabilities with the branch thickness, the training data should include large varieties of branch radius. We model this effect in the virtual plant by sampling the value of radius from a uniform distribution of probable true values. For leaves for instance, we found it essential to model precisely the few tiny leaves on the main stem that have shape, size and orientation different from that of the leaves in the rosette (Figure 9a).
To test the robustness of our approach, we analysed the effect of different modifications of our protocol. First, we generated models of plants with and without small leaves in order to demonstrate the added complexity of training when leaves are present (discussed in the results section). Second, we changed the density of points, which is seen to play a crucial role in training.

Real data
Real data was acquired using the romi plant imager using between 60 and 72 poses around the main stem of an *Arabidopsis* plant (lateral branches were pruned off). The 3D point cloud of the plant was reconstructed using the approach described in section II. The dataset of 13 point clouds was then annotated manually using the open source CloudCompare software.

Training
The model for training synthetic data is initially rendered as a mesh using .obj file format. We sampled 4096 points on the surface of the mesh. Points are shuffled as a pre-processing step so that the network learning is invariant to the order of the points. We generated 1000 virtual plants which were split into 900/80/20 for training, validation and test, respectively. We used the PointNet++ architecture as introduced in (Qi et al., 2017) without batch normalisation (that did not bring any improvement).

Results
We used here the standard mean intersection over union metric to evaluate the result as, mIoU = TP/(TP+FN+FP).

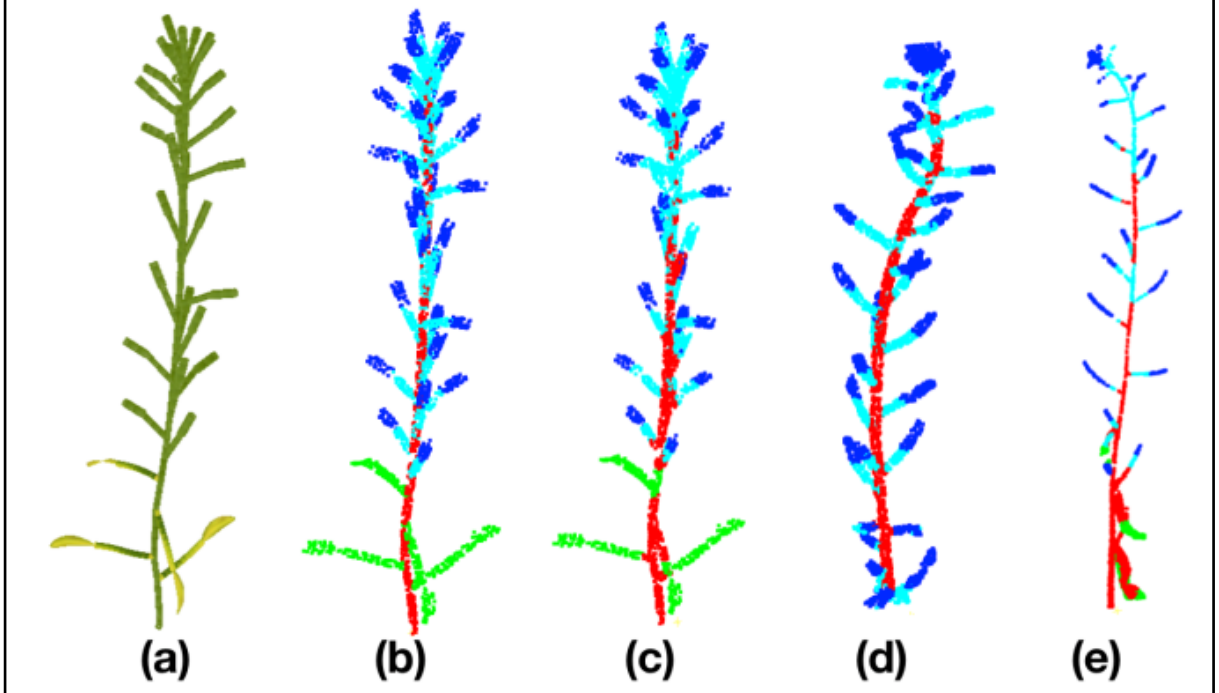

Figure 9: Virtual plant, ground truths and segmentation results of *Arabidopsis* main stem. Sample of a virtual plant model (a), corresponding colour-coded segmented ground truth point cloud (b), corresponding segmentation result with the same colour-code (c), segmentation result on real data without leaf (d), and with leaf (e).

