Ref. Ares(2023)6624275 - 30/09/2023

This project is funded by the European Union
G.A No 101016906

Project Number: 101016906
Start Date of Project: 2021/01/01
Duration: 48 months

Type of document D9.1 – V1.0

Field Validation of the Robot Navigation Functionalities

Dissemination level	PUBLIC
Submission Date	2023-09-30
Work Package	WP9
Task	T9.1, T9.5
Type	Report
Version	1.0
Author	Martina Lippi, Renzo Fabrizio Carpio, Tsampikos Kounalakis and Andrea Gasparri
Approved by	KTH, UNICLAM, DTI

DISCLAIMER:
The sole responsibility for the content of this deliverable rests with the authors. It does not necessarily reflect the opinion of the European Union. Neither the Directorate-General for Communications Networks, Content and Technology, Artificial Intelligence and Digital Industry nor the European Commission are responsible for any use that may be made of the information

This project is funded by the European Union G.A No 101016906

Executive Summary

This deliverable provides a comprehensive overview of the real-world validation process for the navigation capabilities of the robotic mobile bases, as well as the coordination between multiple robots. In particular, the field validation aims to assess the effectiveness and robustness of the localization and mapping algorithm, and of the control, navigation, and obstacle avoidance algorithms. Additionally, the validation of the coordinated maneuvering, which is required for the exchange of a full box with an empty one between the farming and logistic robots is provided.

A video summarizing the overall field validation is available at the link:

Deliverable9_1-Navigation.mp4.

This project is funded by the European Union G.A No 101016906

Table of Content

1 Introduction 6
2 Setup overview 8
2.1 Wi-fi network 8
2.2 Mobile Base Equipment 9
2.2.1 Sensors for Navigation 9
2.2.2 Box-exchange Mechanism and Sensors for Maneuvering 10
2.3 Mobile base kinematic model 12
3 Navigation stack overview 14
4 Simultaneous Localization and Mapping 15
4.1 Perception for obstacle detection 15
4.2 Topological map 17
4.3 Methodology 18
4.3.1 EKF for SLAM 18
4.3.2 Data Association and Planting Pattern Estimation 21
4.3.3 Resulting SLAM Architecture 24
4.4 Experimental Validation 25
4.4.1 Identification of all landmarks 26
4.4.2 Identification of the row orientation 31
4.4.3 Relative localization to the closest landmarks 32
4.4.4 Loop closure 43
5 Local planner and controller 61
5.1 Methodology 61
5.1.1 Finite-horizon optimization problem 61
5.1.2 Possible states 62
5.2 Experimental Validation 64
6 Global planner 71
6.1 Methodology 71

6.1.1 Occupancy Grid 71
6.1.2 Algorithm Overview 75
6.2 Experimental Validation 76
7 Coordinated Maneuvering 81
7.1 Perception system for farming robot localization 81
7.2 Methodology 83
7.3 Experimental validation 87
7.3.1 Perception validation 87
7.3.2 Maneuvering validation 88
7.3.3 Experiment with overall pipeline 93
8 References 95

This project is funded by the European Union G.A No 101016906

Abbreviations and Acronyms

ROS	Robotic Operating System
BEM	Box-Exchange Mechanism
RGB-D	Red Green Blue Depth
FR	Farming robot
LR	Logistic robot

This project is funded by the European Union G.A No 101016906

1 Introduction

The ability of the mobile bases to navigate the vineyard autonomously and safely is an essential prerequisite for accomplishing the majority of the objectives of CANOPIES project. Figure 1 depicts the typical operating scenario envisioned within CANOPIES project. Here, multiple robotic platforms coexist in a vineyard shared with human operators. In this dynamic context, the robots must be able to perceive and navigate around various, possibly dynamical, obstacles, including human operators, structural elements like poles and other robots. Additionally, they must efficiently reach their assigned waypoints within the field. To realize this capability, it is necessary to endow the robot with proper perception systems as well as with an entire navigation stack, including the localization and mapping algorithm, the local planner and controller, and the global planner.

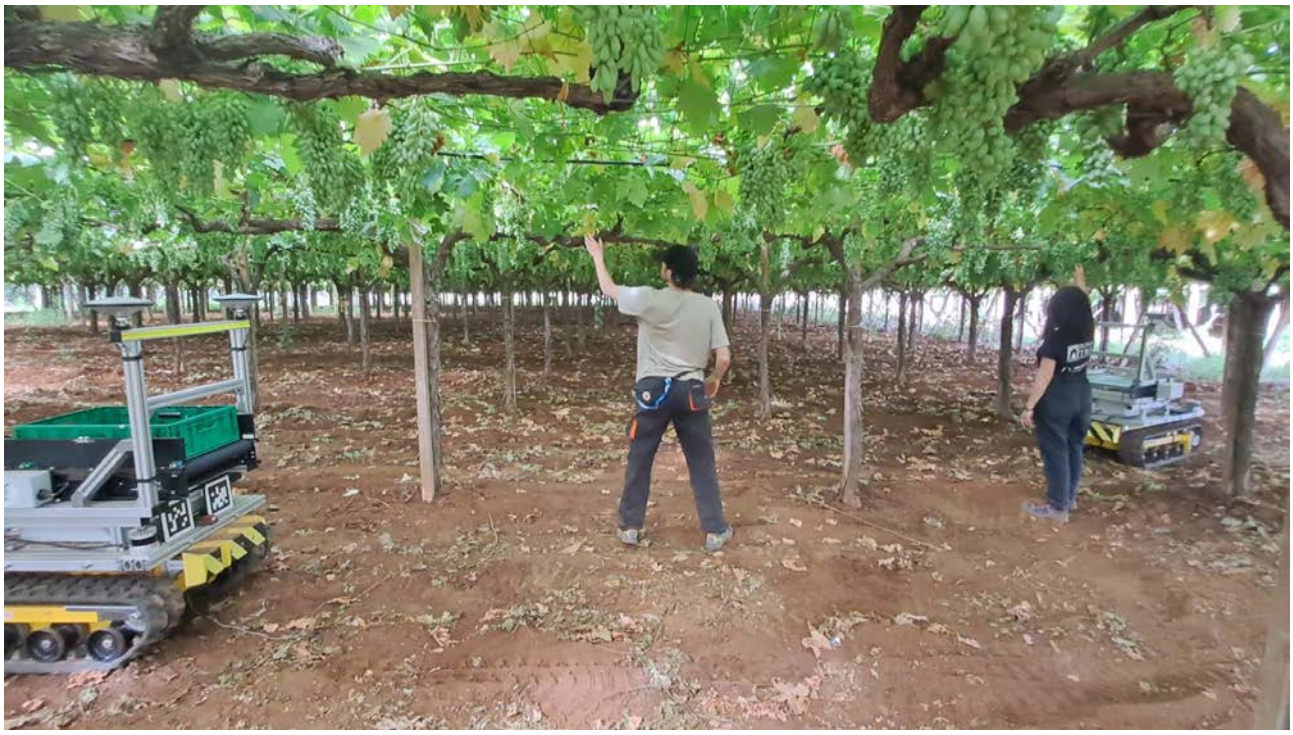


Figure 1 - Illustrative example of the operating scenario considered in CANOPIES project

Building on the autonomous navigation functionalities, the robots are also tasked with achieving coordination. More in detail, a fundamental ability which is envisioned in CANOPIES project is the coordinated maneuvering, which aims to align a logistic robot with a farming one, also referred to as docking. This enables the box-exchange operation between the robots, that is replacing a full box carried by the farming robot with an empty one supplied by the logistic robot. An illustrative example of this operation is provided in Figure 2.

This project is funded by the European Union G.A No 101016906



Figure 2 - Illustration of the box exchange operation

In the following, a brief description of the hardware setup and an overview of the designed navigation stack are provided. Then, each component of the navigation stack along with the respective experimental validation is presented. Finally, the coordinated maneuvering strategy along with its experimental validation, including the box exchange, is described.

This project is funded by the European Union G.A No 101016906

2 Setup overview

In this section, we briefly recall the hardware setup which is used for the navigation stack and the coordinated maneuvering strategy. Further details can be found in Deliverable 7.1 "Coordinated Maneuvering for Box-Exchange Operation".

2.1 Wi-fi network

A wireless mesh network has been installed within the vineyard to provide Wi-Fi network coverage. In this way, it is possible to have communication among the robots in the field, and communication for each robot with the GNSS base station for the RTK localization system. Figure 3 provides a representation of the Wi-Fi mesh network deployed within CANOPIES experimental field (highlighted with a green rectangle) at "Società Cooperativa Agricola Corsira" in the municipality of Aprilia, Lazio, Italy.

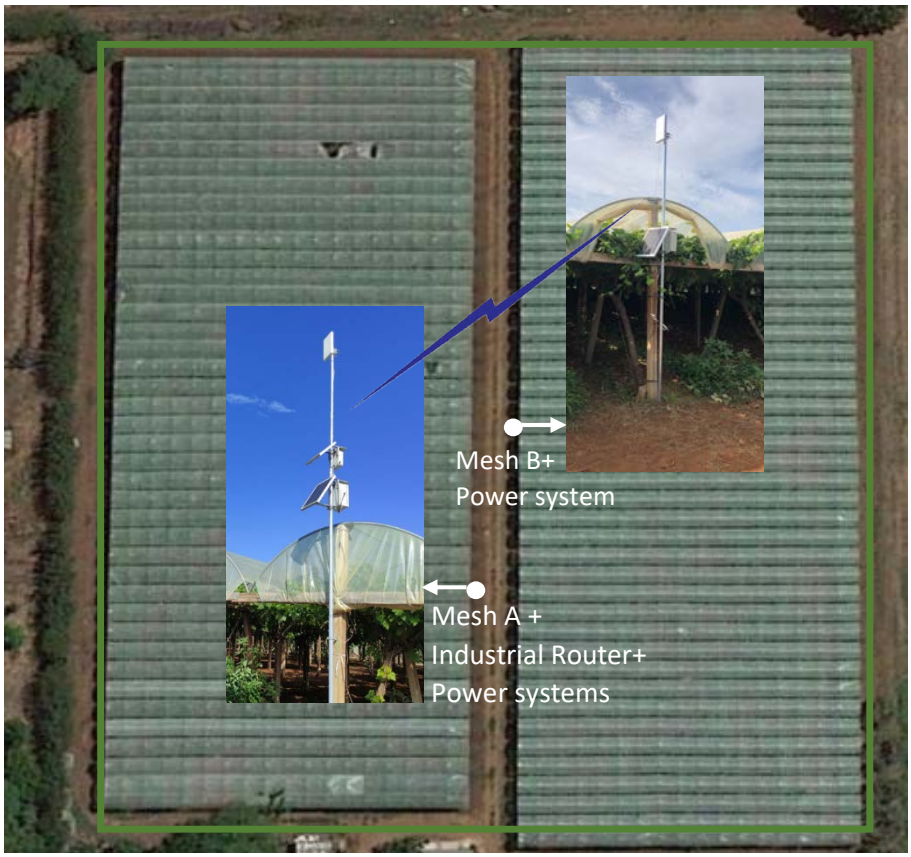


Figure 3 - Wi-fi mesh network within CANOPIES experimental field

Specifically, the Wi-fi mesh network is composed of:

- 2 UNIFI antennas (AC MESH PRO),
- 1 industrial router (IR615-S-EN00-WLAN),

which guarantees coverage over the entire experimental field. Each antenna in the mesh network is equipped with a dedicated power system, featuring a power box with a 12V, 9A sealed lead acid battery, a 300W inverter, a 48V PoE adapter, and a 20W solar panel for battery recharging via the

inverter. Another power box, equipped with a 12V, 7.2A battery, an inverter, and a 10W solar panel, is installed specifically to provide power to the industrial router. This setup ensures reliable and continuous power supply for the mesh operation.

From the software perspective, the UniFi devices are bundled with the UniFi Controller software, which allows managing the UniFi network by resorting to a web browser.

2.2 Mobile Base Equipment

Both the farming and logistic robots are equipped with the same mobile platform, specifically the Alitrak DCT-350P, and utilize the same set of navigation sensors. This choice ensures uniformity in performance across the different types of robots as well as facilitates the development of the navigation stack. Figure 4 shows the mobile platform used in CANOPIES project, its sensors, and the mechanism for the box-exchange. Each mobile base is also provided with an Intel® NUC 11 Pro Mini PC NUC11TNKv7, where the navigation and coordination algorithms with respective data processing are executed, and a Mikrotik RB4011iGS router, which enables communication between different platforms through the mesh network as well as communication between the devices on the base itself.

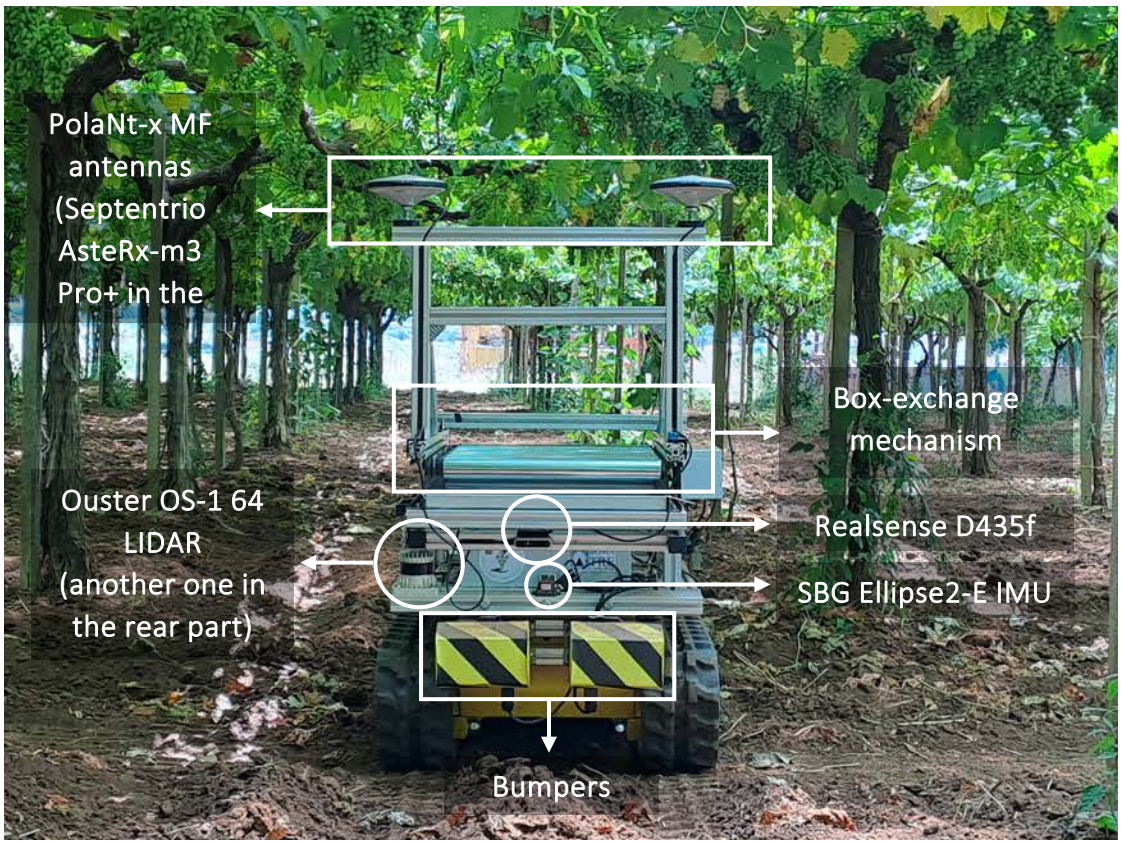


Figure 4 - Overview of equipment of each mobile base

2.2.1 Sensors for Navigation

The following main sensors are available in each platform for navigation purposes:

This project is funded by the European Union G.A No 101016906

- Two Ouster OS1-64 LIDAR sensors (one in the front part and one in the rear part) provide three-dimensional view of the robot's surroundings. The sensors are characterized by 64 channels and offer various resolution configurations, including 512 on a maximum 20Hz setting, 1024 on a maximum 20Hz setting, and 2048 on a maximum 10Hz setting.
- An RTK-GNSS localization system provides georeferenced location of the robot position with great accuracy. The system is composed of a rover module, including a Septentrio AsterRx-m3 Pro+ receiver and two PolaNt-x MF antennas, and a base station module, including a Septentrio AsteRx-U receiver and a PolaNt-x MF antenna. Note that the base station module is common to all platforms and sends corrections through the wi-fi mesh network. According to the datasheet, this setup allows to achieve horizontal accuracy of 0,6 cm + 0,5 ppm and vertical accuracy of 1cm + 1ppm.
- An SBG Ellipse-E IMU sensor provides the platform's inertial state and orientation. According to the datasheet, this sensor allows to obtain error measurements of pitch and roll to 0.1 degrees without any additional aids, and maximum fused output (GNNS + IMU) frequency equal to 200Hz.

2.2.2 Box-exchange Mechanism and Sensors for Maneuvering

The following main sensors are available in each platform for the operation of the Box-Exchange Mechanism:

- The primary element for transferring boxes between the two robots is an Interroll conveyor belt. The selection of these conveyors was based on their ability to function effectively in the field's working conditions, such as humidity and dust. Special attention was given to the conveyors' capacity to utilize friction to aid in loading and unloading operations. One notable distinction between the two robots is that the farming robot is equipped with an extra belt material that enhances the friction even further. A top view of the two robotic platforms equipped with conveyors is provided in Figure 5

This project is funded by the European Union G.A No 101016906



Figure 5 - Mobile bases for the farming (left, no torso included) and logistic (right) robots

- To monitor the status of loading and unloading on each conveyor, a set of retroreflective sensors is employed for each conveyor. Specifically, a retroreflector positioned at the front of each conveyor is activated when its beam is obstructed by boxes being loaded onto or unloaded from the conveyor. This allows the overall system to have a status about the successful loading and unloading of boxes. The retroreflective sensor located at the back of the box enables the BEM system to determine whether a box is present on the conveyor belt, providing information that determines whether the system should proceed with a loading or unloading operation.
- In the case of the logistic robot, the sensory suite for the developed perception system used for farming robot localization, as described in Section 7.1, consists of a Realsense D435f camera. The D435f camera is equipped with an IR pass filter, which enhances the quality of depth noise and extends its performance range, particularly in environments with varying illumination conditions. This camera is mounted on the BEM component of the logistic robot, enabling it to detect fiducial markers, specifically AprilTags, placed on the farming robot.
- Both platforms were outfitted with a prototype bumper design to offer feedback during the docking process. Specifically, the logistic robot features an active bumper prototype that enables linear motion, triggering a limit switch to generate a contact signal. This signal can be utilized as part of the overall state machine to facilitate the docking behavior. On the other hand, the farming robot is equipped with a passive bumper, which lacks sensing capabilities. An illustrative example of the bumpers is provided in Figure 6.



Figure 6 - Overview of the prototype bumper activation. Left: Farming robot (passive bumper). Right: Logistics robot (active bumper)

2.3 Mobile base kinematic model

As reported in Deliverable 7.1 "Coordinated Maneuvering for Box-Exchange Operation", the kinematic model of the mobile bases can be reduced to a unicycle model (shown in Figure 7), i.e., a vehicle that cannot move in the direction orthogonal to its sagittal axis, and is reported in the following for the sake of completeness.

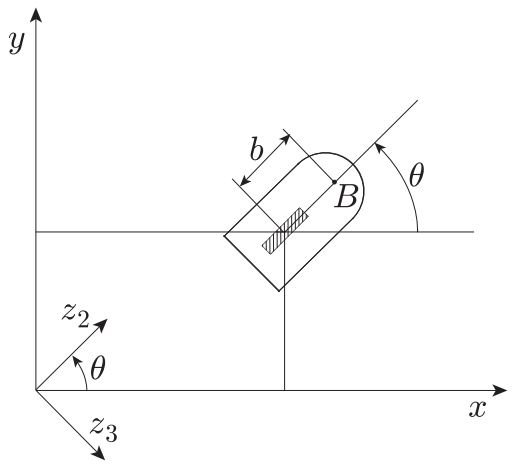


Figure 7 - Unicycle model

The configuration of each mobile base is given by the generalized coordinates $\boldsymbol{q}(t) = [\boldsymbol{p}(t)^{\boldsymbol{T}}, \theta(t)]^T \in \mathcal{R}^3$, where $\boldsymbol{p}(t) = [x(t), y(t)]^T \in \mathcal{R}^2$ represents the Cartesian coordinates of the

This project is funded by the European Union G.A No 101016906

center of the axis of the wheels with respect to a global inertial frame and θ represents the orientation of the vehicle body, with respect to the x axis of the world inertial frame. The kinematic model in the global frame is expressed as follows

$$\dot{\boldsymbol{q}}(t) = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \omega(t) \end{bmatrix} = f\big(\boldsymbol{q}(t), \boldsymbol{u}(t)\big) = \begin{bmatrix} \cos(\theta(t)) & 0 \\ \sin(\theta(t)) & 0 \\ 0 & 1 \end{bmatrix} \boldsymbol{u}(t), \quad (1)$$

where $\boldsymbol{u}(t) = [v(t), \omega(t)]^T \in \mathcal{R}^2$ is the control input composed of the linear $v(t)$ and angular $\omega(t)$ velocities.

This project is funded by the European Union G.A No 101016906

3 Navigation stack overview

The navigation stack consists of three core modules:

1) Simultaneous Localization and Mapping (SLAM): This module is responsible for estimating the robot pose, creating a map of the vineyard as well as providing the list of possible other obstacles in the environment. This is achieved by integrating data from both the navigation sensors and the mobile base encoders.
2) Local Planner and Controller: This module generates optimized trajectories and corresponding velocity commands for the robot to guide the robot to desired waypoints while ensuring obstacle avoidance and minimizing energy expenditure.
3) Global Planner: This module defines the specific waypoints that the robotic platform should follow to reach a desired pose within the vineyard. Therefore, it provides the overarching navigation plan that guides the robot throughout its mission, ensuring it reaches its intended destination.

The overall navigation architecture is depicted in Figure 8.

When a mission is assigned, the global planner takes charge of determining the set of desired waypoints that the robot has to traverse. The current desired waypoint, denoted as q_d, becomes the reference point for the local planner and controller.

Simultaneously, an obstacle detection module analyzes sensor data to identify potential obstacles in the robot's path. This information is then further refined by the SLAM module, which establishes the robot's position and orientation, maps the environment, and outputs the filtered list of obstacles as detailed in Section 4.3.3. These data are used by the local planner and controller to guarantee safe and optimal trajectories.

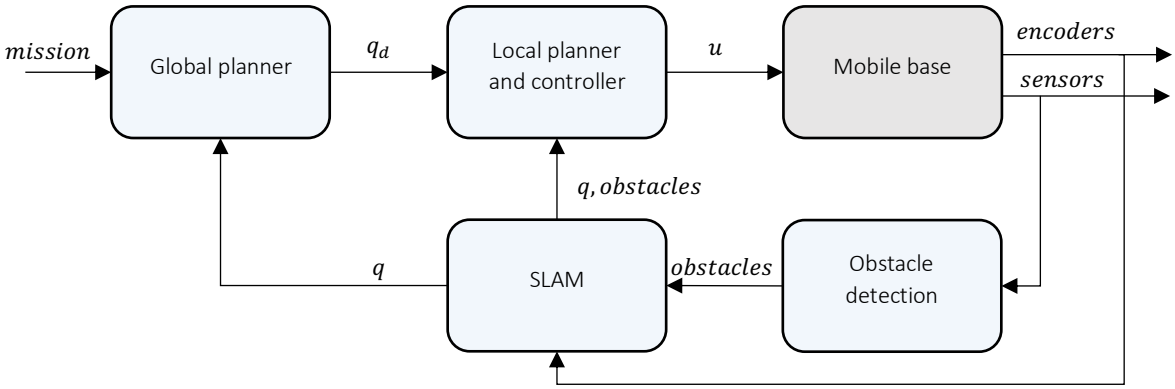


Figure 8 - Navigation stack architecture

This project is funded by the European Union G.A No 101016906

4 Simultaneous Localization and Mapping

In this section, the perception system realized to identify the obstacles and the map formulation are presented; then, the designed SLAM methodology is described, and the experimental results obtained within CANOPIES vineyard are provided.

4.1 Perception for obstacle detection

A LIDAR-based solution has been designed for environment sensing. The aim of this solution is twofold: firstly, to identify the poles where grapevine plants are attached serving as landmarks for the field map, and secondly, to detect any other potential obstacles, including human operators and boxes. In the following, a concise overview of the detection algorithm is provided while a detailed description is available in Deliverable 7.1 “Coordinated Maneuvering for Box-Exchange Operation”.

The basic idea of the detection algorithm is to discern two distinct patterns within the point clouds:

1) lines (vertical or oblique), which are typically associated with poles and humans;
2) general “blobs”, i.e., agglomerate of points, which are associated with possible other objects in the environment such as boxes.

To this aim, the following steps are applied to each point cloud (acquired from each of the two LIDARs):

- *Data filtering*, where a portion of the points within the cloud is selected for processing.
- *Data clustering*, where points are clustered depending on the patterns of interest.
- *Data processing*, where final processing of the clustered data is performed to identify obstacles.

The data filtering step aims to reduce data noise and lighten the computational workload. First, only the points belonging to a restricted number of rings are considered. Second, filtering operations are applied to ignore irrelevant points based on the patterns of interest. Specifically, two point clouds are produced: the “line-oriented” point cloud, where the points belonging to the robot body, those far from the robot and the points with small and high z-axis values are filtered out (associated with the top canopy structure and the ground respectively) and the “blob-oriented” point cloud, which is similar to the previous but the points with small z-axis values are preserved in order to also identify objects lying on the ground. An example of input point cloud collected in the field and the filtered version for line detection is reported in Figure 9.

