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

Type of document D6.1 – V1.0

Robotic Vehicles Field Validation

Dissemination level	PU
Submission Date	2020-04-30
Work Package	WP6
Task	T6:1
Type	Demonstrator + Report
Version	1.0
Author	Ciro Potena, Nicolas Bono Rossello
Approved by	Andrea Gasparri, Emanuele Garone

Executive Summary

This document describes the Robotic Vehicles Field Validation of the project PANTHEON.

In particular, this Deliverable presents the field validation of the unmanned vehicles navigation, monitoring and agronomic intervention capabilities within a real-world (1:1 scale) hazelnut orchard. In this regard, a detailed description of the theoretical achievements along with videos describing the experimental validation are provided.

Note that, regarding the media content this Deliverable is to be considered a live-document and is subject to regular updated to be found on the portal of the project PANTHEON, i.e., http://www.project-pantheon.eu.

Table of Content

Abbreviations and Acronyms

UGV	Unmanned Ground Vehicle
UAV	Unmanned Aerial Vehicle
SLAM	Simultaneous Localization and Mapping
KF	Kalman Filter
EKF	Extended Kalman Filter
ID	Identifier

1 Unmanned Ground Vehicle

One of the objectives of the PANTHEON project is to develop an Unmanned Ground Vehicle (UGV) capable of moving within an orchard, to collect data and perform typical farming operations (such as suckers' treatment and branch marking for pruning) in a fully autonomous manner. Several modules are required to achieve this objective, ranging from the design and implementation of control, navigation, and planning algorithms and a simultaneous localization and mapping (SLAM) framework as well. In the following, we provide a comprehensive description of the main modules we developed together with their experimental validations we carried out within the PANTHEON hazelnut orchards.

1.1 Preliminaries

In this section, we provide a brief overview of the notations and the basic graph theory concepts that will be used in the sequel for deriving the SLAM, the global planning and the navigation architectures. In this work, we make use of a world frame $\mathcal{F}_{\mathcal{W}}$, a robot frame $\mathcal{F}_{\mathcal{B}}$ and a map frame $\mathcal{F}_{\mathcal{M}}$. Moreover, we assume the robot to be equipped with a LiDAR sensor, whose frame is defined as $\mathcal{F}_{\mathcal{S}}$, and with a GPS sensor for which we assume the measurements to be already expressed in $\mathcal{F}_{\mathcal{B}}$.

We represent vectors as bold quantities having a superscript, representing the frame in which they are expressed, and a subscript, indicating the origin and the end of such a vector. For example, the quantity p_{SB}^{W} represents the position of the body frame $\mathcal{B}$ with respect to the sensor frame $\mathcal{S}$, expressed in the world frame $\mathcal{W}$. To simplify the notation, if a vector has no superscript, we assume it to be expressed in the first frame reported in the subscript (*i.e.*, the frame where the vectors origin is). Similarly, we represent matrices as bold upper-case quantities where we encode the starting frame as subscript and the final frame as superscript. For example, the rotation matrix R_{W}^{B} transforms a vector from the body frame $\mathcal{F}_{\mathcal{B}}$ to the world frame $\mathcal{F}_{\mathcal{W}}$. All the remaining matrices are expressed as capital quantities.

To enable the SLAM to estimate the planting pattern of the target field, the field is modeled as a chessboard with rectangular chess. The chessboard model is expressed in the map frame $\mathcal{F}_{\mathcal{M}}$ and define two main chessboard directions along with the trees are arranged.

To achieve a scalable computational complexity in SLAM and to define a global path for the robotic platform, we resort to a topological map encoded by means of a graph. In this regard, let us define a directed graph $\mathcal{G} = \{V, E\}$, where $V = \{v_1, \ldots, v_n\}$ is the set of nodes (*i.e.* trees, in our case) and $E = \{(v_i, v_j)\}$ is the set of pairwise edges among nodes i and j. In addition, with $n = |V|$ and $m = |E|$ we indicate the total number of vertexes and the total number of edges, respectively. Moreover, for any node set $S \subset V$, let $\delta\hat{} + (S)$ denote the set of arcs in V for which tails are in S and for which heads are in $V \setminus S$, and let $\delta^{-}(S)$ denote the set of arcs in V for which the reverse holds. We denote with the symbol $e_k = (v_i, v_j)$ the k-th edge for a directed graph, that is from the vertex v_i to the vertex v_j. For an arc $e_k = (v_i, v_j) \in E$, the cost d_{e_k} represents the travelling distance cost from the vertex v_i to the vertex v_j. Let us define a path between trees i and j as the set of edges through which a tree j can be reached by a tree i. Let us define the length of a path between a pair of vertexes i and j as the number of edges required to build it. Let us denote with $\mathcal{N}(i)^1 = \{j \in V : (i,j) \in E\}$ the 1-hop neighborhood of the tree i, *i.e.* the set of vertexes j that can be reach from vertex i through a path of length 1. In addition, let us denote with $\mathcal{N}(i)^k$ the k-hop neighborhood of tree i, for which we denote with $|\mathcal{N}(i)^k|$ its cardinality. In other words, $\mathcal{N}(i)^k$ is the set of trees j for which there exists a p-hop path from the tree i to the tree j with $p \leq k$. Moreover, let us denote with $\mathcal{N}(i,j)^k = \{\mathcal{N}(i)^k \cup \mathcal{N}(j)^k\}$ as the union between the k-hop neighborhood of trees i and j. Finally, in this work, the 1-hop neighborhood is built according to the Manhattan distance metrics, *i.e.* by considering only the edges along the two principal Cartesian axes.

1.2 Ackermann Steering Kinematics

In this section, we briefly review the basic concept of the Ackermann Steering Geometry. For a full theoretical characterization, we refer the reader to the deliverable 3.1. An Ackermann front-wheel-steering vehicle can be conveniently expressed from a mathematical standpoint in terms of a Bicycle model. In this model, the control inputs are the linear velocity v and the steering angle ϕ. Specifically, the following set of equations can be used to describe the kinematics of a Bicycle model

$$\begin{aligned} \dot{x}(t) &= v(t)\cos\big(\theta(t)\big) \\ \dot{y}(t) &= v(t)\sin\big(\theta(t)\big) \\ \dot{\theta}(t) &= \frac{v(t)}{\mathrm{l}}\tan\big(\phi(t)\big) \end{aligned} \tag{1}$$

where $\boldsymbol{q}(t) = [x(t),\, y(t),\, \phi(t)]^T$ denotes the state vector at time t with $\boldsymbol{p}(t) = [x(t),\, y(t)]^T$ representing the position of the robot in the Cartesian plane at time t w.r.t. the global reference frame $\mathcal{F}_{\mathcal{W}} = \big(x_g,\, y_g,\, \theta_g\big)$, and $\theta(t)$ representing its orientation with respect to the x-axis of global reference frame $\mathcal{F}_{\mathcal{W}}$ at time t, $u(t) = [v(t),\, \phi(t)]^T$ denotes the input vector at time t with $v(t)$ and $\phi(t)$ representing the linear velocity and the steering angle, respectively, at time (t), and finally l denotes the wheelbase, i.e., the distance between the front and rear axles. In the following we will refer to the Bicycle kinematics in a compact form

$$\dot{\boldsymbol{q}}(t) = f\big(\boldsymbol{q}(t), \boldsymbol{u}(t)\big) \tag{2}$$

with $f(\cdot,\cdot)$ the system dynamics defined according to Equation (1). In the sequel, the time-dependence will be omitted if not strictly required for the sake of readability. In this work we assume that the vehicle reference frame $\mathcal{F}_{\mathcal{B}} = (x,\, y,\, \theta)$ is going to be located at the center of the rear axle (see Figure 1 (left)).

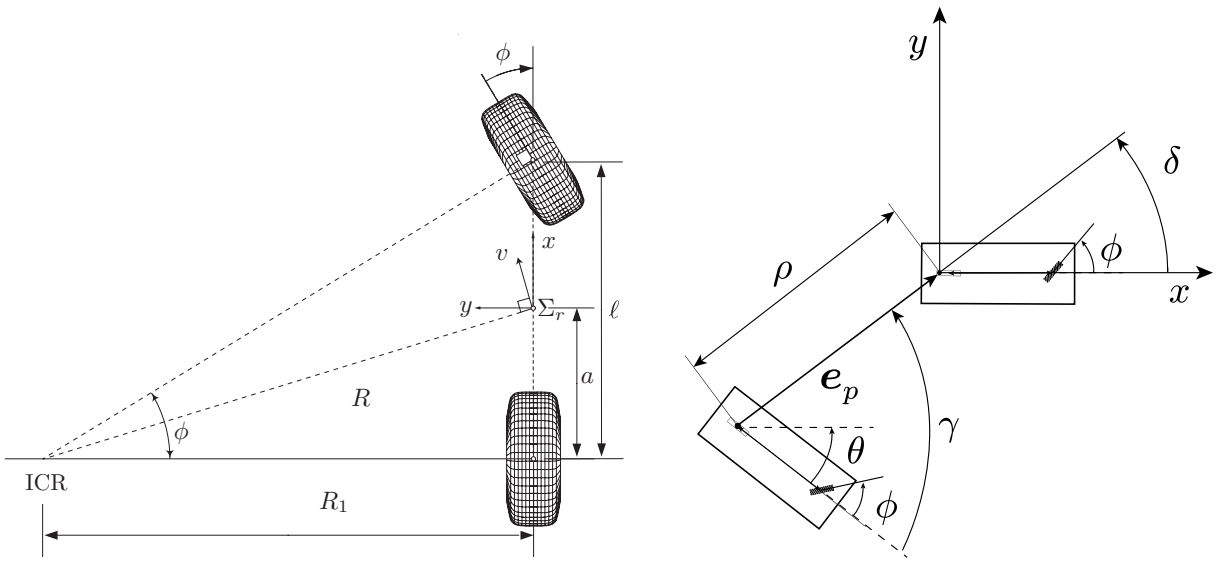

Figure 1 - Left: Equivalent Bicycle model for a front-wheel steering vehicle. Right: definition of polar coordinates for the bicycle model

1.3 Navigation Architecture

The navigation architecture is composed of two major modules: a local planner and a low-level control law. The former generates, in real-time, optimal trajectories by formulating the planning problem as a real-time cost function minimization over a finite time horizon. The latter ensures the robot to closely track the planned trajectory. In the following, we provide a comprehensive description of both the above introduced modules. The proposed navigation architecture has been published in the Robotics and Automation Letters (RA-L) [1] and for presentation at the IEEE International Conference on Robotics and Automation (ICRA), annual flagship conferences of the Robotics & Automation Society (RAS).

1.3.1 Local Planning

The goal of the trajectory planner is to generate optimal trajectories over a time horizon T from any given initial pose and velocity that fulfill the following requirements:

I. the trajectory steers the robot to a given target pose;
II. the trajectory satisfies the Ackermann kinematics constraints and is collision-free;
III. the trajectory is generated in a receding horizon fashion and in real-time.

We reiterate that the latter aspect is essential in the farming context since re-planning in real-time allows the robot to react in time to avoid dynamical obstacles and to handle perception uncertainties as well. The whole problem can be formulated as a cost-function minimization problem over a finite time horizon T. To this end, let $\boldsymbol{o}(t) = \left[o_{1(t)}^T, \dots, o_{\mathfrak{o}(t)T}\right]^T$ be the state vector of the surrounding obstacles where $o_{i(t)} = \left[x_{i(t)}, y_{i(t)}\right]^T$ represents the state of the i-th obstacle considered at time t described by its coordinates, and $\mathfrak{o}(t)$ is the number of detected obstacles at time t. Note that, a time-varying index mapping $i(t)$ is used in the mathematical formulation to model the fact that the i-th obstacle considered for planning the trajectory at time t depends on the actual robot position at that time t.

In addition, let $\boldsymbol{q}(t)$ and $\boldsymbol{u}(t)$ be the state and the input vectors of a robot at time t, respectively, with dynamics defined according to Equation (1) and expressed in a compact form according to Equation (2). The following optimization problem formulation is considered for the navigation of an Ackermann steering vehicle in a high-precision farming environment filled with obstacles

$$\begin{aligned} &\min_{\boldsymbol{q}(t)} h\left(\boldsymbol{q}(t_f)\right) + \int_{t_0}^{t_f} c_n\big(\boldsymbol{q}(t), \boldsymbol{u}(t)\big) + c_o\big(\boldsymbol{q}(t), \boldsymbol{o}(t)\big)\, dt \\ &\text{Subject to} \quad \dot{\boldsymbol{q}}(t) = f\big(\boldsymbol{q}(t), \boldsymbol{u}(t)\big), \\ &\qquad\qquad r\big(\boldsymbol{q}(t_0)\big) = 0, \\ &\qquad\qquad l\big(\boldsymbol{q}(t), \boldsymbol{i}(t)\big) \leq 0 \quad \forall\, i \in \{1, \dots, \mathfrak{o}(t)\} \end{aligned} \tag{3}$$

where $h\left(\boldsymbol{q}(t_f)\right)$ describes the cost in the final state, $c_n\big(\boldsymbol{q}(t), \boldsymbol{u}(t)\big)$ describes the navigation cost, $c_o\big(\boldsymbol{q}(t), \boldsymbol{o}(t)\big)$ encodes the obstacle avoidance cost, $r\big(\boldsymbol{q}(t), \boldsymbol{o}(t)\big)$ and $l\big(\boldsymbol{q}(t), i(t)\big)$ encode the set of equality and inequality constraints to satisfy along the trajectory, and $T = t_f - t_0$ represents the time horizon in which we want to find the solution. In particular, the cost function and the set of equality and inequality constraints are defined as

$$h\left(q(t_f)\right) = \left(q_d - q(t_f)\right)^T Q_f \left(q_d - q(t_f)\right)$$

$$c_o\left(q(t), o(t)\right) = \sum_{i=1}^{\mathfrak{o}(t)} \left(\left(p(t) - o_i(t)\right)^T W_i \left(p(t) - o_i(t)\right)\right)^{-1}$$

$$c_n\left(q(t), u(t)\right) = u(t)^T R\, u(t) + \left(q(t) - q_f\right)^T Q \left(q(t) - q_f\right) \quad (4)$$

$$l\left(q(t), i(t)\right) = d_{min}^T d_{min} - \left(p(t) - o_i(t)\right)^T \left(p(t) - o_i(t)\right)$$

$$\left(q(t_0)\right) = \left(q_i - q(t_0)\right)^T \left(q_i - q(t_0)\right)$$

with q_d representing the goal state, d_{min} safety distance from an obstacle, Q_f, W_i, Q, R weighting matrices, q_i and q_f are the initial pose and desired final pose, respectively.

In addition, it should be noticed that in order to comply with the real-time requirements of the application and to increase the flexibility of the approach, obstacles should be dynamically added and removed according to a proper detection and selection approach, e.g., only detected obstacles within a certain radius from the current robot location are considered for the planning. An obstacle detection algorithm purposely designed for the precision farming setting considered within the project PANTHEON, has been developed inside the Simultaneous Localization and Mapping module. We refer the reader to Section 1.4 for a comprehensive description.

1.3.2 Low-Level Control Law

In this section, we focus on the pose regulation problem for the Bicycle kinematics model given in Equation (1), *i.e.*, the problem of designing a control law which is capable of steering the vehicle from any initial configuration $q_i = [x_i, y_i, \phi_i]^T$ to any desired final configuration $q_d = [x_d, y_d, \phi_d]^T$. To facilitate the analysis, it is convenient to formulate the regulation problem in polar coordinates $\hat{q} = [\rho, \gamma, \delta]^T$. For a given pose $q = [x, y, \phi]^T$ this coordinate transformation is defined as

$$\rho = \sqrt{(x_d - x)^2 + (y_d - y)^2}$$

$$\gamma = atan2\left((y_d - y), (x_d - x)\right) - \theta + \pi \quad (5)$$

$$\delta = \gamma - (\theta_d - \theta)$$

where ρ is the distance between the point (x, y) of the Ackermann reference frame $\mathcal{F}_{\mathcal{B}}$ and the origin of the Cartesian plane, γ the angle between the Cartesian error vector $e_p = p_d - p$ and the sagittal axis of the vehicle, and δ the angle between the same vector and the x-axis. In these polar coordinates $\hat{q} = [\rho, \gamma, \delta]^T$, the kinematic model of the Ackermann steering is

$$\dot{\rho} = -v\cos(\gamma)$$

$$\dot{\gamma} = \frac{\sin(\gamma)}{\rho} v - \frac{v}{l}\tan(\phi) \quad (6)$$

$$\dot{\delta} = \frac{\sin(\gamma)}{\rho} v$$

note that the input vector field associated with $v = 0$ is singular for $\rho = 0$. In the sequel, with no lack of generality we assume that the desired configuration is the origin, that is $q_d = [0, 0, 0]^T$. For a full theoretical characterization of this non-smooth control law with its convergence properties the reader is referred to the deliverable 3.1.

1.3.3 Numerical Validation

In this section, we numerically demonstrate the capabilities of the proposed navigation architecture in a simulated hazelnut farming scenario. The experiments have been carried out through GAZEBO, a robotic simulation toolbox. The non-linear planning problem is set up with the ACADO toolbox and solved by the qpOASES solver [2]. By using the ACADO code generation tool, the problem is exported in a highly efficient c-code that we integrated within a ROS (Robot Operating System) node. We set the discretization step to be $dt = 0.5\ s$ with a time horizon $T = 15\ s$. The planned trajectory, which is described by a discrete sequence of poses, is then fed into the control law node as a sequence of waypoints, while the output Ackermann control inputs are mapped to the wheel motors by a simulated low-level controller running on-board.

The experiment consists of planning trajectories from a given initial pose q_i to a sequence of desired poses $\mathfrak{q}_\mathfrak{b} = (q_{d_1}, \dots, q_{d_M})$. For each pair of poses, the local planner will compute an obstacle-free trajectory described by a sequence of successive waypoints. To ensure smooth navigation with no stop-and-go behavior, a new waypoint is fed into the control law when the distance from the current waypoint has become smaller than a given threshold (set to 0.2 meters in our simulations). For the numerical validation, the control gains are tuned as $(k_1, k_2, k_3) = (1,6,3)$. Indeed, this choice has numerically proven the robot to satisfactory follow the obstacle-free trajectory. It should be noticed that for the numerical validation, a custom-built GAZEBO model of the SHERPA robotic platform has been developed. Briefly, this model is equipped with the same sensor setup as the real platform and moves according to the Ackermann steering kinematics. An example of a planned trajectory in the presence of a dynamic obstacle is depicted in Figure 2 (left). Table I reports trajectory statistics from exhaustive simulations carried out with different setups. Results are sorted by increasing trajectory planning difficulties, either in terms of path length or presence of dynamic obstacles. The success rate is almost $100\ \%$, while it slightly decreases when adding ``last-meter'' dynamic obstacles along the planned route, which may leave no space to corrective maneuver. Nevertheless, the final position and angle errors remain almost unchanged, proving the effectiveness of the control law.

1.3.4 Experimental Setup

Real experiments have been carried out within one of the real-word hazelnut orchards of the PANTHEON project, within the ``Azienda Agricola Vignola'', a farm located in the municipality of Caprarola, in the province of Viterbo, Italy. In particular, the experiments have been carried out in a young orchard (cultivar Nocchione treated as multi stemmed bushes) with a $4.5\ m\ x\ 3.0\ m$ layout. Figure 2 (right) shows the ground vehicle prototype, namely SHERPA HL robotic platform R-A, which has been used for the experimental validation.

Figure 2 - Left: examples of a planned trajectory in a simulated scenario filled with static obstacles (green) and a dynamic obstacle (red). The green obstacles are arranged to resemble the planting layout of hazelnut trees as in the real field. Right: the SHERPA HL R-A robotic platform

Dynamic Obstacle	Trajectory Length [m]	Failure Rate [%]	V[m/s]	ϕ [rad]	Min. Obst. Distance [m]	Position Error [m]	Angle Error [rad]
	5	.02	.34	.75	1.34	.11	.08
	7	.03	.33	.81	1.41	.09	.05
	9	.03	.36	.78	1.36	.07	.07
✓	9	4.06	.4	.98	1.19	.13	.12
✓	12	4.75	.43	1.04	1.25	.11	.09
✓	13	3.97	.41	1.05	1.23	.12	.11

Table I - Simulations: trajectory statistical analysis

The SHERPA HL R-A platform is mechanically designed to operate according to the Ackermann Steering kinematics. The platform is equipped with a Trimble MB-Two GNSS Receiver with GPS-RTK capabilities, a SBG Ellipse2-E IMU with an integrated compass, two Sick S300 Safety Laser Scanners, and a Velodyne VLP-16 Puck LITE 3D LIDAR. The platform also mounts an Intel NUC NF697 with an Intel Core i7-8705G Processor where the low-level software and the proposed control law run. The planner runs on an independent laptop with an Intel i7-8750H processor and the optimal trajectories are sent to the robot through a Wi-Fi connection. The non-linear planner and the control law are set up as described in the previous section.