Results on the test virtual plants dataset

We first considered the segmentation result of the network trained on virtual plants when applied to test on virtual plants (Fig. 9c). We obtained a mIoU of 0.77 for the stem, 0.90 for fruits, and 0.85 for pedicels. This shows that in principle the PointNet++ provides promising results when applied to complete point clouds.

Preliminary results on real plants

We then considered the first application of our method to real plants (Fig. 9d, e), with the aim of analysing how to improve our virtual plants for training so that they optimise the real plant recognition rates. In a preliminary experiment, we could obtain a maximal mIoU of 0.56, suggesting that further improvement of our virtual plants is required. We therefore are currently refining our construction of virtual plants so that mIoU is improved on real plants. In particular, we address the bending of axes that needs to be more realistic with respect to the observed real plants and the number of organs per plant.

Effect of the sampling strategy

We tested 2 sampling strategies to generate point clouds from the mesh of a virtual plant model for training. In the first strategy, we generated a point cloud via Poisson disk sampling of the triangle mesh (Yuksel, 2015). In the second strategy, we used the method proposed in (Chaudhury et al., 2020) to obtain uniform density.

For both cases we performed testing using PointNet++. The mIoU averaged over 20 plants is 0.68 with the first strategy whereas it is 0.84 in the second strategy, showing the importance of using uniform point clouds in our pipeline.

VI. Segmenting point-clouds in space and time

*Work performed by Pan, H., Hétroy-Wheeler, F., Charlaix, J., & Colliaux, D. This work was presented at the **3DV 2021** (Pan et al., 2021). This approach is also extensively described in deliverable **D5.4.***

The objective of the previous segmentation is to identify precisely particular plant parts, so that specific conclusions can be drawn from the phenotyping by the users (farmers, plant scientists). In such scenarios, labelling precisely a fruit or a leaf as such is crucial, because the downstream analysis requires to be focused on such parts. However, there are situations where the precise semantic labelling of botanical elements is not a requisite. For example, to monitor plant growth, identifying the correspondence between two plant parts at two different time points is enough, while the precise botanical nature of this part is not informative in the first place. In the following work, we designed a new algorithm helping in the specific task of segmenting plant architecture to help monitor its growth over time.

A. Multi-scale Space-time Registration of Growing Plants

With the Plant Imager, we acquired *Arabidopsis* plants in 3D twice a day for a week and used this data to develop a 3D+t analysis pipeline. The aim of this analysis is to put points in correspondence across acquisitions. For this, we tested various state of the art methods, based on Hidden Markov Models or Functional Maps for example.

Since the results were not satisfactory (see progress report RP2), we developed a new original method that circumvented the difficulties of existing approaches to correctly cluster the different branches and organs in a plant. Indeed, underlying algorithms mostly rely on local geometric features to register points from one to the next point cloud in the time series: they perform very badly with repetitive structures spread over the entire structure or newly formed shapes. To track growth of plants at the point level despite their multiple branches, leaves, fruits and flowers, as well as the constant organogenesis, we elaborated a hierarchical, multi-scale shape correspondance between two point clouds, using skeletons as a global positioning system. The proposed method starts with the creation of a topological skeleton of the plant at each time step. Any skeleton extraction method can be used, but we found in our experiments that applying iteratively two classical skeleton extraction approaches, namely the Laplacian Contraction (Cao et al., 2010) and the L1-medial Contraction (Huang et al., 2013) methods, allows us to reach satisfying results. This skeleton is then used to segment the plant into parts that we call branches. Then these branches are further divided into smaller segments that possess a simple geometric structure. These segments are matched between two time steps using a random forest classifier based on their topological and geometric features. Then, for each pair of segments matched, a point-wise registration is devised using a non-rigid registration method based on a local ICP (Besl and McKay, 1992).

