UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Tracking of the growth of roots based on multi-object tracking techniques

Dissertation presented by
Carmen Alabart Gutierrez del Olmo

for obtaining the master's degree in
Electrical Engineering

Supervisor(s)
Christophe De Vleeschouwer, Xavier Draye

Readers(s)
Christophe De Vleeschouwer, Xavier Draye, Laurent Jacques
Academic year 2016-2017

Abstract

Multi-object tracking (MOT) is based on the detection of several objects in individual frames at different time instants and then its association across time, obtaining the trajectory of each object. MOT in videos is an important problem in computer vision which has wide applications in various video analysis scenarios, such as visual surveillance, sports analysis, robot navigation, biology,...

This thesis considers MOT applied into a biological scenario, more precisely the analysis of plants root systems. A collection of frames is provided by one single line-scan camera that acquire images periodically. Each image depicts the root of a plant at a given time instant. To ease the detection of the root tips in each image, the plants have been sprayed with water, so that water drops accumulate on each root tip, making their localization easier. Our main purpose is to define a MOT system to track the growing of plant roots, defining the path/trajectory of each root until the plant is completely developed. Our contributions, briefly explained below, are all related to the exploitation of the shortest path obtained for every node in the image in a graph-based framework.

The implementation of MOT can be decomposed in two separate phases. On one hand, a detection phase is implemented, where our main objective is to detect the extremity of each root. The best approach is to define a graph in which we can implement the Dijkstra algorithm for the definition of the shortest path to every node on the image. The graph describes the connectivity of the pixels of the images provided, where each white pixel is considered a node. The extremities of the roots are localized through the analysis of the shortest paths, where each extremity corresponds to a local maxima in a small neighborhood.

The second phase of the MOT, consist in the association of the detections across time to define the shape of each root. A track is obtained for each extremity, measuring how each root has evolved during its growth period. Due to the low number of miss-detections per frame, we can use a tracking algorithm defined over a small time window, because every drop is present in almost all frames. The best algorithm that suits our necessities is the Hungarian algorithm, which performance is based on the definition of a cost-matrix that defines the relation of detections between consecutive frames. For the definition of the cost-matrix we have defined two different scenarios:

- Forward cost. Definition of a Gaussian distribution that predicts the position of a drop in a future frame, given its position in the current frame.
- Backward cost. The cost is based on the distance between a past detection and the shortest path connecting a current detection to the source. This cost matrix is referred to as a backward cost, because the distance is measured from detections in the current frame to detections in the past ones.

Contents

Chapter 1

Introduction and Motivation

1.1 Context and Scope

Nowadays, in many rich and poor nations, plants are cultivated in nutrient-poor soils and/or under conditions of water deficit. In these conditions, soil resource acquisition is a limitation to crop production. Due to the fact that water and nutrients are coil resources, their capture is likely to be determined by the way plants develop their roots in the soil profile and throughout the season. By improving genetically the root system architecture, it could improve crop resource capture. This relies on the identification of the genetic determinism of root growth and development, which, in turn, requires the phenotyping of massive numbers of plant root systems.

In this context, the plant community is constantly developing new tools to observe and describe root system architecture. Many of them are image-based. The information collected from phenotyping platforms and image analysis will allow the identification of the genes involved in the control of root elongation rate, root angles, root branching dynamics,... This can be used by breeding companies to develop more rapidly new varieties that could better cope with soil limitations. [1]

In this project, a multi-object tracking (MOT) system has been defined to analyze the development of root systems. Our priority is to track the behavior of root extremities across time, to collect all the information required to assess the genetic improvement of the root system architecture that define plants. It is based on the detection of the root tips in individual frames at different time instants and their association across time, obtaining the trajectory of each root tip.

1.2 Acquisition System

For the design and validation of the algorithm, we have used three datasets provided by Professor Xavier Draye (*Earth and Life Institute - Agronomy Université Catholique de Louvain*). Each dataset consists in a collection of high resolution 2D images (15300×4096) provided by one single line-scan camera that acquires images periodically, every 2 hours during 3 weeks, along with a backlight configuration. The combination of these two elements provide high contrasted images that are not influenced by the external radiation, which simplifies greatly the image analysis process. However, due to the fact that a line-scan camera reads from top-to-bottom and from bottom-to-top, the collection of images generated are going to be alternatively inverted. As a result, some pre-processing steps (e.g mirroring the images) have to be done to ensure that all the images are equally oriented to guarantee an accurate identification of the root trajectories (Figure 1.1). Each image from the sequence depicts the root of a plant at a given time instant for the reconstruction of root trajectories.

The platform that supports all the plants has been created to favor a non-invasive imaging of root systems growing in aeroponic conditions. However, it uses a structure to support the development of the roots, which is going to be present on every image. Therefore, some post-processing steps have to be done to ensure that the roots tips/drops are not detected on this area. In addition, to ease the detection of the root tips in each image, the plants have been sprayed with water, so that water drops accumulate on each root tip, making their localization easier. Figure 1.2 shows the platform, the plants disposal, the line-scan camera, and structure where the root system is developed.

1.3 Organization and Contribution of the thesis

The thesis is organized as follows. Root tips/drops detection and classification, along with their practical implementation in Section 2. The detection relies on k-shortest paths computation over a graph that connects adjacent root pixels. Then, an in-depth description of the methodology and algorithm used to perform the root tips/drops tracking is provided in Section 3. The association of drops/tips across time builds on the Hungarian algorithm with appropriate cost metrics. Two scenarios are presented, where two different cost matrices are defined: one based on a growing prediction model (forward cost), and another based on the distance between previous tips and the current tip shortest path (backward cost). An experimental validation is presented in Section 4. Finally, Section 5 concludes the thesis with some comments and further work notations.

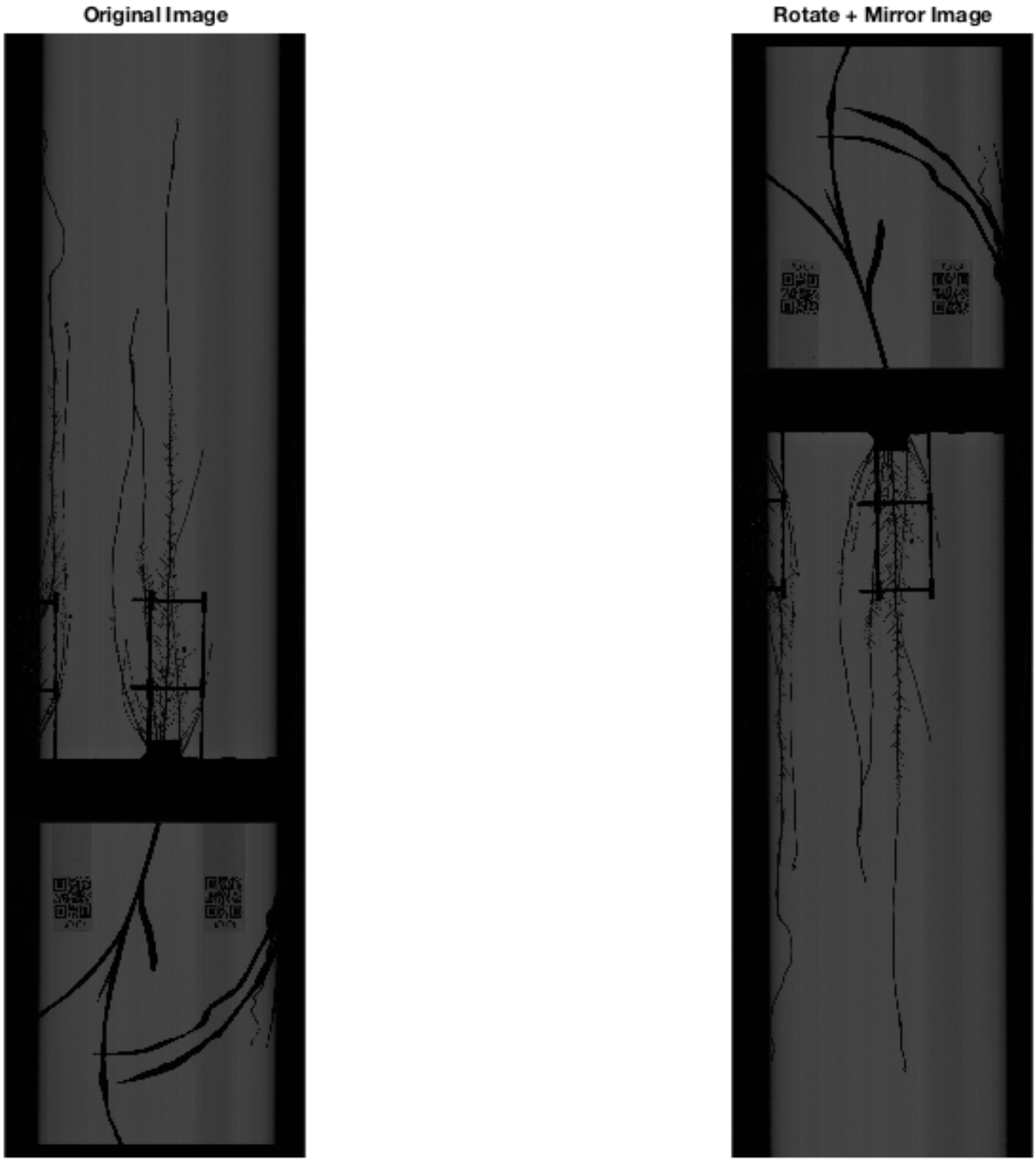
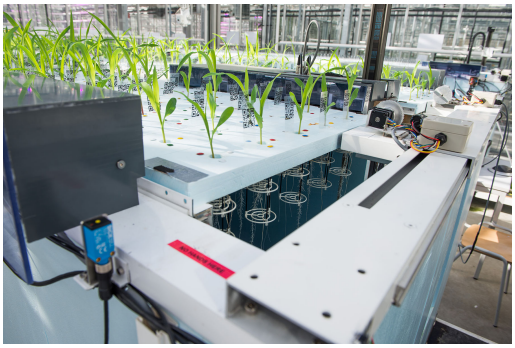


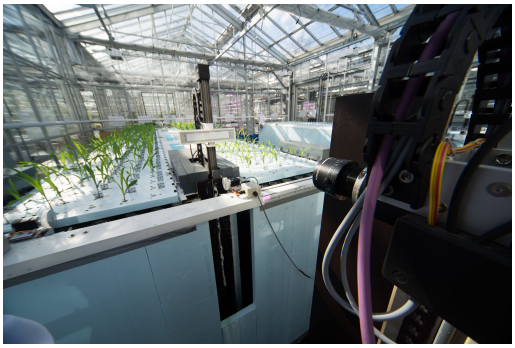
Figure 1.1: On the left, the original image, which has been taken from bottom-to-top. On the right, the original image rotated and mirrored.

(a)

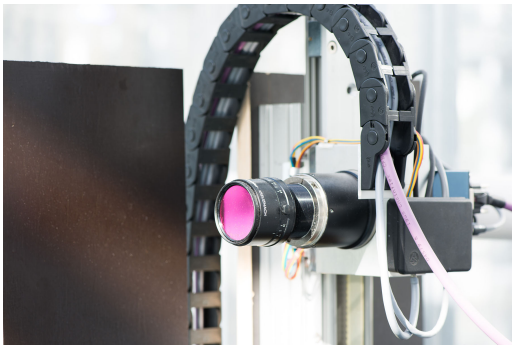


(b)

(c)



(d)



(e)

(f)

Figure 1.2: (a) Collection of plants under study. (b) and (d) platform and scan-line camera used to the study of root systems growing in aeroponic conditions. (c) and (f) support structure for the development of the root system. (e) scan-line camera.

Chapter 2

Root tips/drops detection and classification

In this chapter, we explain the process to detect root tips/drops, and classify them into primary, seminal or branching roots. To detect the root tips, we primarily observe that a tip/drop is localized at the end of a root. This way, if we calculate the distances between the origin of the root and any root pixel, we can observe that the tip always corresponds to the maximum distance. The classification of the detections is based on the length of the shortest-paths connecting the root tips to the source, and the analysis of the amount of nodes shared between paths. These two features make the differentiation between roots possible, due to the fact that primary roots are the longest ones and along with seminal roots they born from the source. Whereas branches are the shortest type of roots and they are born from other roots.

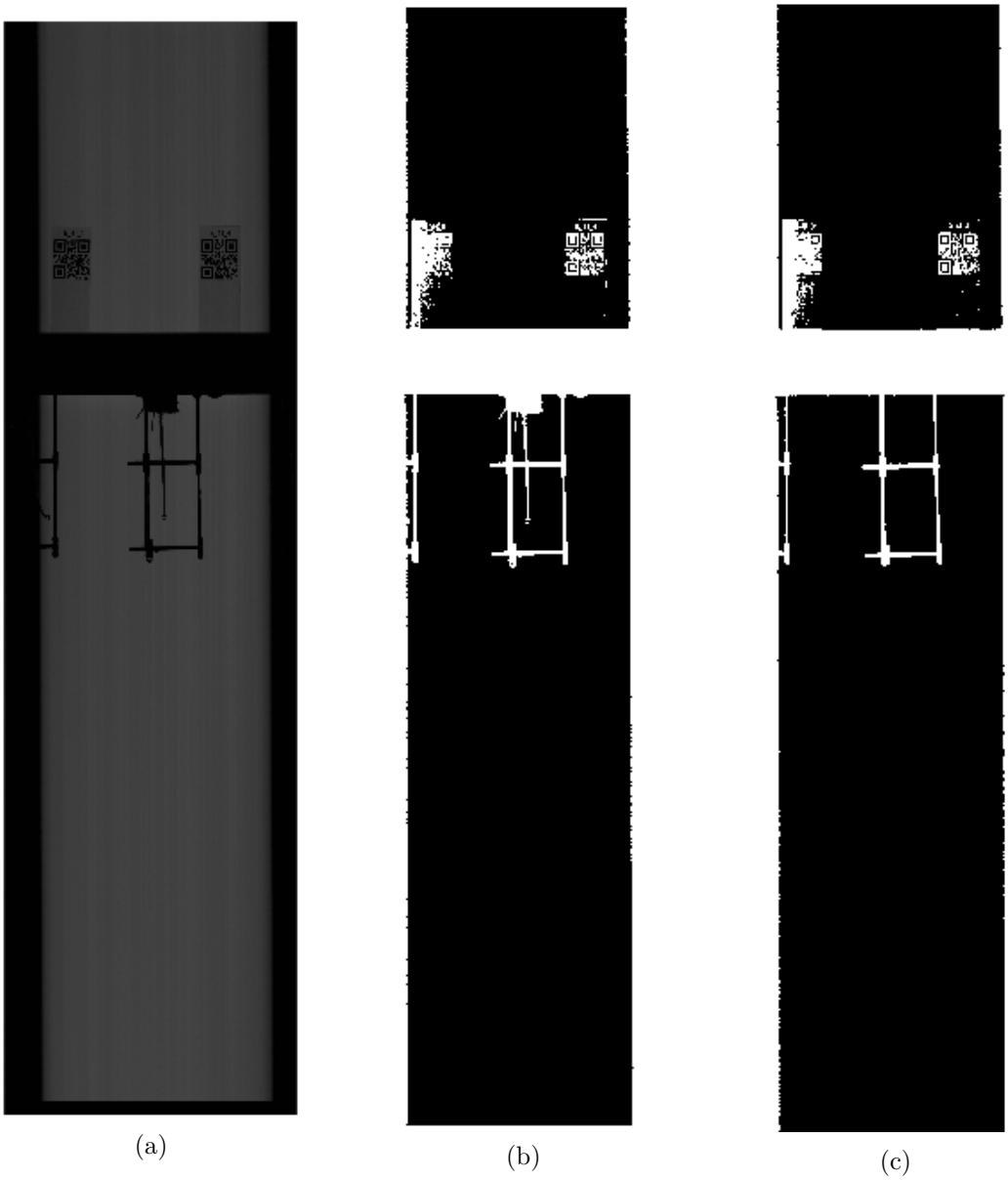
2.1 Root tips/drops detection

The detection of each root tip/drop is based on the segmentation of the root pixels. Given the highly contrasted images provided by the acquisition set-up, the foreground pixels can be identified effectively by thresholding the captured image (see Figure 2.1). In the following, the image I denotes a binary image, in which a pixel is set to one when the corresponding pixel in the input image is above the threshold, and to 0 elsewhere. We introduce another image M, to identify the foreground pixels that are part of the rigid structure supporting the plant while it is growing. M is derived by registering, using a rigid translation, a background reference image (captured before the root starts growing) with the observed image, based on the minimization of the Sum-of-Squares distance (SSD) [3]. Pixels of the foreground I that are part of (or very close to) the registered reference image are set to one in M.

To detect root tips, we consider two features that are specific to tips:

- Feature 1: localization within the root. The main property of a tip/drop is that it is located at the end of a root, which means that each tip/drop is located at a locally maximum distance from the origin. This feature can be exploited to identify a subset of root tips candidates;
- Feature 2: neighborhood of a tip/drop. A root tip/drop is characterized by the fact that in its close surroundings it has no other root tips/drops underneath. This second feature is effective in extracting actual root tips from the set of candidates identified based on the locally-maximum distance criteria.

In practice, the first feature is exploited through the implementation of the Dijkstra algorithm [2]. Therefore, we have to describe a graph that defines the connectivity of the white pixels (equal



(a) (b) (c)

Figure 2.1: From left to right. (a) Original image, (b) original image thresholded (I), and (c) background reference image.

to root pixels) of the image. The construction of the graph $G = (V, E, W)$ assign a node to each white pixel from the image $I(i, j),\ 0 \leq i \leq H,\ 0 \leq j \leq W$. We define the set of vertices V to be the set of root pixels, i.e. $V = \{(i, j)|I(i, j) = 1\}$. We also introduce the line and column indexing functions $i(v)$ and $j(v)$, that are the row and the column indexes respectively of the node $v \in I$. A 4-connected pixels neighborhood N_g has been adopted to define the edges E in the graph for all foreground pixels. For the pixels that belongs to the horizontal areas in the mask M, a 2-connected pixels vertical connectivity is considered to avoid the root support structure serving as shortcuts when connecting some roots to the source (see illustration in Figure 2.2). Finally, W is defined as the weight of each edge, where a unitary weight has been adopted for all edges.