1.3.5 Pose Regulation Experimental Validation

In this section, we experimentally demonstrate the effectiveness of the proposed control law in a real-world high-precision farming scenario, i.e., the hazelnut orchard described in the previous section. In all the regulation experiments the control gains and the desired state are tuned as $(k_1, k_2, k_3) = (0.3, 1.5, 3)$ and $q_d = (q_{x,d}, q_{y,d}, q_{z,d})$, respectively. The pose regulation task is performed with the real robotic platform and for 8 different starting poses q_i derived by sampling a set of positions over a circle with a radius of 6 meters. The trajectories performed by the robot are reported in Figure 3, where it can be noticed that the robot successfully reaches the desired pose from all the initial conditions and shows a symmetrical behavior. Note that, to ease the visualization of this experimental validation the origin of the reference frame is relocated at the desired position p_i of the robot.

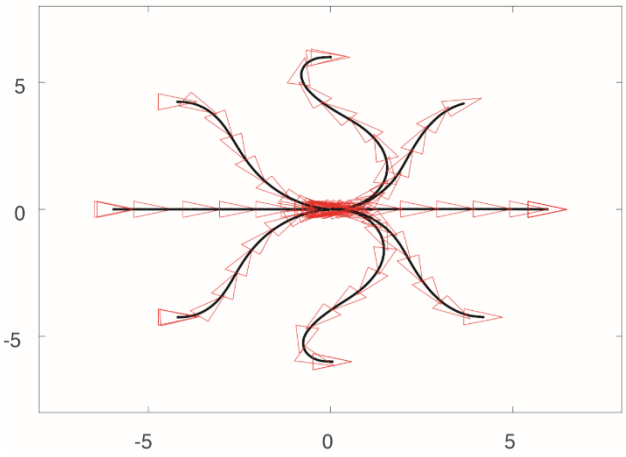

Figure 3 - Trajectories performed by the robot to reach the desired state in the field 16 of the hazelnut orchard. In red, the orientation of the vehicle sampled proportionally to the vehicle velocity.

1.3.6 Navigation Experimental Validation

In this section, we experimentally demonstrate the capabilities of the proposed navigation architecture in a real-world high-precision farming scenario, i.e., the hazelnut orchard described in Section 1.3.4. We demand the robot to move from an initial pose q_i to the desired pose q_d on a different orchard row. Thus, forcing the robot to plan a safe trajectory among trees.

Three examples of trajectories performed by the real robotic platform are illustrated in Figure (4), where in the first case the robot moves forward while in the second and third cases the robot moves backward. It can be noticed that in all three cases, the robot successfully reaches the desired goal by passing in the intermediate point between two adjacent trees when switching from one row to another. In the third case, we spawn a virtual obstacle on the planned route. The robot is then forced to re-plan a different trajectory (depicted in red) to reach the desired goal. As reported in Table II, the robot always manages to reach the goal with a small error. The only exception is the third trajectory where, having delayed the switching maneuver between the rows, the robot does not find enough room to align its yaw with the desired one. Note that, to ease the visualization of these experimental results, in this case, the origin of the reference frame is relocated at the initial position p_i of the robot. In addition, it should be further noticed that the second and the third trajectories depicted in Figure 4 are also reported in the attached multimedia material together with a real footage of the robotic platform.

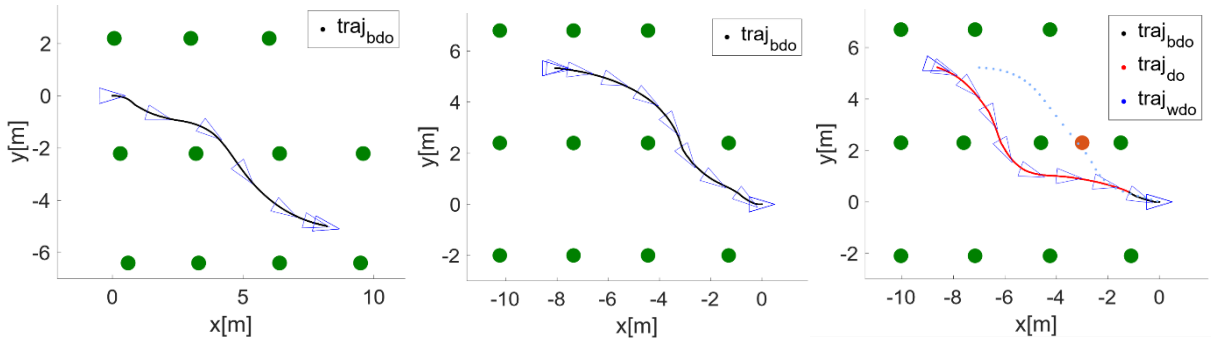

Figure 4 - Trajectories performed by the robotic platform in a real hazelnut orchard (plants are represented by the green circles). From top to bottom, a front-wise maneuver, a rear-wise maneuver and a maneuver with a dynamic obstacle (red dot). The second and third trajectories are also reported in the attached multimedia material together with a real footage of the robotic platform.

Dynamic Obstacle	Trajectory Length [m]	V[m/s]	ϕ [rad]	Min. Obst. Distance [m]	Position Error [m]	Angle Error [rad]
	9.5	.28	.3	1.01	.09	.12
	9.5	.31	.31	1.05	.08	.09
✔	9.5	.29	.41	1.14	.11	.19

Table II - Experiments: trajectory statistical analysis.

1.3.7 Computational Time

In this section, we numerically validate the real-time planning capability of the proposed navigation architecture. To this end, it should be noticed that the computational cost is not constant. Indeed, it may vary according to the specific operating mode of the planner. For this reason, we distinguish among three different planning modes:

- Initial mode (blue): which occurs when the vehicle starts planning a new trajectory.
- Steady mode (green): which occurs when the vehicle is already moving toward the target pose.
- Emergency mode (red): which occurs when a dynamic obstacle suddenly appears along the optimal trajectory.

Figure 5 reports the average computational costs for all the planning phases. The average computational cost is constantly lower than $0.033\ s$. As expected, the steady-planning phase turns out to be the cheapest one since the robot moves slowly and the trajectory to re-plan is close to the previous one. Conversely, the emergency re-planning phase is the most expensive and variable, as the trajectory to re-plan is usually far from the previous one since the dynamic obstacle might drastically change the optimal path.

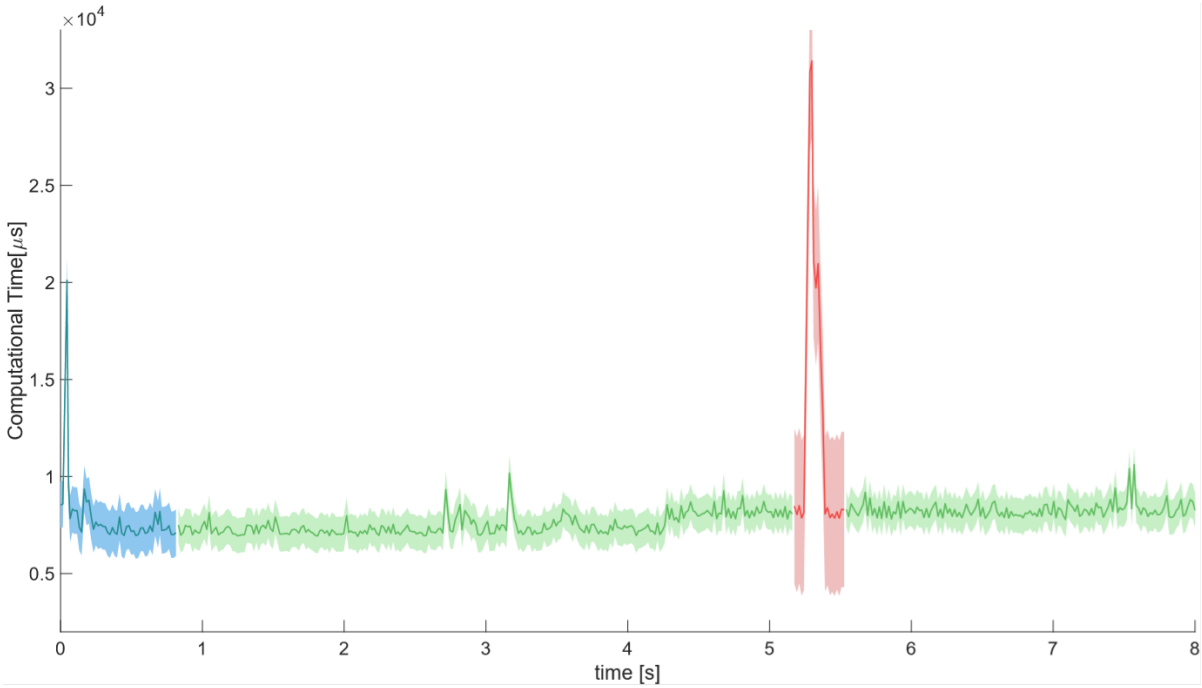

Figure 5 - Average computational time plot across the planning phases over a time horizon of 8 seconds: the (i) planning phase in blue, (ii) the steady-planning phase in green, and (iii) the emergency re-planning phase in red. The shaded areas represent the variance of the average computational time.

1.4 Simultaneous Localization and Mapping

To localize and map the orchard environment we developed a compact and hierarchical Simultaneous Localization and Mapping system purposely designed for large-scale hazelnut orchard farming. The proposed system is composed of two layers: (i) an Extended Kalman Filter (EKF) that performs a SLAM and (ii) a Kalman Filter (KF) that estimates the planting pattern of the target field, *i.e.*, the geometric arrangement of the plants in the orchard. The former formulates the mapping problem as the estimation of a set of landmarks, each one representing an individual tree in the field. To keep the map estimation scalable to the field size, the update process is carried out in a dynamically partitioned manner by considering in the update step only a small number of landmarks in the neighborhood of the moving platform. To achieve this, we consider a topological map encoded by an undirected graph, where the vertexes represent trees and the edges describe 1-hop proximity Manhattan distances. Note that, an undirected graph is a particular instantiation of a directed graph $\mathcal{G} = \{V, E\}$ as defined in Section 1.1, where the existence of an edge $e = (v_i, v_j) \in E$ implies also the existence of an edge $e' = (v_j, v_i) \in E$. The geo-referentiation is finally achieved by a GPS-based correction in the EKF. The latter estimates at the same time the planting pattern of the considered field by exploiting the currently built map as a virtual measurement.

The estimated pattern is then exploited in an outlier rejection process in the EKF, increasing the robustness of the proposed architecture and enabling to distinguish between trees and additional dynamical obstacles in the target field. Real-world experiments within a hazelnut farm located in the municipality of Caprarola, in the province of Viterbo, Italy, are provided. To validate the effectiveness and the robustness of the proposed SLAM architecture under significantly different working conditions, experiments have been carried out across different seasons (see Figure 6). The proposed slam architecture has been submitted and is currently under review at the Journal of Field Robotics (JFR).

Figure 6 - From left to right, images reporting the working scenario in summer and winter, respectively. It can be noticed how the presence of the canopy on the trees significantly changes the working conditions. Indeed, this may significantly affect the SLAM performance.

1.4.1 EKF SLAM in orchards

The goal of a SLAM algorithm is to track, in real-time, the pose of a moving platform, i.e., *localization,* while estimating the position of the surrounding environment, i.e., *mapping*. A SLAM algorithm in an orchard-like scenario must also provide some additional properties: (i) a compact representation of the map, (ii) the scalability of the computational complexity with respect to the map extent, and (iii) the geo-referentiation of each object in the map. The first two properties enable the SLAM to properly work over extended areas and over long operational time since this is the usual working conditions of farming robots in orchard settings. Specifically, the first property enables the map to have a negligible memory footprint, while the second keeps the computational complexity constant with respect to the extent of the map. Finally, the last property is needed by the farmer to have a unique association between the map entries and the real trees of the field to monitor. In this section, we provide a comprehensive overview of the proposed SLAM architecture including also how the required properties are guaranteed. The SLAM problem is formulated in a probabilistic setting be resorting to a landmark-based EKF, considering each tree in the orchard as a landmark to which a unique ID is assigned. The first property is thus guaranteed by the proposed formulation which presents a compact form differently from standard keypoint-based representations of the environment that lead the map size to grow unbounded over time. Specifically, let the 2D position and the heading of the robot be $\boldsymbol{t} = [\, x \; y \,]^T$ and θ, respectively, thus defining the robot pose as $\boldsymbol{p} = [\, x \; y \; \theta \,]^T \in R^3$. Let the i^{th} landmark 2D position be $\boldsymbol{l}_i = \left[\, l_i^x \; l_i^y \,\right]^T \in R^2$. Thus, the full robot state results to be $\boldsymbol{\mu} = [\boldsymbol{p}, \boldsymbol{m}]^T \in \mathcal{R}^{3+2N}$ which considers both the robot pose and the full map $\boldsymbol{m} = [\, \boldsymbol{l}_1, \; \dots \; , \boldsymbol{l}_N]^T \in \mathcal{R}^{2N}$, where N represent the total number of mapped landmarks. Let the input vector be $\boldsymbol{u} = [\, v \; \phi \,]^T \in \mathcal{R}^2$, composed of the linear velocity and the steering angle of the wheels, respectively. Also, let the measurement of the i^{th} landmark be $\boldsymbol{z}_i^L = \left[x_i^l, y_i^l\right]$, given by the 2D coordinates of the sensed plant, and the GPS measurement, consisting into the global robot position be $\boldsymbol{z}^G = [x^G, y^G]$. Finally, for the above introduced observation models, we also define their relative observation matrices $\boldsymbol{C}_L$ and $\boldsymbol{C}_G$, respectively, which model how the external measurements act on the EKF state. Notice that, in case of non-linear observations functions, the observation matrices are obtained through the calculation of the Jacobian with respect to the EKF state variables (see Algorithm I). The transition and observation functions are thus defined as follow

$$\begin{bmatrix} \boldsymbol{p}_{t+1} \\ \boldsymbol{m} \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{p}_t, \boldsymbol{u}_t) \\ \boldsymbol{m} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_t + v_t cos(\theta_t)\Delta t \\ \boldsymbol{y}_t + v_t sin(\theta_t)\Delta t \\ \theta_t + \dfrac{v_t}{l}\, tan(\theta_t)\Delta t \\ \boldsymbol{m} \end{bmatrix} \tag{7}$$

$$h_i^L(\boldsymbol{\mu}) = \boldsymbol{R}_W^B(\boldsymbol{l}_i - \boldsymbol{t}), \boldsymbol{C}_L \in \mathcal{R}^{2\times(3+2N)} \tag{8}$$

$$h^G(\boldsymbol{\mu}) = \boldsymbol{t}, \boldsymbol{C}_G \in \mathcal{R}^{2\times(3+2N)} \tag{9}$$

Where Equation (7) represents the prediction step which only affects the pose of the vehicle since the map m is assumed to be static. The kinematic model has been approximated assuming the input vector u constant for the time Δt. Specifically, the prediction of the vehicle heading θ follows an Ackermann kinematic model, where the angular velocity is derived by $\omega = \frac{v}{l}\tan\phi$ where s is the inter-axes length of the vehicle and ϕ its steering angle. The landmark observation model Equation (8) is formulated as a change of coordinates. Specifically, the i^{th} landmark is transformed from the world coordinate frame $\mathcal{F}_{\mathcal{W}}$ to the robot coordinate frame $\mathcal{F}_{\mathcal{B}}$ with the rotation matrix $\boldsymbol{R}_W^B$. Differently, the GPS observation model Equation (9) does not require any coordinate transformation, since both quantities are expressed in the $\mathcal{F}_{\mathcal{W}}$ frame. Moreover, the GPS-based corrections enable the map to be geo-referenced, thus satisfying the third requirement stated at the start of this section. Prediction and Correction steps are carried out in a standard EKF manner (see [3]) for

which we provide a pseudo-code in Algorithm I for the convenience of the reader. Despite their simplicity, the observation models in Equations (8)-(9) might not properly scale with the field extent. Indeed, the size of the matrices $\boldsymbol{C}_L$ and $\boldsymbol{C}_G$ increases with the dimension of the map, exceeding at a certain point the size for real-time performance. It is well known indeed how the correction step represents one of the main bottlenecks for any SLAM architecture in terms of computational complexity [4] [5]. We tackle this problem by a dynamical partitioned update step, where the neighbors landmark to update are selected trough a k-hop Manhattan proximity distance. To do so, we define the partitioned state $\boldsymbol{\mu}_{\mathcal{N}(i,j)^k}$ and the partitioned covariance matrix $\boldsymbol{P}_{\mathcal{N}(i,j)^k}$, where we recall that with $\mathcal{N}(i,j)^k$ we denote the union of the k-hop neighborhoods of the trees i and j. The partitioning is carried out by exploiting the topological map formulation described in Section 1.1 to select a subset of trees based on the k-hop neighborhood to be used in the partitioned update step. Therefore, the observation models $h^L(\boldsymbol{\mu})$ and $h^G(\boldsymbol{\mu})$ changes into

$$h_i^L\left(\boldsymbol{\mu}_{\mathcal{N}(i)^k}\right) = \boldsymbol{R}_W^B(\boldsymbol{l}_i - \boldsymbol{t}), \boldsymbol{C}_L \in R^{2\times(3+2|\mathcal{N}(i)^k|)} \tag{10}$$

$$h^G\left(\boldsymbol{\mu}_{\mathcal{N}(i)^k}\right) = \boldsymbol{t}, \boldsymbol{C}_G \in R^{2\times(3+2|\mathcal{N}(i)^k|)} \tag{11}$$

The new observation models in Equations (10)-(11) are the core of the proposed SLAM architecture since they do not consider anymore the full state μ, but just a portion of it. This allows keeping the computational burden of the correction step in the real-time domain, enabling the proposed EKF architecture to properly scale with the field extent and to satisfy the second requirement of SLAM in hazelnut orchards. To carry out the EKF update step, we rely on the idea of the Schmidt partial update [6]. An illustration of this strategy is reported in Equations (12)-(15). The main idea is to dynamically perform a coordinate transformation by means of a proper transformation matrix $\boldsymbol{T}$, namely a permutation matrix, in order to transform the state vector μ into a state vector $\hat{\mu}$ composed of two blocks, one block collecting all the states to be updated $\boldsymbol{\mu}_x$ and one block collecting the remaining part of the states $\boldsymbol{\mu}_y$, that is

$$\boldsymbol{\mu} = \begin{bmatrix}\underline{\boldsymbol{p}} & \boldsymbol{l}_1 & \cdots & \boldsymbol{l}_k & \boldsymbol{l}_m & \cdots & \boldsymbol{l}_N\end{bmatrix}^T, \quad \hat{\boldsymbol{\mu}} = \boldsymbol{T}\boldsymbol{\mu} = \begin{bmatrix}\boldsymbol{\mu}_x & \boldsymbol{\mu}_y\end{bmatrix}^T \tag{12}$$

where $\boldsymbol{\mu}_x = \boldsymbol{\mu}_{\mathcal{N}(k,m)^1}$ and μ_y are defined respectively as:

$$\boldsymbol{\mu}_x = \begin{bmatrix}\boldsymbol{p} & \boldsymbol{l}_k & \boldsymbol{l}_m\end{bmatrix}^T, \quad \boldsymbol{\mu}_y = \begin{bmatrix}\boldsymbol{l}_1 & \cdots & \boldsymbol{l}_N\end{bmatrix}^T \tag{13}$$