The results of matching for *Arabidopsis* can be seen in figure 10. We also tested it on other publicly available datasets like maize and tomato. We established three different metrics for 3D point-wise shape correspondence to test the accuracy, continuity, and cycle consistency of the mapping. We then compared our method with the state-of-the-art. Our results show that our approach achieves better or similar results with a shorter running time. We showed that such a registration can be useful for tracking the growth of individual organs or interpolating the point cloud between 2 acquisitions (details are presented in D5.4).

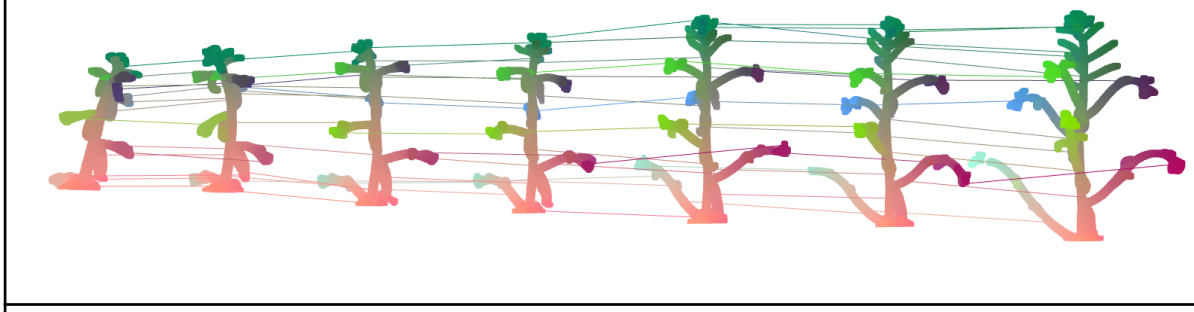

Figure 10: Example of matched point clouds for *Arabidopsis* across a week of daily acquisitions

B. Data-driven modelling of plant growth

The modelling of plant growth is usually seen as a dynamical system, the state at a given time can be estimated through the evolution of a differential equation given an initial state. When considering fine details of the architecture of a plant, describing the initial state of the plant (through its L-string for example) may be not possible. We propose another approach using neural networks trained on virtual plants, where the plant growth prediction is seen as a point cloud completion task. For this, we use a transformer based architecture, called PointTr and we consider 3 classes of objects: regular objects (like cars, chairs,...), random objects generated as trajectories of stochastic processes and plants which are intermediate objects between the 2 previously described classes. A preliminary result of the network on a pine tree is shown on the figure 11 below.

Figure 11: PointTr prediction (red) when the input is the partial point cloud (blue) with the ground truth being the complete point cloud (yellow).

VII. Conclusion and perspectives

As seen in this report, efficient automated phenotyping requires automatic segmentation, i.e. the computer-autonomous recognition of different plant sub-parts meaningful for the experimentalists. This problem is difficult and our work generated different complementary solutions adapted to different phenotyping scenarios and purposes.

As a conclusion, we will present recent work showing that combining the best of different algorithms

might be a reliable solution to solve this very complex problem of plant segmentation.

***Combining the best of geometric and machine-learning methods*.**

In a recent exploratory work, we tested the possibility to increase the resolution of the point clouds obtained from the ml pipeline. If downsampling 2D images is required to format them as input data for the neural network, it is still possible to generate a full-HD point-cloud reconstruction from the raw images, as the geometric pipeline does. Hence, we can use our volume carving technique followed by voxel-to-point cloud conversion (section II.B) in parallel to generate two point clouds: one labelled low-resolution (ml) and another unlabelled high-resolution (geom). Then, by registering the two point clouds together, the labels of the points in ml are transferred to the neighbouring points in geom, resulting in a final high-resolution and labelled point cloud (Figure 12). Our next step is to integrate this label transfer method as a new task in our Plant-3d-Vision library, so that an improved automated ml-pipeline could be created.