This project is funded by the European Union G.A No 101016906

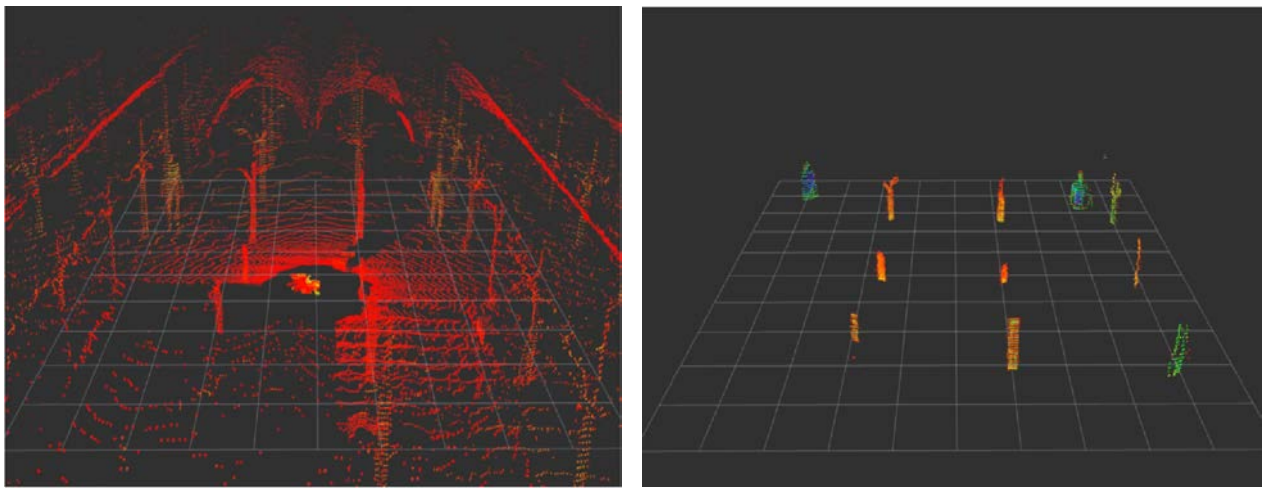


Figure 9 – Example of input point cloud and the filtered version for line detection

The data clustering phase for identifying the line pattern is based on applying a Density Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm to each ring. By doing so, the goal is to ideally obtain a distinct cluster for each grapevine tree and human within every ring. The data clustering phase for identifying blobs relies on a Euclidean Cluster Extraction algorithm applied to the entire "blob-oriented" point-cloud. Figure 10 shows examples of data filtering for lines detection (on the left, in white the centroid of each cluster is shown) and for blob detection (on the right, red spheres mark the cluster centroids). Note that no further processing is needed to identify blobs, but they are obtained during the data clustering phase.

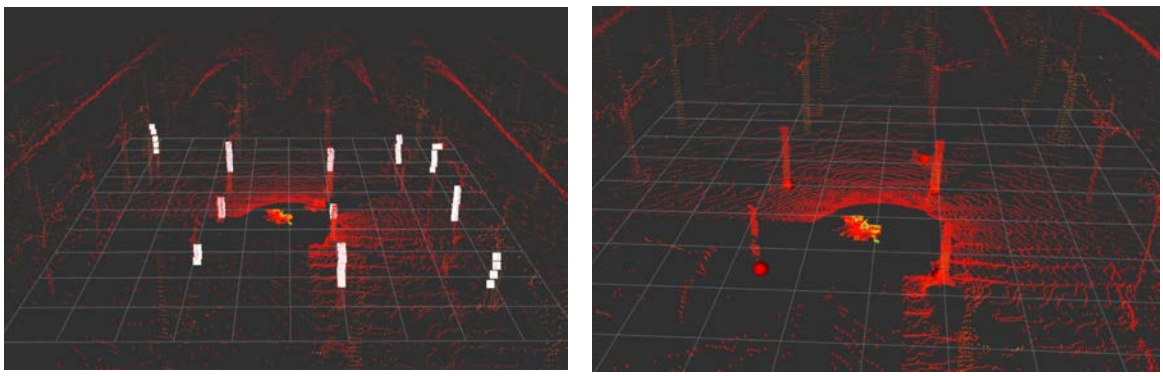


Figure 10 – Data clustering phase for identifying lines (left – white points) and for identifying blobs (right – red spheres)

As final processing step to find lines in the point cloud, a 3D Hough transform for line detection (Dalitz, Schramke, & Jeltsch., 2017) is applied to the clustered point cloud, i.e., the point cloud containing the centroids obtained with DBSCAN. Given the detected lines, these are classified into vertical (in blue in Figure 11) and oblique ones (in green in the figure). The vertical ones can be associated with grapevine trees or humans standing, while the oblique ones can be associated with oblique poles marking the beginning of the rows in the field (on the left) as well as with humans taking into account their natural inclination while walking or performing agricultural tasks. For the sake of completeness, Figure 12 shows the experimental field with respective vertical and oblique poles.

This project is funded by the European Union G.A No 101016906

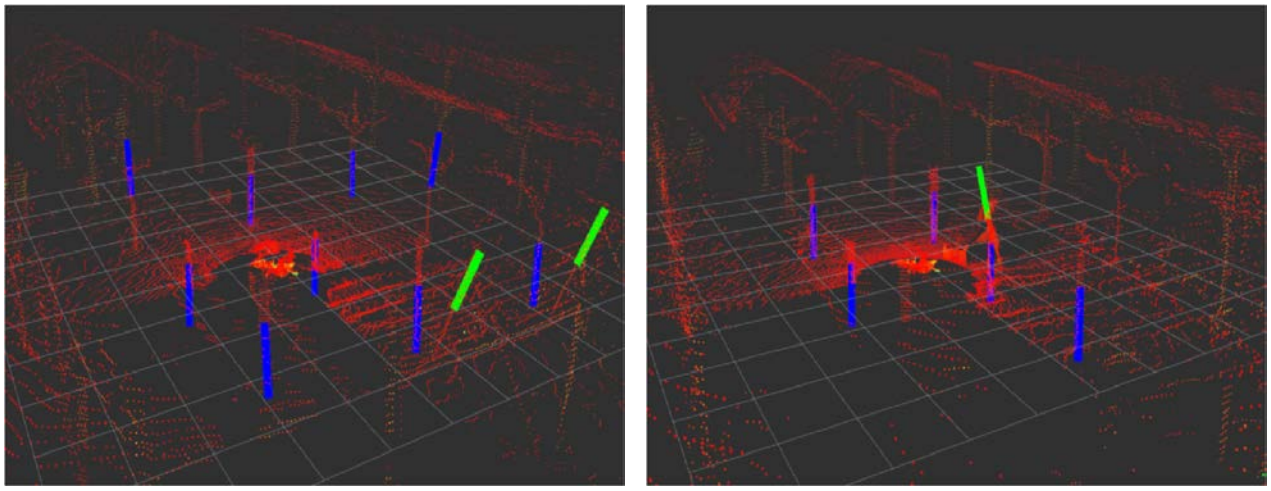


Figure 11 - Lines detection



Figure 12 - Representation of the poles in the field

4.2 Topological map

To achieve a scalable computational complexity in the SLAM, we resort to a topological map encoded by means of a graph. A directed graph $\mathcal{G} = \{V, E\}$ is composed of the set of nodes, or vertices, $V = \{v_1, \dots, v_n\}$ with cardinality $n = |V|$ and the set of edges, or arcs, $E = \{e_1, \dots, e_m\}$ with cardinality $m = |E|$. In our setting, each node is associated with a pole where a grapevine tree is attached, which we simply refer to as tree in the following, while an edge $e_k = (v_i, v_j)$ is built if the tree associated with node v_j is immediately ahead, behind or to the side of the tree associated with node v_i. Figure 13 shows an example of a topological map for the CANOPIES experimental field, where yellow circles denote the nodes and green arrows the edges.

This project is funded by the European Union G.A No 101016906



Figure 13 - Example of topological map for the CANOPIES experimental field. Yellow circles denote the nodes and green arrows the edges.

We define a path between trees i and j as the collection of edges that enables transitioning from node i to node j. The length of the path is determined by the number of such edges. We denote with $\mathcal{N}(i)^k$ the k-hop neighborhood of the node i, i.e., the set of nodes that can be reached from vertex i through a path of maximum length k, and with $|\mathcal{N}(i)^k|$ its cardinality. We additionally introduce the set $\mathcal{N}(i,j)^k = \{\mathcal{N}(i)^k \cup \mathcal{N}(j)^k\}$ representing the union between the k-hop neighborhood of trees i and j.

4.3 Methodology

The objective of the SLAM algorithm is twofold: it aims to determine in real-time the pose of the mobile base (localization), while simultaneously estimating the positions and features of the surrounding environment (mapping). In order to realize an effective SLAM algorithm for farming settings it must provide the following properties: (i) a compact representation of the map, (ii) the scalability of the computational complexity with respect to the map extent, and (iii) the georeferentiation 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 robots in farming 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 to have a unique association between the map entries and the real grapevine trees of the field for monitoring purposes. In this section, we provide a comprehensive overview of the proposed SLAM architecture including also how the required properties are guaranteed.

4.3.1 EKF for SLAM

The SLAM problem is approached within a probabilistic framework using a landmark-based Extended Kalman Filter (EKF). In this formulation, each tree in the vineyard is designated as a landmark and assigned a unique ID. This choice allows the proposed formulation to guarantee the first desired

property and maintain a compact map representation. This stands in contrast to conventional keypoint-based environmental representations, which can result in an uncontrolled expansion of map size over time.

Briefly, the EKF relies on linearizing the nonlinear dynamics around the current state estimate and then propagating the state estimate forward in time using a recursive process. Two phases are recursively executed (see (Thrun, 2002)):

- *Predict*, where the current state estimate and system dynamics are exploited to predict the next state estimate.
- *Update*, where new measurements and the respective observation model are exploited to improve the state estimate.

Two components must be defined:

1. System Transition Function (utilized in the prediction step): This function encapsulates the dynamic behavior of the system, enabling the prediction of the platform's state in the subsequent time step.
2. Observation Functions (employed in the correction step): These functions play a pivotal role in refining the estimation by incorporating sensor measurements, effectively correcting, and updating the system's state.

Let $\boldsymbol{l}_i = \left[l_i^x, l_i^y \right]^T$ be the 2D position of the ith landmark and let N represent the total number of mapped landmarks. The overall state vector which must be estimated by the EKF is $\boldsymbol{\mu} = [\boldsymbol{q}^T, \boldsymbol{m}^T]^T \in \mathcal{R}^{3+2N}$, composed of both the robot pose and the map vector $\boldsymbol{m} = [\boldsymbol{l}_1^T, \dots, \boldsymbol{l}_N^T]^T \in \mathcal{R}^{2N}$. Based on the kinematic model reported in Section 2.3, the transition function representing the prediction step for the EKF is given by

$$\begin{bmatrix} \boldsymbol{q}_{t+1} \\ \boldsymbol{m}_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + v_t cos(\theta_t)\Delta t \\ y_t + v_t sin(\theta_t)\Delta t \\ \theta_t + \omega_t \Delta t \\ \boldsymbol{m}_t \end{bmatrix},$$

where the input vector $\boldsymbol{u}$ is assumed constant for the sampling time Δt and the notation $(\cdot)_t$ denotes the value of the variable $(\cdot)$ at time t. Note the transition function only modifies the pose of the mobile base since the map $\boldsymbol{m}$ is assumed to be static.

As far as the observations are concerned, two kinds of measurements are taken into account: 1) landmark measurements, recognized as straight lines by the line-oriented perception algorithm described previously, and 2) GPS/IMU measurements, obtained through IMU and GNSS receiver. Specifically, we denote by $\boldsymbol{z}_i^L = \left[x_i^l, y_i^l \right]^T$ the measurement vector of the ith landmark given by the 2D coordinates of the sensed tree (specifically, the point obtained as the intersection of the line with the xy-plane is considered), and by $\boldsymbol{z}^G = \left[x^G, y^G, \theta^G \right]^T$ the GPS/IMU measurement, consisting of the global robot position and orientation.

This project is funded by the European Union G.A No 101016906

The respective observation models are

$$h_i^L(\boldsymbol{\mu}) = \boldsymbol{R}_W^B(\boldsymbol{l}_i - \boldsymbol{p}), \qquad \boldsymbol{C}_L \in \mathcal{R}^{2\times(3+2N)},$$
$$h^G(\boldsymbol{\mu}) = \boldsymbol{q}, \qquad \boldsymbol{C}_G \in \mathcal{R}^{3\times(3+2N)},$$

where $\boldsymbol{C}_L$ and $\boldsymbol{C}_G$ represent the observation matrices, modeling how the external measurements act on the EKF state, and how these are used in the computation of the covariance matrix (see (Thrun, 2002)). The landmark observation model $h_i^L(\boldsymbol{\mu})$ is formulated as a change of coordinates. Specifically, the ith landmark is transformed from the world coordinate frame $\Sigma_{\mathcal{W}}$ to the robot coordinate frame $\Sigma_{\mathcal{B}}$ with the rotation matrix $\boldsymbol{R}_W^B$. Differently, the GPS/IMU observation model $h^G(\boldsymbol{\mu})$ does not require any coordinate transformation since all 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. Note 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.

Prediction and Correction steps are carried out in a standard EKF manner (see (Thrun, 2002)). Despite their simplicity, the observation models reported above 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 (Bailey & Durrant-Whyte, 2006) (Cadena, et al., 2016). We tackle this problem by a dynamical partitioned update step, where the neighbors landmark to update are selected trough a k-hop proximity distance. To do so, we define the set $\mathcal{O}$ containing all the *observed* nodes at current time, i.e., visible trees, i.e., for which a measurement is associated as detailed in the next section, and the set $\mathcal{O}^k = \cup_{i\in\mathcal{O}} \mathcal{N}(i)^k$ representing the union of the k-hop neighborhoods of all visible trees. We define the partitioned state $\boldsymbol{\mu}_o$, containing the robot pose $\boldsymbol{q}$ and the position vector of all trees in $\mathcal{O}$, and the respective partitioned covariance matrix $\boldsymbol{P}_o$. This partitioning process leverages the topological map formulation, selecting a subset of trees based on the k-hop neighborhood for use in the partitioned update step. The observation models are modified as follows

$$h_i^L(\boldsymbol{\mu}_o) = \boldsymbol{R}_W^B(\boldsymbol{l}_i - \boldsymbol{p}), \qquad \boldsymbol{C}_L \in R^{2\times\left(3+2\left|\mathcal{O}^k\right|\right)}$$
$$h^G(\boldsymbol{\mu}_o) = \boldsymbol{q}, \qquad \boldsymbol{C}_G \in R^{3\times(3+2|\mathcal{O}^k|)}$$

The above observation models are the core of the proposed SLAM architecture. They focus on specific portions of the overall state $\boldsymbol{\mu}$, rather than the entire state. This strategic approach significantly reduces the computational load during the correction step, ensuring real-time performance, as the decoupling of the graph size from the correction step calculation substantially lowers the algorithm's complexity. Indeed, the increase in graph size only impacts the data association and neighborhood calculation, which have significantly lower costs compared to the correction step. Consequently, the proposed EKF architecture is able to scale with the size of the field, meeting the second requirement for SLAM in farming environments. This balance of

computational efficiency and accuracy is essential for practical implementation in agricultural settings.

To carry out the EKF update step, we rely on the idea of the Schmidt partial update (Brink, 2017). Specifically, a coordinate transformation by means of a transformation matrix $\boldsymbol{T}$, namely a permutation matrix, is dynamically performed to transform the state vector $\boldsymbol{\mu}$ into a state vector $\hat{\boldsymbol{\mu}}$ composed of two blocks: one collecting the components to update $\boldsymbol{\mu_o}$ and one collecting the remaining components, denoted as $\boldsymbol{\mu}_n$, that is

$$\boldsymbol{\mu} = [\boldsymbol{q}^T \quad \boldsymbol{l}_1^T \quad \cdots \quad \boldsymbol{l}_k^T \quad \cdots \boldsymbol{l}_m^T \quad \cdots \quad \boldsymbol{l}_N^T]^T, \quad \hat{\boldsymbol{\mu}} = \boldsymbol{T}\boldsymbol{\mu} = [\boldsymbol{\mu}_o^T \quad \boldsymbol{\mu}_n^T]^T$$

Similarly, the covariance matrix $\boldsymbol{P}$ is re-arranged in a block-wise manner. To associate new LiDAR measurements to the landmarks in the map, we make use of a data association process discussed in the next section.

4.3.2 Data Association and Planting Pattern Estimation

Data association is the critical process of discerning which of the noisy measurements, obtained from the perception detailed in Section 4.1, align with pre-existing landmarks in the vineyard map. When the perceived objects do not align with any known landmarks in the map, the data association process must be able to recognize if they are associated with new landmarks or if they represent outliers, such as people, boxes, or oblique poles unrelated to the map.

Figure 14 shows the steps of the designed data association procedure, where measurements are depicted with red circles, landmarks with trees, the robot with a yellow rectangle and possible association with thin blue lines. As likelihood measure, we consider the Euclidean distance between measurements and landmarks.

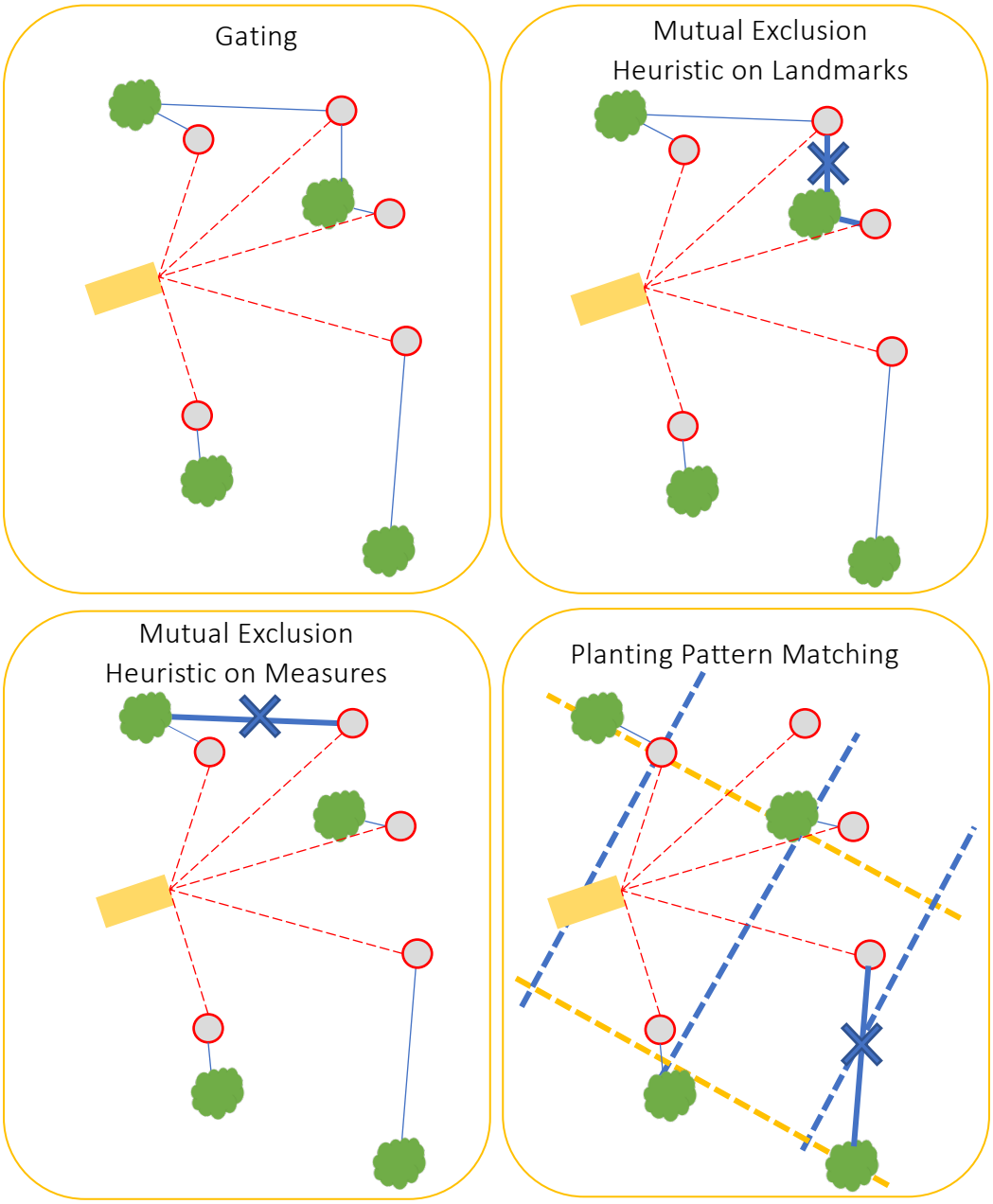


Figure 14 - Data association procedure

Whenever new measurements become available, a *gating* operation is executed (top left of the figure). This operation discerns, for each landmark, which measurements hold the potential to be valid candidates for association based on their Euclidean distance to the landmark. By doing so, we diminish the number of unnecessary computations during the association process.

At this point a maximum likelihood heuristic is applied for both landmarks and measurements. Specifically, this heuristic considers the exclusivity of associations: for any given landmark, only one measurement can be assigned (as depicted in the top right of the figure), and conversely, for any given measurement, only one landmark can be assigned (as shown in the bottom left of the figure).

This approach helps ensure unambiguous associations between measurements and landmarks, enhancing the accuracy of the mapping process.

In order to enhance the robustness of the data association and the outlier rejection, we estimate the field plating pattern, i.e., the geometric arrangement of the plants across the field, and utilize it to refine the association of measurements with landmarks. We refer to this refinement process as *planting pattern matching*.

The planting pattern estimation problem is modeled as a grid rectangular fitting problem. Specifically, we define the planting pattern state as $\boldsymbol{s} = [\,\boldsymbol{c}^T, \phi\,]^T$, where the vector $\boldsymbol{c} = \left[\,c_x, c_y\,\right]^T$ collects the scale factors along the two major grid directions and ϕ is the rotation angle which aligns the grid with the geo-referenced map estimated by the EKF-SLAM. A representation of the planting pattern variables is provided in Figure 15.

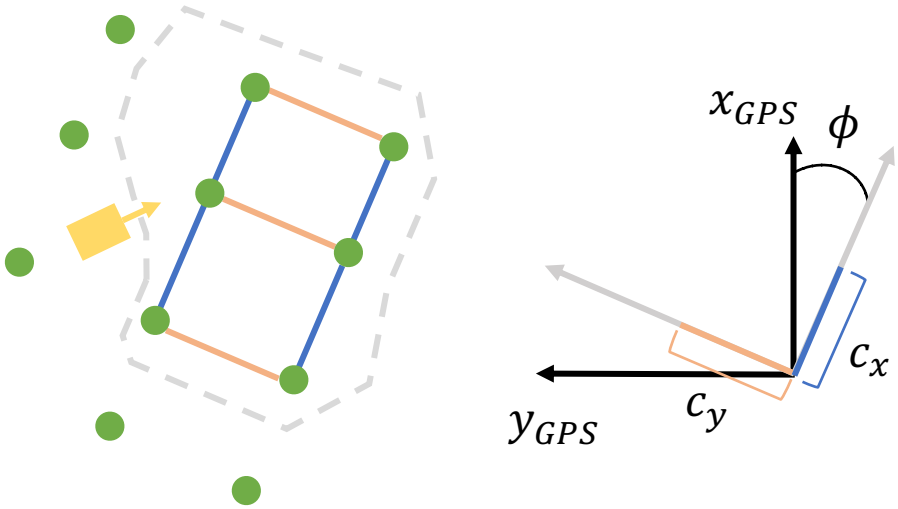


Figure 15 - Representation of the planting pattern state

The fitting process is carried out by formulating a Kalman Filter (KF). Specifically, for the transition model we consider that the planting pattern is constant over time, i.e., it holds

$$\boldsymbol{s}_{t+1} = f_s(\boldsymbol{s}_t) = \boldsymbol{s}_t,$$

while for the observation model we consider that a virtual sensor measures the planting pattern by relying on the k-hop neighborhood of the landmarks, i.e.,

$$h^S(\boldsymbol{s}_t) = \boldsymbol{z}^S_{O,t} - \boldsymbol{s}_{t+1}.$$

Utilizing the estimated planting pattern, we implement an outlier rejection process (depicted in the bottom right of Figure 14). For each tree, we define an obstacle space wherein all LIDAR measurements falling within this space are likely to be attributed to external objects. The union of these obstacle spaces is depicted in red in the representation in Figure 16.

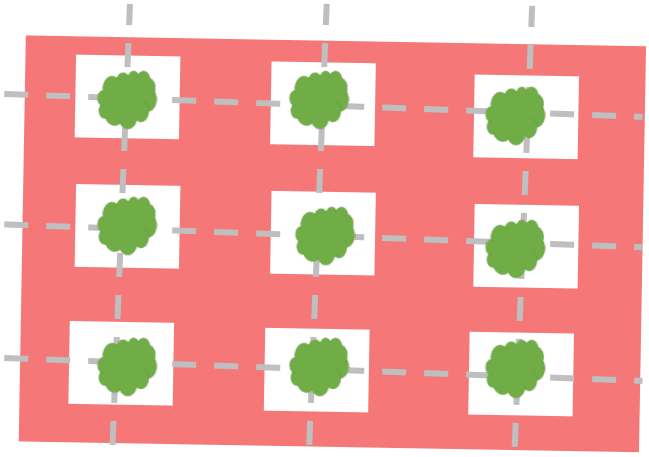


Figure 16 - Outlier filtering based on planting pattern matching

Any measurements identified as external objects are excluded and ignored for the update step of the EKF.