Similarly, the covariance matrix $\boldsymbol{P}$ is re-arranged in a block-wise manner. In particular, the transformed covariance matrix $\widehat{\boldsymbol{P}}$ has a block-diagonal composed of a block $\boldsymbol{P}_{xx}$ which embeds the variables to consider in the update step ($i.e.$ the 2D position of the robot and the k-hop neighbors) and of a block $\boldsymbol{P}_{yy}$ which embeds the variables that will not be taken into account during the update step, that is

$$\boldsymbol{P} = \begin{bmatrix} \underline{\boldsymbol{P}_p} & \boldsymbol{P}_{pl_1} & \cdots & \underline{\boldsymbol{P}_{pl_k}} & \underline{\boldsymbol{P}_{pl_m}} & \cdots & \boldsymbol{P}_{pl_N} \\ \boldsymbol{P}_{pl_1}^T & \boldsymbol{P}_{l_1} & \cdots & \boldsymbol{P}_{l_1l_k} & \boldsymbol{P}_{l_1l_m} & \cdots & \boldsymbol{P}_{l_Nl_1} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \underline{\boldsymbol{P}_{pl_k}^T} & \boldsymbol{P}_{l_1l_k}^T & \cdots & \underline{\boldsymbol{P}_{l_k}} & \underline{\boldsymbol{P}_{l_kl_m}} & \cdots & \boldsymbol{P}_{l_Nl_k} \\ \underline{\boldsymbol{P}_{pl_m}^T} & \boldsymbol{P}_{l_1l_m}^T & \cdots & \underline{\boldsymbol{P}_{l_kl_m}^T} & \underline{\boldsymbol{P}_{l_m}} & \cdots & \boldsymbol{P}_{l_Nl_m} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{P}_{pl_N}^T & \boldsymbol{P}_{l_Nl_1}^T & \cdots & \boldsymbol{P}_{l_Nl_k}^T & \boldsymbol{P}_{l_Nl_m} & \cdots & \boldsymbol{P}_{l_N} \end{bmatrix}, \widehat{\mathbf{P}} = \mathbf{T}\,\mathbf{P}\,\mathbf{T}^{-1} = \begin{bmatrix}\boldsymbol{P}_{xx} & \boldsymbol{P}_{xy} \\ \boldsymbol{P}_{yx} & \boldsymbol{P}_{yy}\end{bmatrix} \tag{14}$$

where $\boldsymbol{P}_{xx} = \boldsymbol{P}_{\mathcal{N}(k,m)^1}$ and $\boldsymbol{P}_{yy}$ are defined respectively as

$$\boldsymbol{P}_{xx} = \begin{bmatrix} \boldsymbol{P}_p & \boldsymbol{P}_{pl_k} & \boldsymbol{P}_{l_k l_m} \\ \boldsymbol{P}_{pl_k}^T & \boldsymbol{P}_{l_k} & \boldsymbol{P}_{l_k l_m} \\ \boldsymbol{P}_{pl_m}^T & \boldsymbol{P}_{l_k l_m}^T & \boldsymbol{P}_{l_m} \end{bmatrix}, \quad \boldsymbol{P}_{yy} = \begin{bmatrix} \boldsymbol{P}_{l_1} & \cdots & \boldsymbol{P}_{l_1 l_N} \\ \vdots & \ddots & \vdots \\ \boldsymbol{P}_{l_1 l_N}^T & \cdots & \boldsymbol{P}_{l_N} \end{bmatrix} \tag{15}$$

Data: State $\boldsymbol{\mu} \in \mathbb{R}^m$, Covariance $\boldsymbol{P} \in \mathbb{R}^{m \times m}$, Observations $\boldsymbol{\mathcal{O}} \in \mathbb{R}^n$
Given
- Transition model: $\boldsymbol{\mu}_{t+1} = f(\boldsymbol{\mu}_t, \boldsymbol{u}_t)$
- Measurement models: $\boldsymbol{y}^L = h^L(\boldsymbol{\mu})$, $\boldsymbol{y}^G = h^G(\boldsymbol{\mu})$
for *each new observation* $\boldsymbol{o}_j \in \boldsymbol{\mathcal{O}}$ **do**
  if $\boldsymbol{o}_j \in$ *wheel encoders* **then**
    $\boldsymbol{u} = \boldsymbol{o}_j$
    $\boldsymbol{N_u} \leftarrow computeInputNoise(\boldsymbol{u})$
    $\boldsymbol{A_t} = \left.\frac{\partial f}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{\mu},\boldsymbol{u}}$, $\boldsymbol{B_t} = \left.\frac{\partial f}{\partial \boldsymbol{u}}\right|_{\boldsymbol{\mu},\boldsymbol{u}}$
    `PredictionStep`$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{u}, \boldsymbol{N}_u, \boldsymbol{\mu}_t, \boldsymbol{P}_t)$
  else if $\boldsymbol{o}_j \in$ *LiDAR* **then**
    $\boldsymbol{z}_j^L = \boldsymbol{o}_j$
    $\boldsymbol{y}_i^L \leftarrow DataAssociation(\boldsymbol{z}_j^L)$
    $\widehat{\boldsymbol{y}}_i \leftarrow EstimatedMeasurement()$
    $\boldsymbol{C}_L = \left.\frac{\partial h^L}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{\mu},\boldsymbol{u}}$
    `UpdateStep`$(\boldsymbol{C}_L, \boldsymbol{y}_i^L, \widehat{\boldsymbol{y}}_i, \boldsymbol{N}_l, \boldsymbol{\mu}_t, \boldsymbol{P}_t)$
  else if $\boldsymbol{o}_j \in$ *GPS* **then**
    if $\boldsymbol{o}_j.Covariance.norm() > tr_{GPS}$ **then**
      continue
    $\boldsymbol{z}^G = \boldsymbol{o}_j$
    $\boldsymbol{N}_g \leftarrow computeGPSNoise(\boldsymbol{z}^G)$
    $\widehat{\boldsymbol{y}} \leftarrow EstimatedMeasurement()$
    $\boldsymbol{C}_G = \left.\frac{\partial h^G}{\partial \boldsymbol{\mu}}\right|_{\boldsymbol{\mu},\boldsymbol{u}}$
    `UpdateStep`$(\boldsymbol{C}_G, \boldsymbol{z}^G, \widehat{\boldsymbol{y}}, \boldsymbol{N}_g, \boldsymbol{\mu}_t, \boldsymbol{P}_t)$
end

`PredictionStep`$(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{u}, \boldsymbol{N}_u, \boldsymbol{\mu}, \boldsymbol{P})$
  $\boldsymbol{\mu} = \boldsymbol{\mu} + A\boldsymbol{u}$
  $\boldsymbol{P} = \boldsymbol{A}\boldsymbol{P}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{N}_u\boldsymbol{B}^T$

`UpdateStep`$(\boldsymbol{C}, \boldsymbol{z}, \boldsymbol{h}, \boldsymbol{N}, \boldsymbol{\mu}, \boldsymbol{P})$
  $\{\boldsymbol{P}_{xx}, \boldsymbol{P}_{xy}, \boldsymbol{\mu}_x\} \leftarrow PartitionsExtraction(\boldsymbol{P}, \boldsymbol{\mu})$
  $\boldsymbol{K} = \boldsymbol{P}_{xx}\boldsymbol{C}^T(\boldsymbol{C}\boldsymbol{P}_{xx}\boldsymbol{C}^T + \boldsymbol{N})^{-1}$
  $\boldsymbol{\mu}_x = \boldsymbol{\mu}_x + \boldsymbol{K}(\boldsymbol{z} - \boldsymbol{h})$
  $\boldsymbol{P}_{xx} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C})\boldsymbol{P}_{xx}(\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C})^T + \boldsymbol{C}\boldsymbol{N}\boldsymbol{C}^T$
  $\boldsymbol{P}_{xy} = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{C})\boldsymbol{P}_{xy}$
  $\boldsymbol{P}_{yx} = \boldsymbol{P}_{xy}^T$

Algorithm I – EKF SLAM Algorithm

Note that, from an implementation standpoint, the computation cost involved in the inversion of the matrix $\boldsymbol{T}$ is almost negligible, even when the map has considerable dimensions, since permutation matrices are orthogonal (*i.e.* $\boldsymbol{T}^{-1} = \boldsymbol{T}^{T}$). The k neighborhood size is a design parameter of the proposed architecture and its choice is a trade-off between the computational burden of the state update step and its optimality. Indeed, as reported in the experimental Section 1.4.5, the smaller the neighborhood size, the lower the resulting accuracy of the map. This simple heuristic allows keeping an almost constant computational complexity. The only operations that grow in complexity as the size of the map increases are the extraction and the update of the $\boldsymbol{P}_{xy}$ partition of the covariance matrix. However, this growth follows a linear trend and the actual computational time involved in these operations is almost negligible.

Despite the simplicity of the proposed update policy, when it is implemented in a real EKF-SLAM architecture some issues may arise. Indeed, it is well-known that in standard EKF-SLAM architectures the covariance matrix can potentially become negative-definite due to numerical approximations or to not well-handled non-linearities [7]. A widely used solution is to rely on the Joseph form update, which is calculated on squared terms, thus guaranteeing a positive solution. For this, in Algorithm I, the $\boldsymbol{P}_{xx}$ update is performed using the latter approach.

We would like to point out that, the prediction step is updated every time a new encoder odometry is available. Normally this rate should be as high as possible in order to reduce the Δt and to have more accuracy in the prediction of the new robot pose. On the contrary, the LiDAR and GPS corrections occur when new data is available and depends moreover on the specifications of the sensors. A further discussion is needed for GPS correction. This step, indeed, could not be always beneficial for the algorithm. The GNSS could be unreliable under foliage, especially when the orchards are in a fully vegetative state, or for a temporary signal loss of RTK correction emitters. On the other hand, the GPS measurement is provided with its own covariance matrix which expresses the level of precision of the current measurement. The GPS correction step is thus performed only when the measurement covariance matrix norm is below a specific threshold of tr_{GPS}. Finally, to associate new LiDAR measurements to the landmarks in the map, as reported in Algorithm I, we make use of a data association process which we discuss in the next section.

1.4.2 Data Association

Data association is the problem of deciding which noisy measurement corresponds to which existing landmark. Notably, this problem is further complicated by considering the possible existence of previously unknown landmarks in the map and the possibility of misleading measurements, i.e., outliers.

Specifically, in the considered working scenario, this process consists of associating the 2D landmarks representing the hazelnut trees with the o objects perceived by the on-board sensors $\boldsymbol{\mathcal{B}} \in R^{o}$. In addition, in case the perceived objects do not belong to the map of the orchard ($e.g.$ rocks, people, or other obstacles), the data association process has to classify them as outliers and send their position to the robot navigation module. We refer the reader to [1] for a comprehensive description of the navigation module installed on the PANTHEON platforms.

The data association is a delicate process and even a few wrong associations could potentially lead the entire SLAM to become unstable and, eventually, to diverge. In this work, to obtain a reliable data association step, we employ a 3D LiDAR sensor that returns, at each new measurement, a set of l 3D points $\mathcal{L} \in R^{l}$. Such points are arranged on n_r planar rings which provides, in total, a vertical Field Of View (FOV) of FOV_v degrees. In addition, each ring has, in general, a planar FOV of 360 degrees. The angular resolution among the rings is FOV_v/n_r degrees. According to the relative mounting orientation of this sensor on the moving platform, there will be a ring which is parallel to the robot chassis, henceforth denoted as horizontal *ring*, and a certain number of rings with a positive angle with respect to the planar ring. In this work, we assume to know the

3D transformation $\boldsymbol{T}_L^B$ that maps the measured set of 3D points from its local reference frame $\mathcal{F}_{\mathcal{M}}$ to the robot body reference frame $\mathcal{F}_{\mathcal{B}}$. The choice of a LiDAR as the main perception sensor comes from different considerations: (i) its usual availability on autonomous vehicles for navigation purposes, thus decreasing the overall cost, (ii) its robustness against the environmental conditions, such as illumination and (iii) the disposition of the sensed 3D points into rings according to a set of scanning angles. The latter property enables the possibility of considering the points as belonging to independent planes. Thus, by removing the z component from each ring, it is possible to reason on different 2D point clouds and to speed up the data processing. On the other hand, point clouds and, in general, 3D data in orchards present some specific challenges to consider:

- the process needs to be independent of the seasonal effects, such as the presence or the total absence of the foliage of the trees (see Figure 6);
- the process needs to be robust to outliers, such as grass, suckers or human operators moving nearby the robotic vehicle.

To handle those challenges, we process the 3D LiDAR data by considering a double-ring clustering approach which can be summarized in the following steps:

I. clusterization of the 2D points lying on a ring with a positive angle α_u w.r.t. the horizontal ring (henceforth denoted as the upper-ring);
II. construction of bounding boxes $\boldsymbol{\mathcal{BB}}$ around the obtained clusters in the upper-ring;
III. projection of the bounding box onto the horizontal (or lower) ring;
IV. calculation of the center of mass for the 2D points belonging to each bounding box.

Data: 3D Point Cloud $\mathcal{L} \in \mathbb{R}^l$, Upper-Ring Angle α_u
Result: 2D Observed Trees Center $\mathcal{B} \in \mathbb{R}^o$
$\{\mathcal{L}_u, \mathcal{L}_l\} \leftarrow RingWiseDataSplit(\mathcal{L})$
$\mathcal{B}_u \leftarrow Clustering(\mathcal{L}_u)$
for *all cluster* $\boldsymbol{b_i} \in \mathcal{B}_u$ **do**
  $\mathcal{BB} \leftarrow BoundingBoxes(\mathcal{B}_u)$
  $\mathcal{BB} \leftarrow LowerRingProjection(\mathcal{BB})$
  for *all lower-ring bounding boxes* $\boldsymbol{bb_i} \in \mathcal{BB}$ **do**
    $\boldsymbol{p_{b,i}} \leftarrow PointsInBoundingBox(\mathcal{L}_l, \boldsymbol{bb_i})$
    $\mathcal{B}.push\_back(CenterOfMassCalculation(\boldsymbol{p_{b,i}}))$
  end
end
return $\mathcal{B}$;

Algorithm II – 3D Data Processing Pipeline

The whole 3D processing pipeline is also reported in Algorithm II. Step (i) enables the clustering procedure to only consider points belonging to the hazelnut trees based on a simple observation: higher the ring, lower the probability to intercept grass, stones, or other ground outliers. Although this approach may increase robustness against false positives, a few additional issues can arise such as in the case of adult trees for which the canopy may overlap and the clustering may yield poor results. Therefore, the choice of the α_u depends on the specific working scenario. Steps (ii), (iii), and (iv), filter out the outliers in the planar ring and enables the process to better estimate the central point of each tree. We point out that, despite the robustness, it is unlikely that the 3D processing pipeline described so far accurately estimates the center of the sensed tree. However, the proposed approach can successfully estimate the actual tree center by exploiting multiple

observations of the same tree from different point of views collected while navigating the orchard in its update process. By doing so, the 3D data processing ends up with 2D positions which roughly indicate the centers of mass of trees trunks whereby we feed the data association step. We remark that, differently from [8], this simple pipeline allows for a flexible and adaptable data association according to the real working conditions of the field, without implicitly assuming the terrain to be weed less.

Finally, the data association is carried out by standard maximum likelihood, mutual exclusion, and negative information heuristics, as described in [9]. Moreover, to increase the robustness of the Data Association we further rely on a plating pattern estimation process, which we discuss in the next section.

1.4.3 Planting Pattern Estimation

A useful cue when developing a SLAM architecture for agricultural robots in orchards is the knowledge of the planting pattern, *i.e.*, the geometric arrangement of the plants across the field. Specifically, by making available this information to the navigating platform enables for a more robust data association and outlier rejection, and also enables the robot to distinguish among planted trees and other obstacles. In this work, we model the planting pattern estimation problem as a chessboard with a rectangular chess fitting process (see Figure 7). Specifically, we use a rectangular grid to fit any possible planting scheme. The fitting process is carried out by relying on a Kalman Filter (KF), considering as state the planting pattern $\boldsymbol{s} = [\,\boldsymbol{c}\ \phi\,]^T$ where $\boldsymbol{c} = \left[\,c_x\ c_y\,\right]^T$ are the scale factors along the two major chessboard directions and ϕ the rotation angle which aligns the chessboard with the geo-referenced map estimated by the EKF-SLAM. The transition and observation models for this KF are defined as follows

$$\begin{aligned} \boldsymbol{s}_{t+1} &= f_s(\boldsymbol{s}_t) = \boldsymbol{s}_t \\ h^S(\boldsymbol{s}_t) &= z^S_{\mathcal{N}(i)^k,t} - \boldsymbol{s}_{i,t+1} \end{aligned} \tag{16}$$

The planting pattern has no dynamics, thus the transition function $f_s(\boldsymbol{s}_t)$ degenerates to the identity matrix $\boldsymbol{I} \in R^{3\times3}$. The observation model results from the difference between $\boldsymbol{z}^S_{N(i)^k,t}$ which can be considered as a virtual sensor measuring the chessboard distances of the k-hops landmarks neighbourhood and the actual chessboard estimate $\boldsymbol{s}_{i,t+1}$.

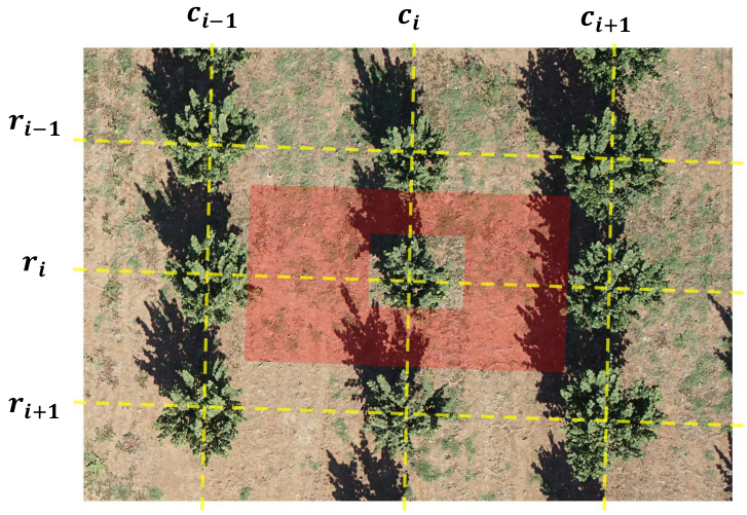

Figure 7 - A bird-eye view of a hazelnut orchard. In particular, the red shaded area represents the obstacle zone O, while the dotted yellow lines describe the planting pattern chessboard directions. For the tree in position (c_i, r_i) the 1-hop neighbourhood is defined as the set of trees: $[(c_i, r_{i+1}), (c_i, r_{i-1}), (c_{i+1}, r_i), (c_{i-1}, r_i)]$.

Data: Measurements $\mathcal{Z} \in \mathbb{R}^n$, Landmarks $\mathcal{L} \in \mathbb{R}^m$
Result: Boolean Vector $\mathcal{B} \in \mathbb{R}^n$
for *all* $z_i \in \mathcal{Z}$ **do**
  $\mathbb{O} \leftarrow computeObstacleSpace()$;
  $z_i \leftarrow sensorToGrid(z_i)$;
  for *all* $l_i \in \mathcal{L}$ **do**
    $l_i \leftarrow mapToGrid(l_i)$;
    $\mathcal{D}.append(distance(l_i, z_i))$;
  end
  $\mathcal{D}.sort()$;
  if $D.firstElement() \in \mathbb{O}$ **then**
    $\mathcal{B}.append(false)$;
  else
    $\mathcal{B}.append(true)$;
  end
  $\mathcal{D}.clear()$;
end
return $\mathcal{B}$;

Algorithm III – Outliers Rejection Heuristic

1.4.4 Outliers Detection

Modelling the orchard environment as a grid allows the robot to make an easier inference about where the trees are located or where they are likely to be located. At the same time, it allows to distinguish objects which do not belong to the orchard by exploiting the planting pattern model.

To exploit the planting pattern as an outlier rejection cue, we define, for each tree, an obstacle space O such that all the LiDAR measurements z_i lying in it are likely to be external objects (see Figure 7). The whole outlier rejection process is reported in Algorithm III. All the measurements marked as external objects are then rejected and not considered in the update step of the EKF. At this point, we reiterate that locating external objects is a useful capability for autonomous platforms. For example, it could prove useful in a Human-Robot Interaction (HRI) farming settings where some specific tasks are still performed manually, and adequate maneuvers are required for a safe interaction with human operators [10], [11]. Finally, depending on the nature and size of the obstacles, the navigation module may need their position to plan a safe trajectory for inspecting the field [1].

1.4.5 Experimental Results

Experiments have been carried out with data acquired in one of the real-word hazelnut orchards of the PANTHEON project, within the "Azienda Agricola Vignola", a farm located in the municipality of Caprarola, in the province of Viterbo, Italy (see Figure 8(right)). More specifically, we used a portion of field 18 with an area of about 1 hectare. The planting pattern of the considered field is $5 \times 5\ m$. Figure 8(left) shows the ground vehicle prototype, namely SHERPA HL robotic platform R-A, which has been used for the experimental validation. The platform is equipped with a RTK-enabled Trimble MB-Two GNSS receiver with a single antenna mounted on the top of the robot, a forward looking Velodyne VLP-16 Puck LITE 3D LIDAR and wheel encoders that provide a rough odometry estimation. Particularly, the latter has been used in the prediction step of the EKF. The EKF SLAM and the planting pattern KF filters have been coded in c++, and the communication with the sensors is implemented through a ROS node. Finally, the clustering of the 3D points in the lower ring has been carried out through the DBSCAN algorithm [12]. To test the proposed approach under different seasonal

conditions, we gathered three different datasets, namely $DataA$, $DataB$, and $DataC$. The total travelled distance for each dataset has been of about 400, 900,and 700 meters, respectively. In particular, $DataA$ has been collected in August which is the harvesting period, while $DataB$ and $DataC$ were collected in January and February, respectively, which are in the vegetation rest period (see Figure 8).