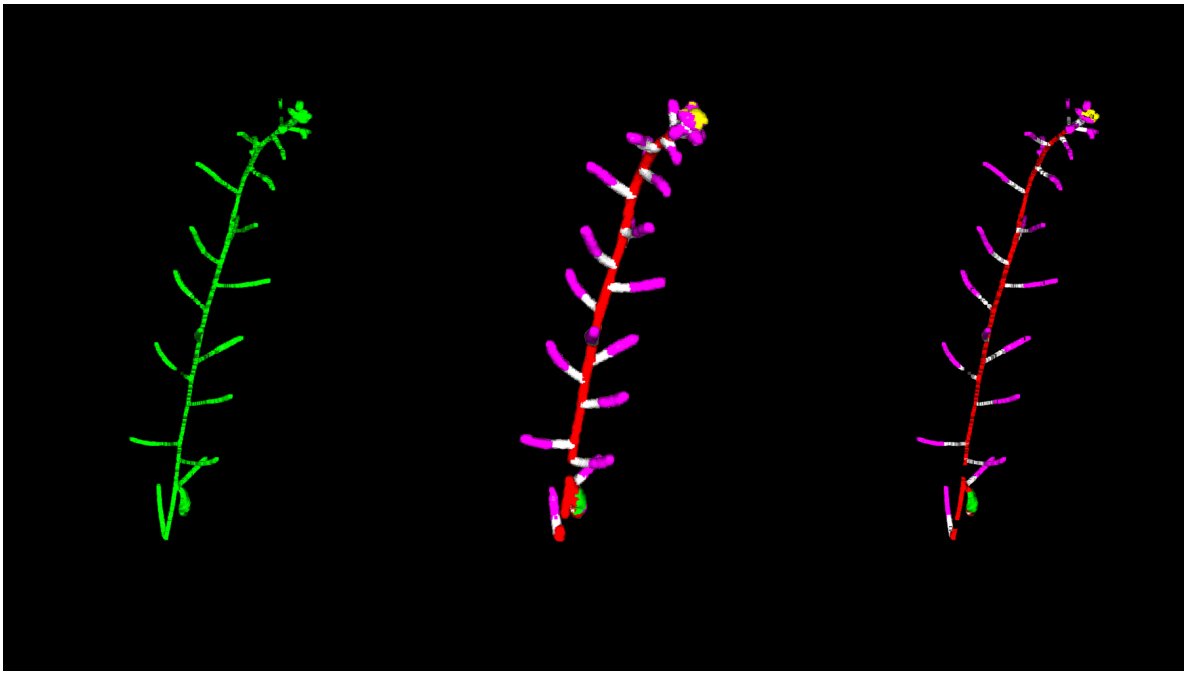

Figure 12: Transfering point label from one point cloud to another. (left), Unlabelled 3D point cloud generated with the geometric pipeline. Observe the thinness of the reconstructed axis (stem, pedicels and fruits). (centre), 3D point cloud generated with the machine-learning pipeline, with much thicker axis. (right), labels from the second thicker point cloud have been transferred to the first thinner point cloud.

3 Bibliography

Besl, P.J., and McKay, N.D. (1992). A Method for Registration of 3-D Shapes. IEEE Trans Pattern Anal Mach Intell *14*, 239–256. .

Boudon, F., Pradal, C., Cokelaer, T., Prusinkiewicz, P., and Godin, C. (2012). L-Py: An L-System Simulation Framework for Modeling Plant Architecture Development Based on a Dynamic Language. Front. Plant Sci. *3*. .

Bradski, G. (2000). The OpenCV Library. Dr Dobbs J. Softw. Tools.

Cao, J., Tagliasacchi, A., Olson, M., Zhang, H., and Su, Z. (2010). Point Cloud Skeletons via Laplacian

Based Contraction. In 2010 Shape Modeling International Conference, pp. 187–197.

Chattopadhyay, S., Akbar, S.A., Elfiky, N.M., Medeiros, H., and Kak, A. (2016). Measuring and modeling apple trees using time-of-flight data for automation of dormant pruning applications. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1–9.

Chaudhury, A., Ward, C., Talasaz, A., Ivanov, A.G., Brophy, M., Grodzinski, B., Huner, N.P.A., Patel, R.V., and Barron, J.L. (2019). Machine Vision System for 3D Plant Phenotyping. IEEE/ACM Trans. Comput. Biol. Bioinform. *16*, 2009–2022. https://doi.org/10.1109/TCBB.2018.2824814.

Chaudhury, A., Boudon, F., and Godin, C. (2020). 3D Plant Phenotyping: All You Need is Labelled Point Cloud Data. In ECCV Workshops, p.