4.3.3 Resulting SLAM Architecture

The resulting SLAM architecture is reported in Figure 17.

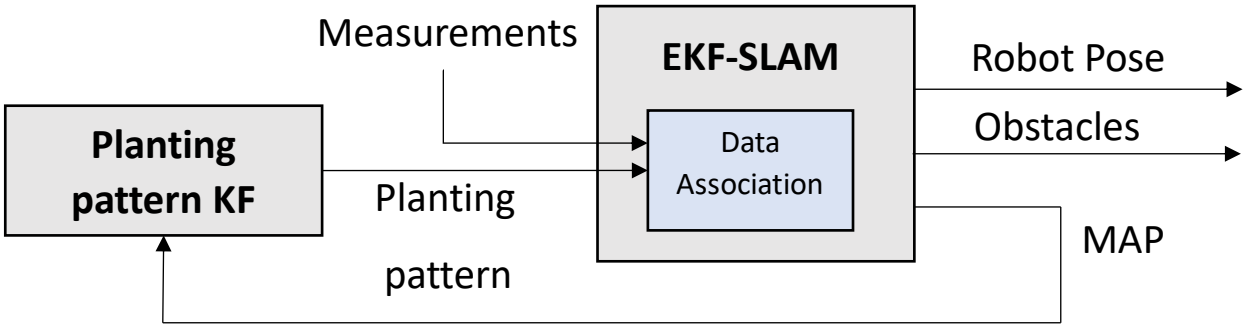


Figure 17 - SLAM architecture

In summary, the SLAM module provides three key outputs:

- Robot Estimated Pose: This represents the calculated position and orientation of the robot within the environment.
- Map: This provides a representation of the trees (representing fixed obstacles of the robot) within the robot's operating space.
- General obstacles: These encompass measurements that were not successfully linked with specific landmarks. They are viewed as generic, potentially dynamic obstacles for the robot and exploited by the low-level navigation algorithm.

Regarding the general obstacles, they can manifest as vertical lines, undergoing filtration by the data association procedure, or as oblique lines (which are not factored into the data association process). For the oblique lines, the output includes both the upper and lower points that belong to each line. Additionally, if the distance between these first two points exceeds a defined threshold, the average

point is also provided. This is motivated by the fact that the subsequent dynamic planner system works in the bi-dimensional xy-plane, and therefore does not consider the height. Therefore, multiple points are necessary to consider the dimensions of the robot's trunk: in fact, if only the point at the base of the oblique line is considered, obstacle avoidance could be not totally guaranteed.

In scenarios where multiple robots coexist in the environment, the poses of these other robots are communicated through Wi-Fi communication and are also included as part of the SLAM output data. Indeed, by incorporating the poses of these additional robots, they can be effectively treated as obstacles for the subsequent navigation algorithms. It is important to highlight that LIDAR measurements identified as pertaining to the bodies of these other robots are deliberately filtered out to prevent any potential ambiguities.

4.4 Experimental Validation

The experiments were conducted at the CANOPIES experimental field located at "Società Cooperativa Agricola Corsira" in the municipality of Aprilia, Lazio, Italy. The table-grape vineyard in question follows a planting pattern of $2.9m \times 2.9\ m$. While the plants adhere to this pattern, some positioning discrepancies are observed, resulting in slight deviations from the ideal layout. The majority of experiments were conducted between the end of July 2023 and the beginning of August 2023, unless stated otherwise. In this period, the vineyard was characterized by prolific vegetation state, featuring dense foliage and bunches as shown in Figure 18, where the areas with Black Magic (left) and White Pizzutello (right) grape varieties are depicted.



Figure 18 - Examples of the operating scenario. The areas with Black Magic (left) and White Pizzutello (right) grape varieties are shown

It is worth noting that these operating conditions are very challenging for the utilization of GPS data in the correction step. The dense vegetation obstructs the consistent reception of accurate GPS signals, leading to instances where reliable positioning through GPS data is impeded. To mitigate this issue, we exploit the GPS measurement covariance matrix in the respective correction step and ignore the measurement when the covariance matrix norm is below a specific threshold.

This project is funded by the European Union G.A No 101016906

The obstacle detection module, the EKF SLAM and the planting pattern KF have been coded in C++, and a ROS node has been realized for each.

To evaluate the effectiveness of the SLAM algorithm we resorted to the benchmarks defined in Deliverable 2.3 "Dimensioning of the Real-World Scenario for Final Demo" which are the following for the robot localization and mapping:

- Identification of all landmarks (vertical pales with the tree trunks tied up to them) in the vineyard with an accuracy of 15 cm.
- Identification of the row orientation with an accuracy of 20°.
- Relative localization to the closest landmarks with an accuracy of 15 cm regarding the position and 20° regarding the orientation.

Additionally, we verified the closed loop property of the SLAM algorithm and evaluated the localization error when returning to the same initial pose. Note that a detailed validation of the perception component can be found in Deliverable 7.1.

4.4.1 Identification of all landmarks

In order to assess the effectiveness of landmark identification, we teleoperated the mobile base through the experimental field, focusing on the left portion illustrated in Figure 3, which is commonly employed for experiments. Specifically, a remote joystick interface was controlled by a human operator while the SLAM algorithm was running on the robotic platform. The latter effectively recognized and mapped all trees in the environment without introducing any erroneous landmarks, even in the presence of moving human operators within the field. An illustration of the resulting map is provided in Figure 19, Representation in Rviz of the overall map, where the spheres identify landmarks and the orange boxes denote obstacles (specifically, there were human operators in the field). Note that some landmarks are missing in the right part as the field was not accessible as shown in Figure 20. The estimated grid distances were $\boldsymbol{c} = [2.8697, 2.899]^T$, which are coherent with the real measurements of the trees within the field.

This project is funded by the European Union
G.A No 101016906

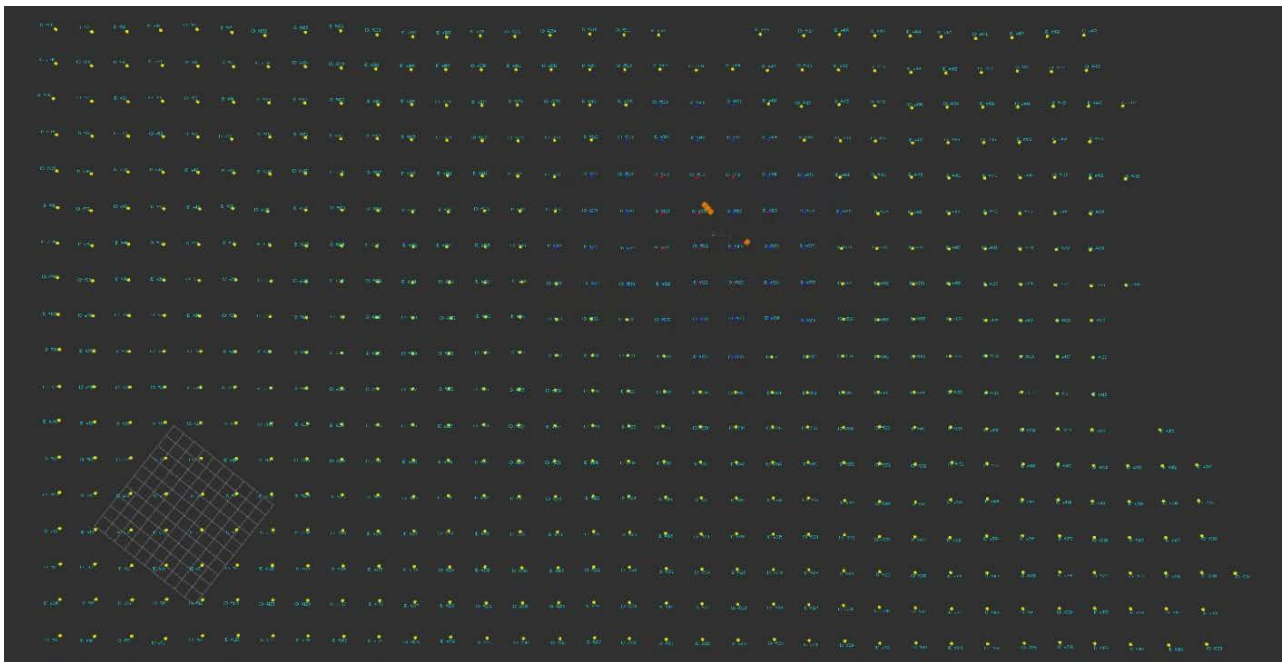


Figure 19 - Representation in Rviz of the overall map, where circles identify landmarks and the orange boxes denote obstacles (specifically, there were human operators in the field)



Figure 20 - Operating conditions in the unmapped part of the field.

For a more comprehensive assessment of the accuracy of landmark positioning, we chose to focus on three complete rows, consisting of 76 landmarks in total. In this evaluation, we manually measured the distances between the trees using a measuring tape. These manual measurements were then compared with the estimated distances obtained from the SLAM module. An overview of the portion of the map utilized in this experiment is provided in Figure 21.

This project is funded by the European Union G.A No 101016906

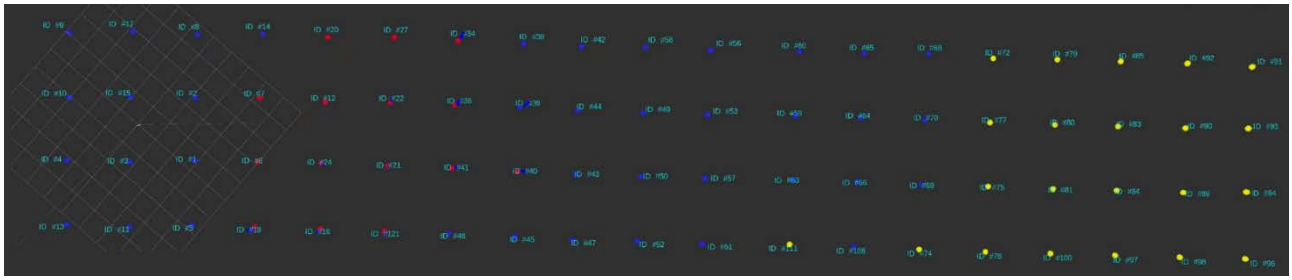


Figure 21 - Portion of the map used to compute the accuracy of the landmark identification

In this regard, a first challenge that arose was how to accurately measure such distances. Indeed, in light of given our line-oriented perception algorithm, each landmark is not directly associated with a single physical object of the vineyard, but, as shown in yellow in Figure 22, the detected point approximately coincides with the centroid among the pole and the trunk points projected in the xy-plane.



Figure 22 - Representation of the approximate landmark point

Accessing such a point for measuring distances between landmarks may not always be feasible. Therefore, we defined a procedure to derive the measurements of interest without the need for direct access to the landmark point. More in detail, let us consider each row divided in blocks with four trees positioned at the corners and let us denote with TL, TR, BR, and BL the top left, top right, bottom right and bottom left landmarks. Our objective is to compute, for each block, the four distances that form the boundaries of the block denoted as $d_{TL,TR}, d_{TR,BR,}, d_{BL,BR}, d_{TL,BL}$ as well as the two diagonals, denoted as $d_{TL,BR}, d_{TR,BL,}$. A top view schematic representation is provided in Figure 23, where for each landmark we report the respective structure in the field composed of trunk and pole.

This project is funded by the European Union G.A No 101016906

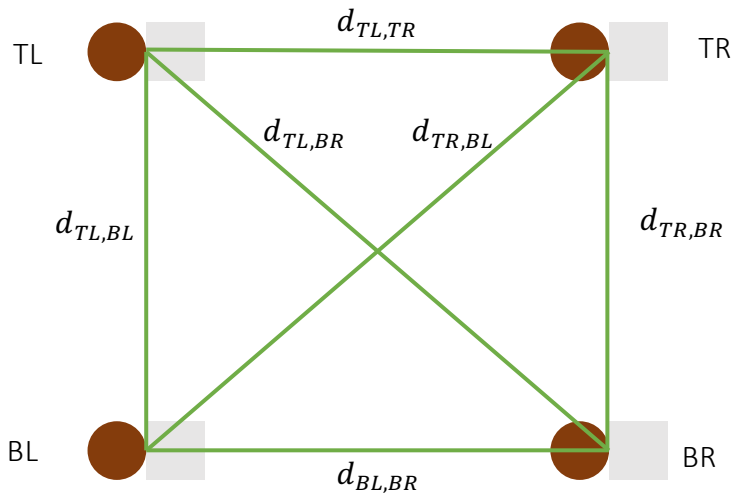


Figure 23 - Desired measures for validating the landmark identification

Note that, as mentioned previously, in general the resulting rectangle enclosing the block is not as precise as in the picture, but misalignments can be observed in the real-world scenario.

To derive the desired distances, we measure the quantities indicated by the dotted blue lines in Figure 24, denoted as $m^*_{(\cdot)}$ and $m_{(\cdot)}$, and for each pole we measured half of its pole width (marked in orange in the zoomed-in view on the right) which we denote as w_{ID} with $ID = TL, TR, BR, BL$. All these measurements are directly accessible with no obstructions. Additionally, we require the knowledge of the angles depicted in orange in the zoomed-in view and denoted as α_{ID} with $ID = TL, TR, BR, BL$. Given the relatively uniform layout of the field, we assume the angle to be constant over the field and equal to $\alpha = 2.356$.

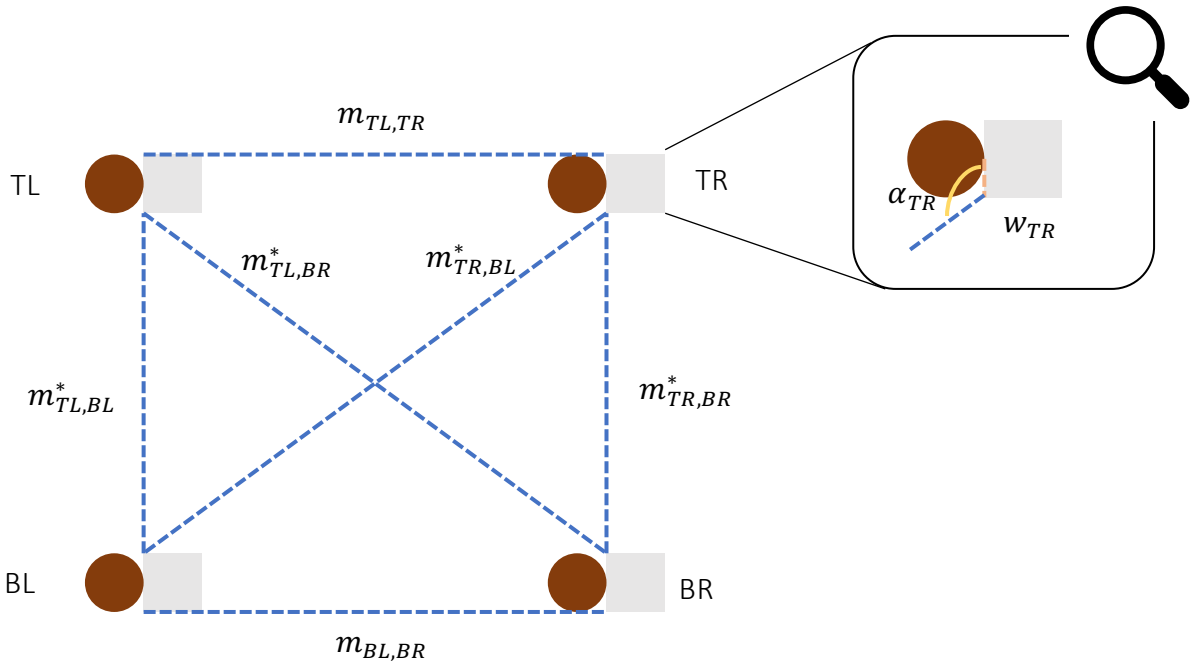


Figure 24 – Actual measurements for validating the landmark identification

By considering the cosine rule, the relation between the measurements and the desired distances is given by:

$$d_{BL,BR} = m_{BL,BR}$$
$$d_{TL,TR} = m_{TL,TR}$$
$$d_{TL,BL} = m^*_{TL,BL} + w_{TL} + w_{BL}$$
$$d_{TR,BR} = m^*_{TR,BR} + w_{TR} + w_{BR}$$
$$d_{TL,BR} = \sqrt{w_{TL}^2 + \frac{m^{*2}_{TL,BL}}{4} - w_{TL} m^*_{TL,BL} \cos(\alpha)} + \sqrt{w_{BL}^2 + \frac{m^{*2}_{TL,BL}}{4} - w_{BL} m^*_{TL,BL} \cos(\alpha)}$$
$$d_{TR,BL} = \sqrt{w_{TR}^2 + \frac{m^{*2}_{TR,BL}}{4} - w_{TR} m^*_{TR,BL} \cos(\alpha)} + \sqrt{w_{BL}^2 + \frac{m^{*2}_{TR,BL}}{4} - w_{BL} m^*_{TR,BL} \cos(\alpha)}$$

Therefore, we collected the field measurements by applying the above procedure to the selected rows and computed the errors between the measured distances and the estimated ones, i.e., for each measurement $d_{ID1,ID2}$ the landmark positioning error is obtained as

$$e^L_{ID1,ID2} = \left\| d_{ID1,ID2} - \hat{d}_{ID1,ID2} \right\|,$$

With $ID1, ID2 \in \{TL, TR, BR, BL\}$ and where $\hat{d}_{(\cdot)}$ denotes the estimate of $d_{(\cdot)}$ according to the SLAM data. Table 1 reports the results on the landmark positioning error. Specifically, when considering all landmarks (first row), we can observe an average error equal to 0.0470 m with standard deviation equal to 0.0389 m and minimum maximum and maximum errors equal to 0.0002 m and 0.1785 m,

respectively, thus confirming the effectiveness of the proposed SLAM architecture (which highly satisfies the benchmark). However, since we noticed that the poles positioned at the beginning and at the end of each row are highly irregular compared to the others, we also evaluated the landmark positioning errors by excluding these landmarks (as reported in the second row of the table). In this case, we can notice that, as expected, all the error measures become even smaller, reaching average value equal to 0.0429 m and number of samples with error greater than 0.1 m equal to 17 (decreased by 8 with respect to the case with all landmarks).

Table 1 - Results for the accuracy related to the landmark identification

Landmarks	Total samples	Mean [m]	Median [m]	Std. dev [m]	Min. [m]	Max. [m]	# samples $e^L > 0.1$m
All	216	0.0470	0.0354	0.0389	0.0002	0.1785	25
No external	192	0.0429	0.0335	0.0348	0.0002	0.1523	17

4.4.2 Identification of the row orientation

To measure the row orientation and then compare it with the one estimated by the planting pattern KF, we used the RTK system with "RTK fixed" status to obtained reliable positioning measures within the field. Specifically, we considered three rows, one in the bottom part, one in the middle and one in the top part of the field, and for each row collected two GPS measures, one at the beginning of the row and one at the end of it. The UTM coordinates of the collected points are reported in Figure 25.

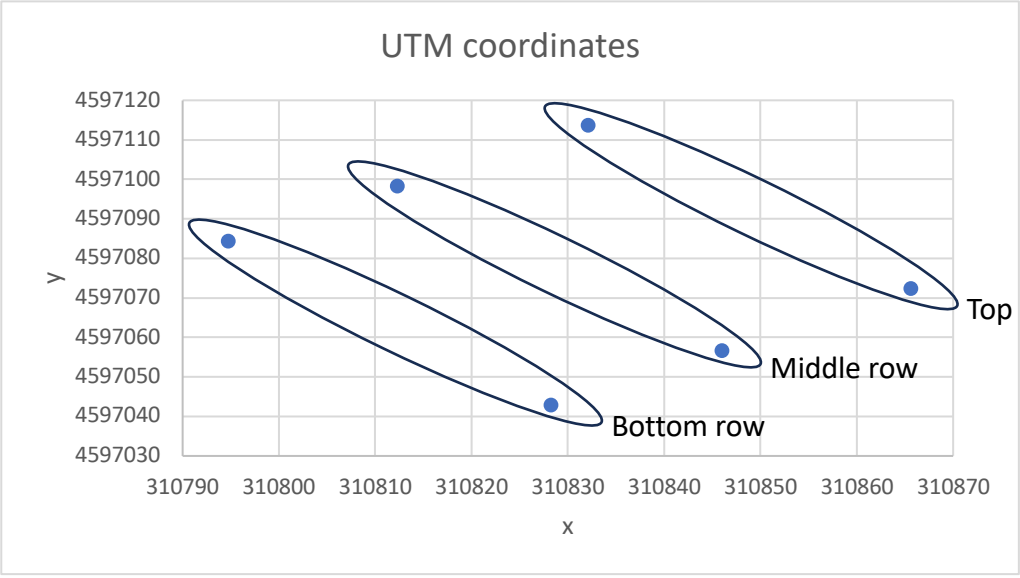


Figure 25 - Points collected to evaluate the estimation of the row orientation

Then, we evaluated for each row the difference between the point at the beginning of the row and the one at the end, obtaining the vectors shown in Figure 26.

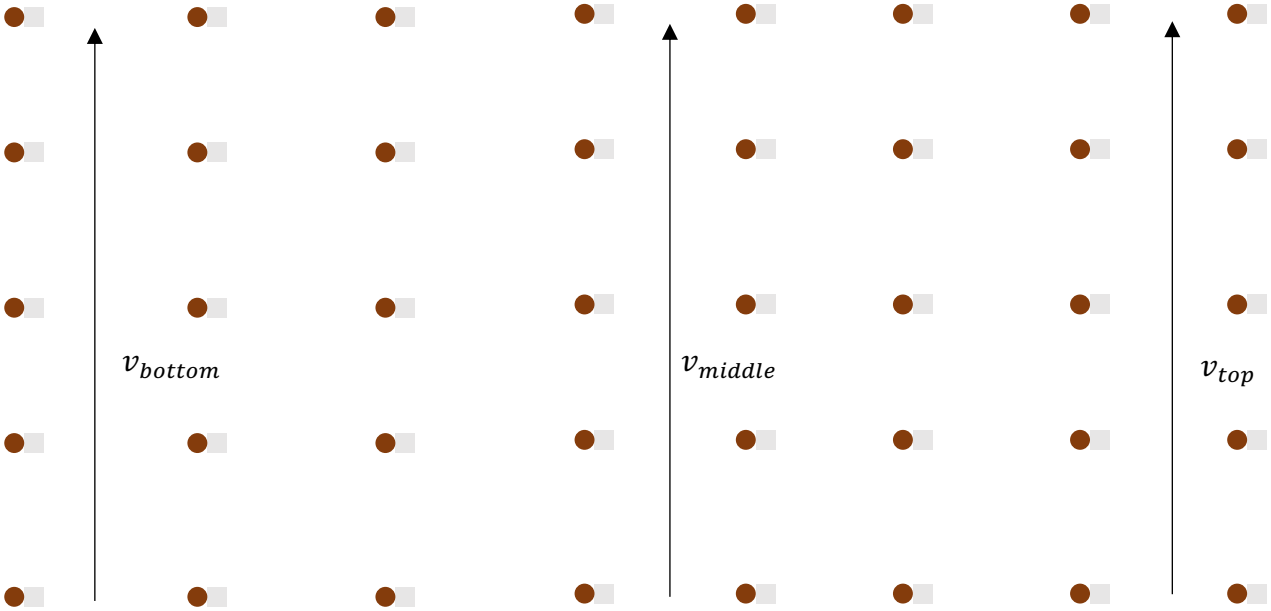


Figure 26 - Vectors utilized to evaluate the estimation of the row orientation

Subsequently, the angle with respect to the x axis of the global reference frame was computed as $\phi_r^o = atan2(v_{r,y}, v_{r,x})$ for each vector v_r, with $r = bottom, middle, top$. Table 2 reports the experimental results. Specifically, the three angle measurements are reported (first three columns), leading to the average angle measurement -0.8902. This value is compared with the estimated angle equal to -0.8817, yielding an average error of 0.008 (that is $0.4870°$). This result aligns perfectly with the benchmark requirements and validate the proposed KF. It is worth mentioning that we opted not to use a compass for this evaluation, as measurements from this type of sensor are more susceptible to inaccuracies.

Table 2 - Results for evaluating the row orientation error

ϕ_{bottom}^o[rad]	ϕ_{middle}^o[rad]		ϕ_{top}^o[rad]	Comparison
-0.8910	-0.8909	-0.8887	**Avg:** -0.8902	
			Estimate: -0.8817	
			Error: 0.008	

4.4.3 Relative localization to the closest landmarks

To assess the relative localization to the closest landmarks, we conducted experiments where the robot was tasked with autonomously navigating from a starting point to a goal one and intermediate measurements along the trajectory were collected. Specifically, we considered four trajectories and stopped the robot three times while following the trajectory, for a total of 20 points (including start and goal) collected for evaluating the accuracy. Note that this choice of the experiments allowed us to simultaneously validate the benchmark related to relative localization to

This project is funded by the European Union G.A No 101016906

the closest landmark and the benchmark related to the localization capabilities during navigation (defined in Deliverable 2.3). Specifically, the latter benchmark states “Capability of safely navigating from one row to another one by doing the related maneuvering operations without losing its localization capabilities within the map, e.g., by keeping its localization accuracy within 15 cm regarding the position and 20° regarding the orientation along the trajectory.” An overview of the points considered as start and goal is provided in Figure 27, while the pictures collected in the field and associated with all five points of the four trajectories are reported in

Table 3 (trajectories 1-2) and Table 4 (trajectories 3-4).