Figure 8 - On the top left, the SHERPA HL robotic platform prototype R-A, on the top right a satellite image on the experimental fields used in the PANTHEON project. On the second column, a hazelnut orchard in the harvesting and vegetation rest.

1.4.6 Evaluation Metrics

In this section, we describe the metrics that have been used to evaluate the performance of the proposed SLAM architecture. In general, the quality of a SLAM approach is defined in terms of localization and mapping accuracy which can be computed by resorting to some ground truth information [13]. In our setting, the only information that can be used as ground truth is data coming from the RTK-GPS. In this regard, since GPS data is used in the EKF update step, we propose three different evaluation metrics to evaluate the performance of the proposed SLAM architecture.

I. the translational drift relative error t_{RMSE};
II. the landmark positioning error m_{RMSE};
III. the planting pattern estimation error pp_{err};

The first two metrics make use of the Root Mean Squared Error ($RMSE$) defined as

$$RMSE(x) = \sqrt{\sum_{i=1}^{N} \frac{(\hat{x} - x)^2}{N}} \tag{17}$$

where $\hat{x}$ is the estimate and x is the parameter to be estimated. We denote the translational drift relative Root Mean Square Error as the t_{RMSE}, and the map Root Mean Square Error as the m_{RMSE}.

The first metric evaluates the SLAM localization capabilities. This is achieved by running the EKF using only LIDAR information while the RTK-GPS data is used as ground truth. In this regard, given the nature of the ground truth information, we compute the t_{RMSE} only with respect to the 2D position of the robotic vehicle, *i.e.* the heading of the vehicle is not considered. Notably, this metrics provides also information regarding the robustness of the proposed architecture, as a low value of the t_{RMSE} index indicates that the proposed algorithm can provide a consistent and reliable estimation even in the case of a temporary malfunction of the RTK-GPS (that may happen in specific working scenarios, such as the correction signal loss in presence of foliage).

The second metric evaluates the SLAM mapping capabilities. This is achieved by running the EKF using the full set of sensors in the SLAM pipeline. As mentioned before, the quality of the map is usually computed by comparing the map obtained by the EKF with some available ground truth information, [14]–[16]. In our setting, given the landmark-based nature of map, we compute m_{RMSE} error metrics by comparing the estimate of a landmark location with an accurate geo-referencing of its actual location.

Finally, the third metric evaluates the planting pattern estimation capabilities. This is achieved by measuring the Euclidean distance pp_{err} defined as

$$pp_{\mathrm{err}} = \sqrt{(\widehat{\boldsymbol{pp}} - \boldsymbol{pp})^2}$$

where $\widehat{\boldsymbol{pp}}$ represents the estimated planting pattern while $\boldsymbol{pp}$ the real one. Differently from the previous metrics, we measure the planting pattern estimation capabilities with and without the GPS update in the EKF to evaluate its effects on the estimation process. We finally remark that the t_{RMSE} is commonly adopted to benchmark SLAM approaches, see for instance [17] [18]. Differently, we purposely chose the other two metrics by considering the specific problem under analysis and the available ground truth information, such as the accurate geo-localization of a portion of the map and the knowledge of the actual planting pattern.

1.4.7 SLAM

According to the above-introduced metrics, we evaluate the proposed SLAM architecture on three datasets, namely $DataA$, $DataB$, $DataC$. Moreover, in order to analyze the effects of the partitioned update on the map and trajectory estimations, for each dataset we run the proposed SLAM approach with both the full map and with the following set of k-hop neighborhood $(1,2,3,4,5,6,8,10)$. The results are reported in Table III where, for each row, we highlight the metrics corresponding to the usage of the full map in the update process and the k-hop neighborhood configuration which results in the lowest error metrics.

A first important outcome regards the t_{RMSE}. In general, the smaller the t_{RMSE}, the closer the estimated trajectory to the ground truth one. We recall that, in this context, the estimated trajectory is obtained by using only the LiDAR in the update step of the EKF, while the ground truth is given by the RTK-GPS. As reported in Table III, the t_{RMSE} remains bounded between 3 and 5 centimeters when using a k-hop neighborhood with $k \geq 6$. The positive entailment of this result is twofold: (i) the proposed SLAM architecture is not highly dependent from the RTK-GPS data, and offers reliable and consistent performance even without its active usage; (ii) the temporary absence of the RTK-GPS update step, such as in presence of RTK signal losses or when the measurement covariance is above the allowed tr_{GPS}, does not represent a real issue.

Notably, the t_{RMSE} is higher in the $DataA$. This can be easily explained by the presence of the sucker at the base of the hazelnut trees, *i.e.* shoots that grow at the base of the tree. Indeed, their presence affects the data association process by introducing noise in the computation of the trunk center of mass. The same trend is indeed confirmed in the m_{RMSE} and pp_{err} metrics which are slightly worse than their respective values in $DataB$ and $DataC$.

			Neighborhood Size								
Dataset	Metrics [m]	GPS	1	2	3	4	5	6	8	10	full
$DataA$	t_{RMSE}		0.21	0.14	0.09	0.06	0.07	0.06	**0.05**	**0.05**	**0.05**
	m_{RMSE}	✓	0.277	0.259	0.259	0.253	0.250	0.249	0.245	**0.244**	**0.239**
	pp_{err}		0.151	0.144	0.128	0.103	0.09	0.082	0.078	**0.066**	**0.051**
	pp_{err}	✓	0.089	0.083	0.072	0.052	0.047	0.044	0.045	**0.041**	**0.043**
$DataB$	t_{RMSE}		0.11	0.09	0.09	0.08	0.06	0.04	0.04	**0.03**	**0.03**
	m_{RMSE}	✓	0.256	0.251	0.243	0.242	0.239	0.240	0.238	**0.231**	**0.228**
	pp_{err}		0.139	0.131	0.126	0.084	0.07	0.051	**0.043**	0.04	**0.042**
	pp_{err}	✓	0.101	0.087	0.072	0.061	0.05	0.042	0.041	**0.04**	**0.037**
$DataC$	t_{RMSE}		0.11	0.12	0.09	0.08	0.06	0.05	**0.04**	**0.04**	**0.04**
	m_{RMSE}	✓	0.256	0.248	0.236	0.238	0.234	0.236	0.235	**0.231**	**0.225**
	pp_{err}		0.141	0.133	0.111	0.075	0.061	0.049	0.044	**0.039**	**0.040**
	pp_{err}	✓	0.102	0.083	0.052	0.052	0.044	0.041	**0.035**	**0.035**	**0.033**

Table III - Evaluation metrics for the three different datasets $DataA$, $DataB$, $DataC$. The results are sorted according to the neighborhood region size, in an increasing order. For each row, we highlight in bold the results obtained with the full map and with the best neighborhood region size.

A qualitative example of estimated trajectories with different path lengths is reported in Figure 9(left). It should be noticed that an image rotation is required to overlay the satellite images with the map and trajectory estimated by the proposed SLAM architecture. Indeed, this rotation can be explained by the fact that satellite images are expressed in terms of the magnetic north pole, while GPS measurements are expressed in terms of the geographic north pole.

Another interesting outcome concerns the overall accuracy of the proposed architecture on all the collected datasets. It is in fact possible to observe in Table III how the error metrics have similar values even among datasets collected in different seasons, proving the robustness of the proposed approach under different working conditions. We recall, indeed, that a hazelnut orchard usually shows drastic appearance variations between the harvesting and the vegetation rest period, such as the loss of the foliage and the presence of weed and external grass vegetation. An example of such a drastic appearance variation is depicted in Figure 8.

A further, noteworthy, result is the evolution of the performance degradation according to the path length used in the neighborhood selection. The smaller the k-hop neighborhood the larger the error metrics. It is possible to observe the same trend across almost all the metrics and the collected datasets. However, the performance degradation rate is small and, even with a k-hop of 1, no drift phenomena appear. In addition, the performance degradation effect is rather negligible for a k-hop neighborhood with $k \geq 8$. As for $k \geq 8$, the error metric is almost equal to the full map setup. This also involves that having $k \geq 8$ does not improve the performance of the proposed SLAM architecture. Moreover, in some specific conditions, such as in $DataC$, the planting pattern estimation error pp_{err} obtained without a partitioned map update results being even smaller than the full map setup. This result may seem counter-intuitive for the reader, as theoretically by letting k to tend to infinity the error of the partitioned estimation tends to converge to the error of the full estimation. However, due to heterogeneity in the plant arrangement across the field (*i.e.*, usually trees are not strictly arranged according to the planting pattern) a local estimation of the planting pattern may be more accurate.

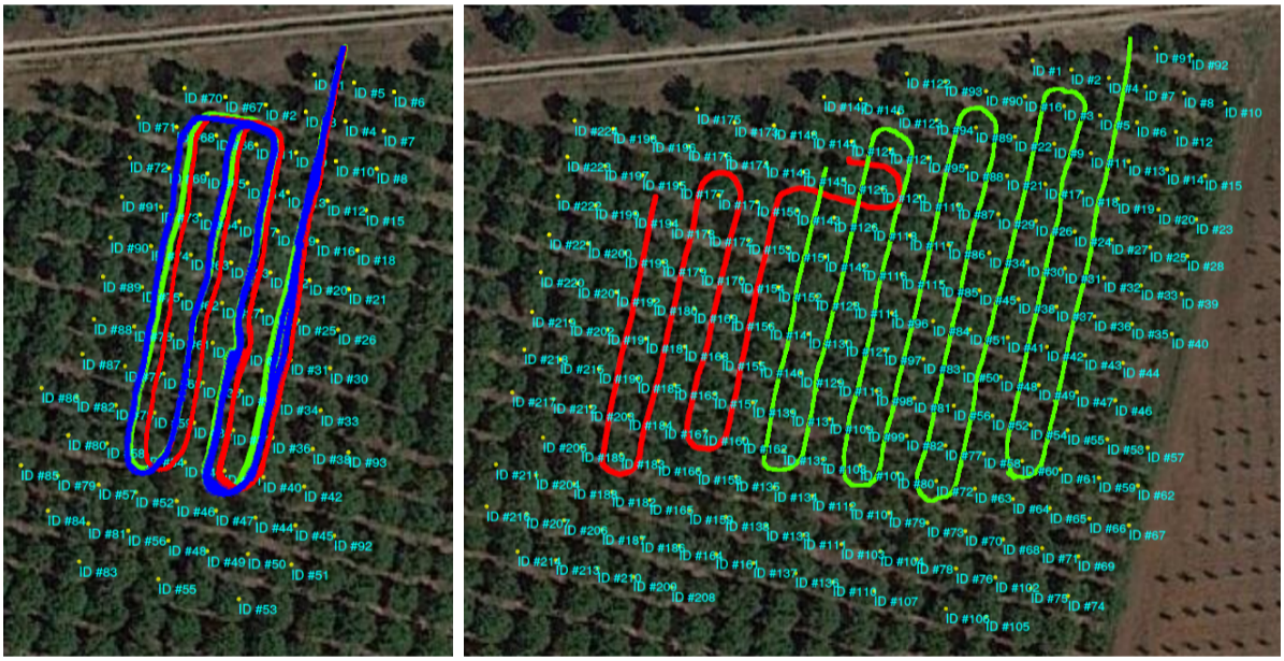

Figure 9 - The left image depicts the estimated trajectory in DataA with different path lengths where the green, blue and red lines represent the RTK-GPS reference and the trajectories estimated with k-hop neighborhood of 5 and 2, respectively. The right image depicts the estimated trajectories in $DataC$. Specifically, the green line represents the estimated trajectory starting with an unknown map of the target orchard, while the red line presents the estimated trajectory starting within the former map and moving in a previously unseen part of the field. In both images, the yellow points represent the estimated landmarks. Finally, the plots are overlaid on geo-referenced satellite images.

A last remark is about the pp_{err}. Differently from t_{RMSE} and m_{RMSE}, this error metrics has been tested with and without the RTK-GPS update step. We recall that a proper estimation of the planting pattern allows for a reliable outlier detection, such as detecting obstacles that do not belong to the orchard. It is possible to observe in Table III how the pp_{err} remains almost unchanged for a k-hop neighborhood with $k \geq 8$, even without the exploitation of the GPS sensor. These results are a further demonstration of the not strict dependence of the proposed EKF from this sensor, and of the reliability of the proposed architecture in estimating the planting pattern.

A further, qualitative, evaluation is reported in Figure 9(right) which shows two estimated trajectories, together with the map built. The whole experiment has been done by using $DataC$. The objective is to show the performance of the proposed SLAM architecture in expanding a map by navigating in a previously unseen portion of the orchard. The green line represents the trajectory followed by the robot when building the initial map, while the red line starts in the map and moves west, adding the new landmarks into the starting map. It is possible to observe how almost all the landmarks are correctly placed on their respective tree centres. The few exceptions are represented by those trees which lie on the boundaries of the map (*e.g.* the trees with IDs $(67, 62, 196, 198, 217)$). Indeed, due to their peripheral position, these trees have been perceived a few times and just by a single side and therefore the SLAM does not have accurate knowledge about their location in the field. Finally, a qualitative evaluation of the obstacle detection capabilities is reported in the attached video.

1.4.8 Computational Complexity Analysis

In this section, we evaluate the real-time capabilities of the proposed SLAM architecture. To do so, we record the time effort required to perform the update step according to the number of landmarks in the map and to the path, the length used to retrieve the k-hop neighborhood. Specifically, we refer to the experiments

carried out with $DataB$ in Table III. The results are reported in Figure 10, where we also plot the time threshold to guarantee real-time performance. The results have been obtained on a laptop with an Intel i7-8750H CPU. The real-time threshold has been computed by considering the LiDAR acquisition rate, which is 10 Hz. Ideally, having the update step computational burden constantly below this threshold means that no LiDAR measurement is lost, and the SLAM approach works at a full rate.

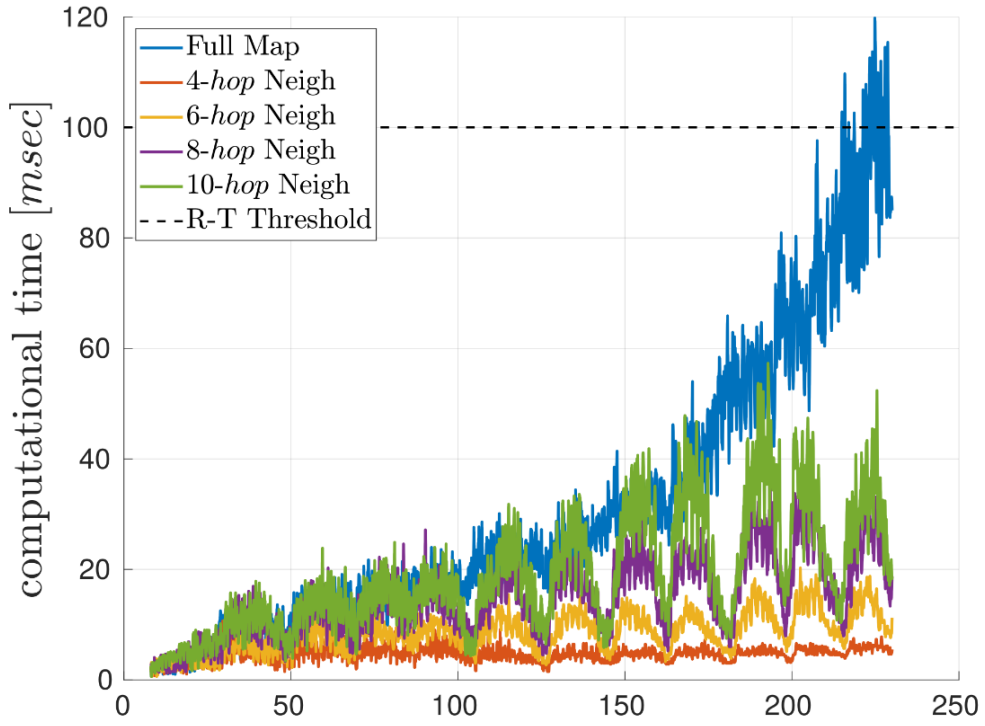

Figure 10 - Plot of the computational time required to perform an update step according to the neighborhood size and to the number of trees in the map. The dashed line represents the real-time threshold which has been computed by considering the LiDAR acquisition frequency of 10 Hz.

As one would expect, the time effort for the full map update grows exponentially, exceeding the threshold once the map contains almost 200 landmarks. Differently, the time effort in the cases of the partitioned map update step is constantly lower than $60\ ms$. This proves the real-time capabilities of the proposed approach since, how we explain in the previous section, having a k-hop neighborhood with $k \geq 8$ does not improve the accuracy in the trajectory and map estimations. Another interesting perspective to analyze the results reported in Figure 10 is by considering the number of trees considered in the update step. The number of trees depends on the k-hop neighbourhood size which, in case of one observed landmark, can be easily obtained as $4(k(k+1)/2)$. The computation of the exact number of trees in case of multiple landmarks is complex and, in general, it is bigger or equal to the case of one observed landmark. Given that, carrying out an update step in the EKF with a k-hop neighborhood length of 10 involves more than 220 trees. Despite this, the proposed SLAM architecture can successfully keep the computational burden well below the real-time threshold. We recall that the extraction and the update of the $\boldsymbol{P}_{xy}$ partition of the full covariance matrix $\boldsymbol{P}$ grows linearly with the field size. We reiterate that, as pointed out in Section 1.4.1, no actual matrix inversion is required being the coordination transformation matrix, a permutation matrix for which the inverse matrix is equal to the transpose matrix.

Other interesting results concern the computational time trend in the case of partitioned update steps. As shown in Figure 10, the time effort follows a sinusoidal-like trend. This can be explained by the trajectory performed by the robot in the orchard. Indeed, when the robot reaches the start or the end of its s-shaped trajectory, namely, before to turn in order to change row, the partitioned map contains fewer landmarks with respect to the case when the robot is in the center of the orchard. This leads to variables size of the partitioned covariance matrix in the update step and, therefore, to a variable update step computational burden.

1.5 Global Planning for Monitoring and Agronomic Interventions

Performing Precision Farming activities requires interventions on individual trees, such as targeted spraying or data collection, or on the entire field, such as pest and disease monitoring. According to the nature of the activity and the needs of the plants, a variable effort may be required for each robotic platform in order to perform a certain task at a given lo-cation. Therefore, it becomes mandatory to design a global planning strategy to optimally move the robotic platforms within the orchard especially when operating in large-scale scenarios. We Developed a global planning strategy specifically designed for targeted activities in orchard settings. The problem is formulated as a novel Multi-Platform Steiner Travelling Salesman Problem (MP-STSP). To guarantee the exploitation of multiple moving platforms and the minimization of the overall operational time, the proposed formulation explicitly considers the time necessary for each task to be performed. By doing so, the computed itineraries attempt to balance the workload among the deployed platforms. More-over, the MP-STSP formulation is general, does not depend on the number of platforms, and well fits with different on-field activities, ranging from full field inspection to targeted per-plant interventions. We report comprehensive simulation results to numerically demonstrate that the proposed planning strategy can be effectively employed to carry out selective precision farming activities within a large-scale orchard. The proposed planning architecture has been submitted and is currently under review at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), an annual flagship conference of Robotics & Automation Society (RAS).

1.5.1 MP-STSP

In this section, we provide the mathematical formulation of the proposed Multi-Platform Steiner Travelling Salesman Problem MP-STSP for Precision Farming. In particular, the goal is to find an optimal path inside the orchard such that all the locations on which a per-plant agronomic intervention has to be performed are visited while the overall operational time is minimized. To achieve this, we propose a graph-based formulation of the MP-STSP by resorting to the graph modeling introduced in Section 1.1. We also assume to deploy multiple autonomous platforms which navigate at a constant velocity v.