Given G, the Dijkstra algorithm is used to compute the shortest distance between each node of

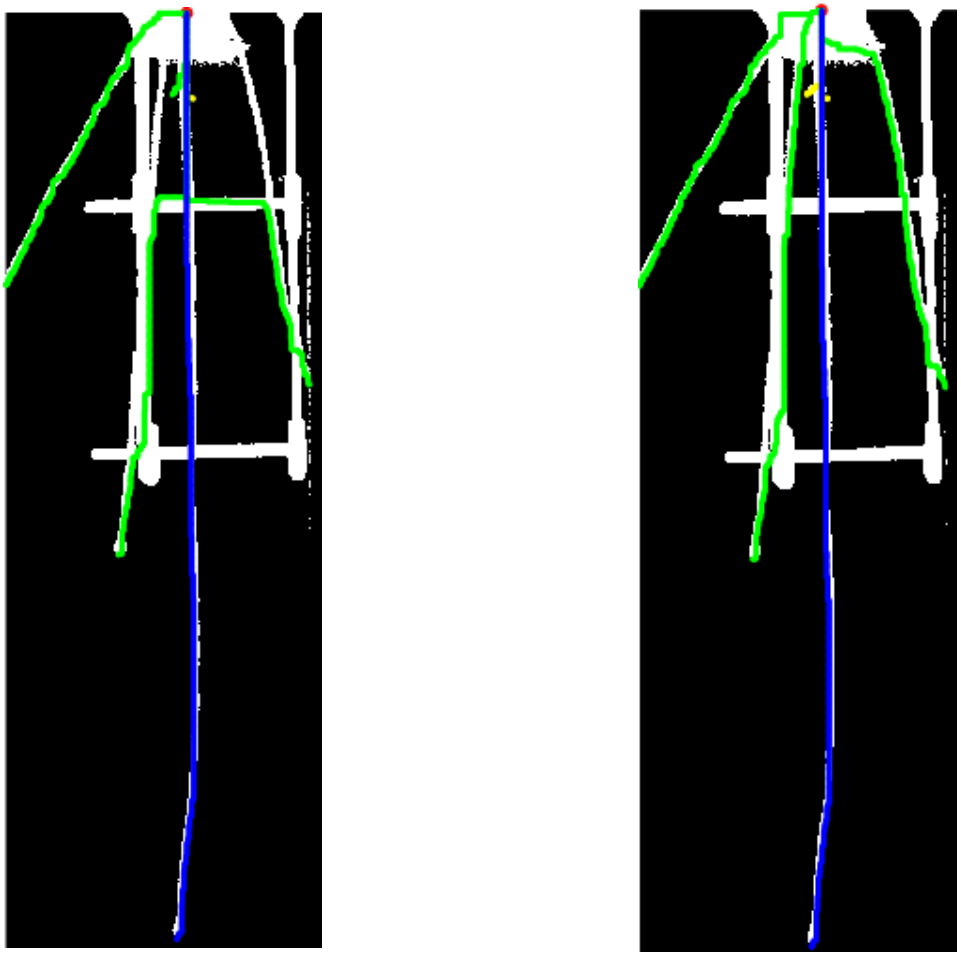


Figure 2.2: On the left, obtention of shortest paths without using a 2-connected pixels vertical connectivity along the horizontal areas in the mask M. On the right, obtention of shortest paths using a 2-connected pixels vertical connectivity along the horizontal areas in the mask M. As we can appreciate, the limitation of a 2-connected pixels vertical connectivity allows a higher accuracy on the definition of the shortest paths that defines each root.

the graph and the source node s. The source node is selected manually, corresponding to the origin of the root system. All distances are stored in a column vector d, where $d(v)$ is the distance from the source s to the node v. All the nodes involved in each shortest path are stored in the vector P, where $P(v)$ is the shortest sequence of nodes between the source s and the node v. This way, if we create a new image (based on the original one) that stores in each white pixel the distance to reach the source node s we can analyze which nodes are maximum within a 8-connected nodes neighborhood N_d. Therefore, the image $H(i,j)$ is created, where each pixel is going to store the distance to s: $H(i(v),j(v)) = d(v) = |P(v)|, \{i(v),j(v)\} \in V$. Afterwards, we are going to look for local maxima within the neighborhood N_d to detect tips/drops candidates $\mathcal{C}$:

$$\mathcal{C} = \{v \in G | v = \underset{n \in N_d(v), v}{\arg\max}\, d(n),\ \ H(i(v)+1, j) = 0\},\ \ \forall j \in [j(v)-1, j(v)+1]$$

As shown in Figure 2.3 some false root tips/drops candidates are produced by: the structure that supports the plant (yellow), the base where roots are born (cyan), or roots asperity (blue). Table 2.1 introduces a number of rules to remove the false-positive cases. The set of drops D is obtained by

removing false-positives from $\mathcal{C}$.

False-Positive Case	Removal Technique
A drop is localized as part of the structure that supports the root system.	All points within the structure defined in the mask M are removed. In addition, two small windows are defined at the bottom of the structure to remove all the points that belong to a drop that has been detected in this area.
A drop is localized as part of the base where roots are born.	A template of the base B is created. Then the collection of white pixel coordinates that belongs to the base B are removed.
A drop is localized as part of the root.	A large rectangular window is considered to analyze the pixels that are below a certain node. This procedure allows the differentiation between false and true tips since a real tip does not have white pixels underneath.

Table 2.1: False-positive cases, and the solution adopted to differentiate them from true positives.

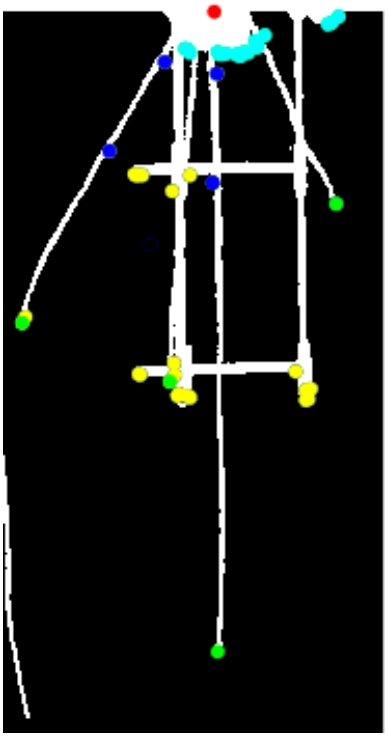


Figure 2.3: Selection of root tips among root pixels with locally maximal distance to the source. The image shows the base where roots born, the structure that supports the plant, and the root system with the complete set of root tips/drops detected (blue + cyan + green + yellow). As we can observe all false-positive cases (part of the structure (yellow), part of the base (cyan), and part of the root (blue)) have been removed, remaining only real root tips cases (green).

2.2 Root tip/drop classification

After the detection of all the tips/drops, we have to classify them into three types (Figure 2.4):

- Primary root. It is the longest root and it grows from the origin point. The tip/drop associated with it is always located in a position that forms an angle of 75 - 110 degrees with respect to the horizontal line located in the source point.
- Seminal roots. These roots are shorter than the primary root. They grow from the origin point too.
- Branches roots. Branches are born from other roots, which means that they do not grow from the origin point. They are shorter than primary and seminals roots.

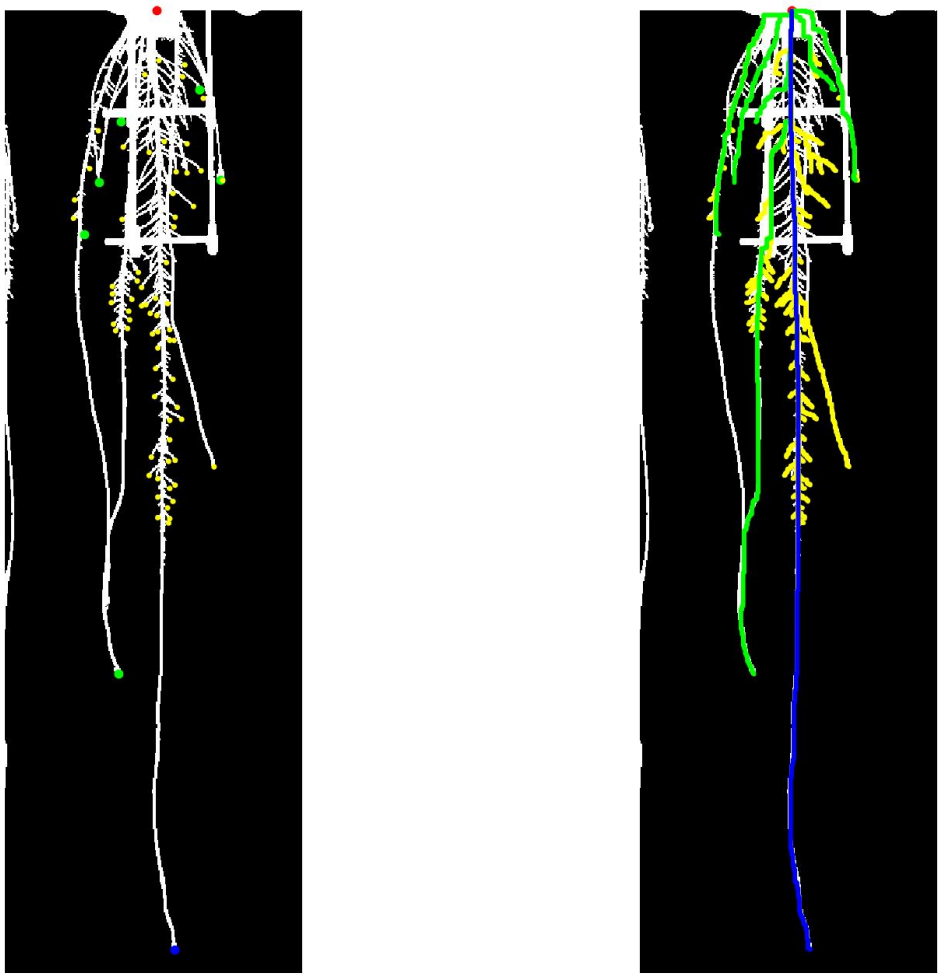


Figure 2.4: On the left, differentiation between detections that belongs to the primary root (blue), seminal roots (green), and branches roots (yellow). On the right, the paths corresponding to each detection.

From the previous section 2.1, we obtained all the possible drops per image D, and the paths followed to reach them P (after the execution of the Dijkstra algorithm). Before the beginning of the

classification process, we sort the drops in descending order based on the length of the shortest path to the source node. As a result, we obtain D_L. This way, the first drop is going to be always the furthest one from s (source node), which usually corresponds to the primary root. To ensure that the first drop belongs to the primary root, we check that the angle with respect to the horizontal line located in the source point is between 75 - 110 degrees (Figure 2.4). The path that belongs to this primary root is going to be denoted as α, and its length is defined as $|\alpha|$. On the other hand, if this condition is not achieved, the root is classified as a seminal root, which path is denoted as β.

Once the longest root is classified as a primary or a seminal root, we proceed with the classification of the rest of the drops/tips. The main difference between a seminal and a branch is the place they were born, from the origin or another root respectively. As a result, if one root was born from another one, they are going to share a high percentage of their shortest path to the source node. This way, we should identify the intersection node between paths, and the percentage of nodes that they shared. To do so we take as a first reference the path of the primary path classified (α or β), and we compare it with the path corresponding to the next drop in D_L. This comparison can be easily achieved from the difference between both paths, denoted as S_x where $x = \{\alpha \text{ or } \beta\}$, depending on the classification of the longest root. When we compute this difference, we can observe that if both paths are equal we obtain a value equal to 0. As a result we can identify easily the intersection between both paths. However, an intersection can lead to two different situations: a crossroad, where the amount of nodes shared is small and both paths get separated later on, and a merge, where the amount of nodes shared is high in comparison with the total length of the path.

$$S_x = P(v_i) \setminus x = \begin{cases} = 0 & \text{if merge/crossroad} \\ \neq 0 & \text{if different paths} \end{cases}$$
$$x = \{\alpha \text{ or } \beta\}$$

Where P is the collection of paths for each drop and v_i is the corresponding drop from D_L. Therefore, $P(v_i)$ is the shortest path of the drop v_i. As we mentioned before, if S_x is equal to zero, it means that a intersection node between both paths has been encountered. To distinguish between a seminal or a branch, we should differentiate the type of intersection: crossroad or merge. To do so, we analyze S_x as a whole, and not node by node. We have to find the furthest position in S_x, that accomplish that all the nodes until that position $[0...k]$ are equal to zero. This means that all the nodes until k have been used by both paths.

$$k^* = \text{ argmax } k$$
$$s.t \sum_{i=0}^{k} S(i) = 0$$

Then we analyze how far is k from the origin. Therefore, if more than the 50% of the path is shared, we will consider that the tip/drop is a branch, which path is denoted as γ. Otherwise it will be a seminal, which path is denoted as β.

$$|P(v_i)| \times 0,5 = \begin{cases} > k^* & P(v_i) \in \gamma \text{ (Branch root)} \\ < k^* & P(v_i) \in \beta \text{ (Seminal root)} \end{cases}$$

In next iterations, we are going to compare the next drops in D_L with the paths stored in α, and β, since branches can be originated from primary and seminal roots. Algorithm 1 describes the classification procedure.

input : D, P
output: Classification between α, β, and γ

Initialization: $\alpha = \{\}$, $\beta = \{\}$, $\gamma = \{\}$;
$D_L \leftarrow SortDescendingOrder(D, P)$;
/* We analyze the furthest node to initialize the process */
$v \leftarrow D_L(1)$;
$p \leftarrow P(v)$;
angle $\leftarrow angleDirection(v)$;
if *(angle* ≥ 75*) and (angle* ≤ 110*)* **then**
    $\alpha_1 \leftarrow p$; // Classified as a primary root
else
    $\beta_1 \leftarrow p$; // Classified as a seminal root
end
for $i = 2 \ldots |D_L|$ **do**
    $v \leftarrow D_L(i)$;
    $p \leftarrow P(v)$;
    $S_\alpha \leftarrow (\alpha - p)$;
    $S_\beta \leftarrow (\beta - p)$;
    $k_\alpha \leftarrow findMax_k(S_\alpha)$;
    $k_\beta \leftarrow findMax_k(S_\beta)$;
    if $(k_\alpha < |p| \times 0,5)$ *or* $k_\beta < |p| \times 0,5)$ **then**
        $\beta_i \leftarrow p$; // Classified as a seminal root
    else
        $\gamma_i \leftarrow p$; // Classified as a branch root
    end
end

Algorithm 1: Classification algorithm.

2.3 Practical implementation

In this section, we explain the algorithm and methods used to achieve a successful detection and classification of roots. As we mentioned in Section 1.2, we should guarantee that all the images are equally oriented. Therefore, we rotate and mirror the inverted frames. In addition, we crop the images to ensure that we perform the detection only in the desired area (area below the horizontal plane), avoiding then possible errors, having a final resolution of 8902 x 2741 pixels. Afterwards, we threshold and normalize the frames to obtain high resolution binary images. This way, we simplify the detection problem into a binary classification problem. As a result, the detection of drops, and the separation between background and foreground is more accurate.

Furthermore, we implement a subsampling process because the amount of data collected per image is very high, increasing then the computational cost of the analysis of the images, making it infeasible. We have created two different resolutions, based on the size of the super-pixel defined: 10x10 and 5x5. In both cases to ensure the connectivity of the pixels, if the super-pixel defined has any white pixel in the border, the whole pixel is going to be considered as white. The aim of distinguishing between two resolutions is that in one case, the image can be analyzed as a whole (10x10), with the cost of a lower resolution. While in the other case (5x5) the image has to be divided into pieces, increasing the computational cost, but it allows us to ensure the reliability of the detections obtained on the low resolution images (10x10). In Table 2.2 we defined the final resolution of the cropped image in low resolution and high resolution.

	Final frame size
Super-pixel size 10x10 (low resolution)	887 x 231
Super-pixel size 5x5 (high resolution)	1773 x 541

Table 2.2: Final resolution depending on the size of the super-pixel.

Low Resolution Analysis

The detection algorithm for the low resolution image is summarized in Algorithm 2.

High Resolution Analysis

The implementation of the high resolution analysis is quite similar to the previous low resolution analysis, but we can find the following differences:

- Due to the fact that the size of the image is still too big to be processed as a whole, we have to divide the image into pieces T_x of 150 rows. The union of all pieces is equal to the original image $I = \bigcup_{i=1}^{X} T_x$. At each piece, we implement the Dijkstra algorithm, but to take into account in each one the distance from s, we have to defined a virtual node s_v that is going to be connected to all the nodes that belongs to the first row of the current piece. This first row is going to be the last row from the previous piece T_{x-1}. This way, we can link all the pieces having the distance of all the nodes to the source s. Each piece, T_x^*, is going to be defined as follows:

$$T_x^* = \begin{bmatrix} \text{Last row previous piece } T_{(x-1)} \\ \text{Current piece } T_x \end{bmatrix}$$

$$T_x = I(i,j), \;\; i \in [(x \times 150) + 1, (150 \times x) + 150] \;\text{ and }\; x = [0, X = H/150]$$

input : Super high resolution image
output: Localization of the drops in the low resolution image

if *image upside-down* **then**
  Rotate and Mirror the image;
  Crop the image;
else
  Crop the image;
end
Image thresholding;
Image normalization;
Subsample the image (super-pixel of size 10x10): I;
Construction of the graph G;
Definition of the source node and execution Dijkstra algorithm: d ;
Assign to each white-pixel (node) the distance to the source node: H;
Search for local maxima and remove false-positive detections: D and P;
Root type classification: α, β, and γ.

Algorithm 2: Detection algorithm for low resolution images

- At each iteration, we create a new piece T'_x, with the same size as the original piece (we remove the additional row from the previous piece), where each white pixel is going to store the distance to s ($T'(i(v), j(v)) = d(v), i, j \in V$, where V is the collection of nodes per piece). Finally, all pieces are concatenated having one single image equivalent to the image H in the previous analysis, but with a higher resolution ($H = \bigcup_{i=1}^{X} T_x$).
- In the high resolution case, to make the differentiation between the three types of roots, we have to do an additional step despite the process describe for the differentiation between drops. In this additional step, we unify the paths across pieces to obtain the whole path from the source node to the drop/tip root. This process is explained in detail in Appendix A.

The detection algorithm for the high resolution image is summarized in Algorithm 3.

Resolution Selection

After the implementation of both resolutions, we have to select one to execute the tracking process. In Figure 2.5 we can observe the time and number of drops collected in the high and low resolution cases. Both graphs are strongly related, due to the fact that the amount of time required to analyze each image increases when the amount of drops per image increases. However, we can observe that the time required for the low resolution case is always lower. The worst case shows a difference of 30 seconds approximately. On the other hand, we can observe that the amount of drops detected in the high resolution case is exponentially higher. The main reason is that the amount of branches detected is better with a higher resolution, due to the proximity between this kind of roots (Figure 2.6). In the low resolution case some of the branches are mixed and their detection is worst.

Due to this results, we have selected the high resolution option to continue in the following chapters. This decision is made based on the fact that the amount of drops detected is higher at the expense of a higher execution time, but this increase of time is low in comparison of the advantages that we have with a higher accuracy on the detection of tips/drops.

In this Chapter, we have described our main contribution to the multi-object tracking problem, which is the exploitation of the shortest path obtained for every node in the image in a graph-based

input : Super high resolution image
output: Localization of the drops in the low resolution image

if *image upside-down* **then**
  Rotate and Mirror the image;
  Crop the image;
else
  Crop the image;
end
Image thresholding;
Image normalization;
Subsample the image (super-pixel of size 5x5): I;
Divide the image into pieces (X pieces): T_x;
Construction of the graph of the first piece ($x = 0$): G_0;
Definition of the source node s and execution Dijkstra algorithm d_0;
Assign to each white-pixel (node) of the piece the distance to the source node, and store the piece in a cell: T'_0;
for $x = 1 \ldots X$ **do**
  Add to the piece the last row of the previous piece ($x - 1$);
  Definition of a virtual node s_v, and construction of the graph: G_x;
  Execution Dijkstra algorithm d_x;
  Assign to each white-pixel (node) of the piece the distance to the source node, and store the piece in a cell: T'_x;
end
Union of all pieces (where each white-pixel represent its distance to the source node) in a single matrix, generating the final image: H;
Search for local maxima and remove false-positive detections: D;
Unification paths through pieces: P;
Root type classification: α, β, and γ

Algorithm 3: Detection algorithm for high resolution images

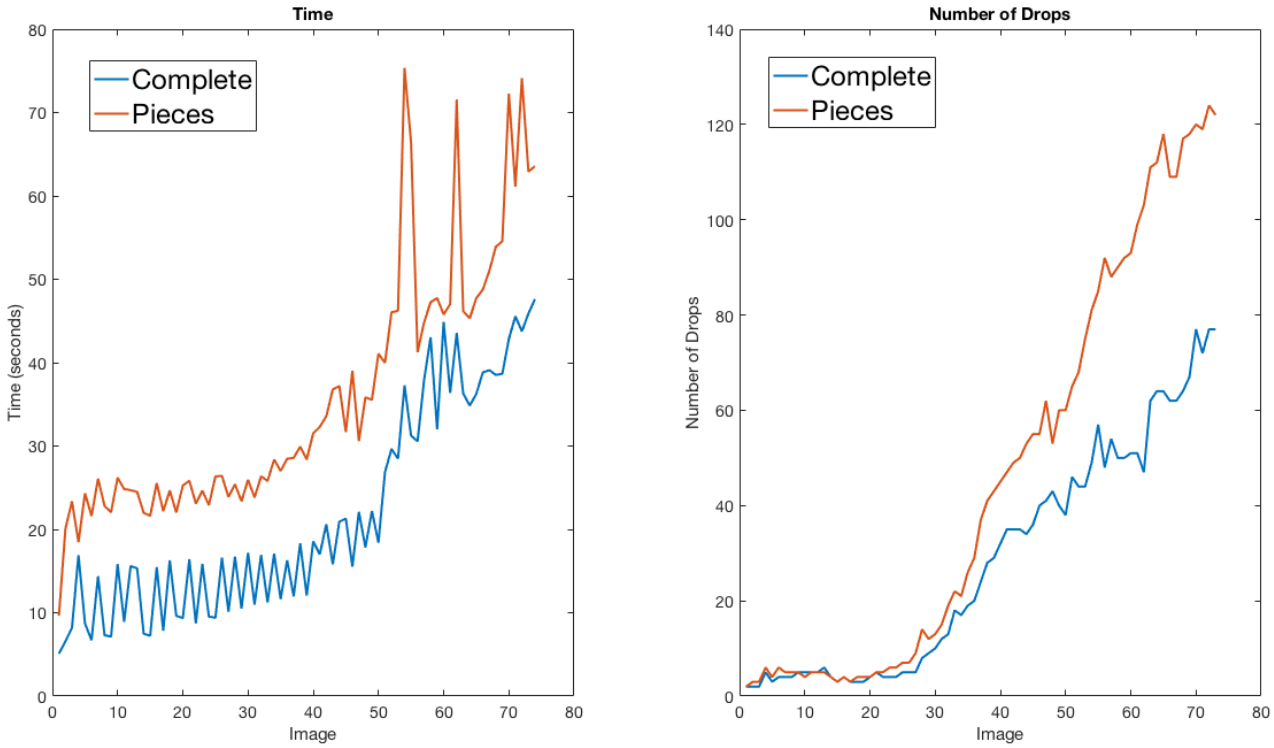


Figure 2.5: On the left, we can observe the time required to execute the detection and classification code for both cases high resolution (complete) and low resolution (pieces). On the right, we can observe the amount of drops detected in both cases high resolution (complete) and low resolution (pieces).

framework. In this first step, we have computed the shortest path to every node (white pixel) in the image through the description of a graph that describes the connectivity of the pixels of the image. The shortest paths have been computed via the implementation of the Dijkstra algorithm. Then we have used the two main features that characterized tip roots (localization within a root and its neighborhood) to detect them. Finally, we have classified all the detections (primary, seminal, and branches) based on the analysis of the intersection between the shortest paths.