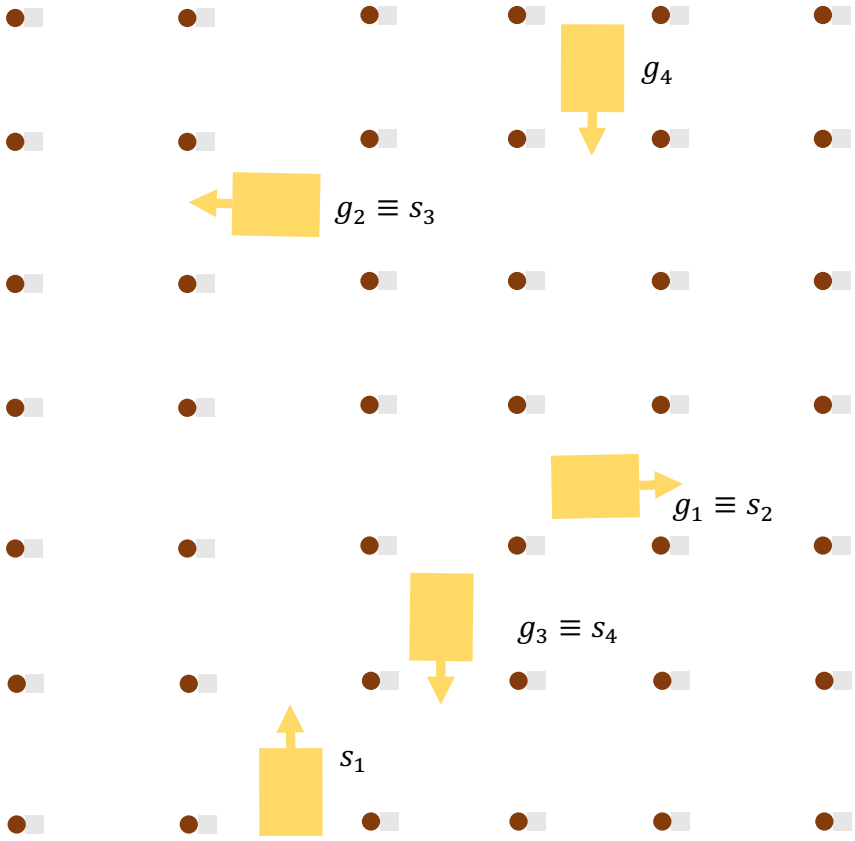


Figure 27 - Overview of the start and goal points considered for testing the landmark relative localization

This project is funded by the European Union G.A No 101016906

Table 3 – Representation of the considered points for the first two test trajectories

	Trajectory 1	Trajectory 2
P1 - Start		
P2		
P3		

This project is funded by the European Union G.A No 101016906

P4		
P5 - goal		

Table 4 - Representation of the considered points for the last two test trajectories

	Trajectory 3	Trajectory 4
P1 - Start		

This project is funded by the European Union G.A No 101016906

P2		
P3		
P4		
P5 - goal		

This project is funded by the European Union G.A No 101016906

To assess the performance of the relative localization, we need the measurement of the distances to the four closest landmarks, denoted as $d_{b,ID}$ with $ID = TL, TR, BR, BL$, and the relative angles, denoted as $\beta_{b,ID}$ with $ID = TL, TR, BR, BL$ and computed as minimum angle between the robot orientation vector and the vector from the robot representative point to the landmark of interest. A representation of the measurements of interest (highlighted in green color) is provided in Figure 28.

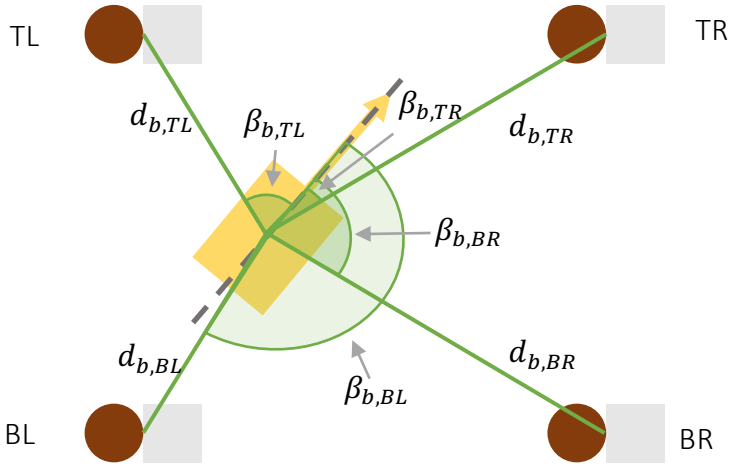


Figure 28 – Representation of the measurements needed to evaluate the accuracy of the relative localization

To obtain angular measurements without relying on the compass, we implemented an additional data collection procedure. This choice was driven by the need to avoid using the compass due to significant accuracy issues stemming from electric fields associated with the soil composition, the cable of the framework supporting the irrigation system, and the presence of the robot. These factors have a substantial impact on measurement repeatability, resulting in oscillations that can exceed 20 degrees of variation. All the measured quantities are indicated in Figure 29 by dotted blue lines.

Specifically, let us define with b and f, two points on the mobile base, that are respectively the origin of the reference frame attached to the robot body, and a point along its sagittal axis. We measured the distances from the robot representative point b to the four landmarks, denoted as $m_{b,ID}$ with $ID = TL, TR, BR, BL$, and from an additional point on the front of the mobile base f to the landmarks, denoted as $m_{f,ID}$ with $ID = TL, TR, BR, BL$. Finally, we measured the distances between the landmarks (excluding the diagonal measurements), denoted as $m_{ID1,ID2}$ with $ID1, ID2 \in \{TL, TR, BR, BL\}$. Note that when collecting these measurements, we approximated the landmark point and did not strictly adhere to the procedure outlined in Section 4.4.1. This simplification was adopted to streamline the data collection procedure, considering the high number of measurements required for each point along the trajectories.

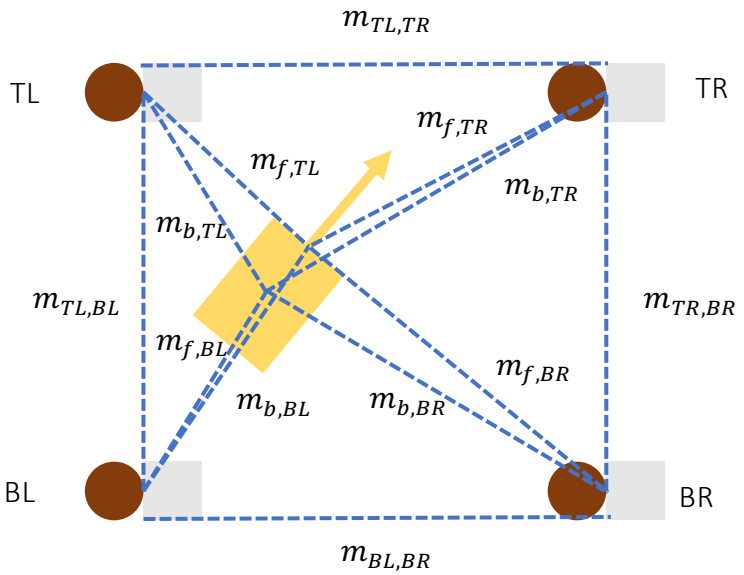


Figure 29 - Representation of the actual measurements collected to evaluate the accuracy of the relative localization

The relation between the measurements and the desired distances is given by:

$$d_{b,BR} = m_{b,BR}$$
$$d_{b,TR} = m_{b,TR}$$
$$d_{b,BL} = m_{b,BL}$$
$$d_{b,BR} = m_{BR}$$

To calculate $\beta_{b,ID}$ with $ID = \{TL, TR, BR, BL\}$, let us denote the following angles:

$$\gamma_{ID1,ID2,ID3} = \text{acos}\left(\frac{{m_{ID1,ID2}}^2 + {m_{ID3,ID1}}^2 - {m_{ID2,ID3}}^2}{(2m_{ID1,ID2}m_{ID3,ID1})}\right)$$

with $ID1, ID2, ID3 \in \{TL, TR, BR, BL, b, f\}$. For the sake of clarity, it should be noticed that the following relationship holds:

$$m_{ID1,ID2} = m_{ID2,ID1}$$

with $ID1, ID2 \in \{TL, TR, BR, BL, b, f\}$.

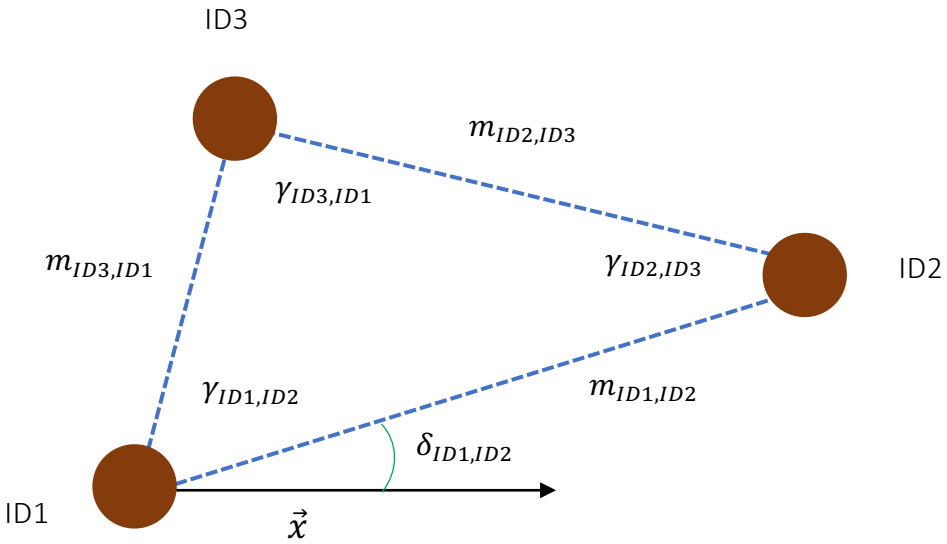


Figure 30 - Relation between angles and position of landmarks, robot, and its front part

In order to compare all estimations regarding the robot position and its front part, it was decided to compute the average of the calculated measurements, assuming that at least one previously computed side remains fixed for the subsequent calculation. In other words, these positions of the robot and its front part are reconstructed four times in the Cartesian coordinate system, considering the four triangles formed as depicted in Figure 29.

The coordinates are calculated using the following equations:

$$\begin{bmatrix} ID1_x \\ ID1_y \end{bmatrix} = \begin{bmatrix} ID1_x \\ ID1_y \end{bmatrix}$$

$$\begin{bmatrix} ID2_x \\ ID2_y \end{bmatrix} = \begin{bmatrix} ID1_x \\ ID1_y \end{bmatrix} + R_{ID1,ID2} \begin{bmatrix} m_{ID1,ID2} \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} ID3_x \\ ID3_y \end{bmatrix} = \begin{bmatrix} ID1_x \\ ID1_y \end{bmatrix} + R_{ID1,ID2}\, V_{D1,ID2,ID3}$$

with

$$R_{ID1,ID2} = \begin{bmatrix} \cos(\delta_{ID1,ID2}) & -\sin(\delta_{ID1,ID2}) \\ \sin(\delta_{ID1,ID2}) & \cos(\delta_{ID1,ID2}) \end{bmatrix}$$

$$V_{D1,ID2,ID3} = \begin{bmatrix} M_{ID1,ID2,ID3} \\ \sqrt{m_{ID3,ID1}^2 - M_{ID1,ID2,ID3}^2} \end{bmatrix}$$

$$M_{ID1,ID2,ID3} = \frac{(m_{ID1,ID2}^2 + m_{ID3,ID1}^2 - m_{ID2,ID3}^2)}{2m_{ID1,ID2}}$$

This project is funded by the European Union G.A No 101016906

where $ID1, ID2, ID3 \in \{TL, TR, BR, BL, b, f\}$ and $\delta_{ID1,ID2}$ is the angle of offset between the axis defined by $ID1$ and $ID2$ and a desired reference axis $\vec{x}$ as shown in Figure 30.

Using the previous equations, we can define the following function $F(\cdot)$, which can be iteratively used to calculate all the relevant points, considering the concatenation of measurements:

$$\begin{bmatrix} p_{ID2_i} \\ p_{ID3_i} \end{bmatrix} = F(ID1, ID2, ID3, ID1_j, \delta_{ID1,ID2}) = \begin{bmatrix} \begin{bmatrix} p_{ID1_x,j} \\ p_{ID1_y,j} \end{bmatrix} + R_{ID1,ID2} \begin{bmatrix} m_{ID1,ID2} \\ 0 \end{bmatrix} \\ \begin{bmatrix} p_{ID1_x,j} \\ p_{ID1_y,j} \end{bmatrix} + R_{ID1,ID2}\, V_{D1,ID2,ID3} \end{bmatrix}$$

Where $p_{ID2_i} = \begin{bmatrix} p_{ID2_x,j} \\ p_{ID2_y,j} \end{bmatrix}$ and $p_{ID3_i} = \begin{bmatrix} p_{ID3_x,j} \\ p_{ID3_y,j} \end{bmatrix}$ are the cartesian coordinates of the two points to be computed, $ID1, ID2, ID3 \in \{TL, TR, BR, BL, b, f\}$, $\delta_{ID1,ID2}$ is the angle of offset between the axis defined by $ID1$ and $ID2$ and the axis defined by BL and BR (set as desired reference axis $\vec{x}$), $ID1_j$ are the coordinates of $ID1$ obtained by the previous j computation, and $j \in \{1,2,3,4\}$ is the id of one of the measurements. Note that, for the sake of simplicity, in the following we use $j = 1$; alternatively, we could utilize the average of the measurements already acquired.

To calculate all the measurements, we will consider the following 8 computations based on the definition of the function $F(\cdot)$ given above, considering the concatenation of previously acquired measurements:

$$\begin{bmatrix} p_{BR_1} \\ p_{b_1} \end{bmatrix} = F(BL, BR, b, BL_1, \delta_{BL,BR})$$

$$\begin{bmatrix} p_{BR_2} \\ p_{f_1} \end{bmatrix} = F(BL, BR, f, BL_1, \delta_{BL,BR})$$

$$\begin{bmatrix} p_{b_2} \\ p_{TL_1} \end{bmatrix} = F(BL, b, TL, BL_1, \delta_{BL,b})$$

$$\begin{bmatrix} p_{TR_1} \\ p_{b_3} \end{bmatrix} = F(BR, TR, b, BR_1, \delta_{BR,TR})$$

$$\begin{bmatrix} p_{f_2} \\ p_{TL_2} \end{bmatrix} = F(BL, f, TL, BL_1, \delta_{BL,f})$$

$$\begin{bmatrix} p_{TR_2} \\ p_{f_3} \end{bmatrix} = F(BR, TR, f, BR_1, \delta_{BR,TR})$$

$$\begin{bmatrix} p_{b_4} \\ p_{TR_3} \end{bmatrix} = F(TL, b, TR, TL_1, \delta_{TL,b})$$

$$\begin{bmatrix} p_{f_4} \\ p_{TR_4} \end{bmatrix} = F(TL, f, TR, TL_1, \delta_{TL,f})$$

with

$$\delta_{BL,BR} = 0$$

This project is funded by the European Union G.A No 101016906

$$\delta_{BL,BR} = 0$$

$$\delta_{BL,b} = \gamma_{BL,BR,b}$$

$$\delta_{BR,TR} = \pi - (\gamma_{BR,b,BL} + \gamma_{BR,TR,b})$$

$$\delta_{BL,f} = \gamma_{BL,BR,f}$$

$$\delta_{BR,TR} = \pi - (\gamma_{BR,f,BL} + \gamma_{BR,TR,f})$$

$$\delta_{TL,b} = \gamma_{BL,BR,b} + \gamma_{BL,b,TL} + \gamma_{TL,BL,b} - \pi$$

$$\delta_{TL,f} = \gamma_{BL,BR,f} + \gamma_{BL,f,TL} + \gamma_{TL,BL,f} - \pi$$

To obtain the estimated coordinates $\widehat{p_{ID}}$ of each ID, we have the following formula:

$$\widehat{p_{ID}} = \frac{\sum_{i=1}^{n} p_{ID_i}}{n}$$

with $ID \in \{TL, TR, BR, BL, b, f\}$, $i \in \{1,2,3,4\}$ and n the number of computations related to ID.

Finally, we obtain the measurements with the following equations:

$$\beta_{b,BR} = atan2\left(\widehat{p_{BR_y}} - \widehat{p_{b_y}}, \widehat{p_{BR_x}} - \widehat{p_{b_x}}\right) - atan2(\widehat{p_{f_y}} - \widehat{p_{b_y}}, \widehat{p_{f_x}} - \widehat{p_{b_x}})$$
$$\beta_{b,TR} = atan2\left(\widehat{p_{TR_y}} - \widehat{p_{b_y}}, \widehat{p_{TR_x}} - \widehat{p_{b_x}}\right) - atan2(\widehat{p_{f_y}} - \widehat{p_{b_y}}, \widehat{p_{f_x}} - \widehat{p_{b_x}})$$
$$\beta_{b,BL} = atan2\left(\widehat{p_{BL_y}} - \widehat{p_{b_y}}, \widehat{p_{BL_x}} - \widehat{p_{b_x}}\right) - atan2(\widehat{p_{f_y}} - \widehat{p_{b_y}}, \widehat{p_{f_x}} - \widehat{p_{b_x}})$$
$$\beta_{b,TL} = atan2\left(\widehat{p_{TL_y}} - \widehat{p_{b_y}}, \widehat{p_{TL_x}} - \widehat{p_{b_x}}\right) - atan2(\widehat{p_{f_y}} - \widehat{p_{b_y}}, \widehat{p_{f_x}} - \widehat{p_{b_x}})$$

Based on the above formulas, we computed the errors between the measured distances and the estimated ones as well as the errors between the measured relative angles and the estimated ones. More in detail, for each measurement $d_{b,ID}$ the relative landmark positioning error is obtained as

$$e^{L}_{b,ID} = \left\| d_{b,ID} - \hat{d}_{b,ID} \right\|,$$

with $ID = TL, TR, BR, BL$ and where $\hat{d}_{(\cdot)}$ denotes the estimate of $d_{(\cdot)}$ according to the SLAM data, while the relative landmark orientation errors are computed as

$$\bar{e}^{L}_{b,ID} = \left| \, |\beta|_{b,ID} - \left|\hat{\beta}\right|_{b,ID} \, \right|,$$

with $ID = TL, TR, BR, BL$ and where $\hat{\beta}_{(\cdot)}$ denotes the estimate of $\beta_{(\cdot)}$ according to the SLAM data.

Table 5 reports the results on the errors for the relative landmark positioning (top) and orientation (bottom) errors. An average positional error equal to 0.0589 m with standard deviation equal to 0.0429 m is observed, which greatly meets the requirements of the benchmark. As far as the orientation is concerned, an average error equal to 0.0404 rad (i.e., $2.3147°$) is achieved, with

This project is funded by the European Union G.A No 101016906

standard deviation equal to 0.0348 rad, fulfilling the benchmark requirements. These experimental results additionally corroborate the effectiveness of the proposed SLAM architecture.

Table 5 - Results of the accuracy for the relative localization to the closest landmarks

Error	Mean	Median	Std. dev	Min.	Max.
Positional	0.0589 [m]	0.0551 [m]	0.0429 [m]	0.0006 [m]	0.1764 [m]
Angular	0.0404 [rad]	0.0291 [rad]	0.0348 [rad]	0.0008 [rad]	0.1895 [rad]

4.4.4 Loop closure

The objective of these experiments is to evaluate the SLAM algorithm's closed-loop performance and assess the localization error when the robot returns to the initial pose. More in detail, for each test, the following procedure was considered:

- Place the mobile base in a specified configuration, marked on the terrain, and store the estimated pose, denoted as $\boldsymbol{q}_s$.
- Teleoperate the mobile base through the experimental field using a remote joystick, while running the SLAM algorithm.
- Return to the same configuration as at the beginning and store the estimated pose, denoted as $\boldsymbol{q}_f$.

We conducted two sets of experiments under different seasonal conditions. Specifically, the first set of experiments was carried out during the winter season in February 2023 (shown in Figure 31), while the second set was carried out in July 2023. Note that, this choice to consider two different seasons for the experiments was motivated by the fact that, during the winter season, the absence of foliage and clusters provides an optimal environment for the GNSS RTK system. Hence, our aim was to compare the system's performance under both ideal and challenging operating conditions.



Figure 31 - Experimental field in February 2023

Two different operating conditions of the SLAM have been tested, i.e., with prior environmental knowledge, utilizing a pre-existing map, and without prior knowledge, i.e., without employing a pre-existing map.

For the experiments conducted during the winter season, Figure 32 - Figure 37 graphically depict the experimental outcomes, while Table 6 provides a numerical analysis highlighting both position and orientation errors according to the two different operating conditions.

Table 6 - Experimental results loop closure during winter season

ID	Position Error	Orientation Error	Know map	Season	Figure
Test1	0.0060 [m]	0.0114 [rad]	No	Winter	Figure 32
Test2	0.0114 [m]	0.0407 [rad]	No	Winter	Figure 33
Test3	0.0096 [m]	0.0262 [rad]	No	Winter	Figure 34
Test4	0.0334 [m]	0.0138 [rad]	Yes	Winter	Figure 35
Test5	0.0209 [m]	0.0103 [rad]	Yes	Winter	Figure 36
Test6	0.0228 [m]	0.0075 [rad]	Yes	Winter	Figure 37

As evident from the table, the positional errors are notably small, measuring always less than 4 cm. As a matter of fact, these errors may also be influenced by various factors, including human error in precisely repositioning the large omni-directional robot in the same location. A similar behavior is observed for what concerns the orientation errors.

Remarkably, the differences between experiments conducted with prior knowledge of the environment and those without such knowledge are insignificantly small. These minute discrepancies preclude any conclusive hypotheses regarding malfunction or performance improvement in one scenario over the other. Thus, demonstrating that the proposed navigation to satisfactorily perform in both operating scenarios.

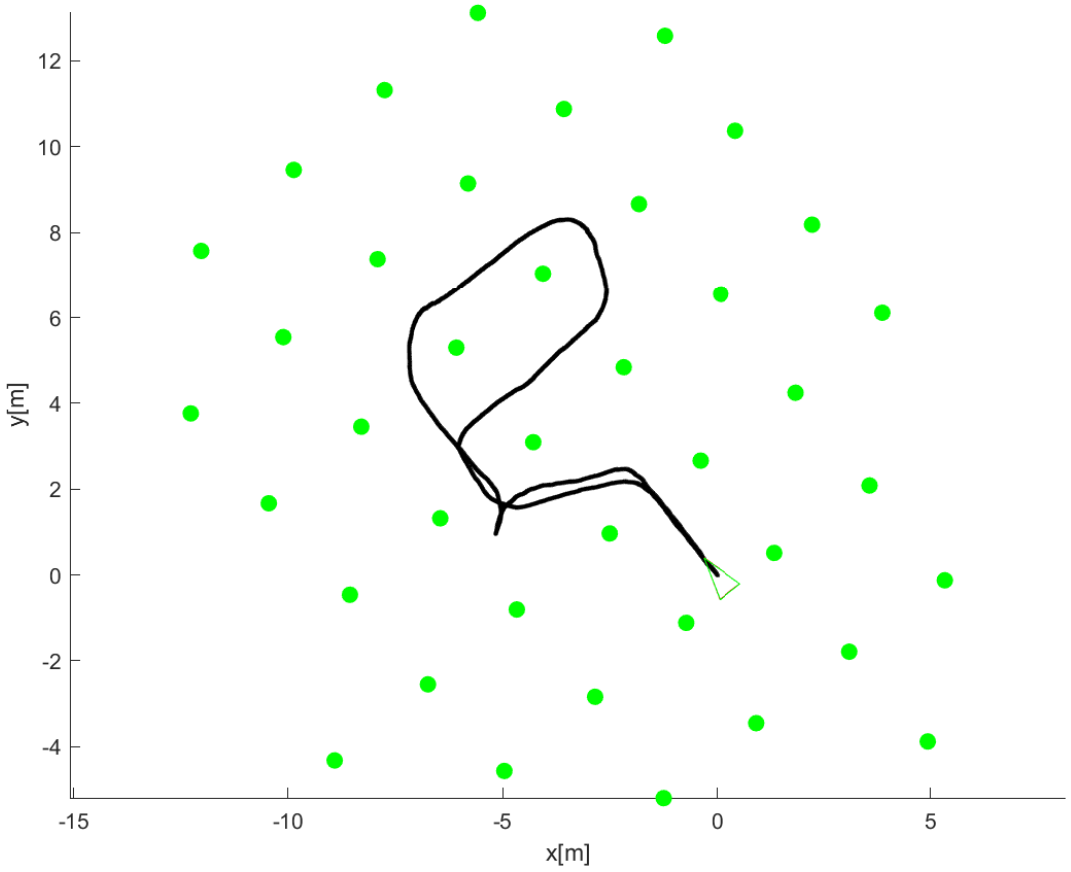


Figure 32 – Test1 - Closed-Loop SLAM Experiment with unknown map during winter season