We denote with $V_r \subseteq V$ the subset of vertexes encoding the locations that must be necessarily visited according to the agronomic task and with $n_V = |V_r|$ its cardinality. Note that, the subset V_r does not include the depot, which we assume, without loss of generality, to be the vertex v_0 for all the robots. Using the same notation as before, we will refer to $n_R = |R|$ as the number of robots, with $R = \{r_1 \dots r_m\}$ its set. Furthermore, we introduce the binary decision variables x_{kj} for each edge $e_k \in E$ defined as

$$x_{kj} = \begin{cases} 1 & if \quad robot \quad \mathrm{r_j} \in \mathrm{R} \quad \text{passes on edge} \quad \mathrm{e_k} \in \mathrm{E} \\ 0 & otherwise \end{cases} \tag{18}$$

In addition, we introduce a binary variable s_{ij} to describe if a robot r_j stops at the required vertex v_i to perform an agronomic task, which is defined as

$$s_{ij} = \begin{cases} 1 & if \quad robot \quad \mathrm{r}_j \in R \quad \text{stops in} \quad v_i \in V_r \\ 0 & otherwise \end{cases} \tag{19}$$

We are now ready to provide the proposed Multi-Platform Steiner Travelling Salesman Problem MP-STSP formulation. Intuitively, the MP-STSP can be thought as an extension for multiple platforms of the STSP over a directed graph $\mathcal{G}$. The reader is referred to [19] for a comprehensive overview of standard STSP formulations. In particular, our formulation follows the approach proposed in [20], where the standard STSP formulation is augmented with a Single-Commodity Flow (SCF) by adding the commodity variables f_a, which we will refer to as f_{kj} in order to take into account the availability of multiple robots. In particular, the following Integer Linear Programming (ILP) formulation can be used to describe the proposed MP-STSP.

Formulation MP-STSP

$$minimize\ T_{max} \tag{20a}$$

Subject to

$$\sum_{r_j \in R} \sum_{e_k \in \delta^+(v_i)} x_{kj} \geq 1 \forall\ v_i \in V_r \tag{20b}$$

$$\sum_{r_j \in R} \sum_{e_k \in \delta^+(v_0)} x_{kj} = n_R \tag{20c}$$

$$\sum_{e_k \in \delta^+(v_i)} x_{kj} = \sum_{e_k \in \delta^-(v_i)} x_{kj}, \quad \forall\ v_i \in V,\ \forall\ r_j \in R \tag{20d}$$

$$\sum_{e_k \in \delta^-(v_i)} f_{kj} - \sum_{e_k \in \delta^+(v_i)} f_{kj} = s_{ij}, \quad \forall\ v_i \in V_r,\ \forall\ r_j \in R \tag{20e}$$

$$\sum_{e_k \in \delta^-(v_i)} f_{kj} = \sum_{e_k \in \delta^+(v_i)} f_{kj}, \quad \forall\ v_i \in V \setminus \{V_r\},\ \forall\ r_j \in R \tag{20f}$$

$$\sum_{e_k \in E} t_{e_k} x_{kj} + \sum_{v_i \in V_r} c_i s_{ij} \leq T_{max}, \quad \forall\ r_j \in R \tag{20g}$$

$$\sum_{r_j \in R} s_{ir} = 1, \quad \forall\ v_i \in V_r \tag{20h}$$

$$s_{ij} \in \{0, 1\}, \quad \forall\ v_i \in V_r,\ \forall\ r_j \in R \tag{20i}$$

$$x_{kj} \in \{0, 1\}, \quad \forall\ e_k \in E,\ \forall\ r_j \in R \tag{20j}$$

$$0 \leq f_{kj} \leq (n_V)\, x_{kj}, \quad \forall\ e_k \in E,\ \forall\ r_j \in R \tag{20k}$$

The optimization problem given in the above formulation can be described as follows. The objective function Equation (20a) encodes the overall time spent by each single robotic platform to perform the assigned tasks at each of the required locations. It is important to highlight that, since the robots are expected to work in parallel, T_{max} represents the overall operational time required by the robots to accomplish the required tasks. We model this by formulating T_{max} as a shared upper bound among the operational time of each deployed robots in Equation (20g). Specifically, we define the operational time of the individual robot as the

sum of the edge traversing time and the time required for each agronomic intervention, which are modeled as $t_{e_k} e_k$ and $c_i s_{ij}$, respectively.

The edge traversing cost d_{e_k} has been embedded into $t\_\{e\_k\} = d\_\{e\_k\}/v$ while with $c_i \in N$ we model the operational time required to perform the i^{th} task. By doing so, we can model different kind of tasks by simply referring to the time required to perform them. In this way, the minimization of T_{max} in the cost function (20a) ensures that the workload is shared among all the robotic platforms, while minimizing the overall operational time.

Another important advantage of the proposed formulation, in contrast to standard mTSP approaches, is that we do not directly constraint the minimum and maximum number of required nodes to be visited by each vehicle. Indeed, the required stops are optimally assigned to the deployed platforms by the minimization of the cost function. As a matter of fact, the resulting assignment will depend on the disposition of V_r inside the orchard and on the time required to perform the agronomic tasks.

Equation (20b) is an extension of the classical Single-Commodity Flow (SCF) formulation [19] for a Multi-Platform framework, where it is necessary to ensure that each robot departs and ends at the depot (see Equation 20c)). The core of this formulation relies on Equation (20e), which now assigns the commodities among all the robots. Given the conditions (20h)-(20i), in fact, all the required vertexes V_r have to be visited, but with commodities that are modified exclusively by a single robot. In particular, if a robot leaves a commodity for a given required vertex, the other platforms are still enabled to pass through the same node, but they will treat it as a non-required vertex, preserving their flow.

Lastly, Equation (20k) activates an edge in the final tour, if any commodity passes through it. This property has now to be ensured for each robot and has been kept as in the classical formulation [19]. Notice that, the upper-bound given in Equation (20k) cannot be made any tighter given the fact that there is no prior knowledge on the distribution of the selected nodes V_r among the robots, as the partitioning of the vertexes to be visited among the robots is part of the optimization itself.

We also highlight how this formulation is versatile, as it enables to compute itineraries for different purposes. Indeed, it allows to describe a selective intervention by properly defining the set of vertexes V_r to be visited, ranging from a small subset of trees to the entire orchard, and it also allows to describe heterogeneous agronomic activities by properly defining their costs c_i encoding the related operational time.

Moreover, it should be noticed that for any configuration of the set of vertexes $V_r \subseteq V$ to be visited, the proposed formulation always finds a feasible solution. On the contrary, this cannot be guaranteed in the mTSP or VRP formulations as passing through the same edge x_{kj} more than once is not allowed. Notably, given the fact that for an orchard, due to their typical planting pattern, the graph encoding the locations from which agronomic interventions must be carried out is usually a grid, no solution can be found for

mTSP and VRP formulations with $n_R > 1$ if the depot is considered as part of the grid graph boundaries. In this regard, one should impose the depot to be external to the graph and connected to different vertexes of the grid. However, in doing so the entries and exits of single robots could be scattered around the grid according to the optimal route. More detail concerning this limitation will be described in the next section, where the proposed approach is compared against a classic VRP. Notably, also in the case of the single platform formulation, such as TSP, the shortest Hamiltonian Path [21] on any $m \times n$ grid with m and n odd, $m \geq 3$, has a lower bound of $mn - 1 + \sqrt{2}$. This would require crossing the orchard in diagonal, which is forbidden in the context of our application scenario. Another commonly used approach for sharing the workload among several platforms is to partition the solution space with some clustering algorithms such as

k-$means$ [22] in order to assign a competence area to each robot. However, as it will be described in the next Section, clustering the required nodes into $k = n_R$ regions via k-$means$ and running the classic STSP algorithm for every robot results into a non-optimal solution for our application. This can be explained by the fact that clustering algorithms, differently from the proposed approach, normally rely solely on Euclidean distances, and do not consider the required time to perform the tasks at the corresponding location in the field. As a matter of fact, this could result into an unbalanced workload distribution among the robots, dependently on how the V_r are located within the orchard.

1.5.2 Numerical Validation

Simulation results are provided to numerically demonstrate the effectiveness of the proposed global planning strategy for targeted per-plant agronomic interventions in large-scale orchards, given in the previous section. We focus on a simulated hazelnut orchard farming scenario and we consider as a case-study an agronomic monitoring task since: (i) it represents a crucial operation in precision farming scenarios for an effective field management, and (ii) the time needed to collect data at a specific location inside the orchard depends on how many plants must be scanned from that specific location (see Figure 12). Specifically, the latter motivation allows defining tasks with different time effort according to their location. Simulations have been carried out by resorting to the Robot Operating System (ROS) framework. We used the GAZEBO robotic simulator for modeling both the orchard environment and the SHERPA HL R-A, both depicted in Figure 11. Specifically, we want to resemble a real data collection campaign where we assume that all the robots are required to perform some tree scanning processes. Furthermore, given the grid-graph encoding the scanning locations within the orchard, it should be noticed that in a single location up to 4 different plants can be scanned, see Figure 12. We are thus considering for this example $c_i = c \cdot k, k \in \{1,2,3,4\}$ with c representing the time required for one full scan procedure and the non-homogeneity of scan times in the map.

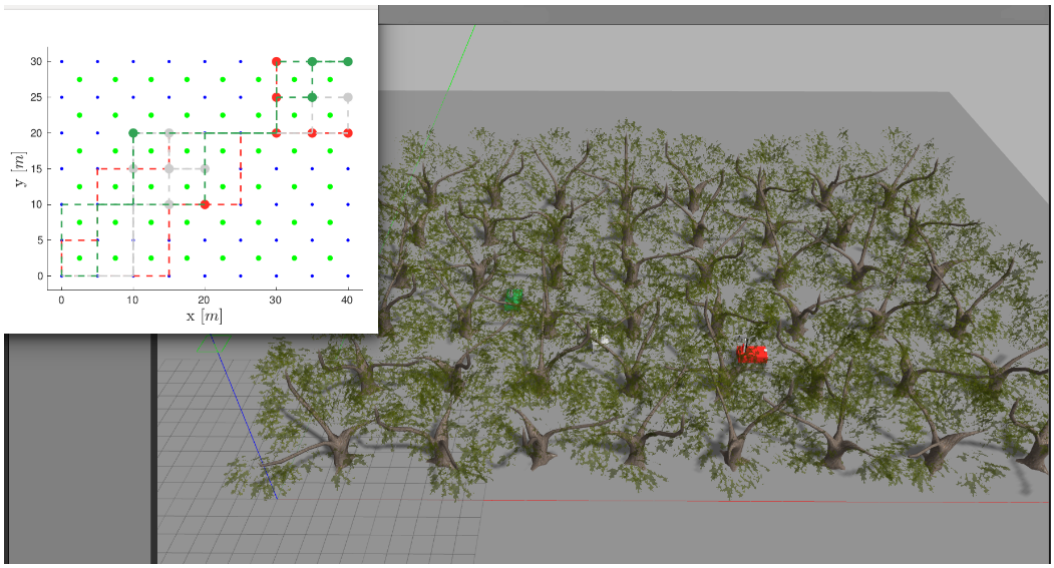

Figure 11 - An illustration of the simulated SHERPA robotic platforms while following the planned path. On the top left, an illustration of the optimal itineraries to follow. Specifically, the dotted lines and the points represent the optimal paths and the nodes to visit and are associated to the respective robots according to their colors.

Figure 12 - Examples of scanning policies in a small patch of the target field. From left to right, the 4 scans that are necessary to 3D reconstruct the tree and the possibility on scan 4 different plants at the same location inside the orchard, respectively.

n_R	VRP		$k-means$		MP-STSP	
	T_{max}	T_{avg}	T_{max}	T_{avg}	T_{max}	T_{avg}
2	875	632.5	790	595	**620**	620
3	460	420	790	416.6	**460**	460
4	550	313.8	450	357.5	**380**	370
5	545	250	410	298	**320**	316

Table IV - Total mission time according to the number of deployed robots and of the resolution methods. For each setup, we highlight in bold the best T_{max}.

The ILP formulation given in the previous section has been written in MATLAB and interfaced with the IBM CPLEX version 12.9 solver. The computation runs entirely offline and takes as an input: i) the graph encoding all the possible locations on which an agronomic intervention may take place; ii) the (sub)set of vertexes describing the locations that must be necessarily visited; and iii) the number of scans c_i to be performed for each location.

The output consists of the optimal way-points sequences as well as the total time cost for each robotic platform. These sequences are then sent to a ROS controller node in order to generate motion control commands for the simulated Ackermann vehicles. For a comprehensive overview of the navigation architecture for Ackermann vehicles that we developed within the H2020 PANTHEON project the reader is referred to [1]. The field has been selected to be composed by $6 \backslash times\ 8$ trees, resulting into a total of $(n_{rows}+1)\cdot(n_{cols}+1)=63$ possible scanning locations, with a planting pattern of $5\times 5\ [m]$ to resemble a typical hazelnut orchard within the experimental site available for the H2020 Project PANTHEON. The required locations and the number of scans for each node have been set to

$$V\_r\ =\ [22\&, 23, 30, 31, 32, 39, 40, 43, 44, 45, 52, 53, 54, 61, 62, 63]$$

$$c = 40\cdot[1,1,1,2,1,1,1,1,2,1,2,4,2,1,2,1]$$

Moreover, since we have chosen a squared planting pattern, the edge traversing time is constant and equal to $t_{e_k}=5\ [sec]$, assuming a constant velocity $v=1\ [m/sec]$ for the robots. We also remark that, in the MP-STSP formulation the starting point is an arbitrary vertex outside the required ones set V_r. For the simulation we assumed the node v_0 or depot to be, without loss of generality, the bottom-left corner of the grid. We selected the scanning locations forming the set of nodes V_r to represent a possible capture campaign for 6

different hazelnut plants. Notice however that out of $|V_R| = 16$ single stops, 5 of them have to perform a double scan (vertexes $\{31,44,52,54,62\}$) and 1 of them (vertex 53) has to perform a quadruple scan.

To highlight the benefits of the proposed approach, we also make a comparison with other approaches. We remark that, despite planning has been widely investigated in robotics, to the best of our knowledge there is no other algorithm that covers the same functionalities as the one proposed in this work. Therefore, to carry out a realistic comparison, we adapted the closest state-of-the-art algorithms, namely a STSP [20] and a VRP [23]. Particularly, to enable the former to handle Multi-Platform problems we exploit a $k\text{-}means$ heuristic to spatially split the V_r nodes to visit into n_R clusters. Each cluster is then handled as an independent STSP problem.

Table IV reports the maximum and the average times, T_{max} and T_{avg} respectively, for different mission setups based on the number of deployed robot and on the resolution methods. In particular, it is possible to appreciate how the proposed formulation allows a better spreading of the worktime by leading to the lowest T_{max} in all the explored configurations. Another interesting outcome derives from the average time T_{avg}. It can be thought as a direct measure of the workload unbalancement. Indeed, the more different T_{max} and T_{avg} are, the more unbalanced the operational times will be. In this regard, the proposed formulation report almost the same maximal and average times in all the explored configurations.

Notably, the same does not hold for the VRP and $k\text{-}means$ setups since they do not take explicitly into account the operational time. Their workload, indeed, is unbalanced, even reporting cases where one of the two statistics is twice the other. Examples of optimal itineraries computed by MP-STSP and the two comparing approaches for $n_R = 3$ are depicted in Figure 13. In particular, the itineraries computed by our method leads the 3 platforms to visit $\{6,6,4\}$ vertexes, respectively. Despite this, the workload is balanced since the third robot has to visit the node 53 in which it has to scan the four neighboring plants, thus requiring the highest operational time. Differently, the $k\text{-}means$ heuristic balances the itineraries on $\{9,4,3\}$, leading to an unbalanced workload. As it is possible to notice in Figure 13, this is mainly due to the first robot which has to visit 9 capturing locations.

To have a coherent comparison, in the VRP simulations reported in Figure 13, the times of Table IV do not consider edges coming/going between the depot and the left-bottom boundaries. One could say that for the carried-out simulations all the left-bottom boundaries are considered as to be the real depot. To accomplish this, the "virtual" depot denoted as 0 has been connected to all the left and bottom boundary nodes with no edge cost. Thus, displayed entries and exits of each single robot have not been forced but they are the result from the optimization. One could think to cast the VRP into a Capacitated (C)VRP instance by assign to each vertex a different demand thus modeling the different scan times. This would indeed improve the optimal path generated for each robot. However, in real settings, this formulation does not reflect our needs since we are not concerned about robots capacities and moreover it suffers from the same problems that are present in the VRP formulation. The last remark is about the possible collisions among the robotic platforms. Collision avoidance for the VRP comes for free, since no robot passes through the same path of the others. The same does not hold for the MP-STSP formulation where multiple robots can pass multiple times on the same edge. However, the locations where the robots stop to run a scanning process are mutually exclusive. Thus, the robots never stop in the same place. On the other hand, to avoid crashes during the routes, we relied on a low-lever planner specifically designed for orchard operations which can handle the presence of dynamic obstacles within the field, see [1] for further details.

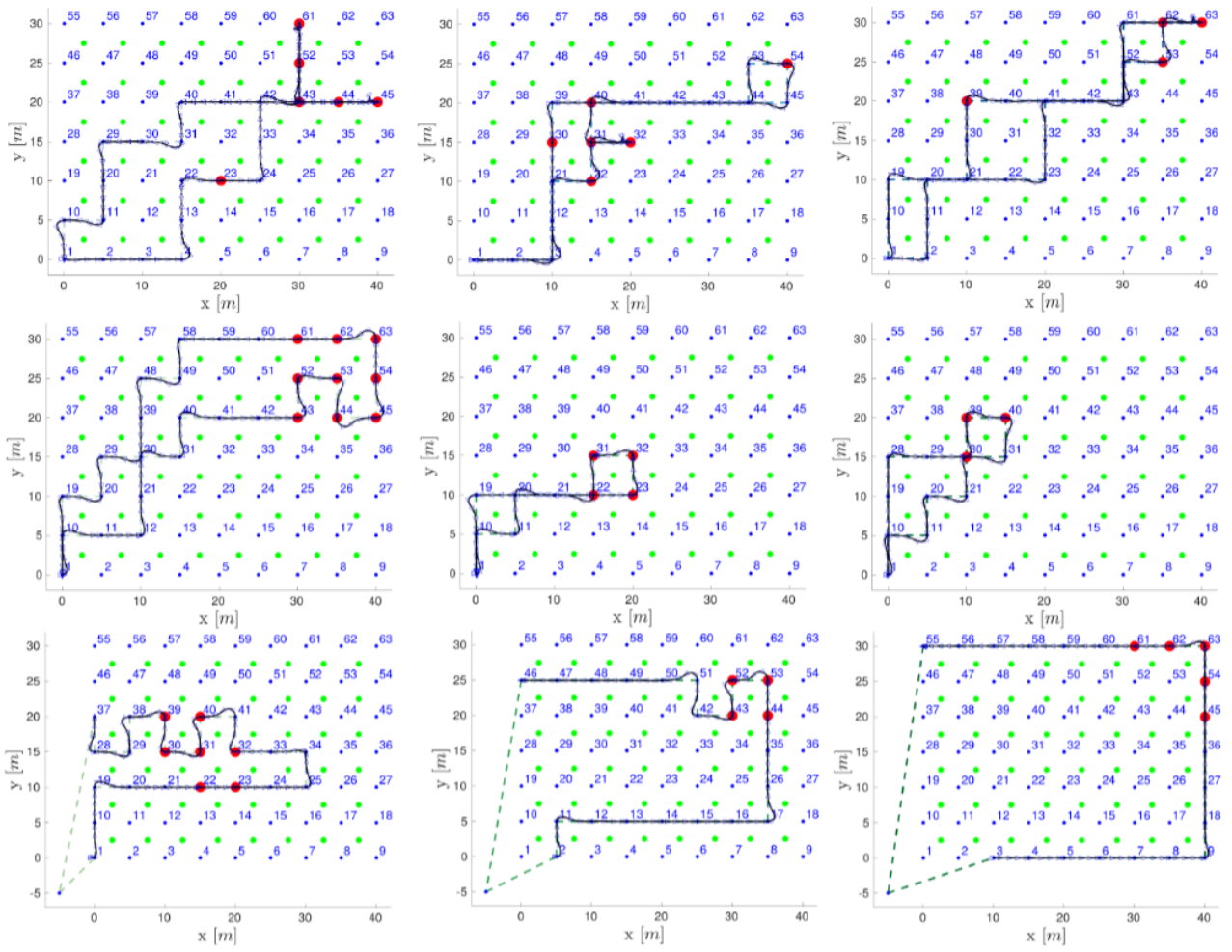

Figure 13 - Optimal solutions resulting from three different approaches with $n_R = 3$. From top to bottom the rows represent results from MP-STSP, k-means and VRP, respectively. The k-means heuristic is obtained by running 3 instances of original STSP algorithm. On the third row, for the sake of VRP needs, we added an external depot shared by the 3.

2 Unmanned Aerial Vehicle

The aerial platform used in the project Pantheon, the drone DJI M600 Pro, has been mechanically modified to host 3 sensors: the Tetracam MCAW 6, the Sony a6100, and a Teax Thermal Capture 2.0. After this customization and the setup of state-of-the-art sensors for localization (D-RTK GPS), the system has already demonstrated to be able to perform the required remote sensing activities.