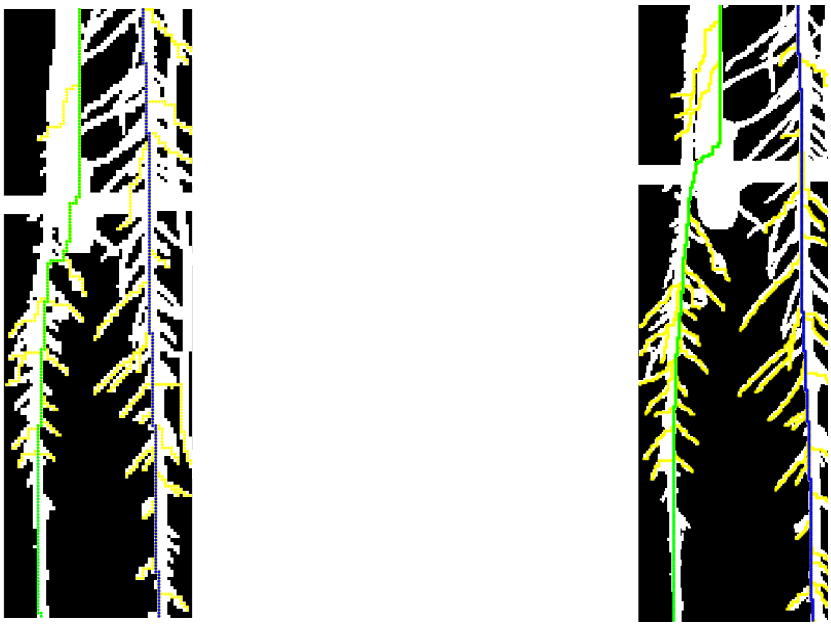


Figure 2.6: On the left, we can observe the low resolution case where the amount of branches (yellow) detected is lower. On the right, we can observe the high resolution case where the detection of branches is more accurate.

Chapter 3

Root tips/drops tracking

3.1 Objectives and priors

In this Chapter, we describe the tracking process as the association of each classified detection across time to obtain the track of every root tip. Figure 3.1, shows an example of the solution to the tracking problem. To do so we have used the Hungarian algorithm to associate detections across time based on the definition of two different cost matrices. Each cost matrix is defined in a different scenario, with the aim to analyze and compare the performance of two different approaches to the same problem. The first scenario defines a forward cost, based on the definition of a Gaussian distribution where we link nodes from previous frames to nodes in the current frame. The second scenario defines a backward cost, based on the minimum distance between nodes from previous frames to the paths of the nodes of the current frame.

Formally, a track L is defined to be a collection of chained detections, i.e., $L = (v^1, v^2, \ldots, v^{|L|})$, $|L|$ being the length of the track. Notice that the chain is ordered in time such that $\mathrm{t}_{v^1} < t_{v^2} < \cdots < t_{v^{|L|}}$. We describe an algorithm based on the fact that the position of the tips/drops has a slight variation between frames, and the number of missed detections is very low. This property allow us to use a method that works with pair of consecutive frames, rather than a long time window. As a consequence, we do not need to bridge long gaps (long periods without detections). An example of an algorithm that works over long time windows is the one described in [8], which solve the tracking problem using the k-shortest paths algorithm. It is based on the definition of a *probabilistic occupancy map*, that is a set of probabilities of presence of objects at a discrete set of locations at each time step independently. However, in our case, for the association of detections between frames we are going to use the Hungarian algorithm [4], [5].

The tracking process is based on the exploitation of two main priors:

– A growing model to predict the position in future frames from past frames (forward cost);

– A path consistency criterion to check that previous tips are not far from the shortest-path computed from the current tip (backward cost).

3.2 Tracking Formalism

For the application of the Hungarian algorithm to our problem, we have to describe two main elements: the entities that we are going to associate, and the cost defined to associate those entities. In our case, we associate detected drops at different time instants. As a result, we match detections at time t (rows of the cost matrix), with detections that belongs to previous frames at time $(t - x)$

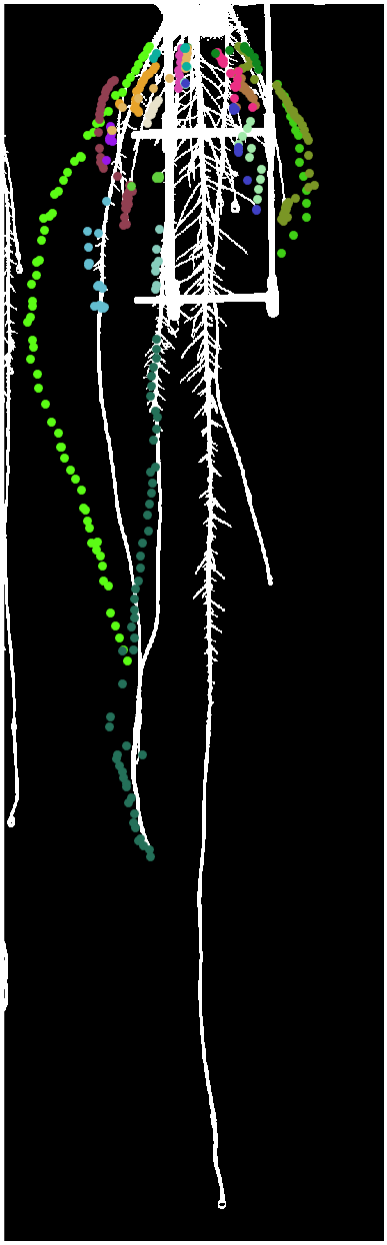


Figure 3.1: Each color represents the set of detections along time that correspond to a (fraction of) seminal root.

(columns of the cost matrix). Due to the small amount of missed detection between consecutive frames the variable x is going to be equal to 1.

3.2.1 Hungarian Algorithm

The classical solution to the assignment problem is given by the Hungarian or Kuhn-Munkres algorithm, originally proposed by H. W. Kuhn in 1955 [6] and refined by J. Munkres in 1957 [7]. The Hungarian algorithm, is a combinatorial optimization algorithm that solve the matching problem between detections. The best way to understand the Hungarian algorithm is through a simple example (Table 3.1), where we want to assign three jobs to three different people at the lower cost. We define a cost matrix C, where the element in the i-th row and j-th column represents the cost of

assigning the j-th job to the i-th worker. We have to find an assignment of the jobs to the workers that minimizes the cost.

	Job 1	Job 2	Job 3
Person 1	$2	$3	$3
Person 2	$3	$2	$3
Person 3	$3	$3	$2

Table 3.1: General Hungarian Algorithm example. The association problem solution is represented in green, where the lower-cost is reached.

After the implementation of the Hungarian algorithm, we can observe that we have obtained the minimum cost ($ 6) when we assigned the job 1 to person 1, job 2 to person 2, and job 3 to person 3. The definition of the cost matrix C is crucial to the correct behavior of the Hungarian algorithm.

Table 3.2 shows a example of the Hungarian algorithm applied to our tracking problem. We want to associate the detections from time t ($V^t = \{v_1^t, v_2^t, v_3^t, v_4^t\}$) with the detections from the previous frame $(t-1)$ ($V^{(t-1)} = \{v_1^{(t-1)}, v_2^{(t-1)}, v_3^{(t-1)}, v_4^{(t-1)}, v_5^{(t-1)}\}$) at the minimum cost.

		$(t-1)$				
		$v_1^{(t-1)}$	$v_2^{(t-1)}$	$v_3^{(t-1)}$	$v_4^{(t-1)}$	$v_5^{(t-1)}$
t	v_1^t	inf	**1**	inf	inf	inf
	v_2^t	inf	inf	inf	**1**	inf
	v_3^t	**1**	inf	inf	inf	inf
	v_4^t	inf	inf	inf	inf	inf

Table 3.2: Hungarian Algorithm example applied to our tracking problem. The association problem solution is represented in green, where the lower-cost is reached.

When we implement the Hungarian algorithm, the lower-cost (3) is obtained when the node v_1^t is associated with the node $v_2^{(t-1)}$, the node v_2^t is associated with the node $v_4^{(t-1)}$, the node v_3^t is associated with the node $v_1^{(t-1)}$, and the node v_4^t is not associated with any previous drop. This way, frame by frame, we can create the tracks that defines the growth of the plant at every time instant. In the case v_4^t, the drop cannot be linked with drops from previous frames, it could be for example a new detection on the sequence of images.

Initially, we have the drops from the whole dataset stored in D, but, as we can observe in the example, to implement the Hungarian algorithm we have to distinguish at each iteration between two sets of drops. The collection of drops that belong to the current frame, denoted as C_t, and the collection of drops that belong to previous frames, denoted as $S_{(t-1)}$. Therefore, the Hungarian problem consists in the association between C_t, and $S_{(t-1)}$, both sets extracted from D. In Table 3.2 we distinguish C_t and $S_{(t-1)}$ as:

$$C_t = \{v_1^t,\ v_2^t,\ v_3^t,\ v_4^t\} \qquad S_{(t-1)} = \{v_1^{(t-1)}, v_2^{(t-1)}, v_3^{(t-1)}, v_4^{(t-1)}, v_5^{(t-1)}\}$$

When we run the Hungarian algorithm, we have as a result a $1 \times |C_t|$ vector A. Each element a_i contains the index of $S_{(t-1)}$ that has been associated with the i-th detection of C_t. In relation with the example above, $A = \{2, 4, 1, 0\}$, which means that v_1^t is associated with the node $v_2^{(t-1)}$, the

node v_2^t is associated with the node $v_4^{(t-1)}$, etc. These set of associated nodes are stored to create frame-by-frame every track $T = \{S_{(t-1)}, C_t\}$, which are stored in M.

However, if $a_i = 0$, it means that the i-th drop of the current frame C_t has not been associated with any drop from the previous frame. All the nodes from $S_{(t-1)}$ that have not been associated, along with all nodes from C_t, are added into S_t in the next iteration to try to associate them in the following iterations until a certain time threshold τ. Therefore, if the node is associated in $t < \tau$ then the node is added into its corresponding track T. On the contrary, a new track will be created.

The algorithm is going to iterate N times, which corresponds to the amount of frames in the dataset. Adding new nodes to the tracks is based on the search of nodes from $S_{(t-1)}$ in M. We search these nodes because they are equal to the nodes in C in the previous iteration ($S_t = C_{(t-1)}$). This overlapping allows the iterative creation of the tracks. Figure 3.2 shows a simple example of how each track is created along time.

	Detection 1	Detection 2	Detection 3	Detection 4
Track 1	(t_1-x)	(t_1) = (t_2-x)	(t_2)	
Track 2	(t_1-x)	**(t_1) = (t_2-x)**	(t_2) = (t_3-x)	(t_3)

Figure 3.2: Each line represents a different track, where every rectangle represent each detection within the track. Each color represents the nodes added to each track in each time instant. As we can observe the nodes added at every t, corresponds to the time (t-x) in the following iteration.

3.2.2 Cost Matrix

The power of the Hungarian algorithm is based on the definition of the cost matrix. It is very important to define a cost matrix that support the correct association between detected drops. The behavior of the Hungarian algorithm is analyzed based on the two priors mentioned at the beginning of this Chapter. Two scenarios are defined:

- Scenario 1: Forward cost. Definition of a Gaussian distribution that predicts the position of a drop in a future frame, given its position in the current frame.
- Scenario 2: Backward cost. The cost is based on the distance between a past detection and the shortest path connecting a current detection to the source. This cost matrix is referred to as a backward cost, because the distance is measured from detections in the current frame to detections in the past ones.

Scenario 1

The first scenario described is based on a model that predicts a detection in the future. First of all, we have analyzed the behavior of the Hungarian algorithm when we implement an isotropic distribution, which is performed with no information a priori. Afterwards, we have analyzed the model based on the definition of an anisotropic distribution to account for the fact that the root is expected to grow in a specific direction, known a priori from the shortest path orientation at current root tip.

input : D, N
output: M

Initialization: $T = \{\}$;
for $t = 1 \dots N$ **do**
  $S_{(t-1)} \leftarrow D(t-1)$;
  $C_t \leftarrow D(t)$;
  $A \leftarrow HungarianAlgorithm(S_{(t-1)}, C_t)$;
  for $j = 1 \dots |A|$ **do**
    if $A(j)$ *equal to* 0 **then**
      $t' \leftarrow getTime(S_{(t-1)})$;
      if $(t - t') > \tau$ **then**
        $T \leftarrow S_{(t-1)}(A(j))$;
      else
        $S_t \leftarrow [S_{(t-1)}(A(j))]$;
      end
    else
      $T \leftarrow [C_t(j), S_{(t-1)}(A(j))]$;
    end
  end
  $M \leftarrow T$;
end

Algorithm 4: Tracks creation algorithm.

For the definition of the cost matrix C we have to translate the probabilities obtained from the distribution mentioned above into costs. Therefore, we use a function $C = f(P)$, based on the fact that higher probabilities should provide lower costs. The function selected is $f(P) = (-\log_{10} P)^{1/n}$, which tends to infinity when the probability P is too low.

Figure 3.3 shows several examples for different values of $n = \{1, 2, 4, 6\}$. As we can observe, the lower is n, for small values of P, the higher is $f(P)$. As a consequence a small difference in P can produce a big difference in $f(P)$. This behavior will have a big impact on the implementation of the Hungarian algorithm, because it could lead to a bad association between detections. If we select a value of $n = 2, 6$, it provides big and small differences respectively, which is going to induce an error during the association process. If the difference is too small, the Hungarian algorithm can choose both detections indifferently because it is not going to have a big impact in the final cost. On the other hand, if the difference is too big, the Hungarian algorithm could choose the incorrect detection because if the other detection is chosen it is going to have a big impact on the final cost. This is why we should select carefully the value of n. For example, $n = 4$, provides a good behavior because it provides a difference sufficiently big when small probabilities are close. In addition, to avoid possible errors we set a certain threshold p, which is going to saturate the function $f(P)$ when the probability is below p. Therefore, if $P < p$, then $f(P) = f(p)$.

As we mentioned before, we defined two distributions: one isotropic, and the other anisotropic. The isotropic distribution reflects that the future position of the tip should remain close to the current one. It is defined without a previous knowledge of the position of the drop in the future, so we do not predict where the drop at time $(t - x)$ is at time t. Therefore, we define a isotropic Gaussian distribution characterized by the fact that the probability distribution is equal in all directions. Its

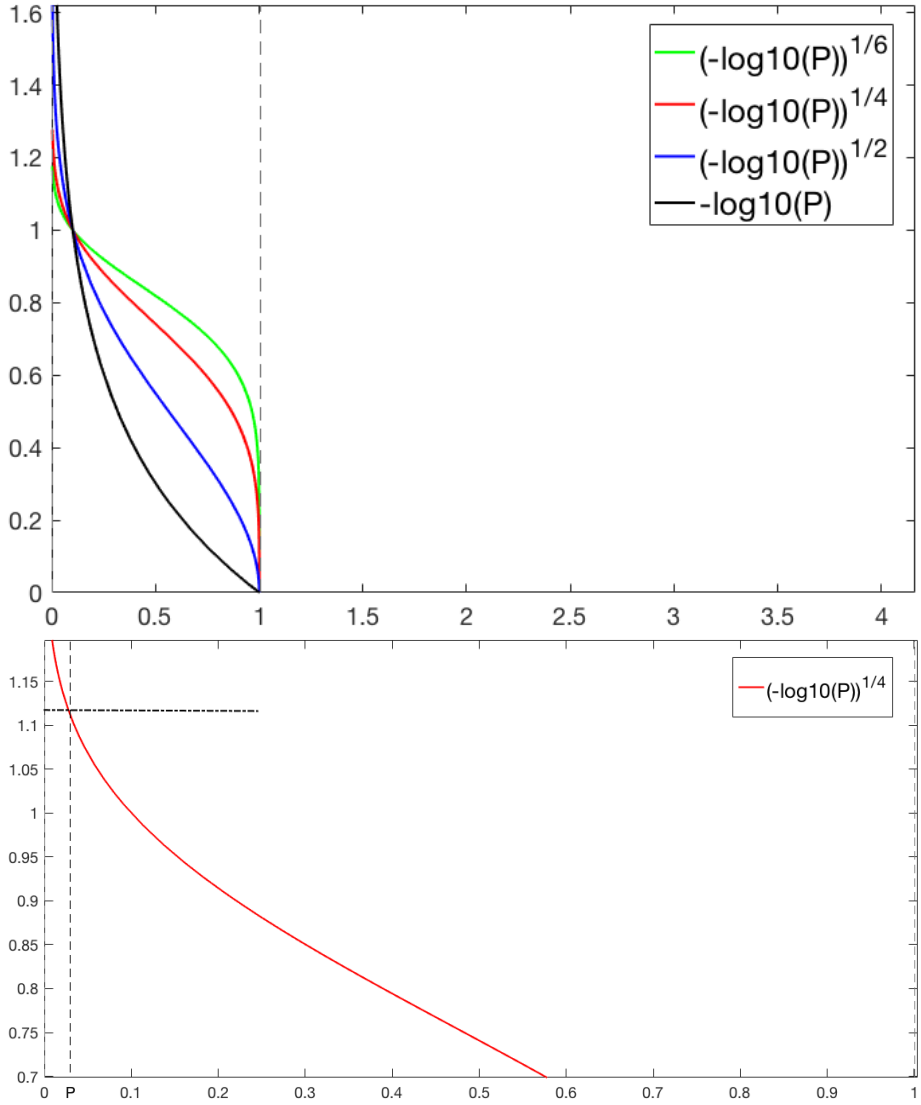


Figure 3.3: Up, we can observe the function $f(P) = (-\log_{10} P)^{1/n}$, for several values of $n = \{1, 2, 4, 6\}$. The lower is n, for small values of P, the higher is $f(P)$. As a consequence a small difference in P can produce a big difference in $f(P)$. Down, we can observe in the case $n = 4$ how the function $f(P)$ is saturated for a certain set of probabilities up to a threshold p.

mathematical expression is:

$$P(v^{(t-x)} = v^t | \mu_{(t-x)}, \sigma^2) = \frac{1}{2\pi\sigma_i\sigma_j} \; exp - \Big(\frac{(i - \mu_{i,(t-x)})^2}{2\sigma_i^2} + \frac{(j - \mu_{j,(t-x)})^2}{2\sigma_j^2} \Big)$$

Where $\mu_{(t-x)} = [\mu_{i,(t-x)}\mu_{j,(t-x)}] = [v_i^{(t-x)}, v_j^{(t-x)}]$ is the mean, and it established the origin of the gaussian distribution at the location of the drop at time $(t-x)$; and σ^2 is the variance that measures how far a set is spread out from the mean at time $(t-x)$. The variance of the distribution is equal in both directions $\sigma_i^2 = \sigma_j^2$.