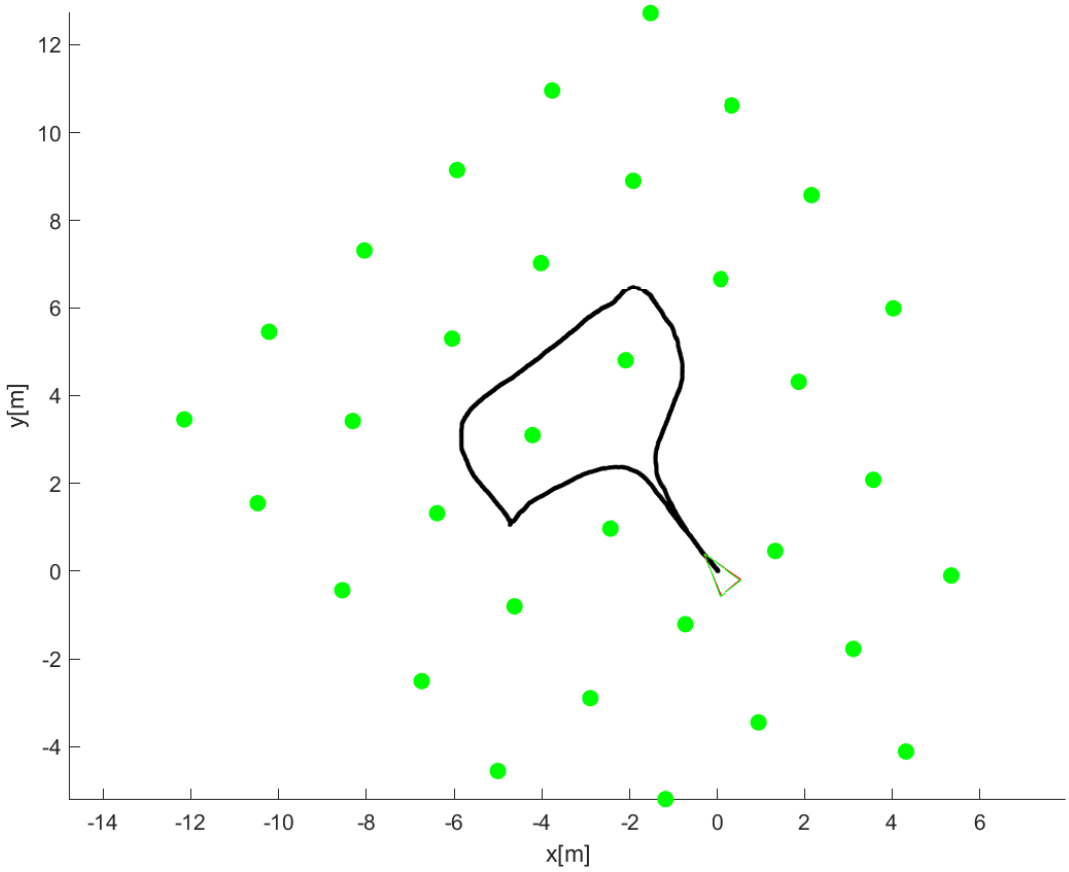


Figure 33 – Test2 - Closed-Loop SLAM Experiment with unknown map during winter season

This project is funded by the European Union G.A No 101016906

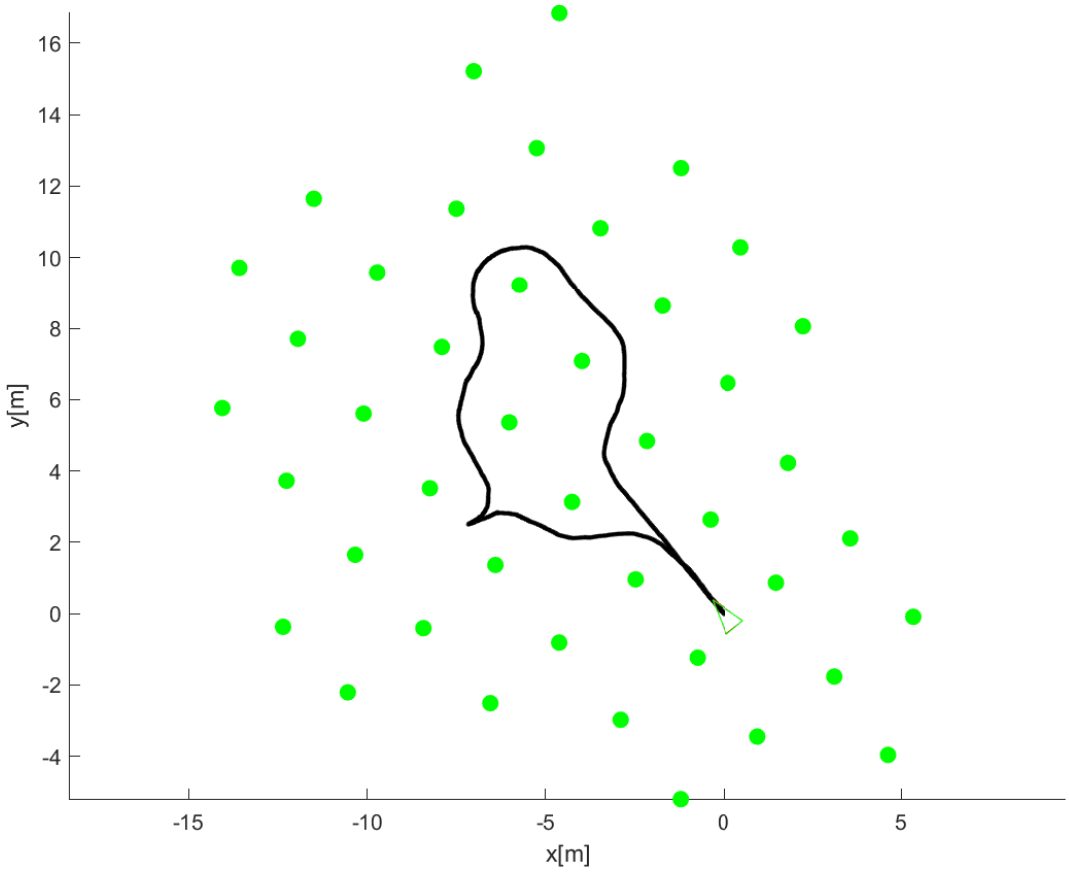


Figure 34 – Test3 - Closed-Loop SLAM Experiment with unknown map during winter season

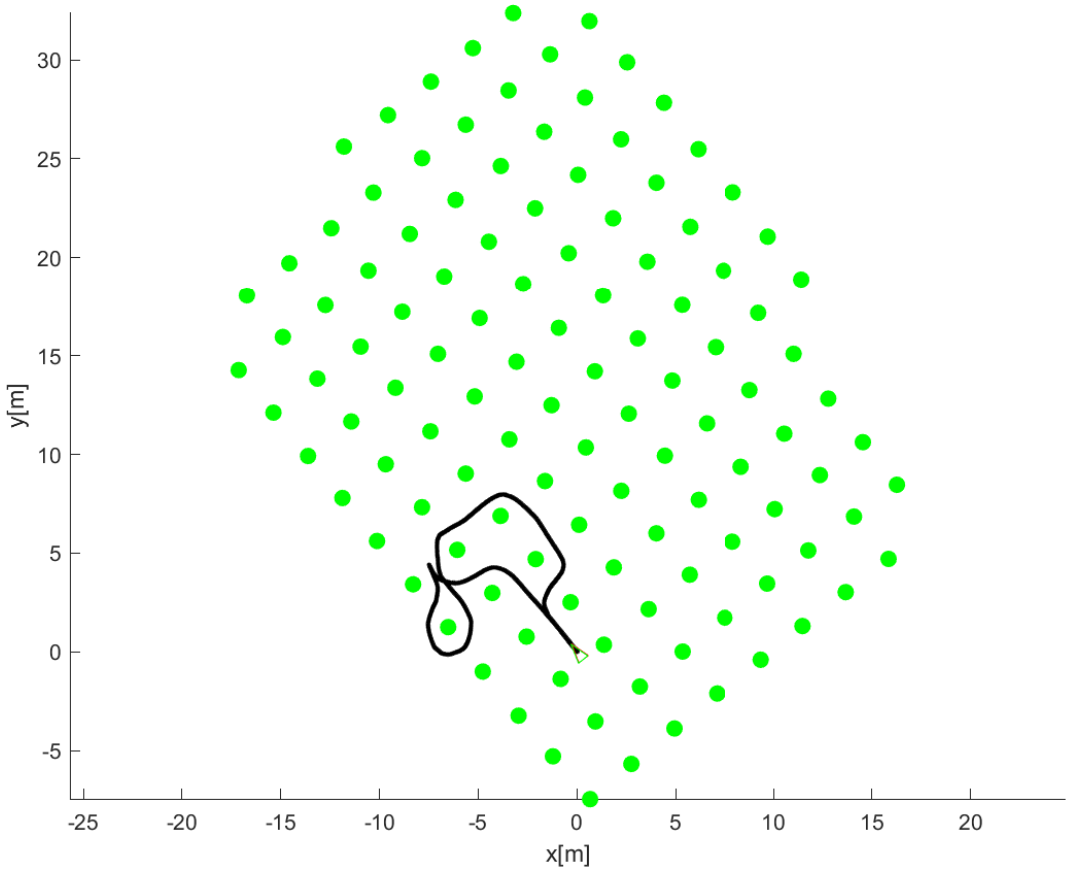


Figure 35 – Test4 - Closed-Loop SLAM Experiment with known map during winter season

This project is funded by the European Union G.A No 101016906

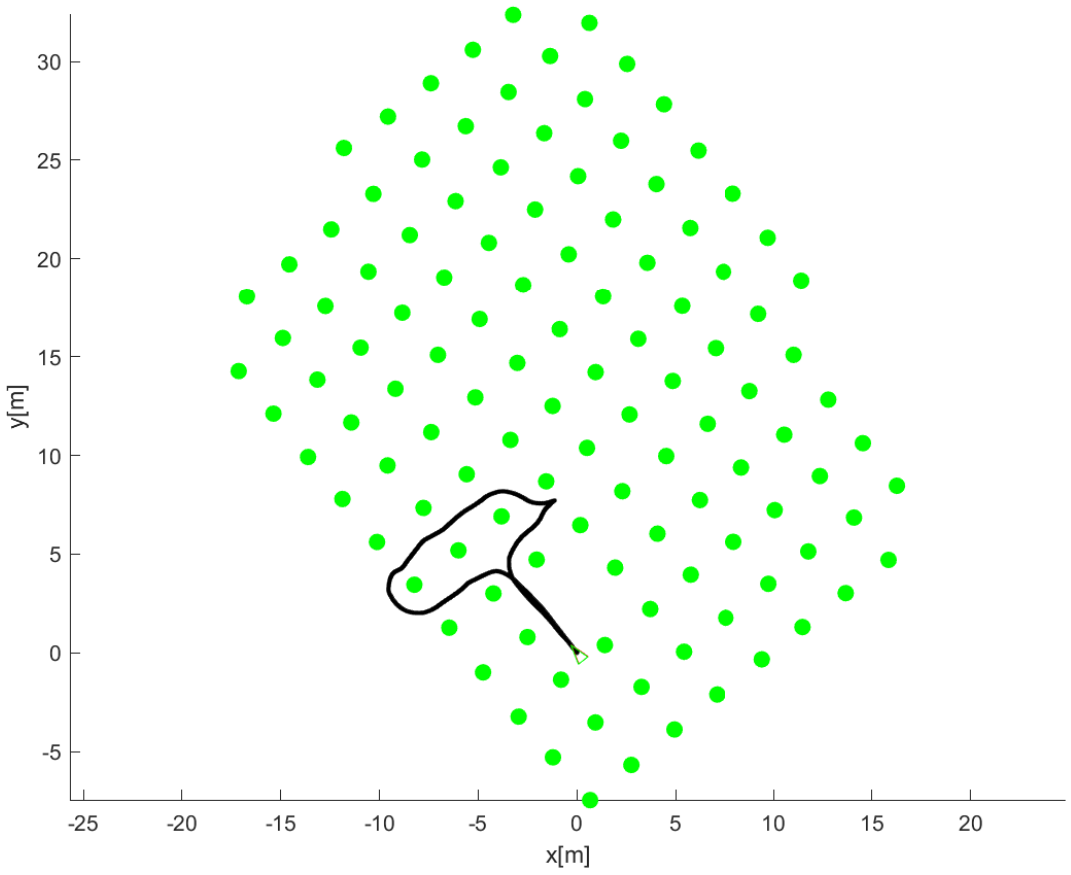


Figure 36 – Test5 - Closed-Loop SLAM Experiment with known map during winter season

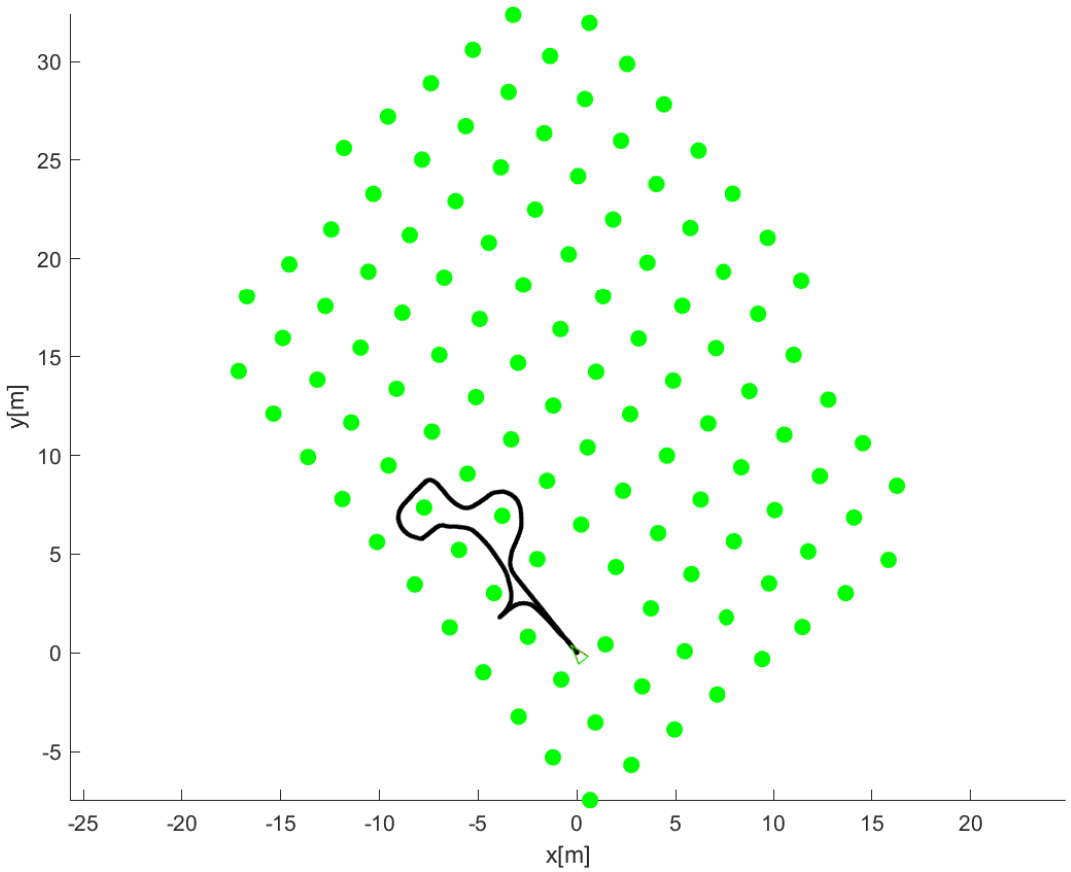


Figure 37 – Test6 - Closed-Loop SLAM Experiment with known map during winter season

For the experiments conducted during the summer season, Figure 38 - Figure 45 graphically depict the experimental outcomes, while Table 7 provides a numerical analysis highlighting both position and orientation errors according to the two different operating conditions.

Table 7 - Experimental results loop closure during summer season

ID	Position Error	Orientation Error	Know map	Season	Figure
Test7	0.0613 [m]	0.0007 [rad]	No	Summer	Figure 38
Test8	0.0161 [m]	0.0127 [rad]	No	Summer	Figure 39
Test9	0.0827 [m]	0.0314 [rad]	No	Summer	Figure 40
Test10	0.0495 [m]	0.0003 [rad]	No	Summer	Figure 41
Test11	0.0054 [m]	7.2989e-05 [rad]	Yes	Summer	Figure 42 Figure 43
Test12	0.0090 [m]	0.0127 [rad]	Yes	Summer	Figure 44 Figure 45

It should be noticed that, in contrast to the winter period, the summer season presents distinct challenges due to the lush foliage that covers the environment. During this time, the landscape becomes densely vegetated, leading to increased occlusion points for satellite-based geolocation. Often, there are instances when GPS signals are either not received or exhibit high covariance, rendering them unsuitable for inclusion during the GPS data validation process before the correction step.

In Test7 in Figure 38, shortly after the initialization process, the GPS signal is lost, resulting in uncertainty that encourages the robot to better adjust to the only available inputs: motor encoder data and observations of landmarks and surrounding obstacles received from the perception system.

It should be noticed that this issue may only occur during the initial phase, that is when the SLAM system does not possess a reliable map with low covariance yet. As a result, it encounters difficulty in interpreting the landmarks, as their sizes and shapes vary, and it does not have a sufficient number of landmarks on the current map yet to accurately localize itself. However, in the Test7 after navigating the first turn, it reestablishes the GPS signal and makes the necessary adjustments. Notably, after a sufficient number of landmarks is obtained and thus a reliable map has been constructed, this issue no longer occurs regardless of wheter the GPS signal is reliable or not.

In Test9 in Figure 40, a similar issue occurs, but once again, after receiving the GPS signal, the SLAM system corrects its position. However, it should be highlighted that in all the experiments carried this

This project is funded by the European Union G.A No 101016906

issue has only affected the initial robot localization while it has never been a cause for any failure in the overal map construction and robot locatization.

In Figure 39 and Figure 41, there are no noteworthy observations. In Test11 in Figure 42, we repeated Test7, this time using a known map with 588 landmarks, to observe the behavior of the SLAM system under the most challenging environmental and computational conditions while ensuring a high level of landmark accuracy around the robot. The results are excellent, as both positional and orientation errors are significantly reduced. In Figure 43, a close-up view reveals that the issue described previously for Test7 in Figure 38 did not occur.

The same applies to Figure 44 and Figure 45, where Test9 is repeated, yielding excellent results once again. It is important to highlight that despite the rise in computational expenses, the implemented optimization guarantees reliable operation.

Finally, it is worth highlighting that, as in the winter case study, no significant difference is recorded between the case of known map and the more challenging case of unknown map. Furthermore, we would like to emphasize that, even under the more demanding conditions of the summer season, we observe comparable results to those achieved during the winter period, thus confirming the effectiveness of the proposed SLAM algorithm.

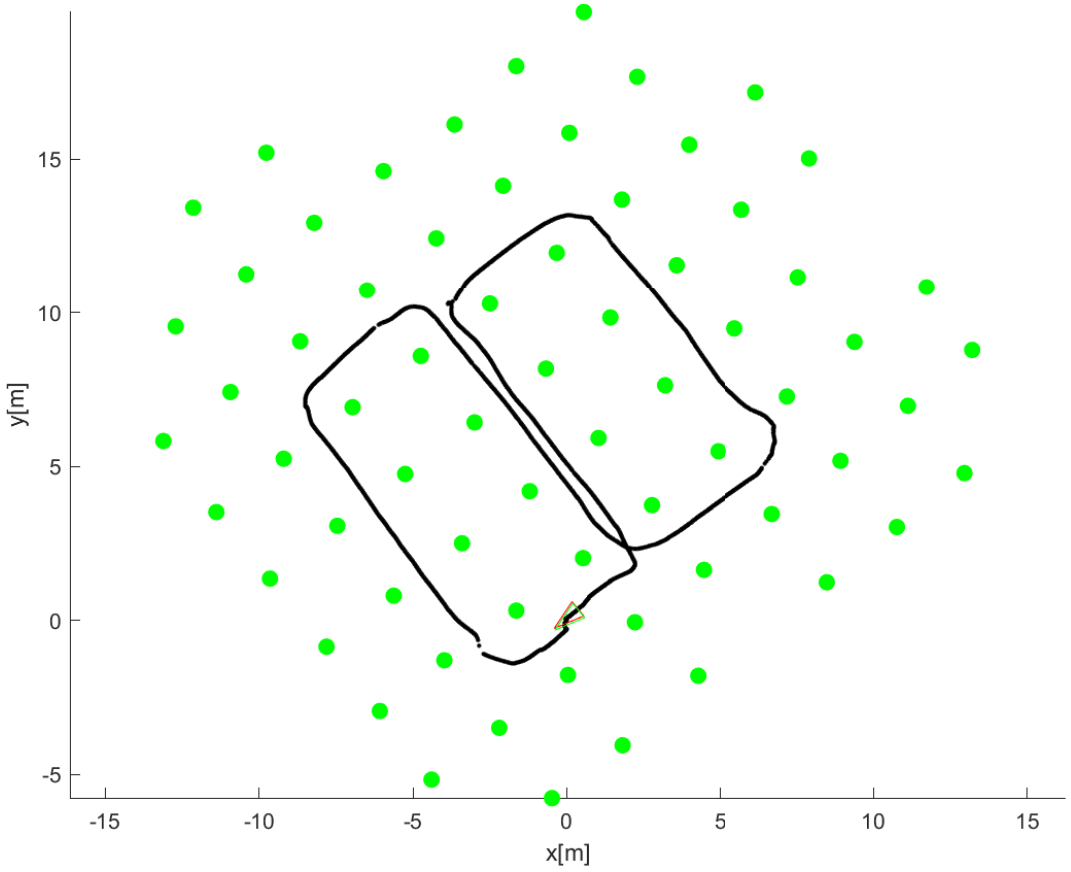


Figure 38 – Test7 - Closed-Loop SLAM Experiment with unknown map during summer season

This project is funded by the European Union G.A No 101016906

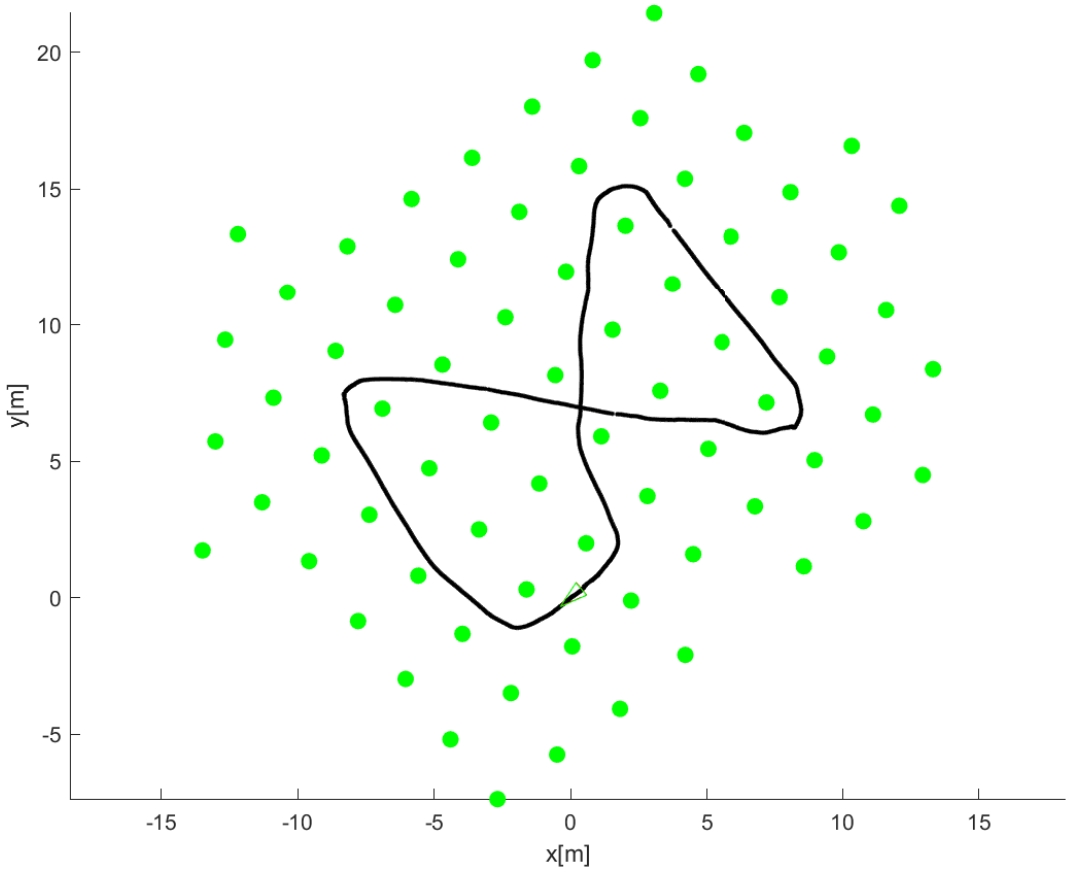


Figure 39 – Test8 - Closed-Loop SLAM Experiment with unknown map during summer season