As a result, the main theoretical research carried out on this platform has been focused on the development of more efficient path planning strategies. This line of research aims at reducing the flying time required to cover a given extension of field and therefore to increase the size of the area covered by flight. The outcome of this research is a journal paper that is currently under submission to the IEEE Transactions on Cybernetics (IEEE TCyb).

2.1 Path planning problem

It is commonly claimed that today the main factors limiting a wide adoption of drones are social and legal issues. However, when working on the field to collect data, one discovers that they are not the only limiting aspect. Indeed, a major aspect that limits the potential effectiveness of drones (and thus their rentability, and ultimately their adoption) is the lack of systematic ways to plan missions so as to maximize the amount of information collected.

In many applications, remote sensing best practices use post-processed information in the form of an orthomosaic [24]. An orthomosaic is essentially a geometrically corrected image obtained thanks to the composition of several overlapped photographs [25]. This technique implies an exhaustive and complete coverage of the area, commonly using boustrophedon patterns as the one shown in Figure 14, to ensure a proper reconstruction of all the elements. In the case of precision farming or other monitoring domains, the creation of a complete orthomosaic can be extremely time consuming and might require several flights to cover relatively small areas, thus limiting the real-world applicability. This situation is encountered in the project PANTHEON, where the farming area exceeds hundreds of hectares.

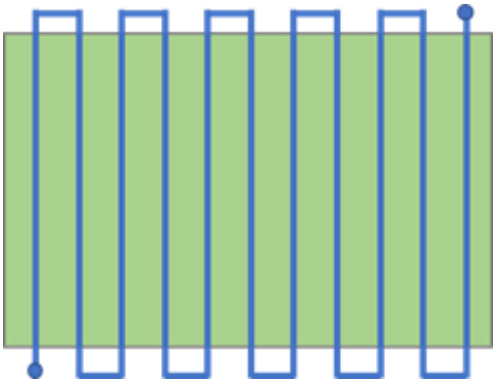

Figure 14 - Common boustrophedon path planning.

Given this common limitation in time and resources, the literature has focused on the definition of optimal policies that partially cover the area of interest. In this regard, many persistent monitoring works rely on graph-based strategies where the latency in between visits to every region is minimized [26]–[28]. However, these strategies consider a static and node-independent distribution of the phenomena, which makes them non suitable for physical systems with significant dynamics.

Alternative approaches base their policies on the spatial correlation between the measurements. In these works, the mission path is computed such it avoids redundant data and thus gathers the maximum amount of information per flight. In [29], UAVs equipped with omnidirectional sensors perform an information-based exploration where the goal is to minimize the time to obtain a predefined measure of data. In environment monitoring, the monitored strategy is defined such that the measurement uncertainty of a Gaussian Process (GP) regression is minimized [30]. Also, in the field of autonomous underwater vehicles (AUVs), multiple AUVs are used to perform the sampling of a scalar field based on the information obtained [31]. These works optimize the path regarding the spatial distribution of the possible measurements. Yet they tend to fail in the evaluation of the temporal correlation with previous data information, which is a meaningful aspect in most persistent monitoring activities.

In PANTHEON, we assumed (realistically) that the phenomena to be monitored have dynamics and statistical properties which are sufficiently well known and that can be exploited in the UAV path planning. In particular we e proposed a path planning strategy where the area of interest is only partially covered, and the remaining elements are estimated based on the dynamics of the system and the spatial correlation between measurements. This approach resembles to a sensor selection problem [32] [33] where the measurement points are considered as the selection or not of available sensors in the observer formulation.

The literature presents a few examples of path planning for spatio-temporal phenomena monitoring [34]–[36]. Binney *et al.* [34] defines a recursive greedy algorithm to compute waypoints based on a given indicator of the estimation process. As example, given a Gaussian Process, the covariance of the estimation of different areas is minimized during a sensing exploration using AUVs. In Garg *et al.* [35] the authors assume a stochastic

dynamic system and perform a multi-vehicle sampling where each robot moves such that the entropy of a particles filter is maximized. Lan *et al.* [36] models the phenomena as Gaussian processes and defines periodic trajectories to minimize the largest eigenvalue of the covariance of a Kalman Filter.

Here, the path planning policy is obtained as part of the estimation process of the monitored phenomena. This is achieved by structuring it as an Orienteering Problem (OP) [37]. The Orienteering Problem is a combinatorial problem which consists of a node selection where the shortest path in between the selected nodes is determined. Given a time or length constraint, the objective is to maximize the score given by the visited points. UAV remote sensing activities, due to the flight time restriction and the discrete nature of the measurements, are likely to be adapted as an OP. In this context, the set of points represents the possible measurement coordinates, the time interval is adapted to the vehicle autonomy, and the cost function is some measure related to the measurement point.

Related works in path planning already rely on an Orienteering problem framework to define tentative information-based policies for monitoring activities. In [38], the information obtained is maximized using a quadratic utility function to represent the spatial relation between the different measurement points. More recently, Bottarelli *et al.* [39] introduce an Orienteering-based path planning used to optimize a level set estimation.

For PANTHEON, we have developed a method where the measurement points are selected such the accuracy of the level sets classification is maximized. The main contribution with respect to the existing literature is the inclusion of the update step of the Kalman filter estimation as part of the Orienteering Problem. This is done by resorting to a formulation based on the Fisher information matrix. The main advantage of this approach is that the path of the mobile sensor is computed considering the process dynamics, the estimation uncertainty and the existing fixed sensing structure. By doing so, it allows to define the optimal combination of the UAV remote sensing with additional sensing devices by resorting to an observer-based architecture.

The developed strategy provides an offline computation of the optimal sensing points over one step ahead horizon of the estimation process. This approach allows to obtain the optimal path in cases where the coverage is done with unknown periodicity or when the time gap between flights is too large.

In order to solve the stated problem, we propose a Mixed-Integer Semi-Definite Programming (MISDP) formulation where the minimum eigenvalue of the information matrix is maximized. This formulation allows to obtain the optimal solution for small instances of the problem. Additionally, for the case of large-scale scenarios, two heuristics are proposed along with an exhaustive computational analysis.

2.2 Path planning for state estimation

Consider a plane partitioned in N areas and let the linear time invariant system

$$\begin{bmatrix} x_{k+1}^1 \\ \dots \\ x_{k+1}^N \end{bmatrix} = A \begin{bmatrix} x_k^1 \\ \dots \\ x_k^N \end{bmatrix} + Bu_k + w_k \tag{21}$$

describe a dynamic phenomenon that we want to observe, where $x_k^i \in \mathbb{R}^n$ represents the states of the system in the ith area and $u_k \in \mathbb{R}^n$ is the vector of the (measured) inputs to the system. A, B are matrices of consistent dimensions. The system is subject to a process disturbance $w_k \sim N(0, Q)$ modelled as a stochastic Gaussian noise with covariance $Q \in \mathbb{R}^{nN \times nN}$, which is typically non-diagonal and has nonzero terms for variables describing adjacent areas.

To estimate the state of this process, two classes of sensors are assumed available: fixed sensors and mobile sensors. Fixed sensors provide continuous measurements of the states in certain specific areas. Mobile sensors are sensors that allow to collect measurements at certain specific times from certain specific locations.

Let us consider the matrix $C_i \in \mathbb{R}^{M_i \times n}$ as the measurement matrix associated to the measurements of the ith area. This matrix can be defined as

$$C_i = \begin{bmatrix} C_i^f \\ C_i^m \end{bmatrix} \quad \forall i = 1, \cdots, N, \tag{22}$$

where $C_i^f \in \mathbb{R}^{f_i \times n}$ denotes the available fixed measurements of the ith area and $C_i^m \in \mathbb{R}^{m_i \times n}$ is the matrix associated to states that can be measured by the use of mobile sensors.

The combination of the measurement matrices of each area provides a time-invariant observation matrix $C_i^m \in \mathbb{R}^{M \times N}$ in the form of a block-diagonal matrix,

$$C = \begin{bmatrix} C_1 & \cdots & 0_{M_i \times n} \\ \vdots & \ddots & \vdots \\ 0_{M_N \times n} & \cdots & C_N \end{bmatrix}, \tag{23}$$

which defines the information of the system that can be accessed to through the two classes of sensors.

In this context, to represent the partial availability of the mobile sensor measurements, we introduce the binary variable γ_k^i to denote the fact that a certain location i is measured at time k by the mobile sensor (in such a case $\gamma_k^i = 1$) or not (in such a case γ_k^i =0). Based on this binary variable, we define the measurement selection matrix Γ_k^i corresponding to each area as

$$\Gamma_k^i = \begin{bmatrix} I_{p_i \times p_i} & 0_{p_i \times m_i} \\ 0_{m_{i,k} \times p_i} & \Lambda_{\mathrm{k}}^i \end{bmatrix} \tag{24}$$

where the first p_i rows refer to the static sensor distribution, constant over time, and $\Lambda_{\mathrm{k}}^i \in \mathbb{R}^{m_{i,k} \times m_i}$ is a row selection matrix associated to the measurements of the mobile sensor at each time k, computed as

$$\Lambda_{\mathrm{k}}^i = \left[e_j^T\right]_{j|\gamma_{i,k}=1} \quad \forall j = 1, \ldots, M_i, \tag{25}$$

where e_j denotes the j-th vector of the standard basis $\mathbb{R}^{M_i}$. Note that this selection matrix is the null matrix $\Lambda_{\mathrm{k}}^i \in \mathbb{R}^{0 \times m_i}$when the area is not measured (γ_k^i =0) and it is the diagonal matrix $\Lambda_{\mathrm{k}}^i \in \mathbb{R}^{m_i \times m_i}$ when it is visited by the mobile sensor.

Given this, the selection of the available measurements at time k is provided by the matrix $\Gamma_{\mathrm{k}} \in \mathbb{R}^{m_i \times m_i}$, which computed as

$$\Gamma_{\mathrm{k}} = \Gamma_{\mathrm{k}}^1 \oplus \Gamma_{\mathrm{k}}^2 \oplus \ldots \oplus \Gamma_{\mathrm{k}}^{\mathrm{N}} \tag{26}$$

where $\oplus$ denotes the direct sum operator. The measurement equation of the entire system is

$$y_k = \Gamma_{\mathrm{k}}(C x_k + v_k) \tag{27}$$

where $y_k \in \mathbb{R}^{M_k}$ is the set of measurements available to the system at time k. The measurement process is also subject to stochastic Gaussian noise $v_k \sim N(0, R)$ with covariance $R \in \mathbb{R}^M$.

In the definition of the problem, the following assumptions are considered:

- The mobile data collection is performed by a UAV with a maximum autonomy $T_{max} > 0$.
- The dynamics of the monitored system are considered to be slower than the process of collecting the data by the mobile sensors.
- The measurements performed by the UAV are always located at the centroid of the area i.

The main goal of this work is to compute the optimal path to be followed by the UAV at each data collection mission, such that the information about the states of the system (21) is maximized given the limited autonomy of the vehicle.

2.3 Path planning for state estimation

Given the fact that the system (21) is a linear system subject to Gaussian noise, a coherent choice is to estimate the states of the system by using a Kalman filter. Due to the occasional availability of the mobile sensing, the state variables are estimated by using a Kalman filter with intermittent observations (see [40], [41]) whose prediction step is

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_k \tag{28}$$

$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q, \tag{29}$$

and the correction step is

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K(y_k - C_k\hat{x}_{k|k-1}) \tag{10}$$

$$K_k = P_{k|k}C_k^T\left(C_kP_{k|k}C_k^T + R_k\right)^{-1} \tag{11}$$

$$P_{k|k} = P_{k|k-1} + K_kC_kP_{k|k-1}, \tag{12}$$

where $\hat{x}_{k|k}$ is the estimated value of the state at time k given the information available at that time and $P_{k|k}$ is the error covariance matrix of the estimation. Note that, for the sake of simplicity, the time-variant matrices C_k and R_k denote the products ΓC and ΓR respectively.

The amount of information regarding the state estimation process is given by the error covariance matrix of the state estimate $P_{k|k}$. This matrix provides an indicator of the performance of the observer after each measurement step. This indicator can be used to evaluate the performance of the monitoring strategy and thus the performance of the path planning.

Combining equations (11) and (12) and applying the matrix inversion lemma, the covariance of the estimation can be expressed as

$$P_{k|k} = \left[P_{k|k-1}^{-1} + C_k^TR_k^{-1}C_k\right]^{-1}. \tag{13}$$

In order to provide a more tractable expression of this measure of performance, let us consider the Fisher Information matrix Y_k. This matrix indicates the quantity of information associated to each variable, and for

the case of a linear system, is equivalent to $Y_{k|k} = P_{k|k}^{-1}$ [42]. Given this property, the post-information matrix of the estimator can be obtained as

$$Y_{k|k} = P_{k|k-1}^{-1} + C_k^T R_k^{-1} C_k, \tag{14}$$

providing an expression where the performance of the estimation is given by the linear combination of the previous information and the sensing structure of the system.

It is worth to notice that, thanks to the statistical properties of the Linear Kalman Filter, the information matrix defined in (14) depends only on the information at time $k-1$ and the sensing structure to be used at time k. This allows to compute this matrix, and therefore a suitable performance indicator, beforehand, given the sensing configuration. Interestingly, this fact can be used to evaluate and select the optimal sensing path of the mobile sensors as an optimization problem without the need of an online implementation.

Note that the matrix $R_k \in \mathbb{R}^{M_k \times M_k}$ is assumed to be a diagonal matrix. Therefore, the Fisher information matrix (14) can be expressed as

$$Y_{k|k} = P_{k|k-1}^{-1} + \sum_{i=1}^{M_k} \frac{C_{k,i}^T C_{k,i}}{r_{k,i}}, \tag{15}$$

where $r_{k,i}$ is the i-th diagonal entry of the matrix R_k and $C_{k,i}$ is the observation matrix when only the measurement of the ith entry is available. Considering the two sources of measurements, from fixed and mobile sensors, we can derive the following expression

$$Y_{k|k} = P_{k|k-1}^{-1} + \sum_{i=1}^{N} C_{f,i}^T R_{f,i}^{-1} C_{f,i} + \sum_{i=1}^{N} \gamma_{k,i} (C_{m,i}^T R_{m,i}^{-1} C_{m,i}), \tag{16}$$

where $C_{f,i}$ represents the observation matrix C whose only non-zeros entries belong to the fixed measurements of the area i and, similarly, $C_{M,i}$ denotes the matrix C with entries associated to the UAV measurements in the i-th area. In this way, the information matrix is defined as a function of the binary variable $\gamma_{k,i}$ and therefore, the areas visited by the UAV.

A common approach to evaluate the performance of the state estimation is to compute the trace of the covariance matrix. However, in the case of the Fisher information matrix, as stated in [43], its trace does not distinguish the gains based on the value of the eigenvalues, and therefore, it fails to provide an appropriate measurement of the information.

In this work, we propose the use of the minimum eigenvalue of the Fisher information matrix to determine the performance of the estimation process. By doing so, the less informative areas are prioritized during the path computation of the mobile sensing. This condition can be expressed based on the variable $\gamma_{k,i}$ as following

$$\begin{aligned} &\max_{\alpha, \gamma_k} \quad \alpha \\ &subject\ to \quad P_{k|k-1}^{-1} + \textstyle\sum_{i=1}^{N} C_{f,i}^T R_{f,i}^{-1} C_{f,i} + \sum_{i=1}^{N} \gamma_{k,i} (C_{m,i}^T R_{m,i}^{-1} C_{m,i}) \ \geq \alpha I \end{aligned} \tag{17}$$

where the measurement areas are selected such that the minimum eigenvalue of the information matrix is maximized. The main merit of (17) is to define the selection of the sensed areas that maximize the performance of the observer as an offline optimization problem.

2.4 Information-based orienteering path planning

This section presents a variant of the Orienteering Problem where (17) is augmented with the limitations in terms of flight time and continuity of a path planning problem. The main difference between the proposed approach and the usual sensor selection problem lies in the fact that the set of chosen points must satisfy the constraints of a feasible path for the UAV.

In this work, the UAV is seen as a point mass particle whose kinematics are the ones of a single integrator with a maximal speed $V_c > 0$. This vehicle is assumed to have a maximum autonomy $T_{max} > 0$ which is not affected by the different maneuvers performed during the flight. On the basis of our first testing campaign in our usual operative condition this assumption is a fair approximation of reality as long as sudden changes of altitude are avoided. Accordingly, the path followed by the UAV during the remote sensing can be defined by a set of straight lines between the different waypoints at the maximum speed allowed for the collection of data. The problem of selecting these waypoints can consequently be addressed as an Orienteering Problem.

2.4.1 Area characterization

The Orienteering Problem considers the case where, due to time or autonomy limitations, we have to select the points to visit as well as in which order for a given set of points. These points, in the case of remote sensing, are the points where the measurements are taken. To ensure the quality of the measurements and the post-process operation, the measurement triggering points must follow certain spatial distribution. To do so, the surface to be estimated is considered as an undirected and connected graph $G =< V, E >$. The set of vertices or nodes, V, represents the centroids of the areas in which the plane is divided and the set of edges E represents the allowed trajectories of the UAV. Consequently, the graph G creates a regular grid covering the whole area of interest which is represented in **Error! Reference source not found.**15.

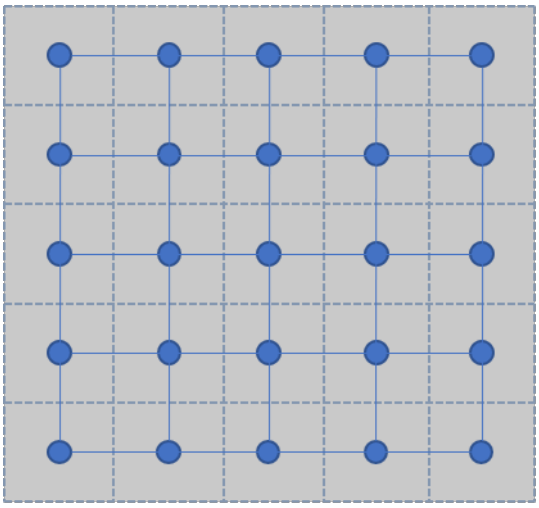

Figure 15 - Example of the regular grid obtained for a monitored area

For the definition of a proper flight mission, the take-off and a landing points must be determined before each flight. Due to the time limiting constraint, the selection of an initial and final point as part of the regular

grid would bias the resultant path, narrowing down the covered area. To avoid this problem, two *external* nodes V_{ini} and V_{fin} with directed edges and connected to all vertices are added to the preliminary graph. This creates an *extended* graph $G^*$where the first measuring point is not necessarily an adjacent node to the takeoff coordinates but part of the most critical area to cover. This concept can be seen in **Error! Reference source not found.**16.

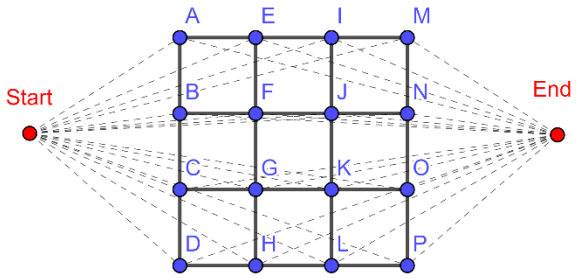

Figure 16 - Schema of the graph distribution G^*.

The edges $(i,j) \in E$ have an associated weight t_{ij} representing the time to travel from vertex i to vertex j. The use of time costs allows the consideration of different speeds during the flight. That is precisely the case of the first and last edge covered, where a higher speed can be considered as there is no sensing constrained related to them.

2.5 Orienteering problem formulation

In this subsection, following an approach inspired by the Miller-Tucker-Zemlin (MTZ) formulation of the TSP [44], the Orienteering Problem structure of the monitoring policy is formulated based on linear integer constraints.

Let us introduce two kind of decision variables; i) the binary variable q_{ij} whose value is 1 if the edge j is visited after the edge i, being 0 otherwise and ii) the integer variable u_i which denotes the visiting order of the node i.

The choice of an initial and final point is enforced by the following constraints

$$\sum_{i=2}^{N} q_{1i} = \sum_{i=1}^{N-1} q_{jN} = 1, \tag{18}$$

$$\sum_{i=2}^{N} q_{i1} = \sum_{i=1}^{N-1} q_{Nj} = 0. \tag{19}$$

For the rest of nodes, we must ensure that each node is visited at most once and that the path obtained is

$$\sum_{i=2}^{N} q_{ik} = \sum_{i=1}^{N-1} q_{kj} \leq 1 \quad \forall k = 2, \dots, N-1. \tag{4020}$$

Due to the restricted flight time, it is necessary to establish a maximum travelling time constraint, which is denote by

$$\sum_{i=2}^{N}\sum_{i=1}^{N-1} t_{ij}q_{ij} \leq T_{max}, \tag{21}$$

where T_{max} is the maximum time given by the autonomy of the UAV. To avoid possible subtours and to ensure the continuity of the path, it must hold that

$$2 \leq u_i \leq N \quad \forall i = 2, \dots, N, \tag{22}$$

$$u_i - u_j + 1 \leq (N-1)(1-q_{ij}) \quad \forall i,j = 2, \dots, N, \tag{23}$$

These constraints ensure the feasibility and continuity of the selected path for the UAV flight. In this regard, Figure 17 shows two examples of feasible and continuous paths.