Chaudhury, A., Hanappe, P., Azaïs, R., Godin, C., and Colliaux, D. (2021). Transferring PointNet++ Segmentation from Virtual to Real Plants. p.

Chu, Y. (1965). On the shortest arborescence of a directed graph. Sci. Sin. *14*, 1396–1400. .

Community, B.O. (2018). Blender - a 3D modelling and rendering package (Stichting Blender Foundation, Amsterdam: Blender Foundation).

Cornea, N.D., Silver, D., and Min, P. (2007). Curve-skeleton properties, applications, and algorithms. IEEE Trans. Vis. Comput. Graph. *13*, 530–548. https://doi.org/10.1109/TVCG.2007.1002.

Dijkstra, E.W. (1959). A note on two problems in connexion with graphs. Numer. Math. *1*, 269–271. https://doi.org/10.1007/BF01386390.

Ghahremani, M., Williams, K., Corke, F.M.K., Tiddeman, B., Liu, Y., and Doonan, J.H. (2021). Deep Segmentation of Point Clouds of Wheat. Front. Plant Sci. *12*. .

Guo, Y., Liu, Y., Georgiou, T., and Lew, M.S. (2018). A review of semantic segmentation using deep neural networks. Int. J. Multimed. Inf. Retr. *7*, 87–93. https://doi.org/10.1007/s13735-017-0141-z.

Hagberg, A., Swart, P., and S Chult, D. (2008). Exploring network structure, dynamics, and function using networkx (Los Alamos National Lab. (LANL), Los Alamos, NM (United States)).

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR 770–778. .

Hétroy-Wheeler, F., Casella, E., and Boltcheva, D. (2016). Segmentation of tree seedling point clouds into elementary units. Int. J. Remote Sens. *37*, 2881–2907. https://doi.org/10.1080/01431161.2016.1190988.

Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. Int J Uncertain Fuzziness Knowl Based Syst *6*, 107–116. .

Huang, H., Wu, S., Cohen-Or, D., Gong, M., Zhang, H., Li, G., and Chen, B. (2013). L1-medial skeleton of point cloud. ACM Trans. Graph. *32*, 65:1-65:8. https://doi.org/10.1145/2461912.2461913.

Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In Proceedings of the Fourth Eurographics Symposium on Geometry Processing, (Goslar, DEU: Eurographics Association), pp. 61–70.

Kutulakos, K.N., and Seitz, S.M. (1999). A theory of shape by space carving. In Proceedings of the Seventh IEEE International Conference on Computer Vision, pp. 307–314 vol.1.

Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. IEEE Trans. Pattern Anal. Mach. Intell. *16*, 150–162. https://doi.org/10.1109/34.273735.

Li, D., Quan, C., Song, Z., Li, X., Yu, G., Li, C., and Muhammad, A. (2021). High-Throughput Plant Phenotyping Platform (HT3P) as a Novel Tool for Estimating Agronomic Traits From the Lab to the Field. Front. Bioeng. Biotechnol. *8*. .

Pan, H., Hétroy-Wheeler, F., Charlaix, J., and Colliaux, D. (2021). Multi-scale Space-time Registration of

Growing Plants. 2021 Int. Conf. 3D Vis. 3DV 310–319. .

Paulus, S. (2019). Measuring crops in 3D: using geometry for plant phenotyping. Plant Methods *15*, 1–13. https://doi.org/10.1186/s13007-019-0490-0.

Potmesil, M. (1987). Generating octree models of 3D objects from their silhouettes in a sequence of images. Comput. Vis. Graph. Image Process. *40*, 1–29. https://doi.org/10.1016/0734-189X(87)90053-3.

Prusinkiewicz, P., and Lindenmayer, A. (1990). The Algorithmic Beauty of Plants. In The Virtual Laboratory, p.

Qi, C., Yi, L., Su, H., and Guibas, L.J. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In NIPS, p.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, N. Navab, J. Hornegger, W.M. Wells, and A.F. Frangi, eds. (Cham: Springer International Publishing), pp. 234–241.

Russell, B.C., Torralba, A., Murphy, K.P., and Freeman, W.T. (2007). LabelMe: A Database and Web-Based Tool for Image Annotation. Int. J. Comput. Vis. *77*, 157–173. .