On the other hand, for the definition of the anisotropic distribution (Figure 3.4), we use some information a priori to predict where the drop at $(t-x)$ is going to be at time t. This prediction

is done based on the knowledge of the shortest path orientation $\bar{u}$ at current root tip. If $P(v) = \{p_1, p_2, \ldots, p_{(L-1)}, p_L\}$ is the set of nodes that shape the path that corresponds to the node v, and L is the length of the path. We can obtain the orientation of the shortest path $\bar{u}$ and the predicted node v_p as:

$$d = p_{(L-25)} - p_L \qquad \bar{u} = \frac{d}{|d|} \qquad v_p = p_L - (x \times \bar{u} \times s)$$

Where d defines the line where the predicted node is going to be, x defines how far in time we are from the current node $(t-x)$, and s is the growing rate of the root. The growing rate value is obtained from the mean of a set of n distances calculated from nodes of consecutive frames: $s = \frac{1}{n} \sum_{i=1}^{n} d(v^{(t-x)}, v^t)$. Once we have calculated the predicted node, we can implement the anisotropic Gaussian distribution centered around this node (Figure 3.7). The mathematical expression of the anisotropic Gaussian distribution is:

$$P(v_p = v^t | \mu, \sigma^2) = \frac{1}{2\pi\sigma_i\sigma_j} \; exp - \Big(\frac{(i-\mu_i)^2}{2\sigma_i^2} + \frac{(j-\mu_j)^2}{2\sigma_j^2} \Big)$$

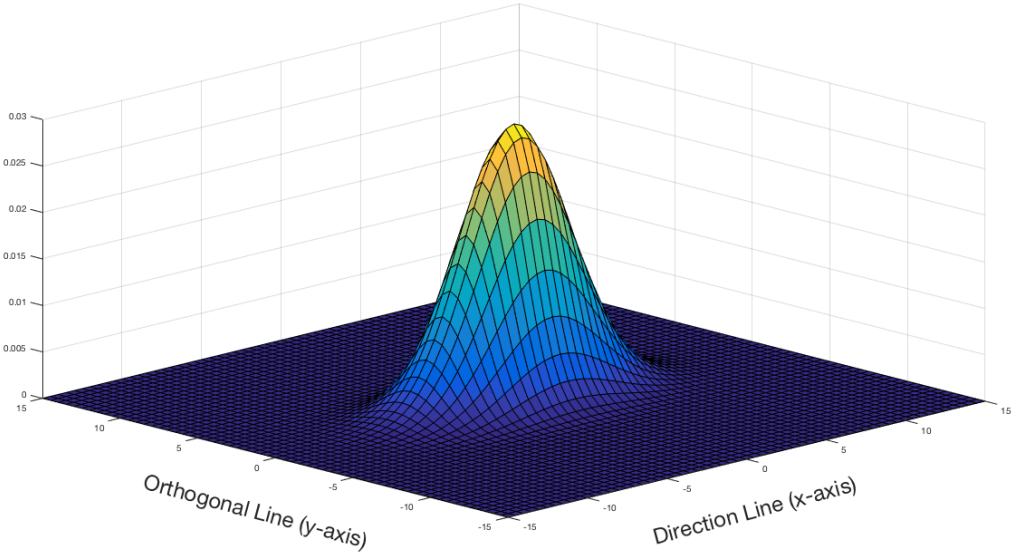


Figure 3.4: Anisotropic Gaussian distribution centered around zero ($\mu = [0, 0]$) and variance $\sigma^2 = [2, 6]$.

Where $\mu = [\mu_i, \mu_j] = [v_{p,i}, v_{p,j}]$ is the mean, and it established the origin of the gaussian distribution in the position of the predicted node; and $\sigma^2 = [\sigma_i^2, \sigma_j^2]$ is the variance that measures how far a set is spread out from the mean at time t. Each component of the variance is defined based on two distances (Figure 3.5), and they are multiplied by a factor $\sqrt{x}$ depending on how far the predicted point v_p is from the current frame $(t - x)$. This factor is applied over σ^2 to take into account the fact that the further the original node used to define the predicted node v_p is from t, the higher is the uncertainty.

On one hand, $\sigma_j^2 = \sigma_j^{2*}\sqrt{x}$ is based on the distance that links the predicted node with the cut-off point ($A = [a_i, a_j]$) between the direction line and the perpendicular line that includes the drop from the previous frame (green distance in Figure 3.5): $d_d(v_p, A) = \sqrt{(v_{p,i} - a_i)^2 + (v_{p,j} - a_j)^2}$. On the other hand, $\sigma_i^2 = \sigma_i^{2*}\sqrt{x}$ is based on the distance that links perpendicularly the drop from the previous

frame with the cut-off point A (magenta distance in Figure 3.5): $d_o(v^t, A) = \sqrt{(v_i^t - a_i)^2 + (v_j^t - a_j)^2}$.

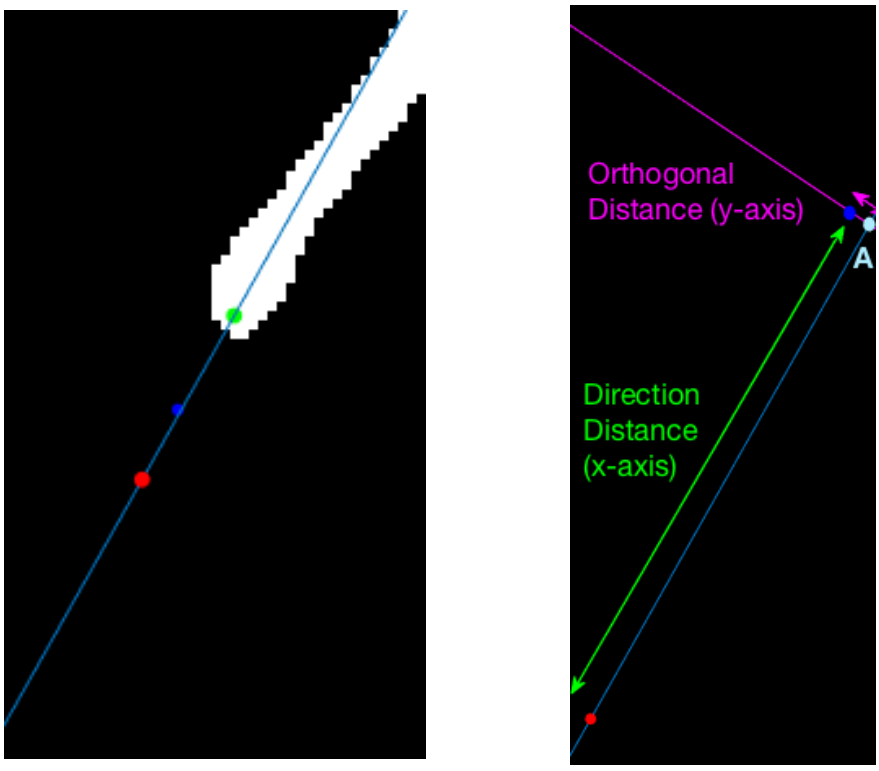


Figure 3.5: On the left, we can observe the prediction node v_p in red, and the nodes at time $(t - x)$ and t in green and blue respectively. On the right, a representation of the distances defined to describe $\sigma^2 = [\sigma_i^2, \sigma_j^2]$. Where σ_i^2 is the orthogonal distance in the y-axis in magenta, and σ_j^2 is the direction distance in the x-axis in green. A (clear-blue) is the cut-off point between the direction line and the perpendicular line that includes the drop from the previous frame

The final value of $\sigma^2 = [\sigma_i^2, \sigma_j^2]$ is calculated as the variance of the distribution obtained for a set of n absolute distances calculated from nodes of consecutive frames (Figure 3.6).

To guarantee that the gaussian distribution and the drops from the current frame are expressed in the same reference system we have to perform an additional step. We have to rotate all the drops from the current frames $V^t = \{v_1^t, v_2^t, \dots\}$ around the predicted drop v_p (Figure 3.7). The rotation of the drops is performed as follows:

$$R = \begin{bmatrix} cos(\theta) & -sen(\theta) \\ sen(\theta) & cos(\theta) \end{bmatrix}$$

$$V^* = (R \times ([V_i^t, V_j^t] - [v_{p,i}, v_{p,j}])) + [v_{p,i}, v_{p,j}]$$

Where R is the rotation matrix, and V^* is the collection of nodes rotated (clockwise) an angle θ (angle between the horizontal axis and the line that describes the orientation of the shortest path).

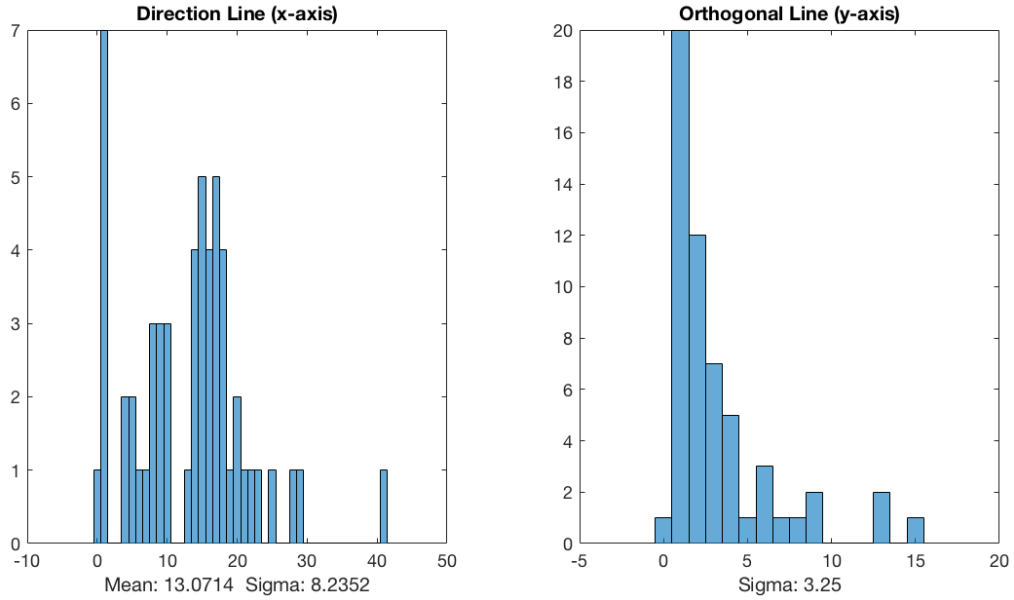


Figure 3.6: Histograms computed for $x = 1$ to obtain the parameters that defines the anisotropic gaussian distribution $\sigma^2 = [\sigma_i^2, \sigma_j^2]$. Where σ_i^2 corresponds to the orthogonal line, and σ_j^2 to the direction line.

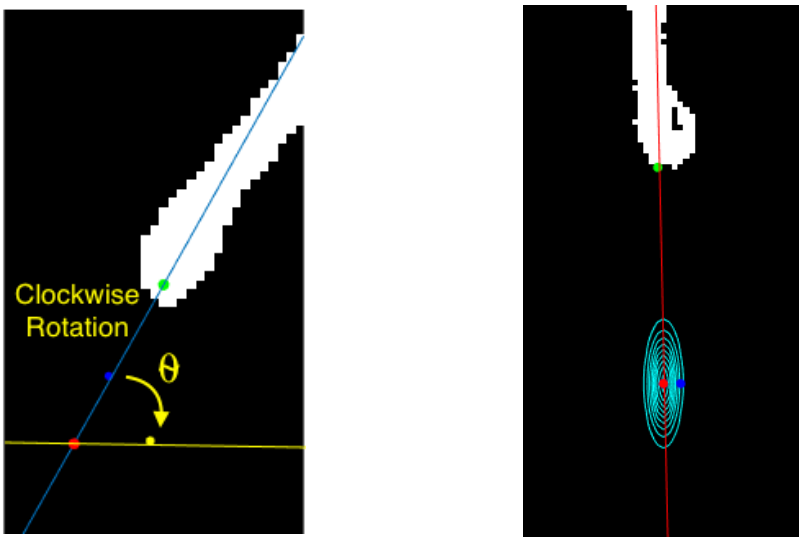


Figure 3.7: On the left, rotation θ degrees of the drop from the current frame v^t (blue) around the predicted drop v_p (red). On the right, the anisotropic Gaussian distribution is plotted centered around the predicted node v_p, where its mean is $\mu = [v_{p,i}, v_{p,j}]$ and its variance is $\sigma^2 = [\sigma_i^2, \sigma_j^2]$.

Scenario 2

The second scenario is described to study the association problem with nodes from previous frames (backward approach). The cost is based on the distance between a past detection and the shortest path connecting a current detection to the source. This cost matrix is referred as a backward cost, because the distance is measured from detections in the current frame to detections in the past ones.

For the definition of the cost matrix C we have to translate the distances obtained into costs. Therefore, we use a function $C = f(d)$, based on the fact that higher distances should provide higher costs. The function selected is $f(d) = d^g$, which tends to infinity when the distance d increases. Figure 3.8 shows several examples for different values of $g = \{2, 3, 4, 6\}$. As we can observe, the higher is g, for high values of d, the higher is $f(d)$. As a consequence if g is too high, a small difference in d (for high values of d) can produce a big difference in $f(d)$. This behavior will have a big impact on the implementation of the Hungarian algorithm, because it could lead to a bad association between detections. Therefore, we should select carefully the value of g. For example, $g = 4$, provides a good behavior because it provides a difference sufficiently big when high distances are close.

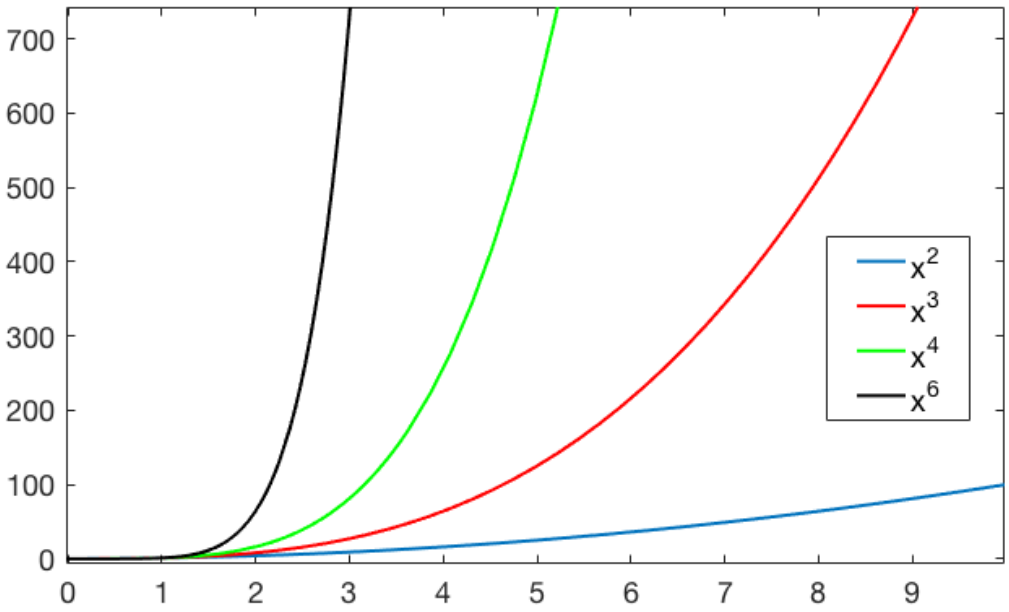


Figure 3.8: We can observe the function $f(d) = d^g$, for several values of $g = \{2, 3, 4, 6\}$. The higher is g, for high values of d, the higher is $f(d)$.

As we mentioned, this scenario is based on the distance from the nodes from previous frames to the shortest path $P(v^t)$ of a node v^t in the current frame t. Where $P(v^t) = \{p_1, p_2, \ldots, p_{(L-1)}, p_L\}$ is the set of nodes that shape the path that corresponds to the node v^t, and L is the length of the path. We do not calculate the distance to all the nodes in $P(v^t)$, we have limited the amount of analyzed nodes to the ones closer to the current drop v^t. This way, given a certain threshold T, we only take into account the nodes $P(v^t) = \{p_{(L-T)}, \ldots, p_{(L-1)}, p_L\}$ from the shortest path (Figure 3.9). The distances are stored in a $N \times K$ matrix S, where $s_{n,k}$ corresponds to the distance from the s-th nodes in the frame at time $(t - x)$ to the k-th node that belongs to the shortest path.

$$P(v^t) = \{p_{(L-T)}, \ldots, p_{(L-1)}, p_L\}, \;\; p = [p_i, p_j]$$

$$S(n,k) = d(v_n^{(t-x)}, p_k) = \sqrt{(v_{n,i}^{(t-x)}) - p_{k,i})^2 + (v_{n,j}^{(t-x)}) - p_{k,j})^2}, \;\; \text{if } n \neq k$$

Then, we obtain for each row the minimum distance to the node v^t,

$$d_{i,j} = \arg\min_k S(n,k), \;\; i \in \{V^{(t-x)}\}, \;\; j = 1 \ldots N$$

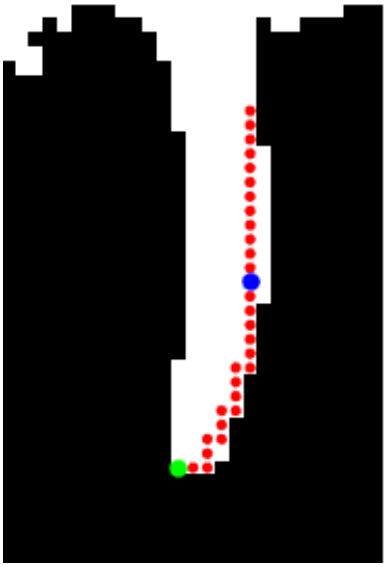


Figure 3.9: We can observe in green the node from the current frame, in red the set of nodes extracted from the shortest path connecting the node in green to the source, and in blue the node from the previous frame. In this example, the distance of the node from the previous frame to the path is equal to zero, due to the fact that the blue node belongs to the path.

Finally, the cost matrix C is defined as follows: $C(i,j) = f(d_{i,j}) = d^g_{i,j}$.

In this Chapter, we have described how we exploit our main contribution to the multi-object tracking problem (shortest paths). The tracking process is based on the utilization of the Hungarian algorithm, which minimize the association cost between detections of consecutive frames. This method can be used because the number of missed detections is very low. As a consequence, we do not need to bridge long gaps (long periods without detections). The power of the Hungarian algorithm lie on the definition of a cost matrix. We have defined two different scenarios where two different cost matrices are described. On one hand, a forward cost based on a model that predicts a detection in the future, trough a anisotropic gaussian distribution. On the other hand, a backward cost based on the distance between a past detection and the shortest path connecting a current detection to the source.

Chapter 4

Validation

4.1 Evaluation Metrics

In this Chapter we are going to assess the correctness of the multi-object tracker described by quantifying its performance. Some evaluation methods, such as [10], compute errors at each frame and then aggregate them along time. Two quantities are defined: *Multiple Object Tracking Precision* ($MOTP$) and *Multiple Object Tracking Accuracy* ($MOTA$). $MOTP$ measures the trackers ability to localize a target, whereas $MOTA$ measures the number of errors committed by the tracker during tracking.

For the implementation of the multi-object tracking evaluation protocol, we need at hand (i) tracker outputs or hypotheses, (ii) a set of ground-truth objects, and (iii) a measure of distance between the estimated and actual state of the target. Let $O_t = \{o_1 \dots o_m\}$ be the collection of tips/drops ground-truth in each frame. The analysis is done every 10 frames. The i-th tip/drop object $o_i = (l_i, x_i)$ has a label l_i and a location x_i. Similarly, $G_t = \{g_1 \dots g_n\}$ represent the hypothesis output by the tracker in each frame. The i-th hypothesis $g_i = (l_i^*, z_i)$ has a label estimate l_i^*, and a location z_i.

The correspondence between a tip/drop object o_i and a hypothesis h_j is based on the calculation of the Euclidean distance $d_{i,j}$ between them. The association should not be made if their distance exceeds a certain threshold T. In the following, we refer to correspondences as valid if $d_{i,j} < T$. The object-hypothesis mapping is stored in the $M_t = \{(o_i, g_i)\}$ matrix. If a correspondence (o_i, g_k) is made that contradicts a mapping (o_i, g_j) in M_{t-1}, replace (o_i, g_j) with (o_i, g_k) in M_t. Count this as a mismatch error and let mm_t be the number of mismatch errors for frame t.