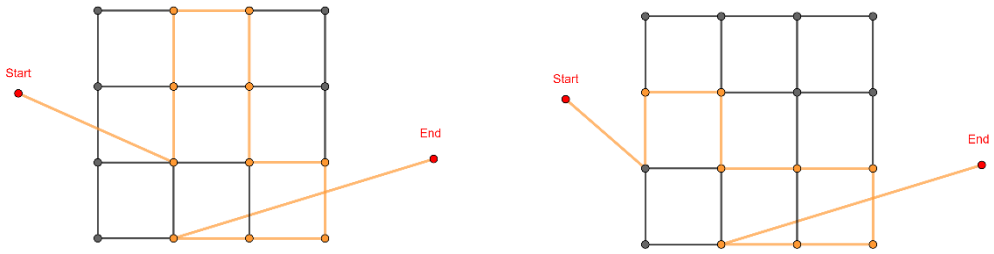

Figure 17 - Examples of feasible paths

Note that the fact of sensing the ith area at time k, previously introduced as $\gamma_{ik} = 1$, is equivalent to the condition $\sum_{j\in N_i} q_{ij} = 1$, where N_i denotes the set of edges with an existing link to i. Therefore, combining the path planning integer constraints (18)-(23) with (17), where γ_{ik} is now represented by $\sum_{j\in N_i} q_{ij}$, we obtain the following optimization problem

$$\max_{\alpha,\gamma_k} \quad \alpha$$

$$subject\ to \quad P_{k|k-1}^{-1} + \sum_{i=1}^{N} C_{f,i}^T R_{f,i}^{-1} C_{f,i} + \sum_{i=1}^{N} \gamma_{k,i}(C_{m,i}^T R_{m,i}^{-1} C_{m,i}) \geq \alpha I$$

$$\begin{aligned} &\sum_{i=2}^{N} q_{1i} = \sum_{i=1}^{N-1} q_{jN} = 1 \\ &\sum_{i=2}^{N} q_{i1} = \sum_{i=1}^{N-1} q_{Nj} = 0, \\ &\sum_{i=2}^{N} q_{ik} = \sum_{i=1}^{N-1} q_{kj} \leq 1 \quad \forall k = 2, \dots, N-1 \end{aligned} \tag{44}$$

$$\sum_{i=2}^{N}\sum_{i=1}^{N-1} t_{ij}q_{ij} \leq T_{max}$$

$$2 \leq u_i \leq N \quad \forall i = 2, \dots, N,$$

$$u_i - u_j + 1 \leq (N-1)(1-q_{ij}) \quad \forall i, j = 2, \dots, N,$$

$$q_{ij} \in \{0,1\} \quad \forall i, j = 2, \dots, N.$$

With this formulation, the information-based path planning is expressed as a Mixed-Integer Semidefinite Programming (MISDP) problem. A MISDP problem that can be optimally solved using commercial solvers, e.g. *SCIP* or *cutsdp [45], [46].* Nevertheless, it remains a NP-hard problem whose solving time grows excessively in the case of large instances of the problem.

2.6 Proposed heuristic

In this section we propose two preliminary heuristic methods to compute a suboptimal path based on the integer relaxation of the mixed-integer problem (44). The goal is to obtain close-to-optimal strategies more adequate for large-scale scenarios.

Consider $q_{ij} \in [0,1]$ and $u_i \in \mathbb{R}$ subject to

$$2 \leq u_i \leq N; \quad \forall i = 2, \dots, N, \tag{24}$$

the problem (44) becomes a Semi-Definite Programming (SDP) problem, which can be effectively and quickly solved by solvers such as *Mosek* [47].

The outcome of this convex problem, q_{ij}^r, provides values between 0 and 1 for the edge , see Figure18, which can be seen as how likely is the edge (i, j) to be taken in the optimal path. Therefore, a suboptimal solution can be obtained by computing heuristics which select the points to visit based on these values [48].

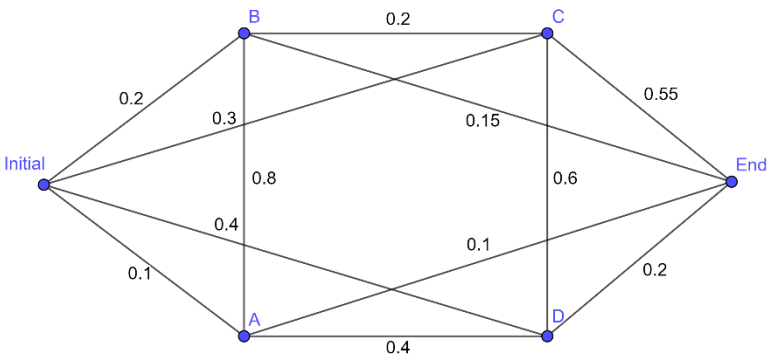

Figure 18 - Example of a possible solution of the relaxed problem.

Algorithm IV selects the path by rounding up each link (i, j) with probability q_{ij}^r. This greedy approach inserts the new edges into the path based on their probability to be part of the optimal solution. To ensure that the obtained path is feasible and continuous, the rounding is done sequentially. The algorithm starts from the

initial point V_{ini} adding edges until the maximum flight time T_{max} is reached. This process is depicted in Figure 19.

Input: Solution G_{rel}
Output: Set of edges X_{path} where $\lambda_1(X_{path})$ is maximized

$X_{path} \leftarrow \emptyset$
for $it \leq N$ **do**
  $X_{temp} \leftarrow \emptyset$
  $i \leftarrow ini$ % Start from initial node ini
  repeat
    Select q_{ij} by rounding adjacent edges
    $X_{temp} \leftarrow q_{ij}$
    $i \leftarrow j$ % Move to next node i
  until $(T_{temp} \geq T_{max})$
  return X_{temp}
  if $\lambda_1(X_{temp}) \geq \lambda_1(X_{path})$ **then**
    $X_{path} \leftarrow X_{temp}$
  end if
end for
Output X_{path}

Algorithm IV: Sequentially edge randomized rounding

To converge to a close-to-optimal solution, the operation is repeated for a fixed number of N iterations. After each iteration, the minimum eigenvalue associated to the computed path, $\lambda_1(X_{path})$, is compared with the previously stored solution. If the new path improves the current stored sequence, the latter is replaced, and the algorithm keeps seeking for alternative paths.

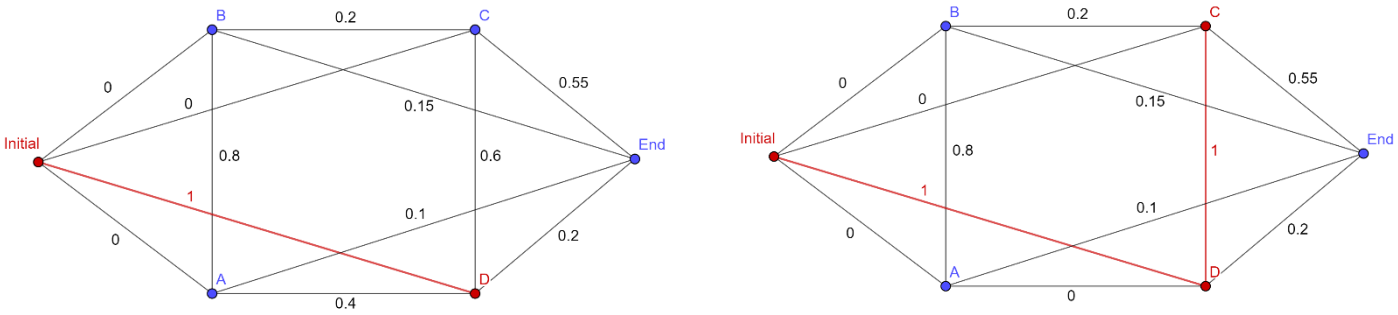

Figure 19 - One step of the randomized rounding algorithm

Algorithm V uses a similar approach focusing, instead, on the probability of selecting given nodes. For this purpose, the values obtained from the SDP problem are moved towards the nodes as an equivalent of a *reward* value. After the solutions q_{ij}^r are obtained, the reward θ_i for $i \in V$ is computed as

$$\theta_i = \sum_{j \in N_i} q_{ji}^r \quad \forall i \in V \tag{46}$$

where N_i represents the set of nodes that have an edge towards the node i. By performing this operation, we obtain a reward for the visit of each node based on the relaxed problem. The main idea behind this second heuristic is to prioritize the nodes with multiple high valued edges.

The insertion of nodes in the final path is done by following a randomized rounding similar to the one used in Algorithm IV. Therefore, once a node is selected, the next iteration chooses within the neighboring nodes N_i. The process is repeated for N iterations, comparing the stored values once T_{max} is reached, and starting the process again from the initial node.

Input: Reward $\theta \in \mathbb{R}^N$ and graph G^*
Output: Set of edges X_{path} where $\lambda_1(X_{path})$ is maximized

$X_{path} \leftarrow \emptyset$
for $it \leq N$ **do**
  $X_{temp} \leftarrow \emptyset$
  $i \leftarrow ini$ % Start from initial node ini
  repeat
    Select $V_j \in N_i$ based on θ_j
    $X_{temp} \leftarrow q_{ij}$
    $i \leftarrow j$ % Move to next node i
  until $(T_{temp} \geq T_{max})$
  return X_{temp}
  if $\lambda_1(X_{temp}) \geq \lambda_1(X_{path})$ **then**
    $X_{path} \leftarrow X_{temp}$
  end if
end for
Output X_{path}

Algorithm V: Sequentially node randomized rounding

2.6.1 Computational analysis of the heuristics

This section presents a study of the scalability, accuracy and computational performance of the two presented heuristics respect to the optimal formulation (44). Simulations are performed given different size areas, ranging from grids of 4×4 up to 6×6 nodes. In order to obtain meaningful results, for each problem size, 100 simulations have been performed varying the information distribution of the process. For this test, the autonomy of the vehicle is such that it allows to visit a maximum of 6 to 9 nodes depending on the size of the grid. This constraint allows to obtain representative differences between the solutions. Table V provides the level of degradation from both heuristics respect to the optimal value. It is noticeable that both heuristics are able to obtain results with less than 5% of degradation respect to the optimal, while providing outperforming results in terms of computational time, as it is shown in Table VI.

Grid size	Algorithm 1	Algorithm 2
4×4	4.80 %	4.74%
5×5	3.82%	3.95%
6×6	3.46%	3.75%

Table V: Solution degradation between the two proposed strategies and the optimal.

Table VI depicts the average time used to obtain the solution to the path-planning problem. In this case, for more than 36 nodes (a grid of 6×6) we can observe how the computational time for the mixed-integer formulation of the problem already increases in an excessive manner. However, for both heuristic strategies,

as **Error! Reference source not found.**20 shows, they provide reasonable computational times even for larger cases with more than 150 nodes.

Grid size	**Algorithm 1** (s)	**Algorithm 2** (s)	**Optimal** (s)
4×4	1.96	1.83	8.12
5×5	1.91	1.77	169.06
6×6	2.20	1.72	372.58
6×6	2.24	2.19	>3600

Table VI: Comparison computational time between the optimal and the two proposed strategies.

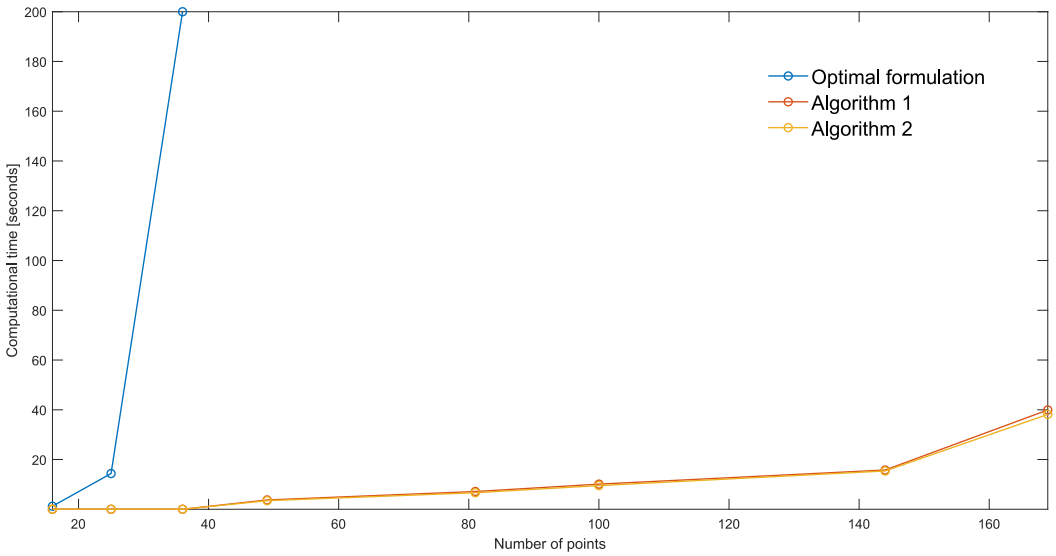

Figure 20 - Evolution of the computational time for the different methods

These results show that both heuristics, while providing sub-optimal results, achieve a level of performance interestingly close to the optimal. Furthermore, their low computational times are promising in their use in large-scale scenarios.

2.7 Simulation results and application

This section provides, through numerical simulations, an illustration of the effectiveness of the information-based path planning introduced. To do so, the adaptability of the path, the evolution over time and the performance are analyzed and compared against a traditional covering strategy. The numerical setting used for the numerical validation mimics the experimental setting proposed in the PANTHEON project. This model was originally presented in [49]. In this setup, we consider fixed soil sensors distributed in some areas of the orchard and the use of an agrometeorological IoT network which provides climate measurements in real time. For an orchard of n plants and divided into N soil parcels, the following system is considered

$$\begin{cases} x_{k+1} = Ax_k + B_d\widehat{d_k} + w_k \\ y_k = \Gamma_k(C_k x_k + v_k) \end{cases} \tag{25}$$

where $x = [x_1^T\ x_2^T\ x_3^T]^T \in \mathbb{R}^{N+2n}$ is the state vector with $x_1 = [\theta_1, \dots, \theta_N]^T$ the soil moisture status, $x_2 = [W_1, \dots, W_N]^T$ the water plant status and $x_3 = \left[W_{rem,1}, \dots, W_{rem,N}\right]^T$ the water status of the leaves, u_k represents the irrigation inputs and $\widehat{d_k}$ the meteorological disturbances.

In this context, while the distributed soil sensors are able to capture the value of soil moisture for the deployed areas, it is assumed that the water status of the leaves can be directly obtained through remote sensing thanks to the use of UAV covering. For further information about the model, the reader is referred to [49].

2.7.1 Adaptability of the path planning

In this section, the proper performance of the path planning is demonstrated. To do so, simulations are carried out while modifying the disposition of the ground sensors and considering multiple days of sensing.

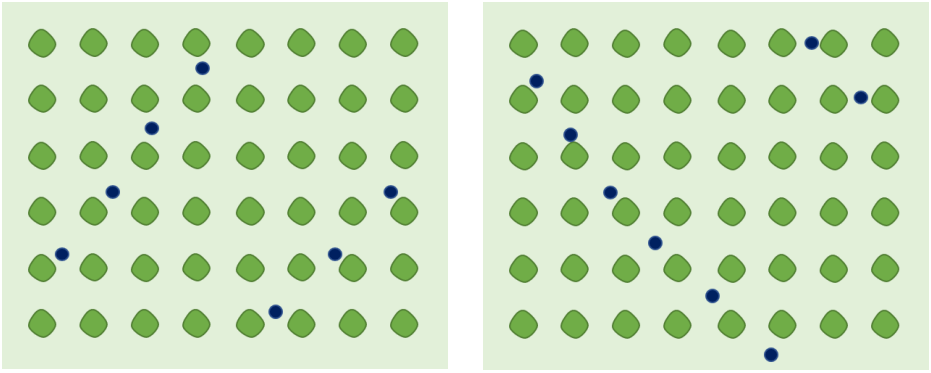

Figure 21- Fixed sensor distributions

In **Error! Reference source not found.**, depicted by blue points, two different soil sensor distributions are presented. In this setup, the areas close to the fixed sensors have more information about the water states than the more isolated areas. Therefore, changing the location of the sensors must change the information distribution regarding the estimation of the states [32] and thus the optimal covering path. The paths obtained for both distributions are depicted in **Error! Reference source not found.**22. These results show how the optimal path correctly changes according to the sensor restructuring. Regarding the use of previous information, a small area of an orchard has been simulated for a longer period. The resulting paths are depicted in **Error! Reference source not found.**23 where successive flights are performed in between periods of time preserving the same fixed sensor distribution. As it can be seen from the four plots, the areas with less information change accordingly to the already covered points and so does the path, demonstrating the spatial and temporal awareness of the presented strategy.

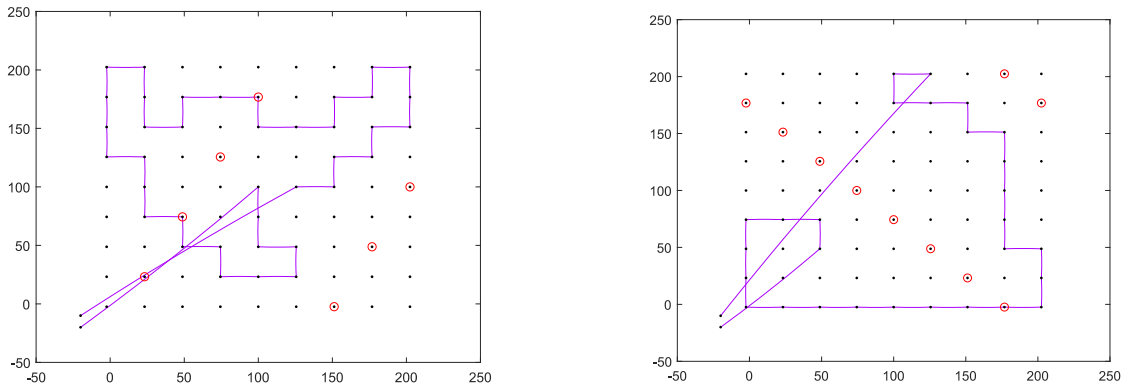

Figure 22 - Comparison of the obtained path according to the soil sensor distribution.

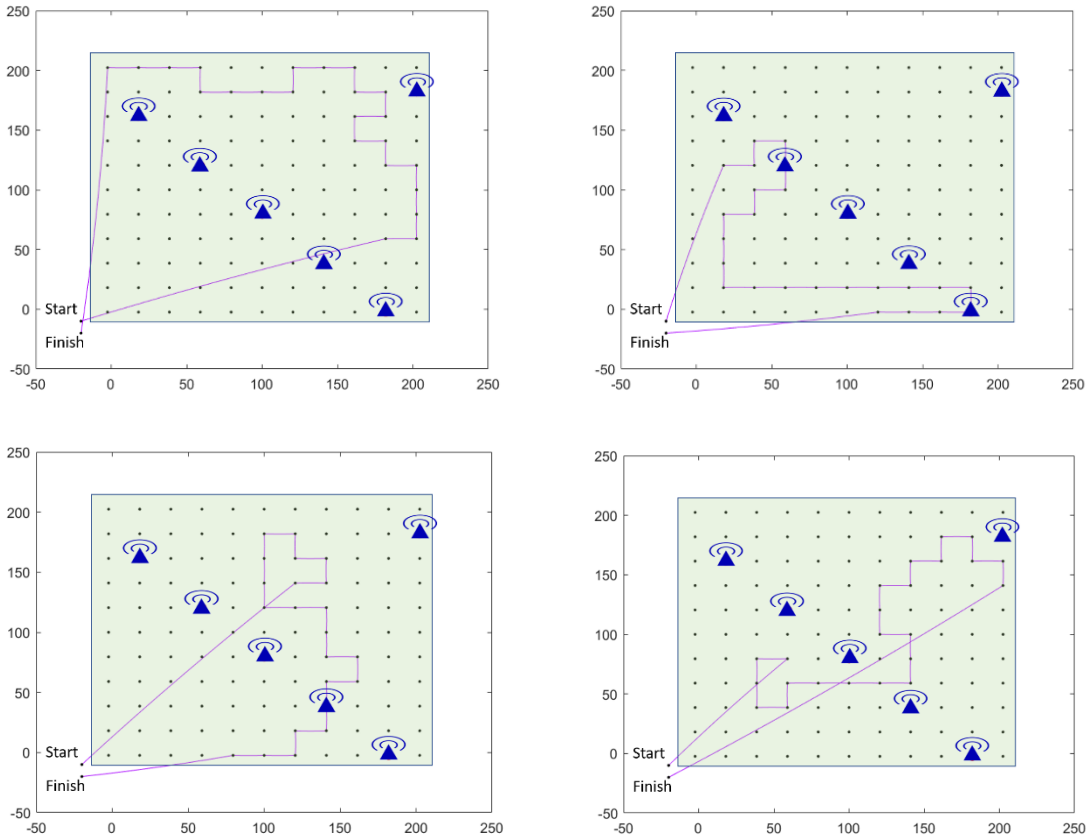

Figure 23 - Iterative path evolution

Remark:

The variation of the path respect to the fixed sensors distribution or the previous measurements depends on the dynamics of the system and the choice of the covariance matrices of the Kalman Filter.