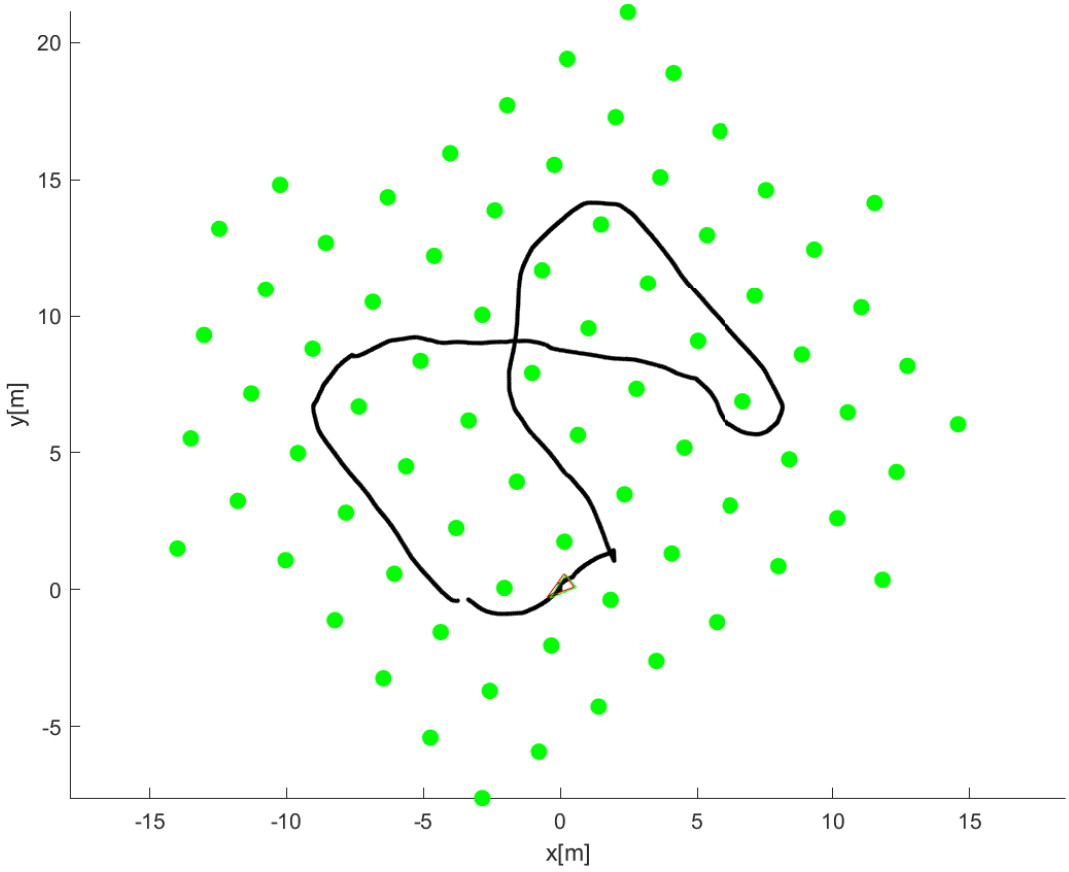


Figure 40 – Test9 - Closed-Loop SLAM Experiment with unknown map during summer season

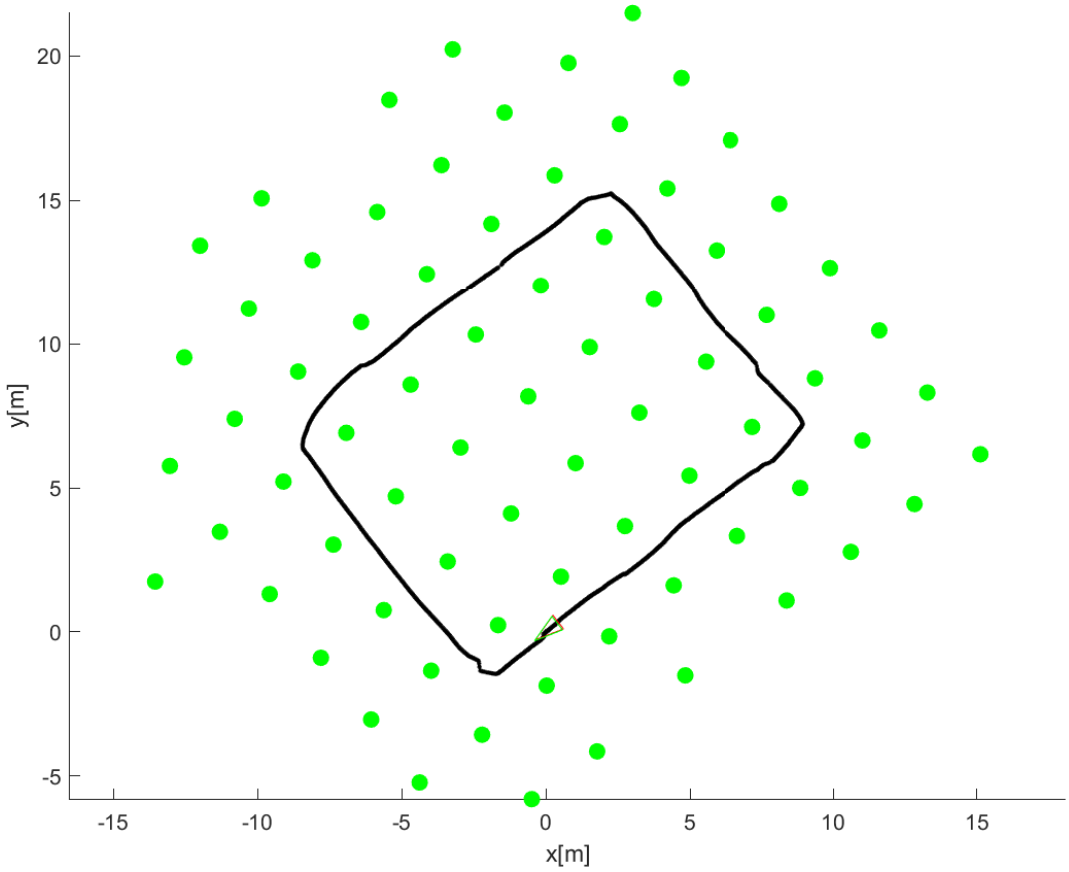


Figure 41 – Test10 - Closed-Loop SLAM Experiment with unknown map during summer season

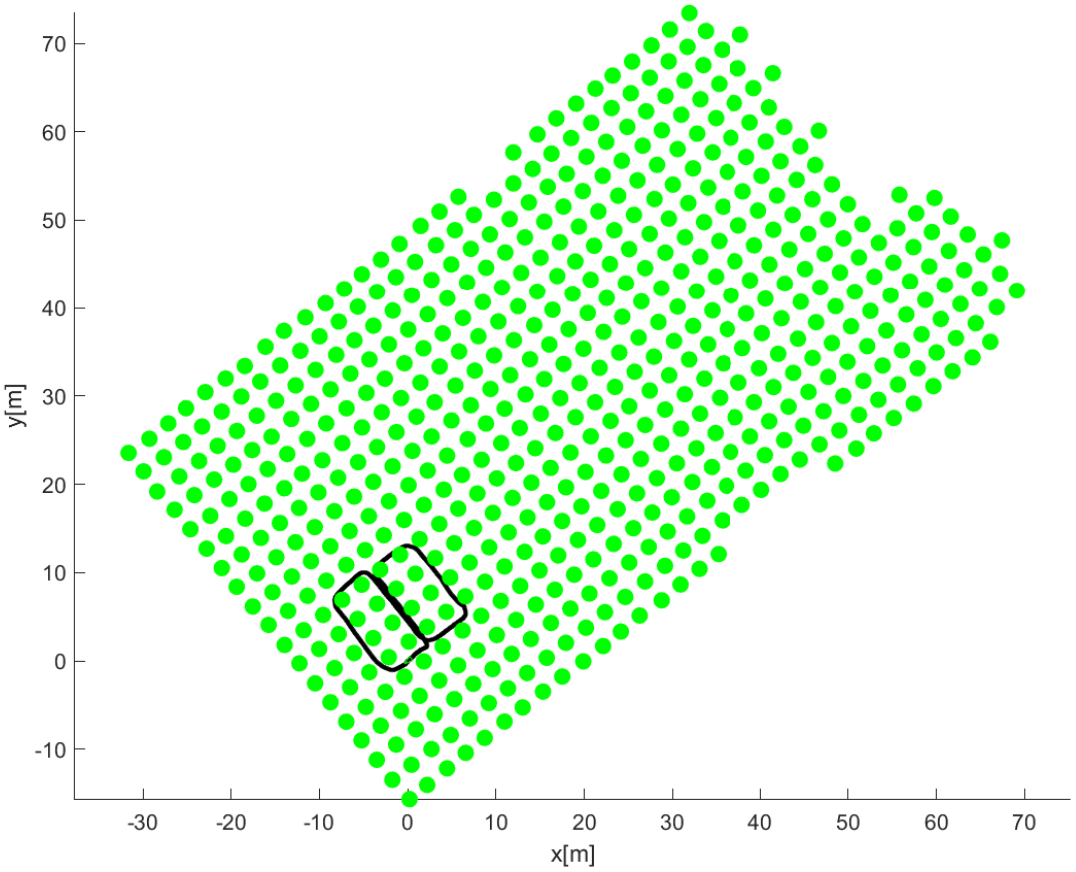


Figure 42– Test11 - Closed-Loop SLAM Experiment with known map during summer season

This project is funded by the European Union G.A No 101016906

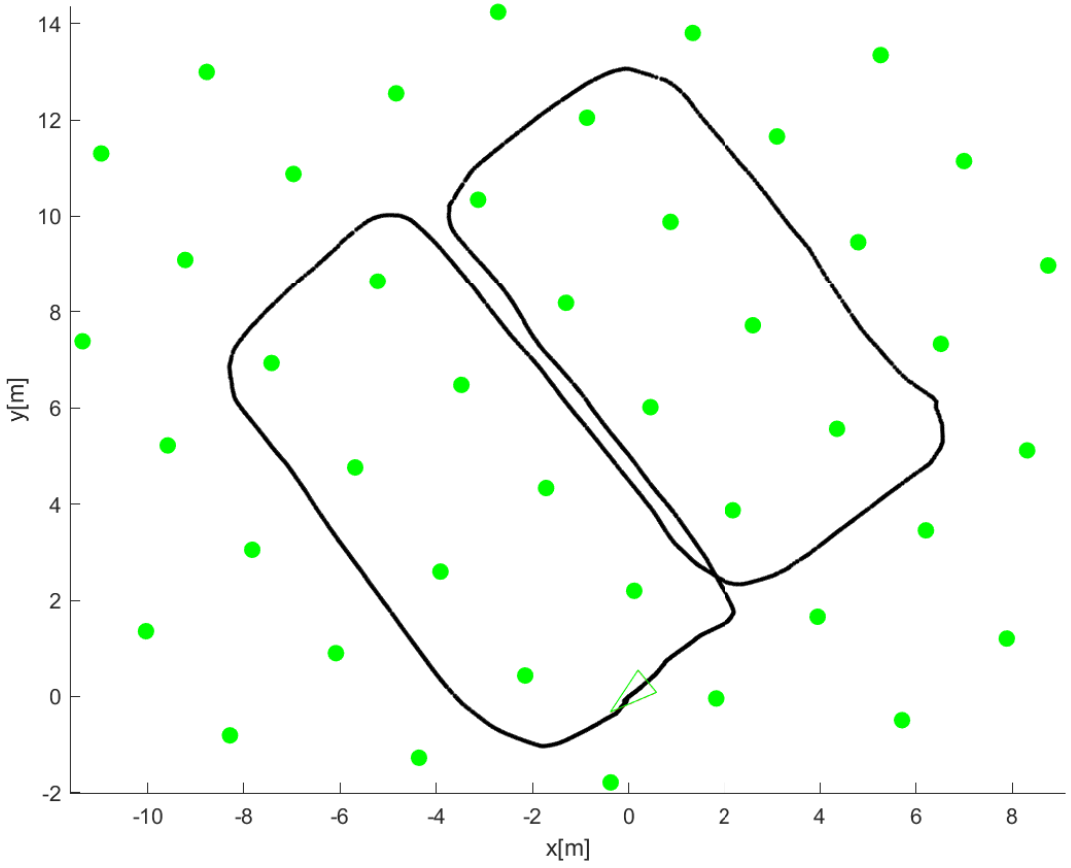


Figure 43– Test11- Closed-Loop SLAM Experiment with known map during summer season. Zoomed in for better visibility

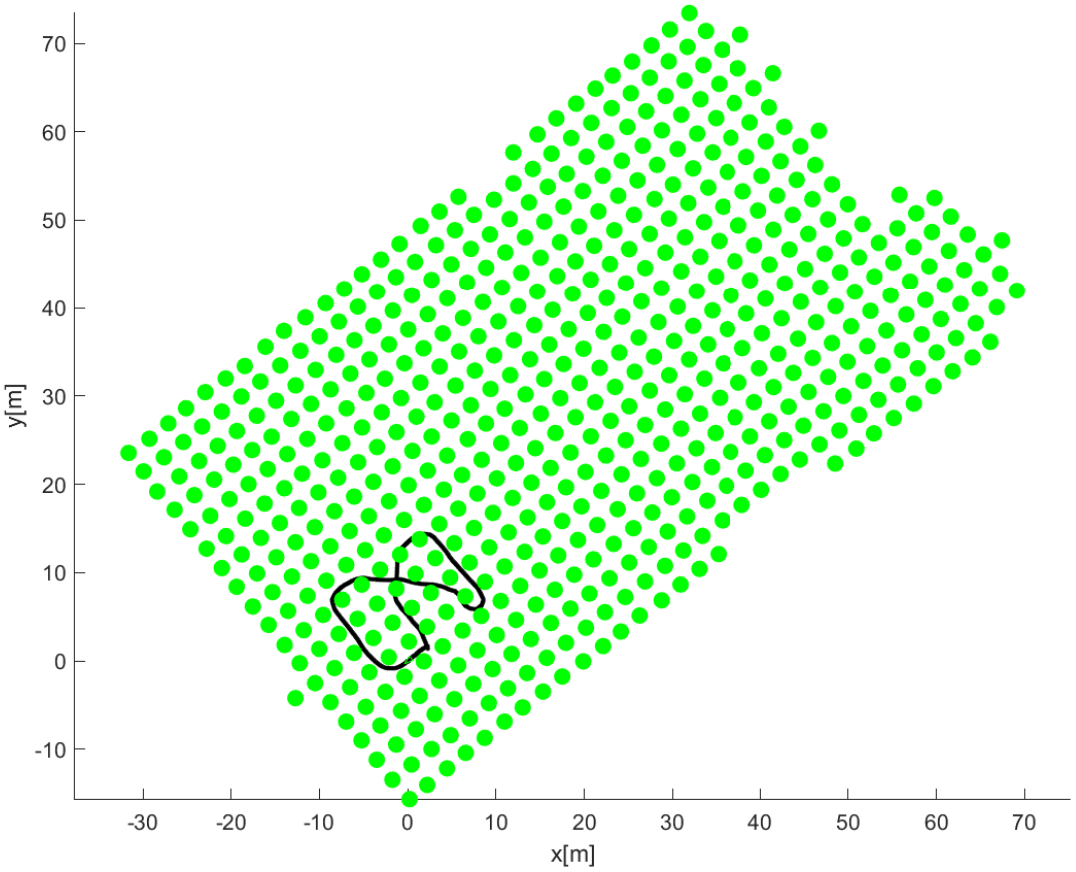


Figure 44– Test12- Closed-Loop SLAM Experiment with known map during summer season

This project is funded by the European Union G.A No 101016906

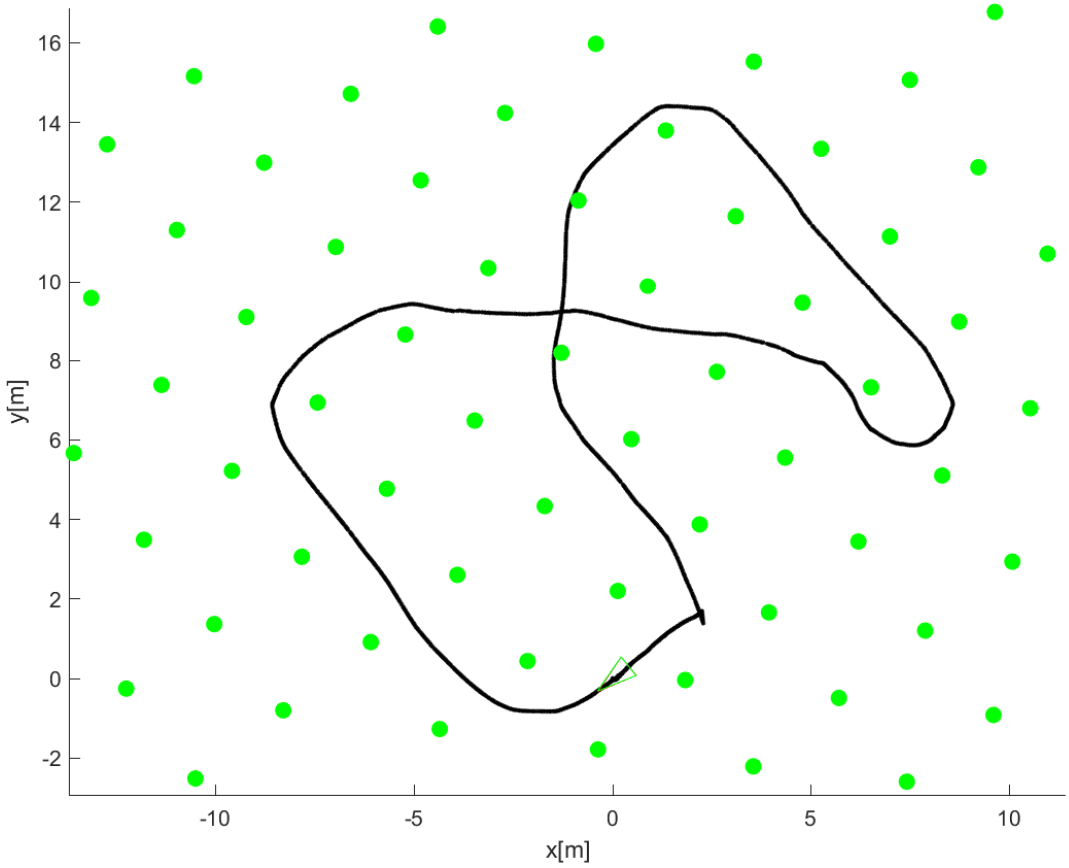


Figure 45– Test12- Closed-Loop SLAM Experiment with known map during summer season. Zoomed in for better visibility

This project is funded by the European Union G.A No 101016906

5 Local planner and controller

The goal of the local planner and controller is to generate optimal trajectories, and respective velocity commands in such a way that:

1. The trajectory steers the robot to a given target pose;
2. The trajectory satisfies the unicycle kinematics constraints and is collision-free;
3. The trajectory is generated in a receding horizon fashion and in real-time.

Note that real-time re-planning plays a pivotal role in enabling the robot to respond promptly to dynamic obstacles. These include human operators who can operate in the same environment. As reported in Section 4.3.3, the set of obstacles in the environment is provided by the SLAM module. This is composed of the static obstacles from the map, possibly dynamic obstacles excluded by the data association process and robots sharing the environment.

It should be notice that according to the preliminary outcomes of the experimental campaign it was decided to simplify the navigation stack architecture compared to the one presented in Deliverable 7.1 "Coordinate Maneuvering for Box-Exchange Operation." In particular, the pose regulation control law has been discarded in favor of a simplified approach that directly generates velocity commands through a local planner and controller formulation. Notably, this adjustment allowed to improve both the efficiency and responsiveness of the system, while maintaining a similar level of accuracy as discussed in Section 5.2.

5.1 Methodology

5.1.1 Finite-horizon optimization problem

The problem addressed by the local planner and controller module can be formulated as a cost-function minimization problem over a finite-time horizon T. Specifically, let $\boldsymbol{o}(t) = \left[\boldsymbol{o}_{1(t)}^T, \ldots, \boldsymbol{o}_{\mathfrak{o}(t)T}\right]^T$ be the state vector of the surrounding obstacles where $\boldsymbol{o}_{i(t)} = \left[x_{i(t)}, y_{i(t)}\right]^T$ represents the state of the i-th obstacle considered at time t described by its coordinates, and $\mathfrak{o}(t)$ is the number of 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. Given a desired pose $\boldsymbol{q}_d$ to reach and the set of obstacles $\boldsymbol{o}(t)$, we introduce the following functions $h\left(\boldsymbol{q}(t_f)\right), c_n\left(\boldsymbol{q}(t), \boldsymbol{u}(t)\right)$, and $c_o\left(\boldsymbol{q}(t), o(t)\right)$ encoding the costs related to the final state, the navigation, and the obstacle avoidance, respectively, and defined as

$$
\begin{aligned}
h\left(\boldsymbol{q}(t_f)\right) &= \left(\boldsymbol{q}_d - \boldsymbol{q}(t_f)\right)^T \boldsymbol{Q}_f \left(\boldsymbol{q}_d - \boldsymbol{q}(t_f)\right), \\
c_n\left(\boldsymbol{q}(t), \boldsymbol{u}(t)\right) &= \boldsymbol{u}(t)^T \boldsymbol{R}\, \boldsymbol{u}(t) + (\boldsymbol{q}(t) - \boldsymbol{q}_d)^T \boldsymbol{Q}\, (\boldsymbol{q}(t) - \boldsymbol{q}_d), \\
c_o\left(\boldsymbol{q}(t), \boldsymbol{o}(t)\right) &= \sum_{i=1}^{\mathfrak{o}(t)} \left(\left(\boldsymbol{p}(t) - \boldsymbol{o}_i(t)\right)^T \boldsymbol{W}_i \left(\boldsymbol{p}(t) - \boldsymbol{o}_i(t)\right) \right)^{-1},
\end{aligned}
$$

This project is funded by the European Union G.A No 101016906

where $t_f = t_0 + T$ is the final time of the finite horizon, with t_0 being the initial time, and $\boldsymbol{Q}_f$, $\boldsymbol{W}_i$, $\boldsymbol{Q}$, $\boldsymbol{R}$ are weighting matrices. The rationale behind the above costs is that $h\left(\boldsymbol{q}(t_f)\right)$ encourages the robot to reach the desired pose at the final time t_f, $c_n\left(\boldsymbol{q}(t), \boldsymbol{u}(t)\right)$ encourages the robot to approach the desired pose while minimizing the control effort and $c_o\left(\boldsymbol{q}(t), \boldsymbol{o}(t)\right)$ promotes avoiding the obstacles in the environment.

The following non-linear optimization problem is formulated:

$$\min_{\boldsymbol{q}(t),\boldsymbol{u}(t)} h\left(\boldsymbol{q}(t_f)\right) + \int_{t_0}^{t_f} c_n\left(\boldsymbol{q}(t), \boldsymbol{u}(t)\right) + c_o\left(\boldsymbol{q}(t), \boldsymbol{o}(t)\right) dt \tag{2a}$$

$$\text{Subject to} \quad \dot{\boldsymbol{q}}(t) = f\left(\boldsymbol{q}(t), \boldsymbol{u}(t)\right), \tag{2b}$$

$$\left(\boldsymbol{q}_0 - \boldsymbol{q}(t_0)\right)^T \left(\boldsymbol{q}_0 - \boldsymbol{q}(t_0)\right) = 0, \tag{2c}$$

$$\left(\boldsymbol{p}(t) - \boldsymbol{o}_i(t)\right)^T \left(\boldsymbol{p}(t) - \boldsymbol{o}_i(t)\right) \geq d_{min,i}^2 \quad \forall\, i \in \{1, \dots, \mathfrak{o}(t)\} \tag{2d}$$

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}(t) \leq \boldsymbol{u}_{max} \tag{2e}$$

where the constraint (2b) represents the compact kinematic model as defined in Section 2.3, the constraint (2c) guarantees continuity of the trajectory imposing that the pose at time $\boldsymbol{t_0}$ is equal to initial pose $\boldsymbol{q}_0$, the constraint (2d) ensures obstacle avoidance, requiring the distance from the robot to the obstacles to be greater than an obstacle-dependent safety distance $\boldsymbol{d_{min,i}}$, and the constraint (2d) ensures that the control input limits are fulfilled, with $\boldsymbol{u_{min}}$ and $\boldsymbol{u_{max}}$ being the minimum and maximum possible control input vectors. Note that the use of an obstacle dependent safety distance allows for tailored avoidance behaviors based on the type of obstacle. For example, higher values should be applied for generic obstacles, which may involve human operators, while lower values can be assigned to static landmarks on the map. This permits the robot to approach poles closely if no collision risk is detected. Further examples are provided in the following section.

To comply with the real-time requirements of the system, a dynamic selection of the obstacles included in the optimization problem is necessary. For instance, only obstacles within a specified radius from the robot's current location might be used for planning. The selection of obstacles should be fine-tuned based on experimental data to strike a balance between computational efficiency and effective planning.

5.1.2 Possible states

We define four distinct states for the local planner and controller, each associated with a specific behavior:

- *Idle*: In this state, the controller is disabled, and zero velocity commands are sent to the motors, implying that no optimization problem is solved.

- *Navigation*: In this state, the robot navigates through the vineyard to reach desired poses and must avoid static, robotic, and other (possibly dynamic) obstacles.
- *Docking*: In this state, the logistic robot (executing the planner) is in the process of docking and needs to approach the farming robot.
- *Out-of-docking*: This state is engaged after the logistic robot has completed the docking operation and needs to move backward away from the farming robot.

In all states, except for the idle, we employ the optimal formulation detailed earlier, with distinct sets of parameters tailored to the specific state's requirements.

More in detail, during navigation we use the following distance thresholds for the static, robotic and generic (possibly dynamic) obstacles $d_{min}^{static} = 0.85$ m, $d_{min}^{robot} = 2$ m and $d_{min}^{generic} = 1.3$ m, respectively, while during docking and out-of-docking we lower d_{min}^{robot} to 0.85 m. This decision is driven by practical considerations. During the docking operation (and the subsequent out-of-docking phase), it is essential to achieve a close proximity between the robotic platforms. Conversely, while navigating, higher values are necessary to prevent the two robots from aligning side by side in the same row, accounting for their respective structural dimensions.

Different limits for the control inputs are taken into account in different phases. Specifically, during navigation, we set $\boldsymbol{u}_{min} = [-0.5\frac{m}{s},\ -0.5\frac{rad}{s}]$ and $\boldsymbol{u}_{max} = [0.5\frac{m}{s},\ 0.5\frac{rad}{s}]$,. In contrast, during the docking phase, we consider $\boldsymbol{u}_{min} = [-0.1\frac{m}{s},\ -0.1\frac{rad}{s}]$ and $\boldsymbol{u}_{max} = [0.1\frac{m}{s},\ 0.1\frac{rad}{s}]$ to reduce the velocity commands, considering the critical nature of the maneuver. Moreover, different weight matrices are considered as reported in the following table:

Navigation weights	Docking/out-of-docking weights
$\boldsymbol{Q}_f = diag(500,500,100)$, $\boldsymbol{R} = diag(200,200)$, $\boldsymbol{Q} = diag(150,150,100)$, $\boldsymbol{W}_{static} = \boldsymbol{W}_{robotic} = \boldsymbol{W}_{generic} = diag(1,1)$	$\boldsymbol{Q}_f = diag(500,500,1000)$, $\boldsymbol{R} = diag(200,200)$, $\boldsymbol{Q} = diag(150,150,200)$, $\boldsymbol{W}_{static} = diag(0.5,0.5)$, $\boldsymbol{W}_{robotic} = \boldsymbol{W}_{generic} = diag(1,1)$

More in detail, the docking/out-of-docking weight matrices encourage a more precise orientation positioning of the robot compared to the navigation case. This is essential to achieve a correct alignment of the platforms.