Some other variables are defined to evaluate the performance of the algorithm:

- Number of matches found at each time instant (c_t).
- Number of false-positive detections (fp_t). All remaining hypothesis that are not linked to any object are considered false-positives.
- Number of miss-detections (m_t). All remaining objects that are not linked to any hypothesis are considered misses.
- Number of objects present at time instant (g_t).

This way, MOTP and MOTA expressions are as follows:

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_t c_t}$$

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mm_t)}{\sum_t g_t}$$

The $MOTA$ can be seen as composed of 3 error ratios: the ratio of misses in the sequence, computed over the total number of objects present in all frames, the ratio of false positives, and the ratio of mismatches.

$$\overline{m} = \frac{\sum_t m_t}{\sum_t g_t} \qquad \overline{fp} = \frac{\sum_t fp_t}{\sum_t g_t} \qquad \overline{mm} = \frac{\sum_t mm_t}{\sum_t g_t}$$

4.2 Evaluation Metrics Results

The proposed algorithm has been evaluated on three datasets, whose general properties are described in Table 4.1, provided by the acquisition system described in section 1.2. The detection and classification process, along with the scenarios within the tracking process have been implemented on Matlab running on a 1,7 GHz Intel Core i7, with a 8 GB RAM. In addition, for the implementation of the Hungarian algorithm it has been used the algorithm provided in [9].

Average number of frames	147
Resolution of frames [pixeles]	15300 x 4096
Average time between frames	2 hours

Table 4.1: Properties of the dataset

The running time of the algorithm for each phase is shown in Table 4.2.

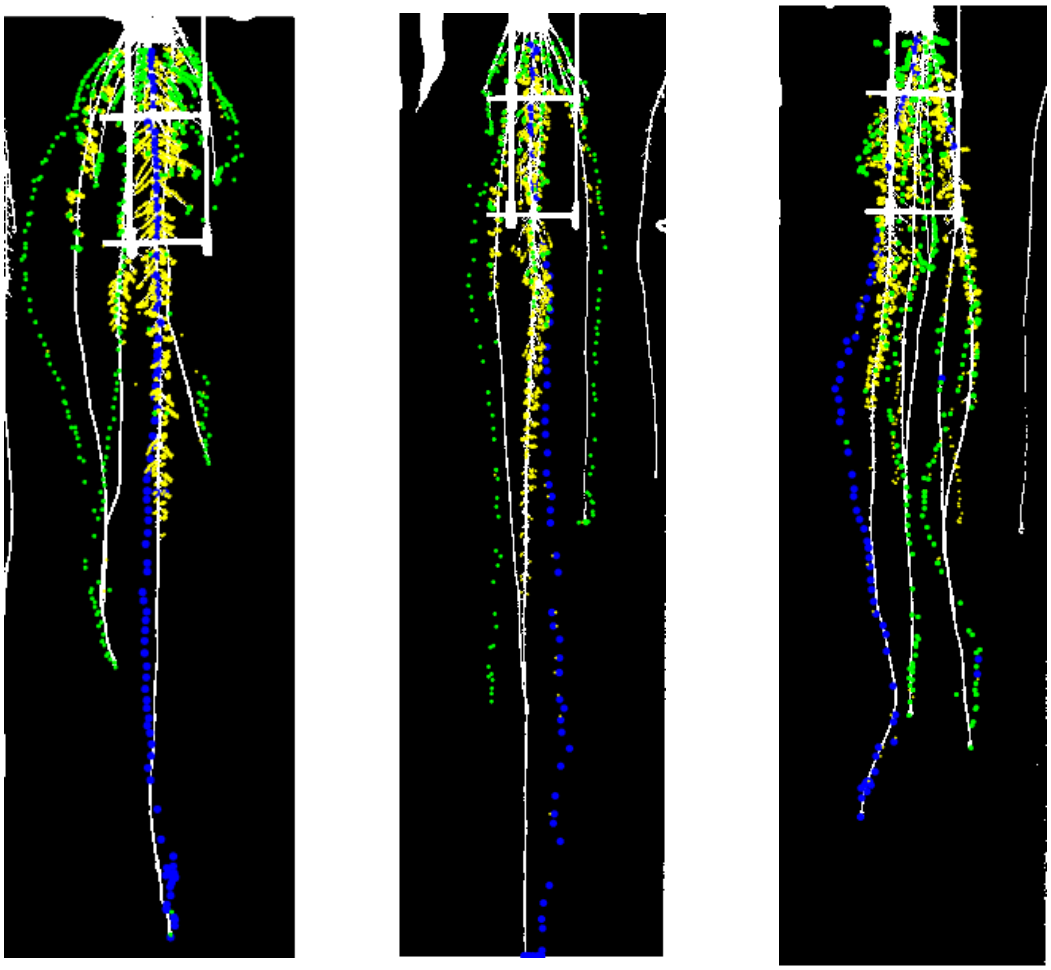
Detection and Classification Phase	$2,6299 \times 10^3$s (43,72 min)
Tracking Phase (Forward Case)	224,2741 s (3,73 min)
Tracking Phase (Backward Case)	27,6927 s (0,46 min)

Table 4.2: Running time of each phase of the algorithm.

The first phase of the algorithm is the detection and classification of all the tips/drops in three types of roots (primary, seminals and branches). In Figure 4.1 we can observe all the detections classified in each dataset.

The tracking process is performed in two different scenarios based on the definition of two different costs: forward and backward costs. In Figures 4.2, 4.3, and 4.4 we present the tracking solution of the first dataset for each root type.

To evaluate the results obtained in the Figures presented, we have implemented the evaluation protocol defined in the previous section. In Table 4.3 we analyze globally the performance of the algorithm, and in Table 4.4 we analyze the performance of the algorithm for each root type. We make



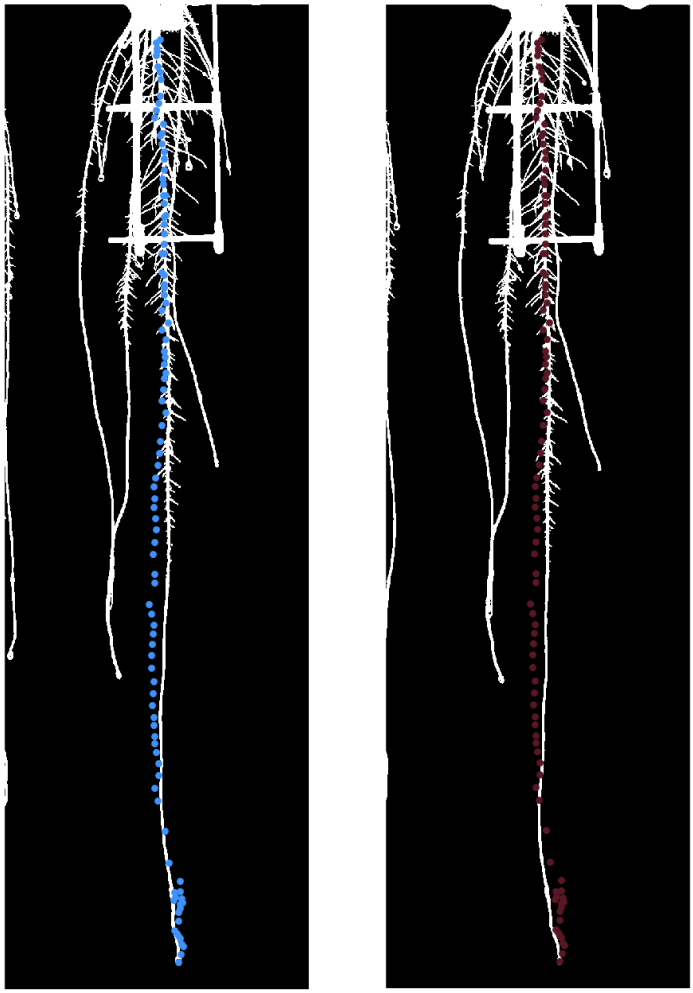
(a) Dataset 1 (b) Dataset 2 (c) Dataset 3

Figure 4.1: Classification of all the detections in each dataset: primary (blue), seminals (green), and branches (yellow).

this differentiation to evaluate on one hand the performance of the tracking process by analyzing all tracks simultaneously, and on the other hand the performance of the classification process by analyzing each root type independently. In both tables we analyze the performance of the algorithm in the case of the forward and backward costs. In addition, to have a better measurement of the accuracy of the algorithm, we have filled the gaps of each track (frames without detections) by interpolating these values from detections of previous and future frames within the track. The results of the metric obtained for each time instant for each dataset are in Appendix C.

Dataset 1 is the one with the best results due to the definition of its root system, which makes a good differentiation between primary and seminals, and the space between branches is enough to detect them correctly. On the other hand, the disposition of the roots in Dataset 2 and 3 makes its analysis more difficult. In Dataset 3 most of the seminal roots are merged and the space between seminals and branches is very small, which leads to some errors during the classification and tracking process. In Appendix C we can observe a set of Figures that shows the tracking solution for each Dataset.

As can be seen, in all the datasets, the tracker is very precise at estimating locations (average



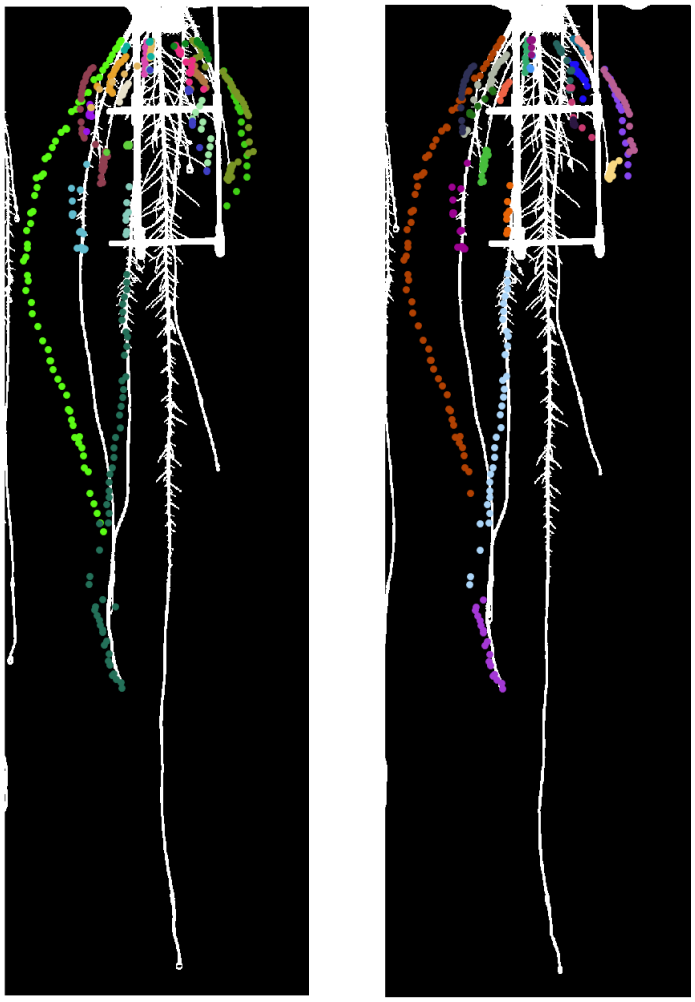
(a) Primary Root (b) Primary Root

Figure 4.2: Dataset 1. On the left. (a) Tracking of primary root (scenario 1). On the right (b) Tracking of primary root (scenario 2). Each color represent different tracks, and each node of the track belongs to a different time instant.

Dataset	Scenario	$MOTP$	$\overline{m}$	$\overline{fp}$	$\overline{mm}$	$MOTA$
1	Forward	1,7788	24,23 %	4,29%	9,51 %	61,96%
	Backward	1,7067	10,74 %	7,06%	13,8 %	68,4%
2	Forward	0,4165	22,75 %	7,78%	14,97%	54,49%
	Backward	0,3525	17,37%	22,16%	13,17%	47,31%
3	Forward	0,2814	17,67%	10,34%	19,83%	52,16%
	Backward	0,333	8,3%	20,96%	18,78%	51,97%

Table 4.3: Evaluation metric results for the tracking problem: MOTP, misses ($\overline{m}$), false-positives ($\overline{fp}$), miss-matches ($\overline{mm}$), and $MOTA$.

error $< 0,8114$). The high percentage of missed-detections (m) is due to the fact that nodes that are very close to the structure supporting the root system or very close between each other are not detected. This error is specially notorious when the amount of branches start to increase (last frames of the sequence), where the tracking problem is more complex due to the small space between roots and the occlusions between them. Some false-positive (fp) errors are produced because they are not



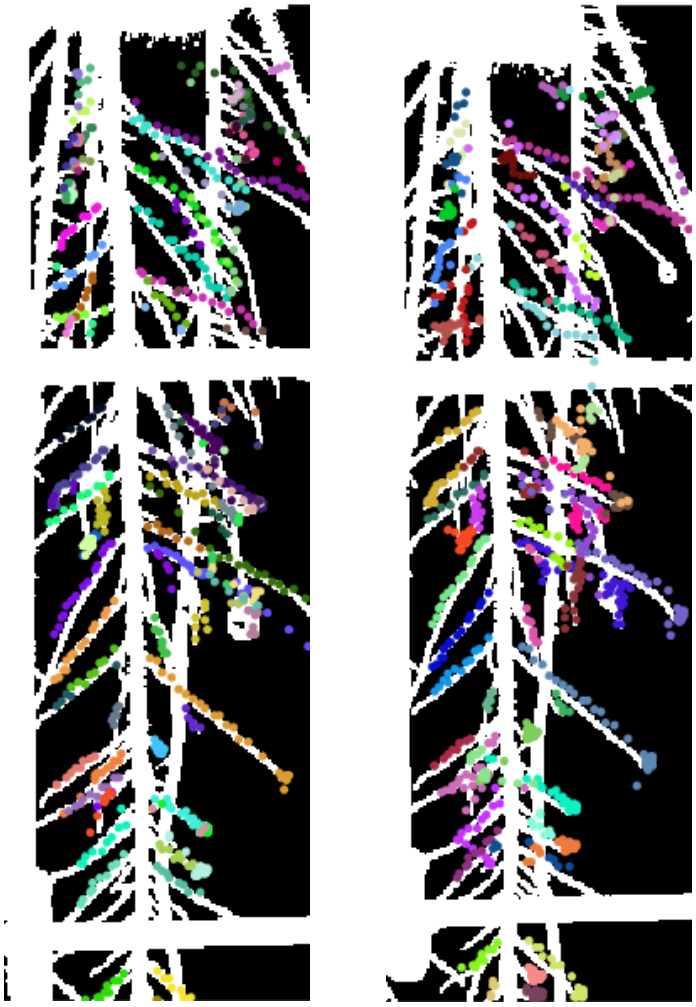
(a) Scenario 1 Seminal Roots (b) Scenario 2 Seminal Roots

Figure 4.3: Dataset 1. On the left (a) Tracking of seminal roots (scenario 1). On the right (b) Tracking of seminal roots (scenario 2). Each color represent different tracks, and each node of the track belongs to a different time instant.

included in the ground-truth. The definition of the ground-truth is a difficult task because some roots have a high amount of crossroads with other roots (specially in the case of branches). This produces some difficulties during the labelling of each track, and some errors may occur. Finally, we can also find miss-match (mm) errors, which are produced meanly for two reasons: crossroad/merges between roots, and gaps that has not been bridge between frames. This errors produce splits within the tracks.

In the seminal case most of the false-positive (fp) cases presented are due to the miss-classification between seminal and branches, which is specially notorious in Dataset 3 with a percentage of $90-95\%$. This occurs because in some occasions branches are longer than expected, and they are miss-classified as seminals. In addition, as we mentioned before, the creation of the ground-truth is a difficult task and in the branches case some nodes are missing. As a result, the errors in the classification process increase the percentage the false-positives and miss-detections (m), because these nodes are not being taken into account in its corresponding root type analysis.

On the other hand, regarding the association problem between nodes, we can observe in the for-



(a) Scenario 1 Branches Roots (b) Scenario 2 Branches Roots

Figure 4.4: Dataset 1. On the left (a), tracking of branches roots (scenario 1). On the right (b), tracking of branches roots (scenario 2). Each color represent different tracks.

ward scenario a high dependency on the growth speed s of the roots. It predicts the position of the future node based on s. Therefore if the speed growth has a high variance between consecutive frames, it is going to lead to miss-match errors (mm) because the nodes cannot be associated. This can be shown for example in Figure 4.5(b,d), where the track of the Primary root in Dataset 2 is split in two.

Moreover, we can also observe the sensitivity of each tracking scenario to the relative displacement of the roots between frames. In the forward scenario, the predicted node is allocated based on the orientation of the root. As a result, if a root from future frames suffer a high displacement, the positions of the nodes that belong to this root are going to be far from the predicted node and the probability calculated is going to be almost zero. In the backward scenario, if the displacement of a root is too big between consecutive frames, the distance between the nodes from previous frames and the shortest path of a node from the current frame is going to be high. Therefore if the displacement of roots between consecutive frames is too high, it is going to lead to miss-match errors (mm) for both scenarios.

Dataset	Scenario	Root type	$MOTP$	$\overline{m}$	$\overline{fp}$	$\overline{mm}$	$MOTA$
1	Forward	Primary	2,619	0 %	0%	0 %	100%
		Seminals	2,0636	15,64%	59,38%	6,25%	18,75%
		Branches	1,7148	31,8%	3,89%	7,07%	57,24%
	Backward	Primary	2,619	0 %	0%	0 %	100%
		Seminals	2,0636	15,61%	40,62%	15,62%	28,12%
		Branches	1,6589	15,19%	8,48%	15,19%	64,66%
2	Forward	Primary	0,3194	0 %	0%	14,29 %	85,71%
		Seminal	0	30%	35%	15%	20%
		Branches	0,5050	28,57%	10%	13,57%	47,86%
	Backward	Primary	0,3194	0 %	0%	42,86 %	57,14%
		Seminal	0	30%	30%	20%	20%
		Branches	0,4128	21,43%	27,14%	21,43%	42,14%
3	Forward	Primary	0	14,29%	14,29%	14,29%	57,14%
		Seminal	0	18,18%	95,45%	9,09%	-22,73%
		Branches	0,3005	25%	16,67%	8,33%	50%
	Backward	Primary	0	14,29%	0%	28,57%	57,14%
		Seminal	0	18,18%	90,91%	13,64%	-22,73%
		Branches	0,3766	17%	23%	16%	44%

Table 4.4: Evaluation metric results for the tracking problem for each root type: MOTP, misses ($\overline{m}$), false-positives ($\overline{fp}$), miss-matches ($\overline{mm}$), and $MOTA$.

However, the backward scenario is more sensitive to this kind of errors. This behavior is due to the functions selected in each scenario to obtain the cost matrices. In the forward scenario we have used $f(P) = (-\log_{10} P)^{1/n}$, while in the backward scenario we have used $f(d) = d^g$. In scenario 2, when the distance d is higher than expected, and we raised it to the power of n we make this value bigger. As a result, the cost is going to be too high to perform the association properly. On the contrary, in the forward scenario when we have a low probability, due to the saturation of the function selected despite having a extremely low (almost zero) probability, the cost obtained is going to be high but not high enough to produce an error during the association process. This can be shown for example in the tracking of the Primary root in Dataset 2 (Figure 4.5(a,c)).

Finally, by the comparison of both Tables we can observe that in general terms the forward scenario has a better performance, but it is not far from the performance of the backward scenario. The accuracy of each method depends on the properties of the dataset under study, we must take into account how it evolves along time to use one scenario or another due to the drawbacks of each scenario. If we analyze in detail each root type we can observe that in the case of primary roots the performance is better in the forward scenario, because the horizontal displacement of the primary root is higher than in the case of seminal and branches roots, which results in a lower accuracy in the backward scenario. In the case of seminal and branches roots the performance of each scenario varies, as it was mentioned before, based on the disposition and evolution of the roots within the dataset.