2.7.2 Performance analysis

In this subsection, an iterative implementation of the presented path planning strategy is compared against a common strategy in persistent monitoring. This strategy divides the area into equal number of partitions based on the flight autonomy which are sequentially covered, ensuring the minimum latency in between visits. To obtain a fair comparison, all strategies provide the observation matrix C_k to a Kalman filter with identical parameters. The maximum flight time considered is also common to all of them. By doing so, the evaluation focuses on the best path strategy assuming the same state estimation process. The model is simulated for 350 hours and the data used for the meteorological disturbances comes from direct measurements from the experimental field. Stochastic flights are performed within a range from 35 to 70 hours, taking into consideration logistics and weather uncertainty.

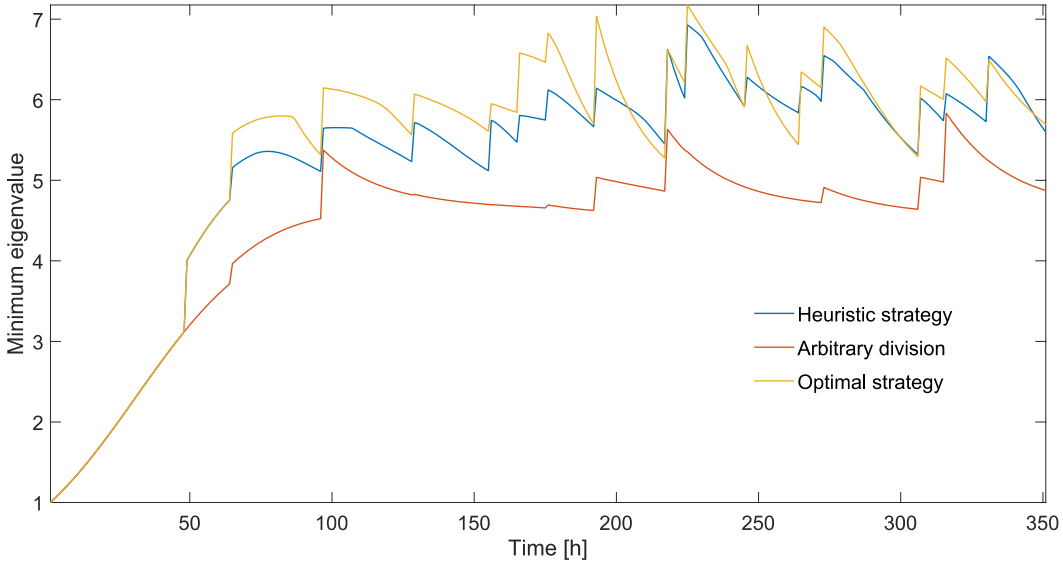

Figure 24 - Evolution of the minimum eigenvalue of the information matrix in a grid of 6 × 6 nodes.

Firstly, to include the performance of the optimal strategy proposed in (44), a simulation for a simpler grid of 6×6 nodes is performed. In Figure 24, the evolution of the minimum eigenvalue of the information matrix is shown. In this figure, the traditional strategy is referred as *Arbitrary division* and the heuristic shown corresponds to the Algorithm IV. From this plot, it can be seen that the presented strategies, optimal and suboptimal, clearly provide better results than the regular division. It is also important to notice how the optimal formulation presents some instants underperforming the heuristic strategy (Algorithm IV). This is due to the myopic approach followed in the development of the formulation, which can lead to less favorable situations in the case of short or regular periodicity in the remote sensing.

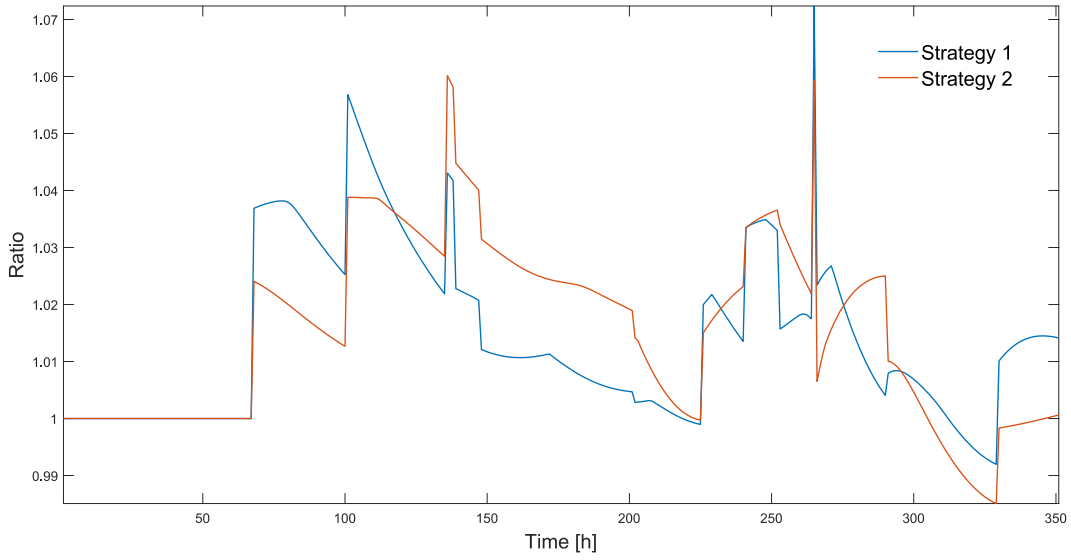

Figure 25 - Ratio between the performance obtained for the different strategies.

Error! Reference source not found., depicts the results obtained from a larger scenario, where the area dimensions correspond to a real case of large-scale hazelnut orchards, as shown in **Error! Reference source**

not found.. In this case, the values represented in the plot denote the ratio between the performance of the proposed strategies and the regular division of the area which is computed as

$$R(t) = \frac{\lambda_{1,heur}(t)}{\lambda_{1,arbt}(t)} \tag{26}$$

where $\lambda_1(t)$ represents the minimum eigen value at time k of the Fisher information matrix associated to each strategy. **Error! Reference source not found.**27 provides, for a similar setup, the case where the maximum flying time for the information-based strategy is reduced by 50%. From this plot, it can be seen that after reducing by 50% the resources, the information-based strategy is still able to outperform most of the time the regular coverage strategy.

Figure 26 - Area considered for the simulation.

This result supports one of the main claims of this contribution, which is that an information-based approach can help to reduce the resources put into the monitoring while keeping a similar performance. The results presented in this section demonstrate that, for a limited amount of flying time and using the same estimation methodology, the use of the proposed information-based path planning is able to improve the information regarding the estimation of the states. Additionally, besides the myopic approach thought for single and isolated missions, this last subsection shows how an iterative implementation provides interesting results.

Remark:

The results shown in this subsection are obtained in the case of a static sensor distribution and with a fairly symmetric structure. In the case of less symmetric structures or changing distribution the difference between the two approaches becomes larger and even more significant.

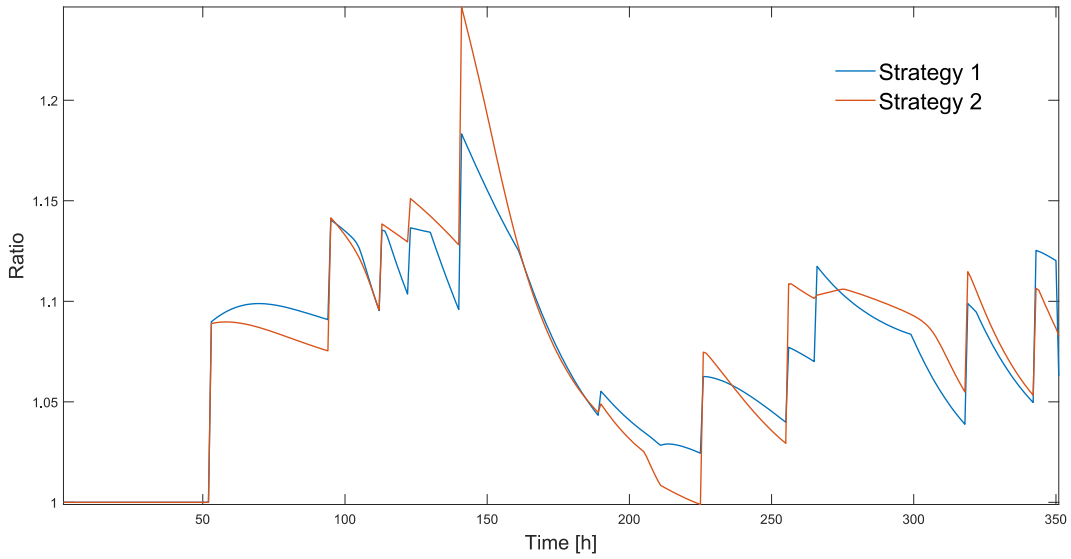

Figure 27 - Evolution of the ratio of performance. Case with reduction of 50% in flying time for the information-based approach.

3 Reference to Media Content

3.1 Unmanned Ground Vehicle

Title	Description	URL
UGV Field Validation	Experimental Validation of the UGV Platform	https://youtu.be/57Pa9xu2lvs

3.2 Unmanned Aerial Vehicle

Title	Description	URL
UAV Field Validation	Experimental validation of the UAV Platform	https://youtu.be/RU1eR1uzCuc

4 Bibliography

[1] R. F. Carpio *et al.*, "A Navigation Architecture for Ackermann Vehicles in Precision Farming," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1103–1110, 2020.

[2] H. J. Ferreau, "An online active set strategy for fast solution of parametric quadratic programs with applications to predictive engine control," *University of Heidelberg*, 2006.

[3] S. Parsons, "**Probabilistic Robotics** by Sebastian Thrun, Wolfram Burgard and Dieter Fox, MIT Press, 647 pp., $55.00, ISBN 0-262-20162-3," *The Knowledge Engineering Review*, vol. 21, no. 3, pp. 287–289, Sep. 2006, doi: 10.1017/S0269888906210993.

[4] C. Cadena *et al.*, "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016, doi: 10.1109/TRO.2016.2624754.

[5] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, Sep. 2006, doi: 10.1109/MRA.2006.1678144.

[6] K. M. Brink, "Partial-Update Schmidt–Kalman Filter," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 9, pp. 2214–2228, 2017, doi: 10.2514/1.G002808.

[7] D. Simon, *Optimal State Estimation: Kalman, H∞, and Nonlinear Approaches*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2006.

[8] F. Auat Cheein, G. Steiner, G. Perez Paina, and R. Carelli, "Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection," *Computers and Electronics in Agriculture*, vol. 78, no. 2, pp. 195–207, Sep. 2011, doi: 10.1016/j.compag.2011.07.007.

[9] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, Sep. 2003, vol. 2, pp. 1985–1991 vol.2, doi: 10.1109/ROBOT.2003.1241885.

[10] F. A. Cheein *et al.*, "Human-robot interaction in precision agriculture: Sharing the workspace with service units," in *2015 IEEE International Conference on Industrial Technology (ICIT)*, Mar. 2015, pp. 289–295, doi: 10.1109/ICIT.2015.7125113.

[11] J. P. Vasconez, G. A. Kantor, and F. A. Auat Cheein, "Human–robot interaction in agriculture: A survey and current challenges," *Biosystems Engineering*, vol. 179, pp. 35–48, Mar. 2019, doi: 10.1016/j.biosystemseng.2018.12.005.

[12] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial Databases with Noise," Jan. 1996, doi: null.

[13] A. Filatov, A. Filatov, K. Krinkin, B. Chen, and D. Molodan, "2D SLAM quality evaluation methods," in *2017 21st Conference of Open Innovations Association (FRUCT)*, Nov. 2017, pp. 120–126, doi: 10.23919/FRUCT.2017.8250173.
[14] L. Fang *et al.*, "Comparative evaluation of time-of-flight depth-imaging sensors for mapping and SLAM applications," in *ICRA 2016*, 2016.
[15] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct. 2013, pp. 1–6, doi: 10.1109/SSRR.2013.6719348.
[16] M. Rojas-Fernández, D. Mújica-Vargas, M. Matuz-Cruz, and D. López-Borreguero, "Performance comparison of 2D SLAM techniques available in ROS using a differential drive robot," in *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, Feb. 2018, pp. 50–58, doi: 10.1109/CONIELECOMP.2018.8327175.
[17] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.
[18] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.
[19] B. Fleischmann, "A cutting plane procedure for the travelling salesman problem on road networks," *European Journal of Operational Research*, vol. 21, no. 3, pp. 307–317, Sep. 1985, doi: 10.1016/0377-2217(85)90151-1.
[20] A. N. Letchford, S. D. Nasiri, and D. O. Theis, "Compact formulations of the Steiner Traveling Salesman Problem and related problems," *European Journal of Operational Research*, vol. 228, no. 1, pp. 83–92, Jul. 2013, doi: 10.1016/j.ejor.2013.01.044.
[21] A. Fischer and P. Hungerländer, "The traveling salesman problem on grids with forbidden neighborhoods," *J Comb Optim*, vol. 34, no. 3, pp. 891–915, Oct. 2017, doi: 10.1007/s10878-017-0119-z.
[22] L. Wu, M. Á. G. García, D. P. Valls, and A. S. Ribalta, "Voronoi-based space partitioning for coordinated multi-robot exploration," *Journal of Physical Agents*, vol. 1, no. 1, pp. 37–44, Sep. 2007, doi: 10.14198/JoPha.2007.1.1.05.
[23] J. Conesa-Muñoz, J. M. Bengochea-Guevara, D. Andujar, and A. Ribeiro, "Route planning for agricultural tasks: A general approach for fleets of autonomous vehicles in site-specific herbicide applications," *Computers and Electronics in Agriculture*, vol. 127, pp. 204–220, Sep. 2016, doi: 10.1016/j.compag.2016.06.012.
[24] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 92, pp. 79–97, 2014, doi: https://doi.org/10.1016/j.isprsjprs.2014.02.013.
[25] M. Díaz-Cabrera, J. Cabrera-Gámez, R. Aguasca-Colomo, and K. Miatliuk, "Photogrammetric Analysis of Images Acquired by an UAV," in *Computer Aided Systems Theory - EUROCAST 2013*, Berlin, Heidelberg, 2013, pp. 109–116.
[26] S. Alamdari, E. Fata, and S. L. Smith, "Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 138–154, Jan. 2014, doi: 10.1177/0278364913504011.
[27] Christos. G. Cassandras, X. Lin, and X. Ding, "An Optimal Control Approach to the Multi-Agent Persistent Monitoring Problem," *IEEE Trans. Automat. Contr.*, vol. 58, no. 4, pp. 947–961, Apr. 2013, doi: 10.1109/TAC.2012.2225539.
[28] J. Scherer and B. Rinner, "Persistent multi-UAV surveillance with energy and communication constraints," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 1225–1230, doi: 10.1109/COASE.2016.7743546.

[29] A. T. Klesh, P. T. Kabamba, and A. R. Girard, "Path planning for cooperative time-optimal information collection," in *2008 American Control Conference*, Jun. 2008, pp. 1991–1996, doi: 10.1109/ACC.2008.4586785.

[30] K.-C. Ma, Z. Ma, L. Liu, and G. S. Sukhatme, "Multi-robot Informative and Adaptive Planning for Persistent Environmental Monitoring," in *Distributed Autonomous Robotic Systems: The 13th International Symposium*, R. Groß, A. Kolling, S. Berman, E. Frazzoli, A. Martinoli, F. Matsuno, and M. Gauci, Eds. Cham: Springer International Publishing, 2018, pp. 285–298.

[31] R. Cui, Y. Li, and W. Yan, "Mutual Information-Based Multi-AUV Path Planning for Scalar Field Sampling Using Multidimensional RRT*," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 46, no. 7, pp. 993–1004, Jul. 2016, doi: 10.1109/TSMC.2015.2500027.

[32] V. Tzoumas, A. Jadbabaie, and G. J. Pappas, "Sensor placement for optimal Kalman filtering: Fundamental limits, submodularity, and algorithms," in *2016 American Control Conference (ACC)*, Jul. 2016, pp. 191–196, doi: 10.1109/ACC.2016.7524914.

[33] S. Joshi and S. Boyd, "Sensor Selection via Convex Optimization," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 451–462, Feb. 2009, doi: 10.1109/TSP.2008.2007095.

[34] J. Binney, A. Krause, and G. S. Sukhatme, "Optimizing waypoints for monitoring spatiotemporal phenomena:," *The International Journal of Robotics Research*, Jul. 2013, doi: 10.1177/0278364913488427.

[35] S. Garg and N. Ayanian, "Persistent Monitoring of Stochastic Spatio-temporal Phenomena with a Small Team of Robots," in *Robotics: Science and Systems X*, Jul. 2014, doi: 10.15607/RSS.2014.X.038.

[36] X. Lan and M. Schwager, "Planning periodic persistent monitoring trajectories for sensing robots in Gaussian Random Fields," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 2415–2420, doi: 10.1109/ICRA.2013.6630905.

[37] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics (NRL)*, vol. 34, no. 3, pp. 307–318, 1987, doi: 10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D.

[38] J. Yu, M. Schwager, and D. Rus, "Correlated Orienteering Problem and its Application to Persistent Monitoring Tasks," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, Oct. 2016, doi: 10.1109/TRO.2016.2593450.

[39] L. Bottarelli, M. Bicego, J. Blum, and A. Farinelli, "Orienteering-based informative path planning for environmental monitoring," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 46–58, Jan. 2019, doi: 10.1016/j.engappai.2018.09.015.

[40] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1453–1464, Sep. 2004, doi: 10.1109/TAC.2004.834121.

[41] E. Garone, B. Sinopoli, A. Goldsmith, and A. Casavola, "LQG Control for MIMO Systems Over Multiple Erasure Channels With Perfect Acknowledgment," *IEEE Trans. Automat. Contr.*, vol. 57, no. 2, pp. 450–456, Feb. 2012, doi: 10.1109/TAC.2011.2167789.

[42] M. Segal and E. Weinstein, "A new method for evaluating the log-likelihood gradient, the Hessian, and the Fisher information matrix for linear dynamic systems," *IEEE Transactions on Information Theory*, vol. 35, no. 3, pp. 682–687, May 1989, doi: 10.1109/18.30995.

[43] B. Grocholsky, A. Makarenko, and H. Durrant-Whyte, "Information-theoretic coordinated control of multiple sensor platforms," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, Sep. 2003, vol. 1, pp. 1521–1526 vol.1, doi: 10.1109/ROBOT.2003.1241807.

[44] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer Programming Formulation of Traveling Salesman Problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, Oct. 1960, doi: 10.1145/321043.321046.

[45] T. Gally, M. E. Pfetsch, and S. Ulbrich, "A framework for solving mixed-integer semidefinite programs," *Optimization Methods and Software*, vol. 33, no. 3, pp. 594–632, May 2018, doi: 10.1080/10556788.2017.1322081.

[46] C. Rowe and J. Maciejowski, "An efficient algorithm for mixed integer semidefinite optimisation," in *Proceedings of the 2003 American Control Conference, 2003.*, Jun. 2003, vol. 6, pp. 4730–4735 vol.6, doi: 10.1109/ACC.2003.1242470.
[47] L. Vandenberghe and S. Boyd, "Semidefinite Programming," *SIAM Review*, vol. 38, no. 1, pp. 49–95, 1996.
[48] P. Raghavan and C. D. Tompson, "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 4, pp. 365–374, Dec. 1987, doi: 10.1007/BF02579324.
[49] N. Bono Rossello, R. Fabrizio Carpio, A. Gasparri, and E. Garone, "A novel Observer-based Architecture for Water Management in Large-Scale (Hazelnut) Orchards," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 62–69, Jan. 2019, doi: 10.1016/j.ifacol.2019.12.498.