One significant distinction between the states is the introduction of an additional safety constraint during the docking phase. By considering that the docking is realized by aligning the front parts, this constraint ensures that the logistic robot is prevented from entering the half-plane occupied by the

farming robot. This half-plane is defined by the line passing through the front of the farming robot as shown in red in Figure 46.

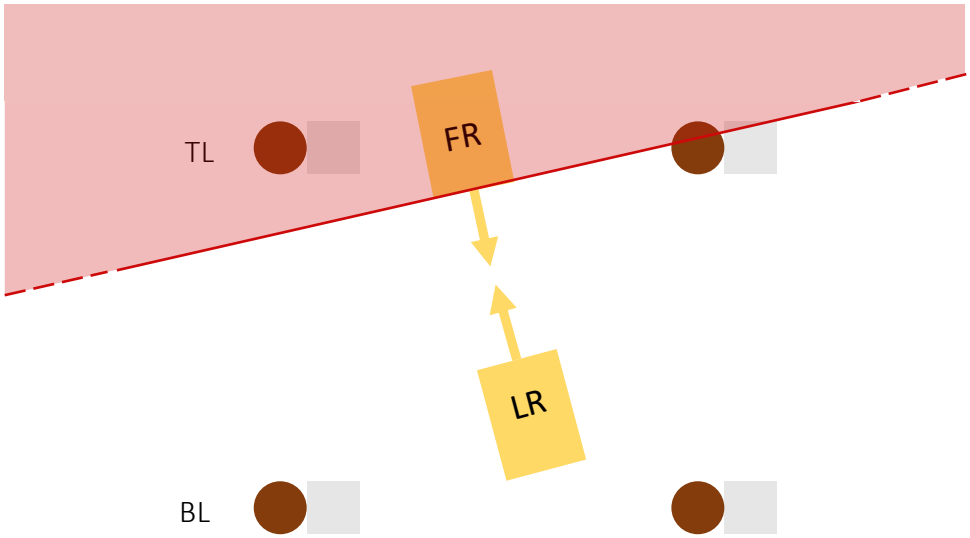


Figure 46 - Representation of the additional constraint during docking. The forbidden area is shown in red

In conclusion, depending on the task assigned to the robot and the specific phase of the task, a different state of the planner and controller may need to be activated to achieve the desired behavior, as exploited for the coordinated maneuvering procedure detailed in Section 7.2.

5.2 Experimental Validation

The local planner and controller module has been implemented in C++ and a dedicated ROS node has been developed, as detailed in Deliverable 7.1, which includes a Graphic User Interface (GUI), shown in Figure 47 for improved human comprehension and monitoring.

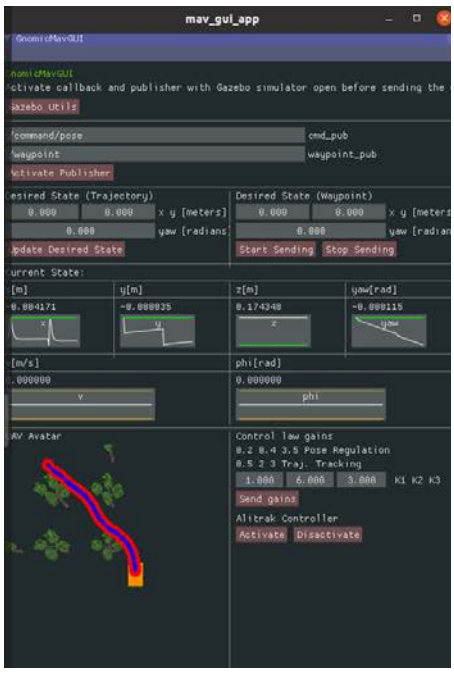


Figure 47 - GUI for the dynamic planner and controller

The non-linear planning problem is set up with the ACADO toolbox[1] 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 is integrated within the ROS node. The planner ROS node subscribes to the topics published by the SLAM node containing the odometry and the obstacles. Several ROS interfaces have been exploited to request the attainment of desired goals. Specifically, three interfaces are available that are:

1. *Topic-based:* In this case, the goal is published on a topic, which the planner ROS node subscribes to. Whenever the goal on the topic is updated, the optimization problem adjusts accordingly. The acknowledgment of goal achievement is communicated through a separate topic. This interface is suitable when the acknowledgment is not a primary concern, as it necessitates subscribing to the acknowledgment topic and actively checking it for updates.
2. *Action-based:* A ROS action server[3] is running, enabling the request of (custom) actions containing the desired goal. The acknowledgment process is fully integrated within the ROS action architecture by providing feedback and results. In this case, the update of the goal cannot be "continuous"; instead, a new action request must be made each time, preempting any previous request. This interface is appropriate when the robot should reach a predefined pose that does not change over time.
3. *Action-and-topic-based:* A further ROS action server allows to request the attainment of a goal, but this can be updated over time using a topic. In this setup, the acknowledgment system remains available, and the action feedback and result are associated with the latest

[1] https://acado.github.io/
[2] https://github.com/coin-or/qpOASES
[3] http://wiki.ros.org/actionlib

goal sent over the topic. This interface is suitable when both receiving acknowledgments and continuously updating the goal are important.

A goal is considered successfully reached if both positional and orientation errors fall below predefined thresholds, and the robot linear and angular velocities are under predefined limits. If the robot velocities drop below the specified thresholds, but the errors exceed the maximum allowable values, a timer is activated: if the robot remains in this state for more than 10 seconds, a failure is registered. This precaution helps prevent unnecessary attempts to reach the desired pose when the robot is potentially obstructed by terrain issues (see Figure 48 for an example). Moreover, a failure is declared if no feasible solutions for the optimization problem (implying no feasible trajectory for the robot) are found for over 10 seconds.



Figure 48 - Example of terrain issues preventing the robot navigation

For the experiments reported in the following, we used the action-and-topic-based interface to send goal commands and considered the navigation state of the local planner and controller. Additionally, a finite time horizon $T = 15$ s was set.

To evaluate the effectiveness of the planner and controller algorithm we resorted to the respective benchmarks defined in Deliverable 2.3 “Dimensioning of the Real-World Scenario for Final Demo” which are the following:

1. Pose regulation with an accuracy of 15 cm regarding the position (x,y) and 20° regarding the orientation.
2. Capability of safely navigating from one row to another one by doing the related maneuvering operations without losing its localization capabilities within the map, e.g., by keeping its localization accuracy within 15 cm regarding the position and 20° regarding the orientation along the trajectory.

As mentioned in Section 4.4.3, we verified the second benchmark in conjunction with the third benchmark related to the SLAM. Specifically, we required the robot to autonomously navigating from a starting point to a goal one, avoiding the obstacles in the environment, and intermediate measurements along the trajectory were collected to verify the localization capabilities. The respective experimental results are reported in Table 5, showing the fulfillment of the benchmark. Therefore, in the following we focus on the pose regulation benchmark.

To assess the pose regulation accuracy, we computed the errors obtained by reaching a specified desired configuration from four distinct starting configurations. We repeated the overall experiment

This project is funded by the European Union G.A No 101016906

five times, i.e., selected five different points and, for each of them, evaluated the errors obtained from four starting configurations. For the selection of starting configurations, we considered the grid layout of the field. The robot was placed in the four nearest cells along the diagonals relative to the cell containing the desired state. The exact position within these cells, as well as the orientation of the robot, were chosen randomly. A visual representation of this selection process is provided in Figure 49.

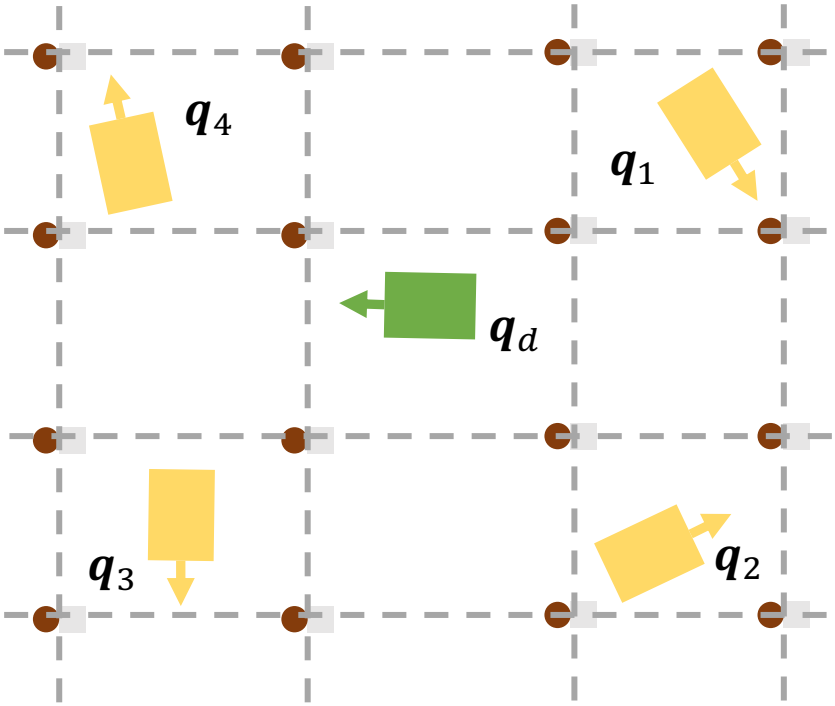
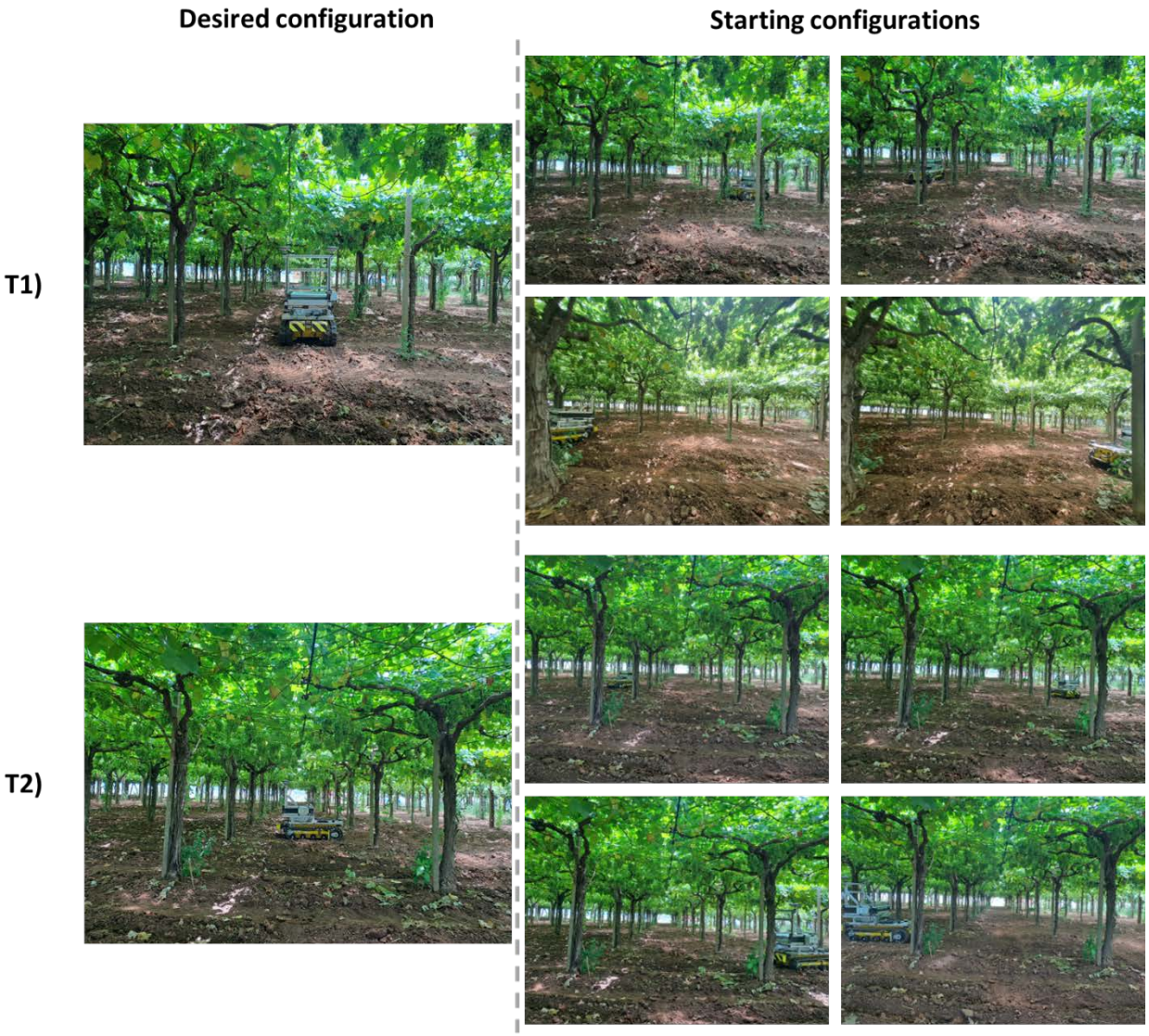


Figure 49 - Depiction of the configuration selection for evaluating the pose regulation accuracy

The pictures of the five considered desired configurations, along with the respective starting configurations, are shown in Figure 50.

This project is funded by the European Union G.A No 101016906



This project is funded by the European Union
G.A No 101016906

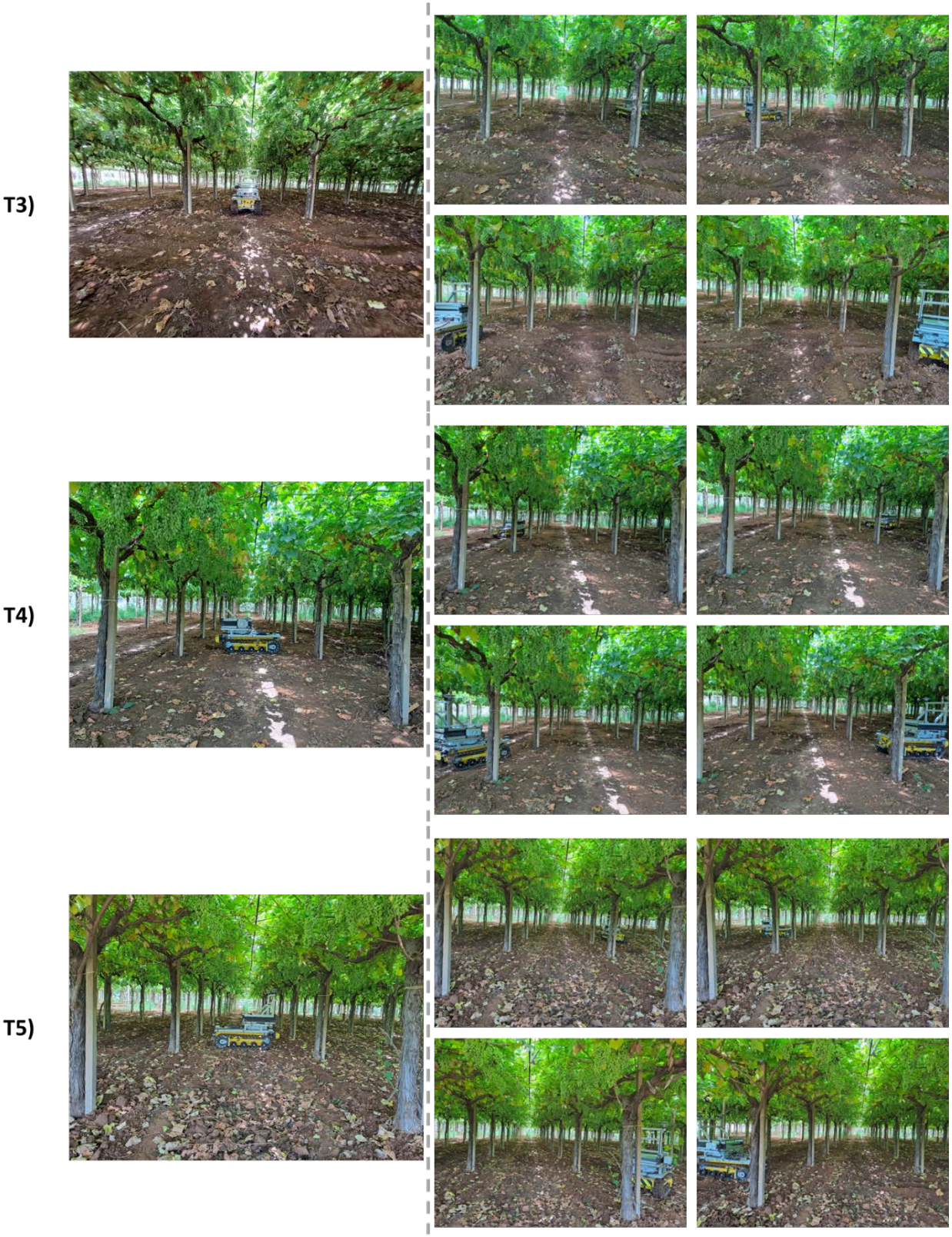


Figure 50 - Pictures related to the five tests for evaluating the pose regulation accuracy

This project is funded by the European Union G.A No 101016906

The accuracy results are summarized in Table 8. In particular, the top row shows the statistics related to the positional errors while the bottom row to the orientation errors. An average positional error equal to 0.0481 m with standard deviation equal to 0.0425 m is observed, which greatly meets the requirements of the benchmark. Such requirements are also fulfilled in the worst-case scenario, i.e., with the maximum error (equal to 0.1094 m). Regarding the orientation, an average error equal to 0.0318 rad (i.e., $1.8236°$) is achieved, with standard deviation equal to 0.0342 rad, fulfilling the benchmark requirements.

Table 8 - Results of the accuracy for the pose regulation

Error	Mean	Median	Std. dev	Min.	Max.
Positional	0.0481 [m]	0.0425 [m]	0.0272 [m]	0.0120 [m]	0.1094 [m]
Angular	0.0318 [rad]	0.0342 [rad]	0.0190 [rad]	0.0013 [rad]	0.0696 [rad]

It is worth noting that implicit validations of the planner and controller are also carried out in the following sections, as it serves as a basic component for testing the global planner and coordinated maneuvering. More in detail, in Section 7.3.3, an experiment involving human operators in the environment In Section 7.3.3, an experiment involving human operators in the environment is also reported.

This project is funded by the European Union G.A No 101016906

6 Global planner

In the high-level navigation of our robotic system, a fundamental component is the Global Planner. This component, responsible for determining optimal paths for the navigation, is built upon the A* methodology. The A* Algorithm is widely recognized as one of the most effective tools for solving optimal pathfinding problems in real-world contexts, and it has been selected as the core of our high-level navigation approach.

In the following chapter, we will thoroughly examine the details of the A* algorithm and its practical application within our Global Planner. Notably, we have developed two distinct variants not on the algorithm itself but on the method of generating inputs for it. Each variant is tailored to address specific navigation tasks and adapts to the characteristics of the respective robot in use. Through these methodologies, we aim to ensure that our robotic system can plan and follow optimal paths, by considering the physical constraints of the robot in the real world.

6.1 Methodology

The A* algorithm is a widely employed heuristic search algorithm in the field of robotics and artificial intelligence for solving pathfinding and graph traversal problems. Its primary objective is to determine the most efficient path from a given starting point to a specified goal point in a graph or grid, while considering the associated costs.

The A* algorithm is characterized by its feature to find the optimal solution by evaluating nodes n in a way that combines the cost to reach a node $g(n)$ and a heuristic estimate of the cost to reach the goal from that node $h(n)$. These two components are combined to calculate the $f(n)$ value, which guides the algorithm's exploration.

The A* algorithm receives several inputs, including the goal location, the current position of the robot, the planting pattern of the SLAM system, and an occupancy grid. Regarding the occupancy grid, its specific configuration depends on the required task. For instance, in the case of the logistic robot, there are no restrictions on grid traversal. However, for the robot equipped with the manipulator, motion is constrained to minimize the risk of collisions with the metal frames between rows of trees.

6.1.1 Occupancy Grid

The occupancy grid serves as a representation of the environment. It plays a key role in enabling our navigation systems to make an informed decision about the robot motion. The occupancy grid provides a spatial depiction of the surroundings based on the landmarks and obstacles provided by the SLAM system, helping the robot discerns areas that are free from obstructions and those that are occupied by obstacles.

The generation of the occupancy grid is carried out by aligning the map generated by the SLAM system with the frame of the new occupancy grid being created. We discretize the space by dividing it into discrete cells. In our case, given the specifications of our experimetal field this discretization has been chosen with a step size of 0.5 meters. Based on the presence of obstacles, we mark the

This project is funded by the European Union G.A No 101016906

corresponding grid cells as occupied. Additionally, we have explored the option of partial marking for cells adjacent to obstacles, discouraging trajectories that come too close to obstacles.

To create the columns that span the entire rows, we utilized the orientation of the map as a guiding factor. This orientation helped us determining the specific axis along which these columns needed to be anchored and extended for comprehensive coverage. In practical terms, it allowed us to align the columns with the predominant direction of motion.

In Figure 51 and Figure 52, we showcase the generated grid, specifically designed for our logistics transport robot. In such scenarios, the occupancy grid does not impose significant restrictions on grid traversal. This allows the robot to navigate freely through the environment, optimizing its path to reach its destination efficiently while avoiding the static obstacles, i.e., poles.

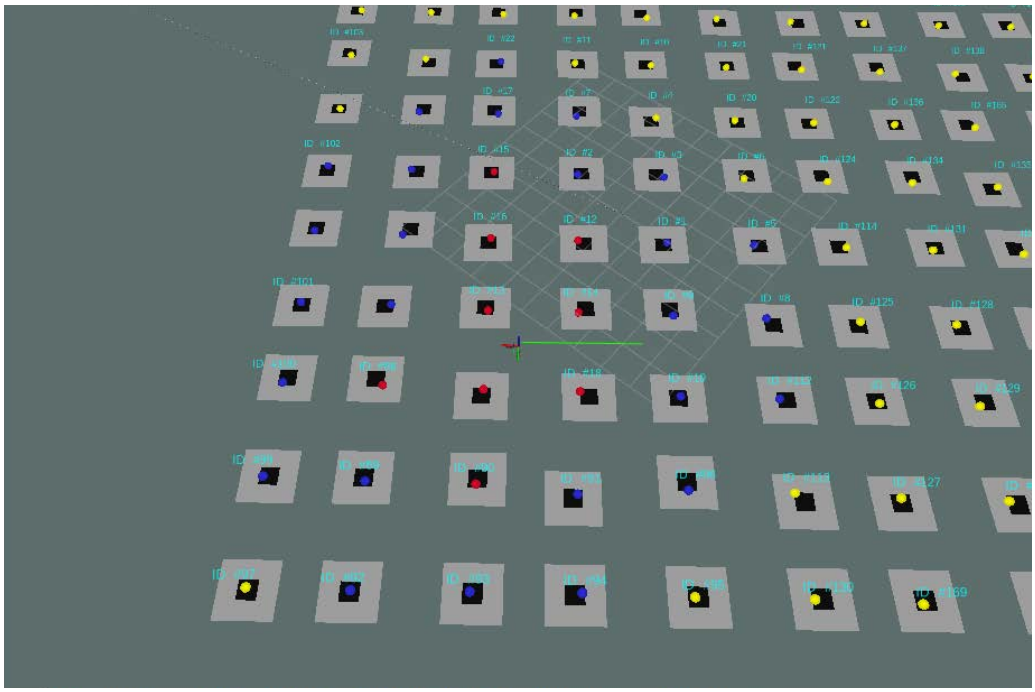


Figure 51 - Global Planner for logistic robot row changing.

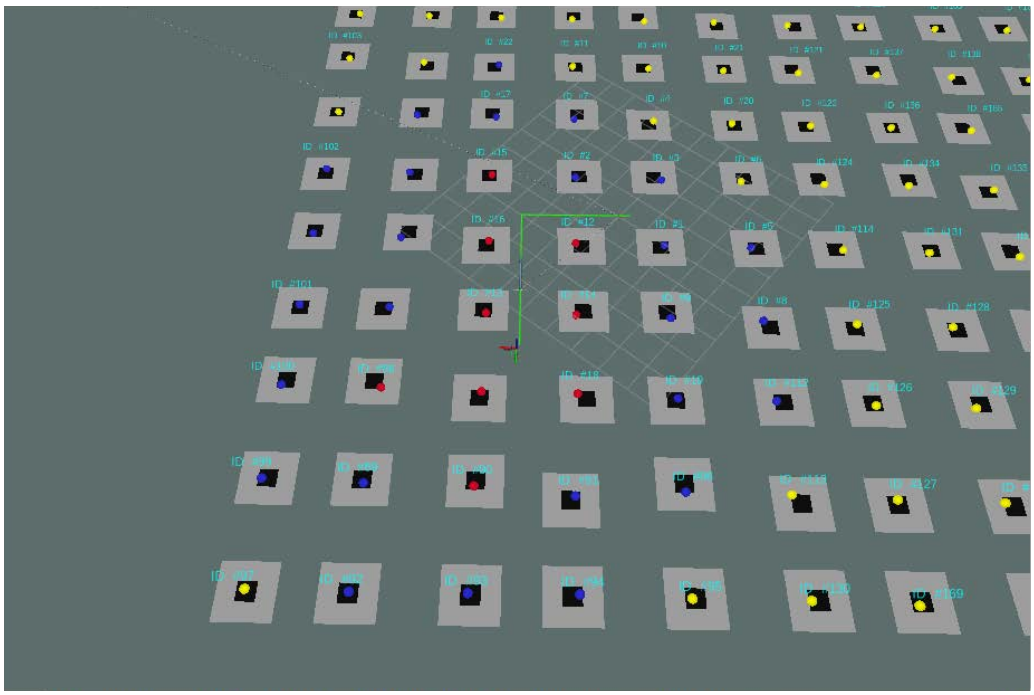


Figure 52- Global Planner for logistic robot row changing.