4.3 Confidence Metric

In the previous Section, we have analyzed the correctness of the multi-object tracker described by quantifying its performance. Now we want to evaluate the confidence/reliability of the constructed tracks, which is going to be measured separately in the case of primary and seminals roots, and

branches roots due to the different features that characterize each type. We have defined the following cues to asses the reliability of the tracks:

– Time. In the initial frames the accuracy of the tracking process is really high, but it decreases when the branches start growing due to the higher complexity of the disposition of the roots. The reliability of constructed tracks until a certain time instant T is higher. Figure 4.6 and Table 4.5 shows how the error ratio evolve across time.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
Scenario 1	0	0	0	1	0	0	0	2	7	11	38	49
Scenario 2	0	0	0	1	0	0	1	0	10	12	31	46

Table 4.5: Dataset 1. Error ratio at every time instant. The analysis is performed over every root type and scenario.

– Length. Formally, a track L is defined to be a collection of chained detections, i.e., $L = (v^1, v^2, \ldots, v^{|L|})$, $|L|$ being the length of the track. In the case of primary and seminal roots, they are long roots, which means that short tracks tends to be unreliable. As a result, a long track is more likely to be a correct track. On the contrary, in the branches case, they are short roots, which means that long tracks tends to be unreliable. As a result, a short track is more likely to be a correct track. This behavior can be seen in Figure 4.7.

– Association ambiguity. The uncertainty of an association can be described based on the existence of a cost of an alternative association that is close to the cost of the selected association. Hence, the uncertainty can be measured by comparing the real cost of the association between two nodes $[v_m, v_n]$ (solution of the Hungarian algorithm), and the smallest cost of the association between the nodes $[v_m, v_k]$, where $k \neq n$. For example, given the cost matrix in Table 4.6 we want to associate the detections from time t ($V^t = \{v_1^t, v_2^t, v_3^t, v_4^t\}$) with the detections from the previous frame $(t-1)$ ($V^{(t-1)} = \{v_1^{(t-1)}, v_2^{(t-1)}, v_3^{(t-1)}, v_4^{(t-1)}, v_5^{(t-1)}\}$) at the minimum cost.

		$(t-1)$				
		$v_1^{(t-1)}$	$v_2^{(t-1)}$	$v_3^{(t-1)}$	$v_4^{(t-1)}$	$v_5^{(t-1)}$
t	v_1^t	inf	**1**	inf	inf	inf
	v_2^t	inf	**1.1**	inf	**1**	inf
	v_3^t	**1**	inf	inf	inf	**4**
	v_4^t	inf	inf	inf	inf	inf

Table 4.6: Hungarian Algorithm example. The association solution is represented in green, where the lower-cost is reached. Then we can see in red the cost that is close to the green case it is corresponding line, and in yellow the ones that are far.

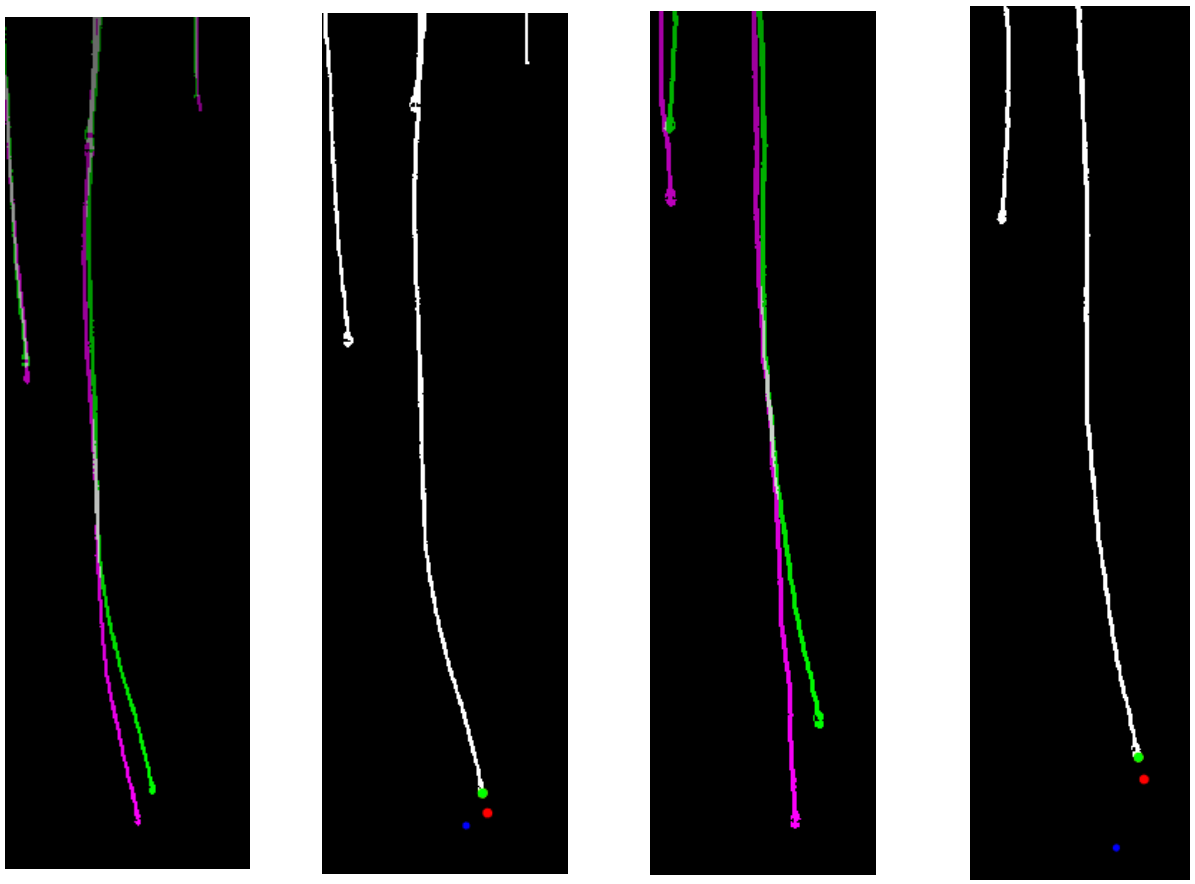
From the table above we can observe, that the node v_2^t is associated with $v_4^{(t-1)}$, which cost is very close to the cost of the node $v_2^{(t-1)}$. If we compute the difference between both costs we are going to obtain a small value, which means that $v_2^{(t-1)}$ and $v_4^{(t-1)}$ are very close to v_2^t. This means that there are certain ambiguity in their association. On the contrary, the node v_3^t is associated with $v_1^{(t-1)}$, which is far from the cost of $v_5^{(t-1)}$. If we compute the difference between both costs we are going to obtain a high value, which means that $v_5^{(t-1)}$ is far from v_3^t.

This means that there are not any uncertainty in the association. Formally, this computation is expressed as follows:

$$d_i = |c_{m,n} - c_{m,k}|, \text{ where } k \neq n$$

Given a certain track L_i, the collection of cost differences is $D_i = \{d_1, d_2, \dots\}$. If the difference is low it means that the costs are similar because a different node (v_k) is close to the node under study (v_m). Therefore there is some ambiguity on the association problem. To measure the uncertainty of the track we are going to look for the minimum value of these differences, which is given by $d^* = \arg\min D_i$.

In this Chapter we have described the evaluation protocol to asses the correctness of the algorithm proposed based on two different quantities: MOTA and MOTP. Where the MOTA is the sum of three different error ratios that takes into account the number of miss-detections, false-positive, and miss-matches errors, and MOTP measures the accuracy of the position of the detections. During the evaluation of the algorithm, we have observed some drawbacks in both scenarios: dependency of the forward cost scenario on the growth speed, and dependency of the backward cost scenario on the horizontal displacement of the roots. In addition, we have also described some cues to determine if a track is reliable. These cues are the time, length of the tracks, and ambiguity of the association between detections based on minimum difference among costs within a track. The analysis of these features are done separately on one hand for primary and seminals roots, and on the other hand for branches roots due to the different characteristics between them.



(a) BC: not associated. (b) FC: associated. (c) BC: not associated. (d) FC: not associated.

Figure 4.5: In this set of images we can observe the two main errors that leads to miss-match errors in both scenarios (BC - Backwards Cost, and FC- Forward Cost). The images belongs to Dataset 2. (a) Fusion of two consecutive frames. In this case the backward cost cannot associate the detections due to the horizontal displacement, while the forward cost can make the association as shown in (b). In the second image (b), we can observe in green the node at time $(t-x)$, in red the predicted node at time t, and in blue the real node at time t. (c) shows again the fusion of two consecutive frames, but in this case we can observe that the speed growth has vary a lot from one frame to another. As a result, as we can observe in (d) the association in the forward case cannot be done, but it is not achieved neither in the backward case because it suffers also from a horizontal displacement.

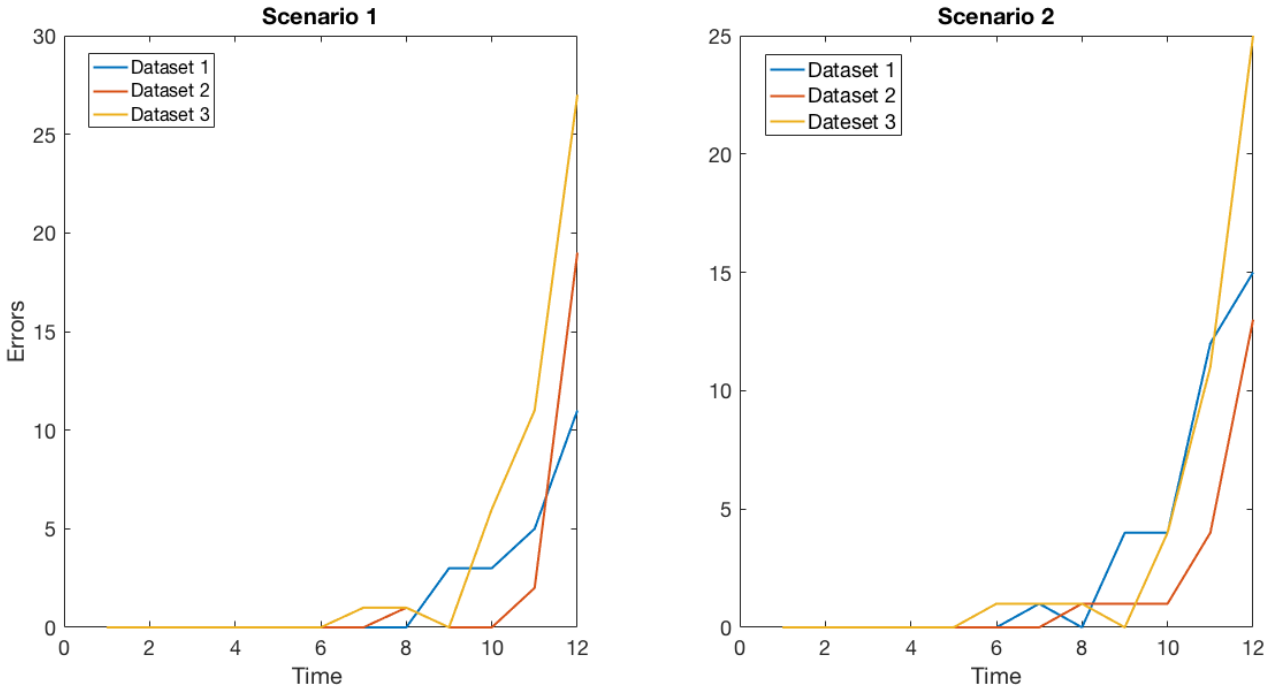


Figure 4.6: Tracking errors along time in both scenarios. At time $T = 9$ the amount of errors start increasing, reaching the maximum at the end of the sequence of frames. Each time instant is separated 10 frames.

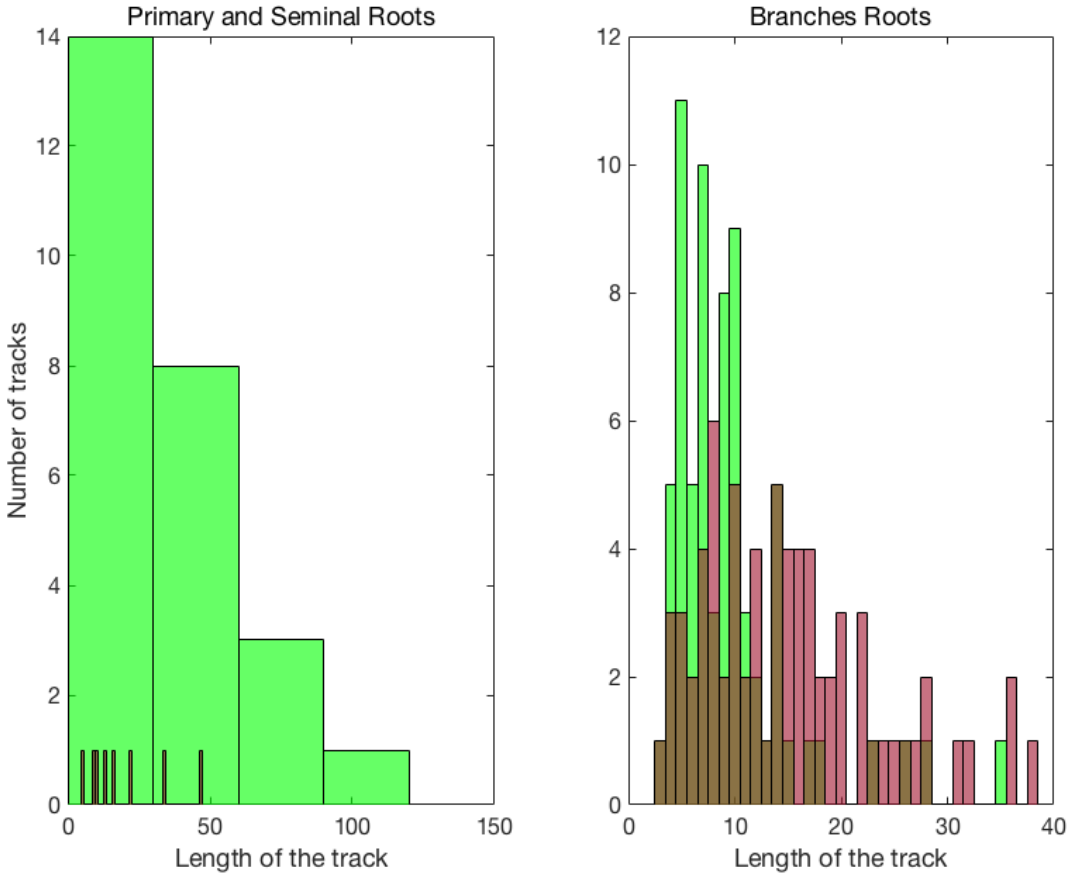


Figure 4.7: For the construction of the graphs it has been taken 60 samples from each dataset (180 samples in total). On the left, the histogram shows in green the amount of primary and seminal tracks that are correct depending on its length, and in red the amount of incorrect cases. As we can observe, when the length increases the amount of errors decreases. On the right, the histogram shows in green the amount of branches tracks that are correct depending on its length, and in red the amount of incorrect cases. We can appreciate that when the length increases the amount of error increases too.

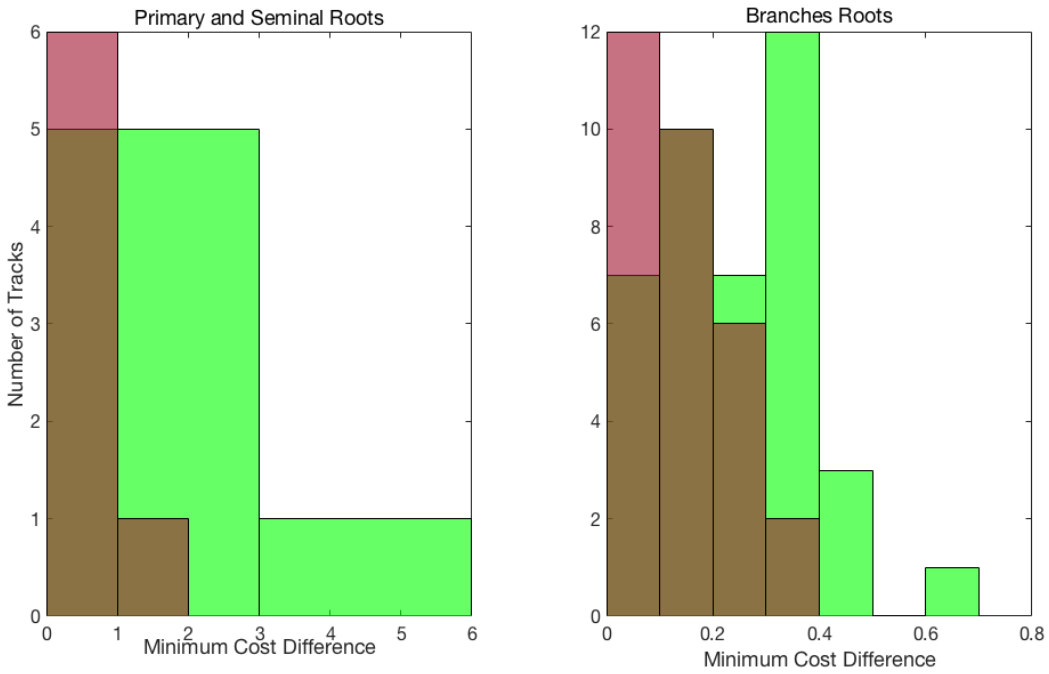


Figure 4.8: For the construction of the graphs it has been taken 60 samples from each dataset in the forward case (180 samples in total). On the left, the histogram shows in green the amount of primary and seminal tracks that are correct depending on the minimum cost difference, and in red the amount of incorrect cases. On the right, the histogram shows in green the amount of branches tracks that are correct depending on the minimum cost difference, and in red the amount of incorrect cases. We can appreciate that in both cases the errors are concentrated in a small range of costs. This set of costs defines the tracks that have nodes which association have some uncertainty due to the fact that it has other nodes almost at the same distance as the one being associated with.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

Multi-object tracking (MOT) is an important issue in computer vision. It has numerous applications in multiple disciplines such as player tracking in sport analysis, people tracking in surveillance, root detection in biology, crack evolution in material science,...

We focused on detection-based multi-object tracking approach in a biological scenario for the tracking of the extremities of roots plants to describe their root system architecture. In such approach, our priority is to track the behavior of root extremities across time, to collect all the information required to assess the genetic improvement of the root system architecture that define plants. It is based on the detection of the root tips in individual frames at different time instants and their association across time, obtaining the trajectory of each root tip.

In this thesis, we have addressed the MOT problem in two well-defined phases. First of all, a detection and classification phase described in Chapter 2, based on a shortest path framework using a graph-based formalism. In this first step, we compute the shortest path to every node (white pixel) in the image through the description of a graph that describes the connectivity of the pixels. The shortest paths are computed via the implementation of the Dijkstra algorithm. Then we use the two main features that characterized a root tip (localization within a root and its neighborhood) to detect them. Finally, we classify all the detections between primary, seminal and branches roots.

The second phase of the MOT problem, described in Chapter 3, is based on the link between all the detection at different time instants. To do so we use an algorithm based on a small time window, the Hungarian algorithm, due to the fact that almost all tips are present along the complete sequence without disappearing for long periods. Furthermore, we have analyzed the performance of the Hungarian algorithm over two different scenarios, based on the definition of two different cost matrices:

- Forward cost, where we have defined a Gaussian distribution that predicts the position of a drop in a future frame, given its position in the current frame.
- Backward cost, where the cost is based on the distance between a past detection and the shortest path connecting a current detection to the source. This cost matrix is referred to as a backward cost, because the distance is measured from detections in the current frame to detections in the past ones.

As we can observe we have exploited our main contribution, the shortest paths, in detection, classification and tracking phases:

– Identification of the tips/drops candidates.

– Classification of the tips/drops based on their paths.

– Definition of the forward cost, based on the prediction on a node, which direction is obtained from its shortest path.

– Definition of the backward cost, based on the distance of nodes from previous frames to the path of the node in the current frame.

As we can observe the calculation of the shortest paths brings several advantage in both phases, allowing us not only to detect the extremities of the roots, but to track them also. In addition, thanks to the calculation of the k-shortest paths, we can extract some additional information needed to perform the genetic improvement of the root architecture, which is our main goal. Some of this information is for example, the root angles, root direction, localization of intersection points between roots,...

5.2 Future Work

In this Section we are going to mention some of the improvements that can be applied on the proposed algorithm.