Sanders, P., and Schulz, C. (2011). Engineering Multilevel Graph Partitioning Algorithms. In Algorithms – ESA 2011, C. Demetrescu, and M.M. Halldórsson, eds. (Berlin, Heidelberg: Springer), pp. 469–480.

Schönberger, J.L., and Frahm, J.-M. (2016). Structure-from-Motion Revisited. 2016 IEEE Conf. Comput. Vis. Pattern Recognit. CVPR https://doi.org/10.1109/CVPR.2016.445.

Schönberger, J.L., Zheng, E., Frahm, J.-M., and Pollefeys, M. (2016). Pixelwise View Selection for Unstructured Multi-View Stereo. In Computer Vision – ECCV 2016, B. Leibe, J. Matas, N. Sebe, and M. Welling, eds. (Cham: Springer International Publishing), pp. 501–518.

Sethian, J.A. (1998). Fast marching methods and level set methods for propagating interfaces. p.

Sorkine, O. (2005). Laplacian mesh processing. Eurographics State Art Rep. *4*. .

Tabb, A., and Medeiros, H. (2018). Fast and Robust Curve Skeletonization for Real-World Elongated Objects. In 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1935–1943.

Tagliasacchi, A., Alhashim, I., Olson, M., and Zhang, H. (2012). Mean Curvature Skeletons. Comput. Graph. Forum Proc Symp. Geom. Process.

The CGAL Project (2022). CGAL User and Reference Manual (CGAL Editorial Board).

Turgut, K., Dutagaci, H., Galopin, G., and Rousseau, D. (2022). Segmentation of structural parts of rosebush plants with 3D point-based deep learning methods. Plant Methods *18*, 1–23. https://doi.org/10.1186/s13007-022-00857-3.

Van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., and Yu, T. (2014). scikit-image: image processing in Python. PeerJ *2*, e453. .

Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nat. Methods *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2.

Wintz, T., Colliaux, D., and Hanappe, P. (2018). WINTZ, COLLIAUX, HANAPPE: EXTRACTION OF TRAITS FROM ARABIDOPSIS.

Xu, H., Gossett, N., and Chen, B. (2007). Knowledge and heuristic-based modeling of laser-scanned trees. ACM Trans. Graph. *26*, 19-es. https://doi.org/10.1145/1289603.1289610.

Yuksel, C. (2015). Sample Elimination for Generating Poisson Disk Sample Sets. Comput. Graph. Forum *34*. .

Zhou, Q.-Y., Park, J., and Koltun, V. (2018). Open3D: A Modern Library for 3D Data Processing. ArXiv:1801.09847.

Ziamtsov, I., and Navlakha, S. (2019). Machine Learning Approaches to Improve Three Basic Plant Phenotyping Tasks Using Three-Dimensional Point Clouds. Plant Physiol. *181*, 1425–1440. https://doi.org/10.1104/pp.19.00524.

4 Annexes

Table of main third-party computer vision software used in our analysis pipelines:

Software	Description	Use in Romi
Colmap	Structure-from-Motion (SfM) and Multi-View Stereo (MVS) pipeline	Estimating camera poses and matrices
opencv	open-source library of computer vision algorithms.	Undistorting images
scikit-image	collection of algorithms for image processing in python	Applying binary masks on images to select the plant
scipy	Fundamental algorithms for scientific computing in Python	Fast-marching algorithm in the Level-Set method (from voxel to point cloud)
CGAL	software project providing geometric algorithms (C++ library)	Computing a Curve skeleton from point cloud
Open3D	modern library for 3D data processing	Computing a mesh from the point cloud
Networkx	Python package for network analysis	Computing a rooted tree graph form the curve skeleton

Table of other third-party software used in our analysis pipelines:

Software	Description	Use in Romi
Blender	free and open-source 3D computer graphics software	Generate 2D images from virtual plants
Lpy / OpenAlea	open source platform providing an easy-to-use environment for plant modelling	Generating virtual plants
LabelMe	open annotation tool	Manual annotation of 2D images for training

CloudCompare	Open source software for 3D point cloud and mesh processing	Manual annotation of 3D point clouds for training