On the one hand, on the detection and classification phase new features can be considered to reduce the errors produce during the classification process. One of these features is for example the diameter of the root. In addition, the running time used to implement this first stage of the algorithm can be decreased by reducing the amount of data used by subsampling the image. A trade-off between time and number of detections should be found.

On the other hand, on the tracking phase, several improvement can be done:

– In the first scenario the position of the predicted node can be improved in different ways. We have observed a high dependency on the growth rate, due to the fact that this is a fixed value calculated a priori it can induce some errors. This can be improved, by updating this value iteratively frame-by-frame based on the analysis of previous frames. This way we can adapt the growing rate to the growing stage of the plant. Furthermore, other parameters can be also used to increase the accuracy of the position of the node, for example by taking into account the action of gravity.

– During the implementation of both scenarios, we have observed the dependency on the horizontal displacement of the roots, specially in the case of the primary root. Some models can be developed to take into account this displacement to compensate it, avoiding errors in the tracking process.

– Refinement of the cost definition, by adapting it to each root type.

– Definition of a new scenario where both cost matrices are combined, using simultaneously the backward and forward cost.

– Post-processing techniques to link the tracks that have been split.

Bibliography

[1] Jonathan P. Lynch *Steep, cheap and deep: an ideotype to optimize water and N acquisition by maize root systems.* Department of Plant Science, The Pennsylvania State University, University Park, PA 16802, USA.

[2] Douglas B.West. *Introduction to graph theory.* Second Edition, Pearson Education.

[3] Roger D. Eastman, Nathan S. Netanyahu, and Jacqueline Le Moigne. *Survey of image registration methods.* Sept. 15, 2010.

[4] Mike Dawes, *The Optimal Assignment Problem.* Course notes, University of Western Ontario.

[5] G. Ayorkor Mills-Tettey, Anthony Stentz, M- Bernardine Dias. *The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs.* Robotics Institute Carnegie Mellon University Pittsburgh, Pennsylvania. CMU-RI-TR-07-27. July 2007

[6] Kuhn,H.W. *The hungarian method for the assignment problem.* NavalResearchLogistics Quarterly 2 (1955), 83?97.

[7] Munkres, J. *Algorithms for the assignment and transportation problems.* Journal of the Society for Industrial and Applied Mathematics 5, 1 (March 1957), 32?38.

[8] Jérôme Berclaz, François Fleuret, Engin Türetken, and Pascal Fua. *Multiple Object Tracking using K-Shortest Paths Optimization.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 2011, pp. 1806-1819.

[9] Yi Cao *Munkres' Assignment Algorithm, Modified for Rectangular Matrices.* http://csclab.murraystate.edu/bob.pilgrim/445/munkres.html. Cranfield University on 11th September 2011

[10] Keni Bernardin, Alexander Elbs, Rainer Stiefelhagen *Multiple Object Tracking Performance Metrics and Evaluation in a Smart Room Environment.* Institut fur Theoretische Informatik Interactive Systems Lab, Universitat Karlsruhe, 76131 Karlsruhe, Germany

[11] Seung-Hwan Bae and Kuk-Jin Yoon *Confidence-Based Data Association and Discriminative Deep Appearance Learning for Robust Online Multi-Object Tracking.* IEEE Transactions on Pattern Analysis and Machine Intelligence, April 2017, pp. 99.

Appendices

Appendix A

Matlab Tips Detection and Classification

In this appendix, we are going to describe some of the functions and methods used to implement the detection and classification algorithm:

- Rotation of the image 180 degrees: `imrotate(im,180)`
- Mirror the image: `fliplr(im)`
- Crop the image: `imcrop(im,[xmin ymin width height])`
- Identification of all the nodes in the subsampled image: the nodes of the image corresponds to the pixels which values are equal to 1 ($I(i,j) = 1$), this could be done with the function `[i,j] = find(im_subsampled)`.
- Identification of the nodes that belongs to the structure that supports the plant: `nodes_struct = find(ismember([i,j],struct_horizontal_nz,'rows'))`, where the variable `struct_horizontal_nz` corresponds to the coordinates that define the horizontal sections of the structure.
- Creation of the graph (G): to build this matrix we should firstly calculate the distance between each node in the i-direction and j-direction with the $l1 - norm$:
  $||v_1 - v_2||_1 = d_1(v_1, v_2) = |v_1^i - v_2^i| + |v_1^j - v_2^j|$.

  To obtain the first and second terms of the sum in Matlab, we are going to apply the following functions respectively: `i_dist = uint8(abs(bsxfun(@minus, i, i')))` and `j_dist = uint8(abs(bsxfun(@minus, j, j')))`. Once we have the distance between all nodes, we have to filter them to get only the nodes with a 4-connectivity (N_g) as we can see in Figure A.1. To achieve this, we are going to sum $\Delta x + \Delta y$:

  $$\Delta x + \Delta y = \begin{cases} < 2 & \text{if 4 - connectivity} \\ \geq 2 & \text{rest} \end{cases}$$

  In Matlab is implemented as follows: `graph = double((x_dist + y_dist) < 2)`.

  In addition, when the roots cross the horizontal sections of the structure, it was stablished that the connectivity should be 2-connectivity ($\Delta x = 0, \Delta y = 1$). To obtain the nodes that

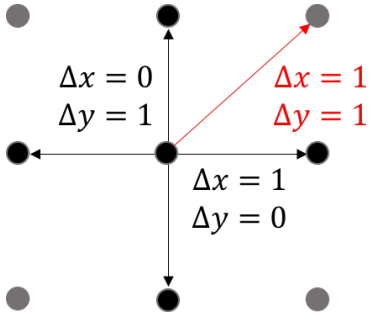


Figure A.1: Definition of the connectivity (black nodes) for adjacent nodes in the image.

accomplish this constraint we are going to multiply the matrices where $\Delta x = 0$ and $\Delta y = 1$, the result is going to be a matrix where each element is equal to one when both conditions are true (AND logical operation):
`is_connectivity_2 = (x_dist(nodes_struct,:)==0).*(y_dist(nodes_struct,:)==1)`.
Then we are going to apply this changes to the graph:
`graph(nodes_struct,:) = is_connectivity_2`.

Finally, we should ensure that the diagonal of $A(G)$ is equal to zero, and all the elements different from 1s are equal to `inf`.

- Implementation of the Dijkstra Algorithm with the Matlab function:
  `[dist, path,~] = graphshortestpath(sparse(graph),source,'Method','Dijkstra')`
- Creation of the distance image $H(i(v), j(v)) = d(v)$: `im_weighted(im_weighted == 1) = dist`
- Calculation of the angle between the horizontal line where the source node is located and the first drop (primary/seminal). The equation that defines the angle between two lines is:

$$\alpha = cos^{-1}(\frac{vect_1 vect_2}{|vect_1||vect_2|})$$

First of all we should define each line ($y = ax + b$), taking into a count that a line can be defined by two points, we use the function `polyfit (x_points, y_points,1)` to obtain the values $[a, b]$. Then, we create a vector based on each line, and perform the scalar product between them with the function `dot(vect1, vect2)`. To obtain the module of each vector we use the function `module = sqrt(sum(vect.^2))`. Finally, depending on the direction of the root ($vect_2$) the calculation of the angle varies as follows:

$$\alpha = \left\{ \begin{array}{c} \texttt{acos(scalar\_prod/(mod\_1*mod\_2))} \text{ if } a_2 < 0 \\ \texttt{pi - acos(scalar\_prod/(mod\_1*mod\_2))} \text{ the rest} \end{array} \right\}$$

- For the storage of each piece T_x, we are going to use create a cell with the function `cell`, and we are going to unified all the cells to form the final image H with the function `cell2mat(cell)`.
- For the high resolution paths unification we should analyze the pieces in decreasing order (from X to 0), working on each piece from bottom-to-top for the detection of drops/tips roots. So first of all, we should convert the coordinates of the drops/tips to check to which piece a drop/tip belongs to. Once we detect the set of drops that belong to a piece, we track its path from the

drop/tip node to the node on the first row of the piece (P_x). Once we have located the position where the tip/drop paths begins, we have to link it to with the next piece to know where do we have to search to obtain the whole path through each piece. Then, we will add the last row of the previous piece into the first row of the current piece to make the connection between pieces.

$$T_x^* = \begin{bmatrix} \text{Last row previous piece } T_{(x-1)} \\ \text{Current piece } T_x \end{bmatrix}$$

$$T_x = I(i,j), \;\; i \in [(x \times 150) + 1, (150 \times x) + 150] \text{ and } x = [X = H/150 \ldots 1]$$

This process is repeated until we reach the first piece, where the source node is located. Finally, the union of all the paths tracked through the pieces defines the path from the source node to the drop/tip root ($P = \bigcup_{i=1}^{X} P_x$).

– To achieve the differentiation between drops into the three possible types of roots: ***primary*** (longest root, it is the main root), ***seminals*** (they are not born from other roots) and ***branches*** (shorter roots, those which emerge from the main or seminal roots), we are going to define a matrix *nodes_main_drops*, which is going to store the primary root and seminal roots.

$$nodes\_main\_drops = \begin{bmatrix} path_1 = \{v_1, v_2, ..., v_D\} \\ path_2 = \{v_1, v_2, ..., v_{D'}, 0, 0, ...\} \\ path_i = \{v_1, v_2, ..., v_{D'}, 0, 0, ...\} \end{bmatrix}$$

This matrix is going to help us to identify if a root was born on the origin or it was originated from another root. Therefore, we are going to analyze if a merge between paths has been produced or not M. The length of the primary root is going to define the size of this matrix ($number\ of\ primary + seminal\ roots \times length\ longest\ root\ (D)$). This matrix is going to be filled iteratively as follows:

- Firstly, we should obtain the difference between the length of the path of the first drop ($path_1 = \{v_1, v_2, ..., v_D\}$), which corresponds to the length of the furthest node (primary root), and the current drop, this value will define the padding that we should add to the path of the current drop, being its length D' ($path_i = \{v_1, v_2, ..., v'_D(k)\}$). Where $D' < D$, then apply the padding to $path_i$. Creating then a new vector $path'_i$, which has the same size (D) as $path_1$. The aim of this process is the creation of two matrices of the same size.

  In the vector $path_i$ the nodes are stored from the source to the drop (top-to-bottom), which means that $n_1(k)$ is always going to be closest node to the source node. To detect a merge or a crossroad between roots, we analyze the paths of the tips/drops from bottom-to-top, so we flip this vector to analyze easily each case ($path_i = \{v'_D, v_{D'-1}, ..., v_1\}$).
- Then, we add the first drop to the matrix *nodes main drops*, due to the fact that the first drop its always going to be a primary or a seminal root.
- Hereunder, we can perform the difference between both matrices. This way if we obtain a zero, then it means that there is a coincidence between paths, which could correspond to a merge or to a crossroad between roots. Therefore, we are going to analyze the following values obtained from the difference between paths: if they are all zeros, it means that it is a merge. However, we have considered a coincidence window of 8 nodes to decide if it is a merge or not, because initially all roots are born from the same origin so they can share some nodes. If the amount of share nodes is greater than 8, it means that its finally a merge.

- For the differentiation between seminal and branches, we analyze if two drops share more or less than the 50 % of their paths. The drop is classified as a seminal drop if they share less than the 50 %, on the contrary it will be classified as a branch drop. On the other hand, if we find again non-zero values in the following values obtained from the difference between paths, it means that what we found was just a crossroad between roots, and we keep analyzing in case that there is a merge further on. If there is no more zero-values, or there are not merges detected, the drop is classified as a seminal drop and it is added to the matrix *nodes_main_drops*.

Appendix B

Matlab Tips Tracking

In this appendix, we are going to describe some of the functions and methods used to implement the tracking algorithm:

- Calculation of the bivariate gaussian distribution: `mvnpdf(cost_matrix,mu,Sigma)`
- Calculation of the euclidean distance between two set of coordinates: `pdist2(X,Y)`
- Calculation of the mean from a gaussian distribution: `[mu, sigma] = normfit(data)`

Appendix C

Validation Results

In this appendix, we are going to analyze the results of the evaluation method at each time instant (every 10 frames) in each dataset. As we described in Chapter 4 the evaluation protocol used defines the following variables to evaluate the performance of the algorithm:

- Number of matches found for time, c_t.
- Number of objects present at time t, g_t.
- Number of false- positives fp_t: all remaining hypothesis that are not linked to any object are considered false positives.
- Number of miss-detections m_t, all remaining objects that are not linked to any hypothesis are considered misses.
- Number of mismatch errors mm_t

C.1 Dataset 1

In Table C.1, we can observe the values obtained for the tracking problem globally, and in Table C.2 we can observe the values obtained for the tracking problem on each root type.

C.2 Dataset 2

In Figures C.1, C.2, and C.3 we present the tracking solution of the second dataset for each root type.

In Table C.4, we can observe the values obtained for the tracking problem globally, and in Table C.3 we can observe the values obtained for the tracking problem on each root type.

C.3 Dataset 3

In Figures C.4, D.1, and C.6 we present the tracking solution of the third dataset for each root type.

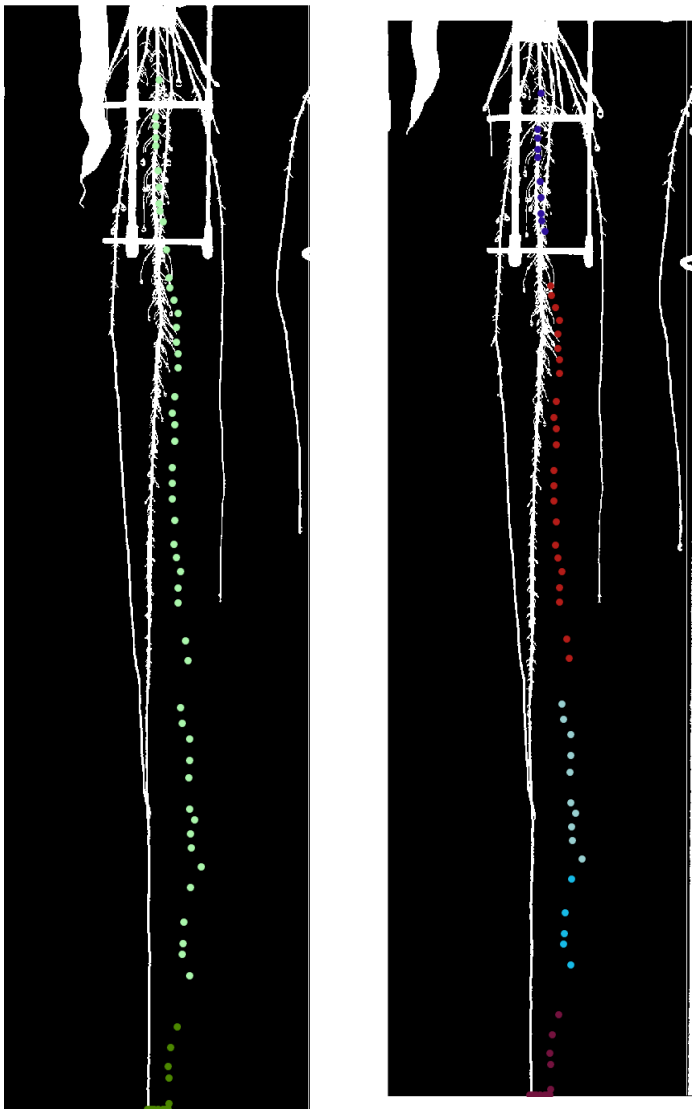
In Table C.5, we can observe the values obtained for the tracking problem globally, and in Table C.6 we can observe the values obtained for the tracking problem on each root type.

Forward Scenario												
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
g_t	0	1	1	3	2	4	4	14	41	59	91	106
c_t	0	1	1	2	2	4	4	15	36	47	61	76
m_t	0	0	0	1	0	0	0	1	5	12	30	30
fp_t	0	0	0	0	0	0	0	1	2	2	6	3
mm_t	0	0	0	0	0	0	1	0	3	3	8	16
Backward Scenario												
g_t	0	1	1	3	2	4	4	14	41	59	91	106
c_t	0	1	1	2	2	4	4	15	40	56	79	89
m_t	0	0	0	1	0	0	0	1	1	3	12	17
fp_t	0	0	0	0	0	0	0	0	7	5	6	5
mm_t	0	0	0	0	0	0	1	0	5	7	14	18

Table C.1: Dataset 1. Number of mis-detections (m_t), false-positives (fp_t), and mis-matches (mm_t) at every time instant. The analysis is performed over every root type and scenario.

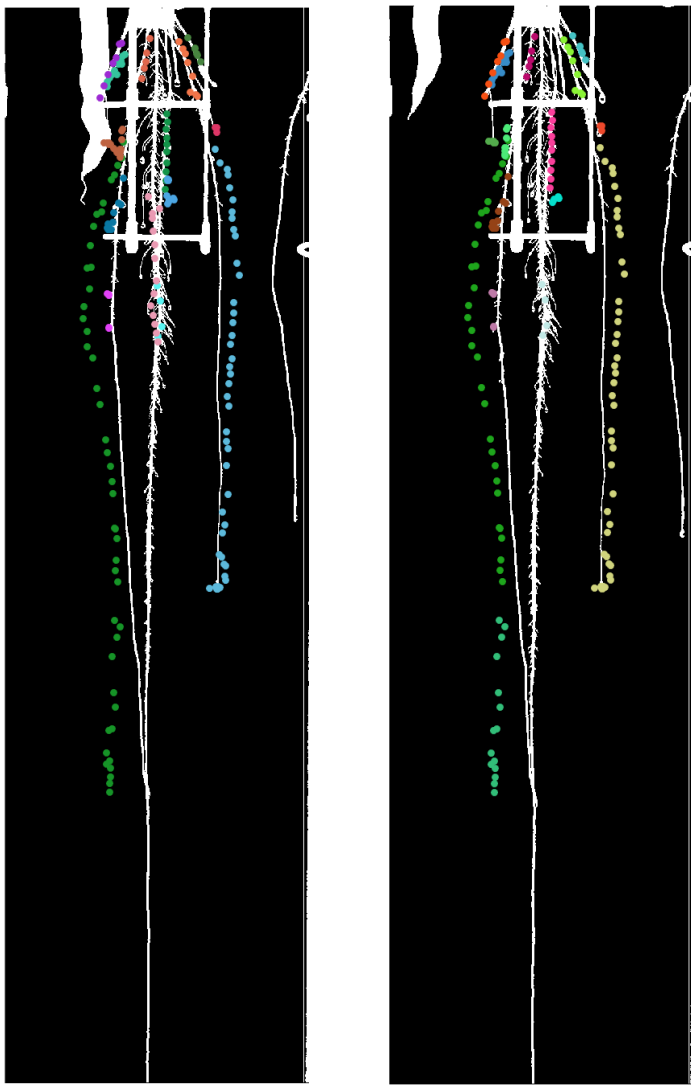
Forward Scenario												
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}
Primary Root												
g_t	0	1	1	1	1	1	1	1	1	1	1	1
c_t	0	1	1	1	1	1	1	1	1	1	1	1
m_t	0	0	0	0	0	0	0	0	0	0	0	0
fp_t	0	0	0	0	0	0	0	0	0	0	0	0
mm_t	0	0	0	0	0	0	0	0	0	0	0	0
Seminal Roots												
g_t	0	0	0	2	1	3	3	3	5	5	6	4
c_t	0	0	0	1	1	3	3	3	5	5	3	3
m_t	0	0	0	1	0	0	0	0	0	0	3	1
fp_t	0	0	0	0	0	0	0	0	2	4	8	5
mm_t	0	0	0	0	0	0	1	0	0	0	1	0
Branches Roots												
g_t	0	0	0	0	0	0	0	10	35	53	84	101
c_t	0	0	0	0	0	0	0	11	28	38	51	67
m_t	0	0	0	0	0	0	0	1	7	15	33	34
fp_t	0	0	0	0	0	0	0	1	2	1	4	3
mm_t	0	0	0	0	0	0	0	0	2	2	5	11
Backward Scenario												
Primary Root												
g_t	0	1	1	1	1	1	1	1	1	1	1	1
c_t	0	1	1	1	1	1	1	1	1	1	1	1
m_t	0	0	0	0	0	0	0	0	0	0	0	0
fp_t	0	0	0	0	0	0	0	0	0	0	0	0
mm_t	0	0	0	0	0	0	0	0	0	0	0	0
Seminals Roots												
g_t	0	0	0	2	1	3	3	3	5	5	6	4
c_t	0	0	0	1	1	3	3	3	5	5	3	3
m_t	0	0	0	1	0	0	0	0	0	0	3	1
fp_t	0	0	0	0	0	0	0	0	2	4	4	3
mm_t	0	0	0	0	0	0	1	0	1	0	3	0
Branches Roots												
g_t	0	0	0	0	0	0	0	10	35	53	84	101
c_t	0	0	0	0	0	0	0	11	32	46	70	83
m_t	0	0	0	0	0	0	0	1	3	7	14	18
fp_t	0	0	0	0	0	0	0	0	7	5	7	5
mm_t	0	0	0	0	0	0	0	0	3	5	8	17

Table C.2: Dataset 1. Number of mis-detections (m_t), false-positives (fp_t), and mis-matches (mm_t) at every time instant. The analysis is performed over every root type and scenario.



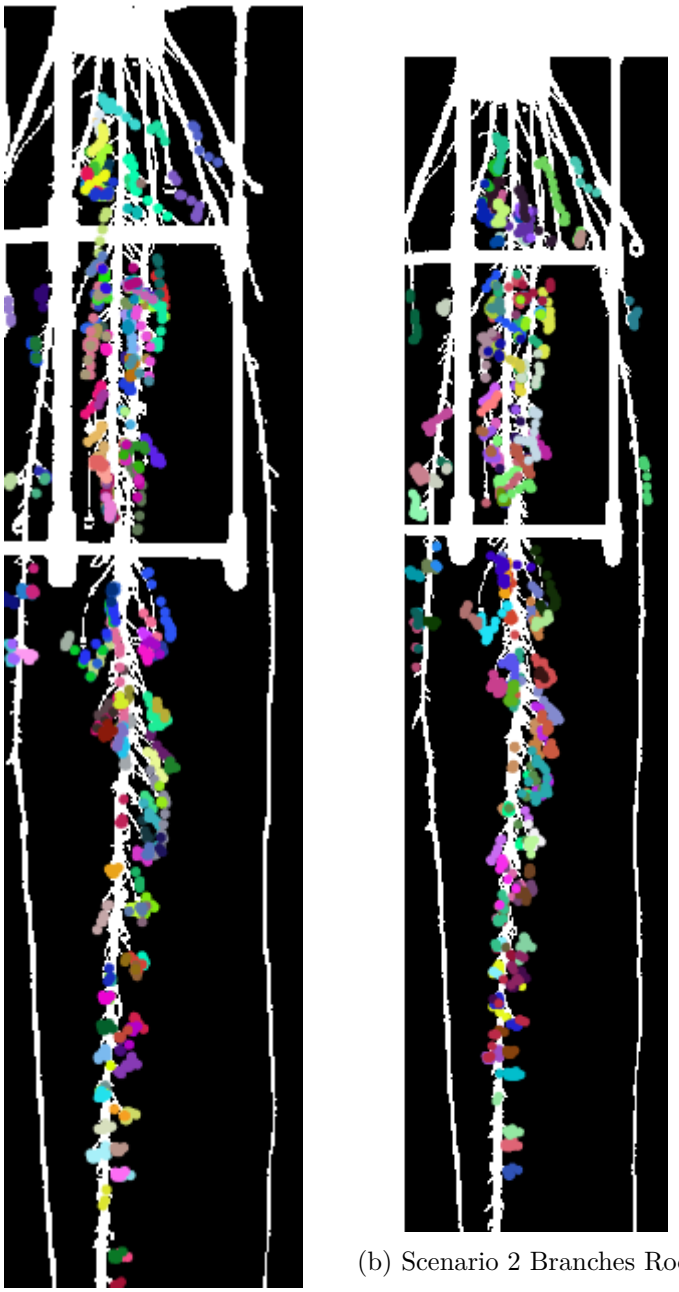
(a) Primary Root (b) Primary Root

Figure C.1: Dataset 2. On the left. (a) Tracking of primary root (scenario 1). On the right (b) Tracking of primary root (scenario 2). Each color represent different tracks, and each node of the track belongs to a different time instant.



(a) Scenario 1 Seminal Roots (b) Scenario 2 Seminal Roots

Figure C.2: Dataset 2. On the left (a) Tracking of seminal roots (scenario 1). On the right (b) Tracking of seminal roots (scenario 2). Each color represent different tracks, and each node of the track belongs to a different time instant.



(a) Scenario 1 Branches Roots

(b) Scenario 2 Branches Roots

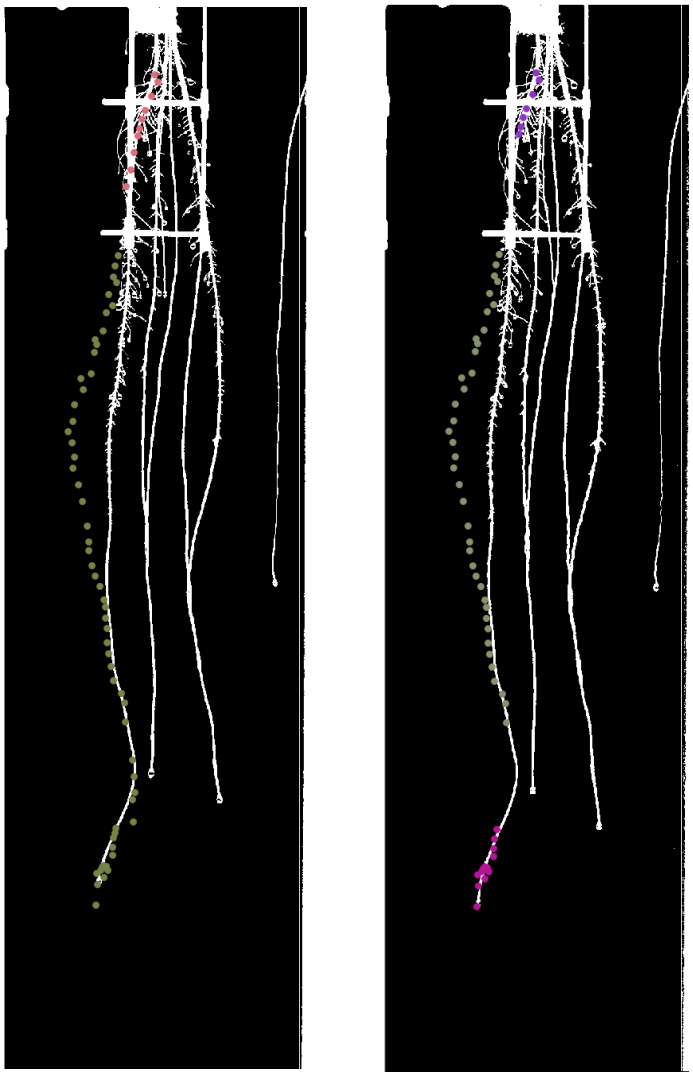
Figure C.3: Dataset 2. On the left (a), tracking of branches roots (scenario 1). On the right (b), tracking of branches roots (scenario 2). Each color represent different tracks.

Forward Scenario								
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
g_t	0	1	3	4	5	30	47	77
c_t	0	1	2	4	5	23	40	54
m_t	0	0	1	0	0	7	7	23
fp_t	0	0	0	1	0	0	4	8
mm_t	0	0	0	1	1	0	2	21
Backward Scenario								
g_t	0	1	3	4	5	30	47	77
c_t	0	1	2	4	5	25	36	65
m_t	0	0	1	0	0	5	11	12
fp_t	0	0	0	0	0	2	12	23
mm_t	0	0	1	1	1	1	4	14

Table C.3: Dataset 2. Number of mis-detections (m_t), false-positives (fp_t), and mis-matches (mm_t) at every time instant.

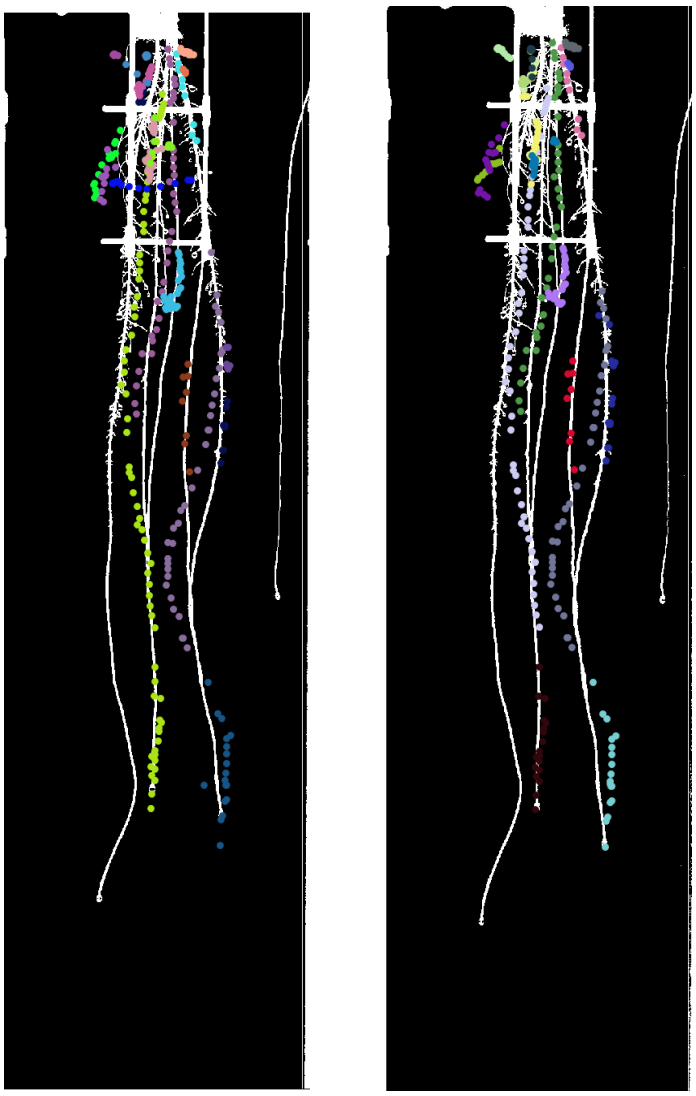
Forward Scenario								
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
Primary Root								
g_t	0	1	1	1	1	1	1	1
c_t	0	1	1	1	1	1	1	1
m_t	0	0	0	0	0	0	0	0
fp_t	0	0	0	0	0	0	0	0
mm_t	0	0	0	0	0	0	1	0
Seminal Roots								
g_t	0	0	2	3	3	6	3	3
c_t	0	0	1	3	3	3	2	2
m_t	0	0	1	0	0	3	1	1
fp_t	0	0	0	0	0	0	3	4
mm_t	0	0	0	1	0	0	0	1
Branches Roots								
g_t	0	0	0	0	1	23	43	73
c_t	0	0	0	0	1	17	34	48
m_t	0	0	0	0	0	6	9	25
fp_t	0	0	0	1	0	2	4	7
mm_t	0	0	0	0	0	0	1	18
Backward Scenario								
Primary Root								
g_t	0	1	1	1	1	1	1	1
c_t	0	1	1	1	1	1	1	1
m_t	0	0	0	0	0	0	0	0
fp_t	0	0	0	0	0	0	0	0
mm_t	0	0	1	0	0	1	1	0
Seminals Roots								
g_t	0	0	2	3	3	6	3	3
c_t	0	0	1	3	3	3	2	2
m_t	0	0	1	0	0	3	1	1
fp_t	0	0	1	0	0	0	2	4
mm_t	0	0	0	1	1	0	1	1
Branches Roots								
g_t	0	0	0	0	1	23	43	73
c_t	0	0	0	0	1	19	31	59
m_t	0	0	0	0	0	4	12	14
fp_t	0	0	0	0	0	4	12	22
mm_t	0	0	0	0	0	0	2	11

Table C.4: Dataset 2. Number of mis-detections (m_t), false-positives (fp_t), and mis-matches (mm_t) at every time instant. The analysis is performed over every root type and scenario.



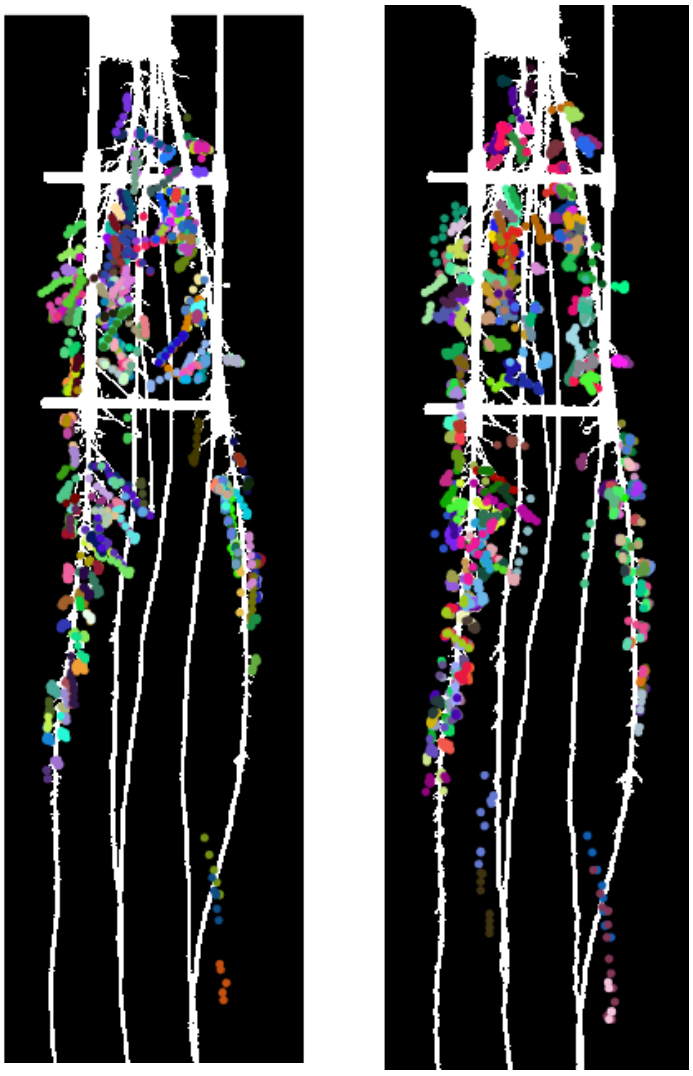
(a) Primary Root (b) Primary Root

Figure C.4: Dataset 3. On the left. (a) Tracking of primary root (scenario 1). (b) Tracking of primary root (scenario 2). Each color represent different tracks, and each node of the track belongs to a different time instant.



(a) Scenario 1 Seminal Roots (b) Scenario 2 Seminal Roots

Figure C.5: Dataset 3. On the left (a) Tracking of seminal roots (scenario 1). On the right (b) Tracking of seminal roots (scenario 2). Each color represent different tracks, and each node of the track belongs to a different time instant.



(a) Scenario 1 Branches Roots (b) Scenario 2 Branches Roots

Figure C.6: Dataset 3. On the left (a), tracking of branches roots (scenario 1). On the right (b), tracking of branches roots (scenario 2). Each color represent different tracks.

Forward Scenario									
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
g_t	0	2	3	3	6	18	33	95	72
c_t	0	2	1	3	5	17	29	75	59
m_t	0	0	2	0	1	1	4	20	13
fp_t	0	0	1	1	1	1	1	5	14
mm_t	0	0	0	1	1	0	6	11	27
Backward Scenario									
g_t	0	2	3	3	6	18	33	95	72
c_t	0	2	2	3	5	18	30	88	62
m_t	0	0	1	0	0	0	1	7	10
fp_t	0	0	1	1	1	1	5	15	24
mm_t	0	0	1	1	1	0	4	11	25

Table C.5: Dataset 3. Number of mis-detections (m_t), false-positives (fp_t), and mis-matches (mm_t) at every time instant. The analysis is performed over every root type and scenario.

Forward Scenario									
	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Primary Root									
g_t	0	1	0	1	1	1	1	1	1
c_t	0	1	1	1	1	1	1	1	1
m_t	0	0	1	0	0	0	0	0	0
fp_t	0	0	1	0	0	0	0	0	1
mm_t	0	0	0	1	0	0	0	0	0
Seminal Roots									
g_t	0	1	3	2	4	3	4	3	2
c_t	0	1	1	2	4	2	3	3	2
m_t	0	0	2	0	0	1	1	0	0
fp_t	0	0	1	1	0	3	4	7	5
mm_t	0	0	0	0	1	0	0	1	0
Branches Roots									
g_t	0	0	0	0	0	14	26	91	69
c_t	0	0	0	0	0	11	19	64	51
m_t	0	0	2	0	0	3	7	27	18
fp_t	0	0	0	0	1	1	1	5	14
mm_t	0	0	0	0	0	0	3	8	26
Backward Scenario									
Primary Root									
g_t	0	1	0	1	1	1	1	1	1
c_t	0	1	1	1	1	1	1	1	1
m_t	0	0	1	0	0	0	0	0	0
fp_t	0	0	0	0	0	0	0	0	0
mm_t	0	0	0	1	0	0	0	0	1
Seminals Roots									
g_t	0	1	3	2	4	3	4	3	2
c_t	0	1	1	2	4	2	3	3	2
m_t	0	0	2	0	0	1	1	0	0
fp_t	0	0	1	0	0	3	5	7	4
mm_t	0	0	0	0	1	0	0	2	0
Branches Roots									
g_t	0	0	0	0	0	14	26	91	69
c_t	0	0	0	0	0	12	21	78	55
m_t	0	0	2	0	0	2	5	13	14
fp_t	0	0	0	1	1	1	5	14	24
mm_t	0	0	0	0	0	0	2	7	23

Table C.6: Dataset 3. Number of mis-detections (m_t), false-positives (fp_t), and mis-matches (mm_t) at every time instant. The analysis is performed over every root type and scenario.

Appendix D

Dataset Comparison

In this appendix we are going to compare the collection of detections obtained with our algorithm, and the collection of detections provided in the Dataset 1. In the following figures we can observed the combination of both sets of detections.

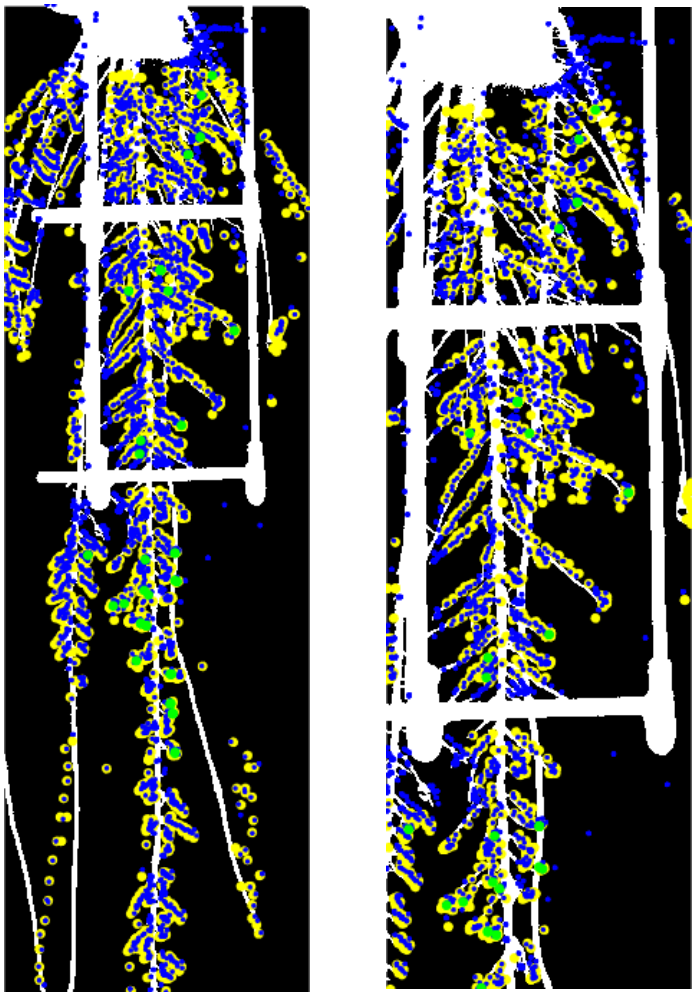


Figure D.1: We can observe in yellow the detections obtained with our algorithm, in blue the detections provided in Dataset 1, and in green the exact matches between both set of detections.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgium **www.uclouvain.be/epl**