Figure 53 and Figure 54 illustrate the generated grid for our farming robot equipped with a manipulator involving motion restrictions. This robot is employed in environments where the presence of obstacles, such as metal frames between rows of trees, necessitates careful motion planning. In this context, the occupancy grid is configured to limit robot motion strategically, minimizing the risk of collisions with these critical obstacles. Note that, due to the discretization of the environment, the rows of the field do not appear to be perfectly straight in Figure 53. However, the discretization step size was carefully selected to strike a balance between accuracy and computational complexity.

This project is funded by the European Union G.A No 101016906

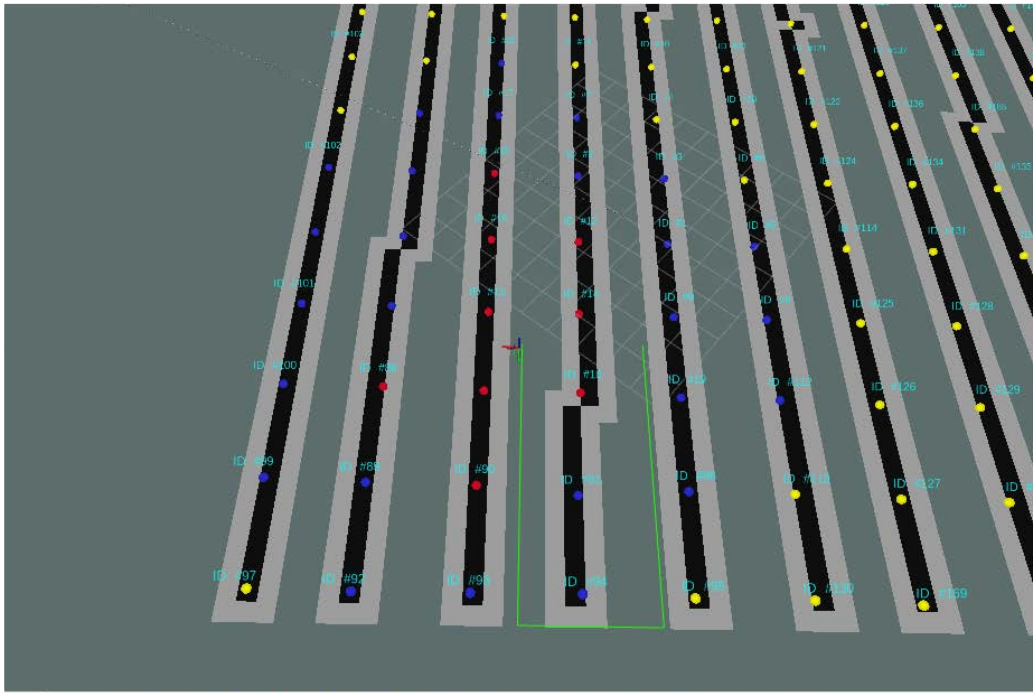


Figure 53- Global Planner for farming robot row changing.

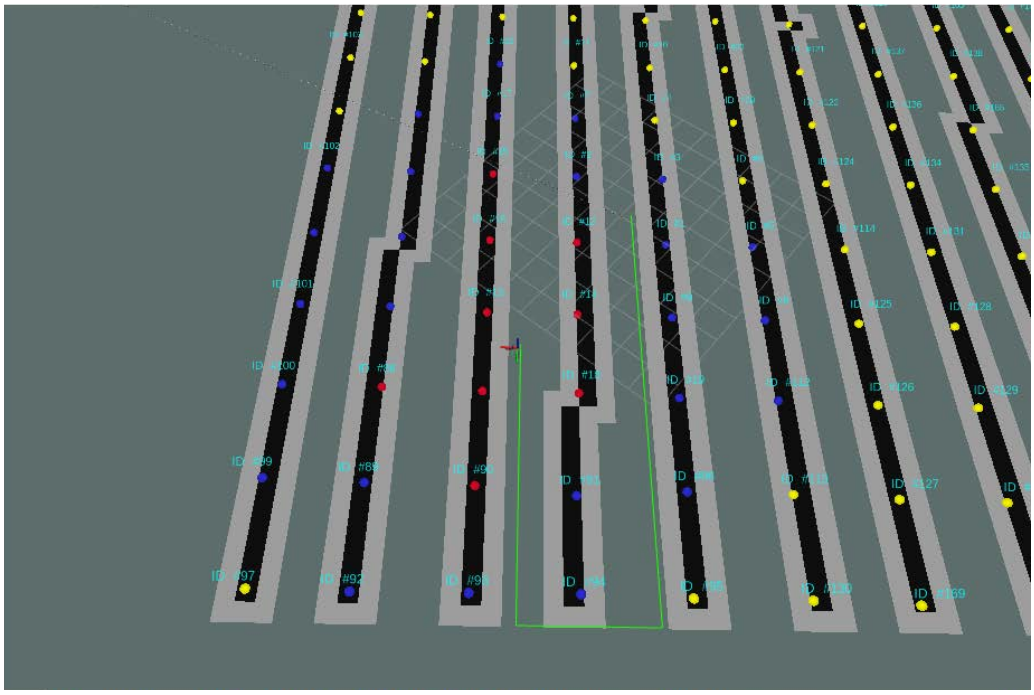


Figure 54- Global Planner for farming robot row changing.

This project is funded by the European Union G.A No 101016906

6.1.2 Algorithm Overview

The A* algorithm, for ease of understanding, simplifies the terminology by referring to the cells of the Occupancy Grid as nodes n. This algorithm is structured around the following steps:

- Initialize two empty sets S_{open} and S_{close}, where S_{open} contains nodes to be evaluated and S_{close} contains nodes already evaluated.
- Add the starting node to S_{open} with an initial $f(n)$ value set to zero. Calculate the heuristic $h(n)$ value for the starting node.
- Repeat the following until S_{open} is empty or the goal is reached:
    - Select the node with the lowest $f(n)$ value from S_{open}.
    - Move the selected node from $S_{open}$to S_{close}.
    - For each neighboring node of the selected node:
        - If the neighbor is in S_{close}or is impassable, skip it.
        - If the neighbor is not in S_{open}, calculate its $g(n)$, $h(n)$ and $f(n)$ values and add it to S_{open}.
        - If the neighbor is already in $S_{open}$and the new $g(n)$ value is lower, update its $g(n)$ and $f(n)$ values.
- When the goal node is reached or $S_{open}$is empty, the algorithm terminates. If the goal node is reached, the optimal path can be reconstructed by following the parent pointers from the goal node to the start node.

Specifically, let n be the coordinates of a node, n_{from} the coordinates of the node from which n arrives, $n_{neighbor}$ the coordinates of the neighbor node of n, n_0 the coordinate of the start node, c the cost from one node to another, c_{max} the max value assigned to a node in the occupancy grid, $c_{occ}(n)$ the value assigned to the node on the occupancy grid, $c_{grid}(n)$ the cost of n based on the value in the occupancy grid $c_{occ}(n)$, k_{grid} the cost of passing through an occupied grid, k_{dir} the cost of change direction, $c_{dir}(n, n_{neighbor})$ the cost related to the change direction required to go from n to $n_{neighbor}$, d_{prev} the direction used to arrive in n, d_{next} the direction required to go to $n_{neighbor}$, and α_{turn} the angle difference between d_{prev} and d_{next}.

Specifically, $f(n)$, $g(n)$, and $h(n)$ are calculated as follows:

$$f(n) = g(n) + h(n)$$

$$g(n_{neighbor}) = g(n) + c(n) + c_{grid}(n) + c_{dir}(n, n_{neighbor})$$

$$h(n) = distance_{euclidean}(n, goal)$$

with

$$c = 1$$

$$c_{grid}(n) = k_{grid} * \frac{c_{occ}(n)}{c_{max}}$$

$$c_{dir}(n, n_{neighbor}) = \begin{cases} 0, & if\ \alpha_{turn} > \frac{\pi}{2} \\ k_{dir}, & otherwise \end{cases}$$

$$\alpha_{turn} = \frac{acos(d_{prev} \cdot d_{next})}{(\|d_{prev}\| \|d_{next}\|)}$$

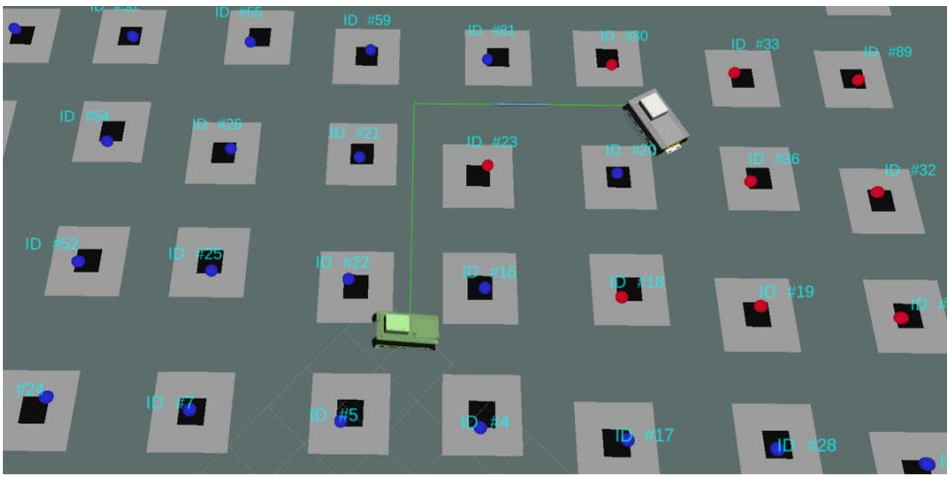
$$d_{prev} = \begin{cases} \begin{bmatrix} cos(\theta) \\ sin(\theta) \end{bmatrix}, & if\ n = n_0 \\ n - n_{from}, & otherwise \end{cases}$$

$$d_{next} = n_{neighbor} - n$$

6.2 Experimental Validation

The global planner module has been implemented in Python and a dedicated ROS node has been developed, which receives desired poses and sends intermediate waypoints to the local planner and controller.

An example of navigation achieved by employing the global planner is reported in Figure 55, where Rviz screenshots of the navigation process (realized within the experimental field) are depicted. Specifically, the first screenshot reports the starting configuration (gray robot configuration), and the planned path (green line) to reach the desired pose (green robot configuration). Furthermore, the occupancy map of the environment is shown where black cells, where the landmark lie, denote areas with maximum cost, while grey cells denote areas close to obstacles where navigation should be avoided. The following screenshots report the trajectory followed by the robot, to finally reach the desired goal.



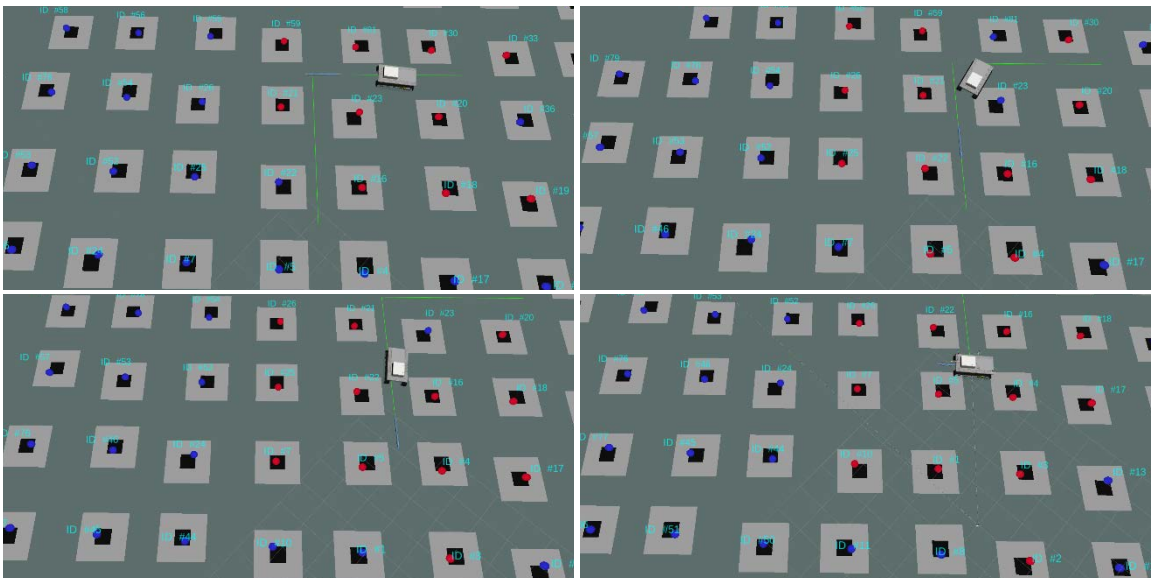


Figure 55 – Example of navigation sequence obtained using the global planner. The starting configuration is reported in the first picture along with the desired goal (green robot) and the planned path (green line)

The accuracy of the global planner was validated using a similar approach as for pose regulation accuracy, i.e., we selected five distinct points and, for each, assessed the errors from four starting configurations. To determine these starting configurations, we once again considered the grid layout of the field, this time opting for cells along the diagonals that were two steps away from the cell containing the desired state. The exact position within these cells, as well as the orientation of the robot, were chosen randomly. A visual representation of this selection process is provided in Figure 56.

This project is funded by the European Union G.A No 101016906

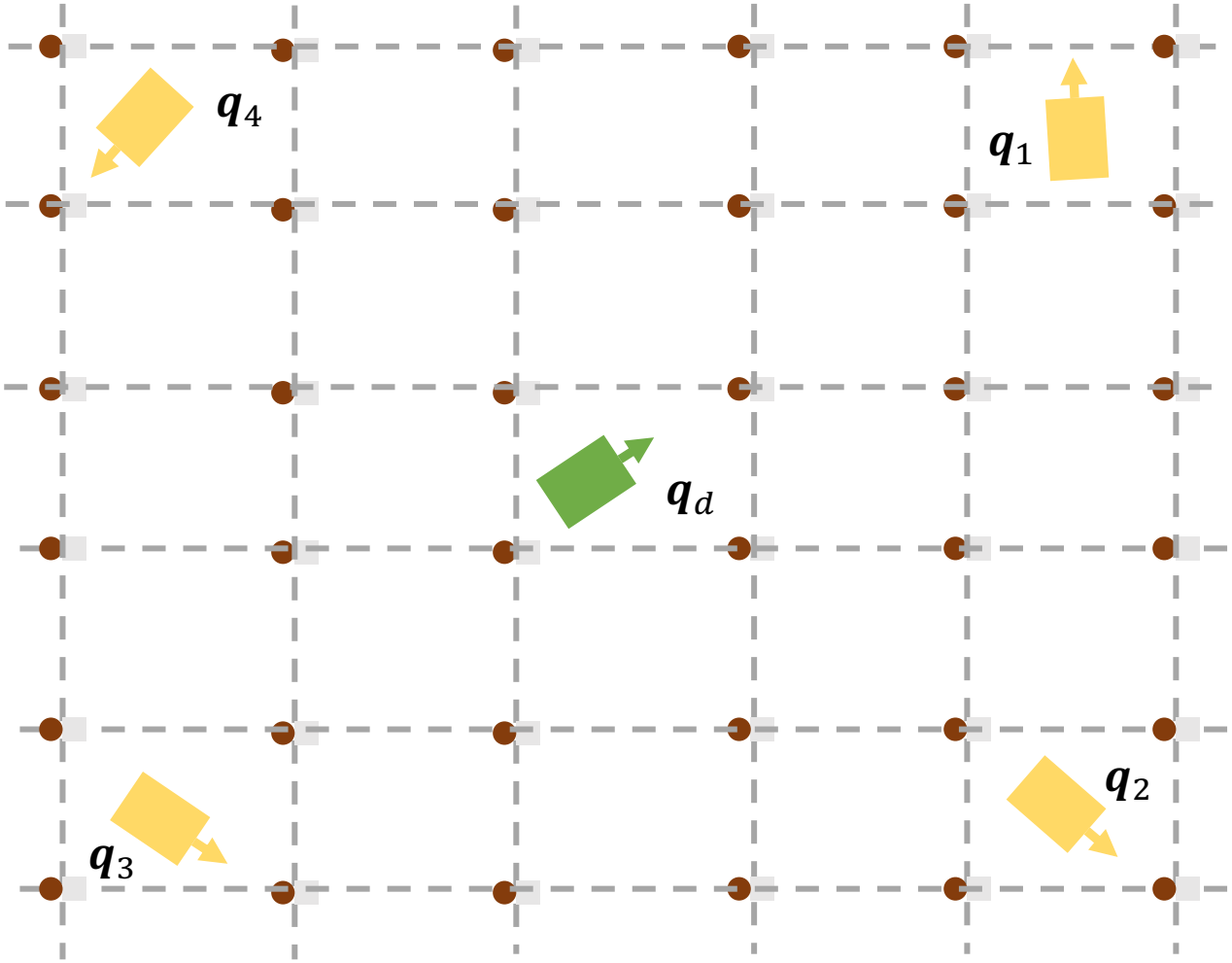
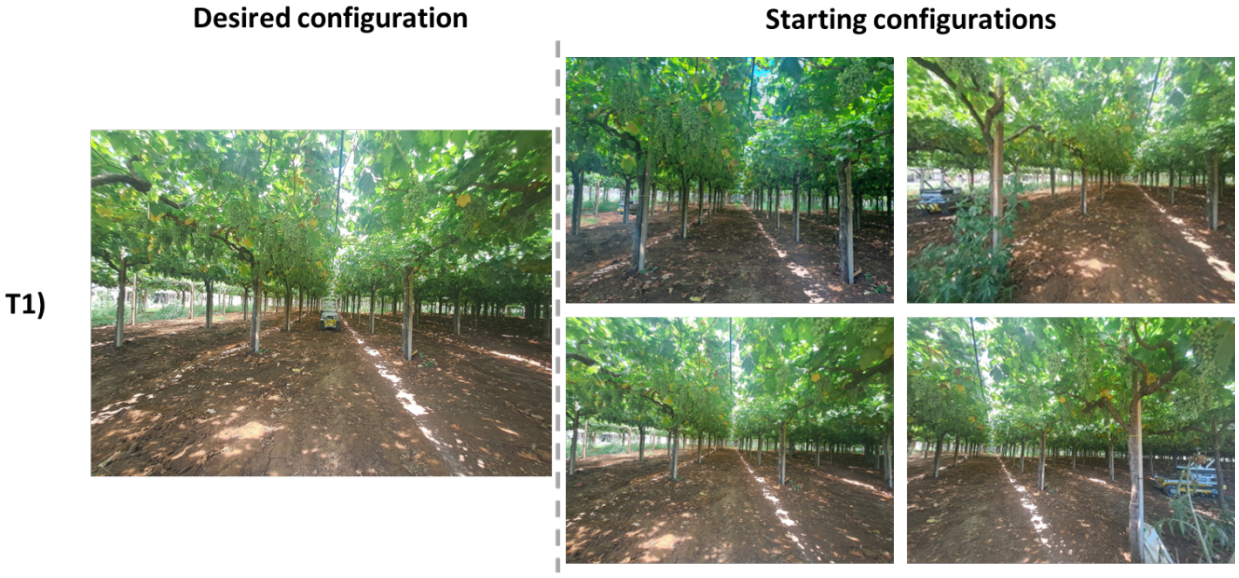


Figure 56 - Depiction of the configuration selection for evaluating the global planner accuracy

The pictures of the five considered desired configurations, along with the respective starting configurations, are shown in Figure 57.



This project is funded by the European Union
G.A No 101016906

T2)











T3)











T4)













Figure 57 - Pictures related to the five tests for evaluating the global planner accuracy

The accuracy results are summarized in Table 9, where the top row reports the statistics related to the positional errors while the bottom row to the orientation errors. Results similar to the local planner and controller are registered. Specifically, the positional error is equal to 0.0526 m on average with standard deviation of 0.0336 m and reaches minimum and maximum values equal to 0.0062 m and 0.1340 m, respectively. Regarding the orientation, an average error equal to 0.0357 rad (i.e., $3.0109°$) is obtained, with standard deviation equal to 0.0335 rad, while minimum and maximum values are equal to 0.0022 rad and 0.0962 rad, respectively, confirming the effectiveness of the proposed approach.

Table 9 - Results of the accuracy for the global planner

Error	Mean	Median	Std. dev	Min.	Max.
Positional	0.0526 [m]	0.0505 [m]	0.0336 [m]	0.0062 [m]	0.1340 [m]
Angular	0.0357 [rad]	0.0195 [rad]	0.0335 [rad]	0.0022 [rad]	0.0962 [rad]

This project is funded by the European Union G.A No 101016906

7 Coordinated Maneuvering

The coordinated maneuvering represents the fundamental step to accomplish the box exchange between farming and logistic robots. According to the BEM described in Section 2.2.2, the two robots must be aligned along their sagittal axis and oriented in such a way that the front parts face each other, as illustrated in Figure 58.



Figure 58 - Alignment for box exchange

In the following, the perception system to accurately locate the farming robot is first reported. Then, the methodology to realize the alignment and exchange the boxes is described and finally the experimental results are discussed.

7.1 Perception system for farming robot localization

To facilitate the docking procedure between the logistics and farming robots during the box exchange, it is crucial to detect the relative pose between the two robots. There are various sensors and associated data available on the robots that can be utilized for this purpose. A docking perception system was developed and implemented, which involves a RGB camera and fiducial markers, as described in Section 2.2.2. This system was deployed to support the design of the alignment planning strategy, which is the primary objective of Task T7.1. The AprilTag algorithm and its library of fiducial

markers were employed for this purpose. Additionally, the ROS wrapper package[4] was utilized in conjunction with the system.

The fiducial markers used in the system are black and white visual markers designed to assist the algorithm. The algorithm initially calculates the gradient at each pixel to extract line segments and then employs a graph-based clustering method to identify the quadrilateral regions of the tag. By knowing the physical size of the tag in the real world and successfully detecting the appropriate tag, the pose of the tag relative to the camera can be determined using homography and extrinsics estimation. For more in-depth information about the AprilTag system, please refer to the reference (Wang & Olson, 2016).

The AprilTag ROS wrapper offers detection data, specifically the relative pose between the tag and the camera, in a format compatible with ROS. In ROS, all coordinate frames are interconnected in a tree structure known as the tf-tree. Figure 59 illustrates a simplified version of the tf-tree, showcasing the Alitrak robot, LIDAR (os_sensor), camera, and a detected AprilTag marker.

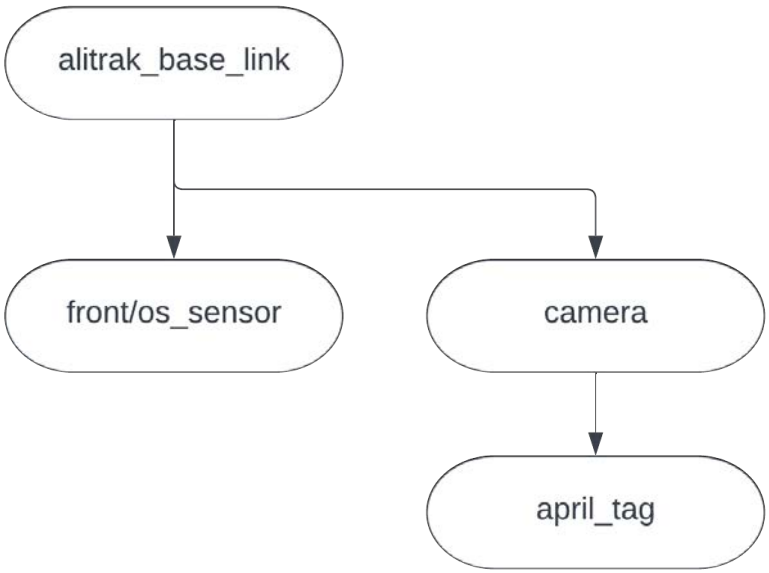


Figure 59 - Coordinate frame transformation tree without odometry

In a docking procedure where the docking target remains mostly stationary, such as when the logistics robot is performing the docking with the farming robot, relying solely on relative pose detection from a potentially noisy system with occasional occlusion is not practical. Notably, the robot is equipped with the SLAM algorithm which combines several sensors, including IMU, RTK GNSS, and 3D LIDAR, to provide a high-quality estimation of the robot's localization.

This motion information is encoded in the tf-tree through the relationship between the odometry frame and the base frame. The pose of the robot (alitrak_base_link) in the odometry frame is smooth

[4] https://github.com/AprilRobotics/apriltag_ros

and continuous, without abrupt jumps, making it the frame in which most control algorithms are applied. By utilizing the tf-tree, the pose of the AprilTag can be calculated relative to the odom frame, as depicted in Figure 60.

When the docking target is stationary, this approach offers the advantage of reducing the impact of poor detection, such as outliers or temporary loss of the target due to occlusion. Even if the detection system loses track of the target, the last known pose of the target in the odom frame remains a valid target for the robot. Additionally, the robot continuously updates its own pose in the odom frame, enabling it to continue the docking procedure even if the target is temporarily lost.

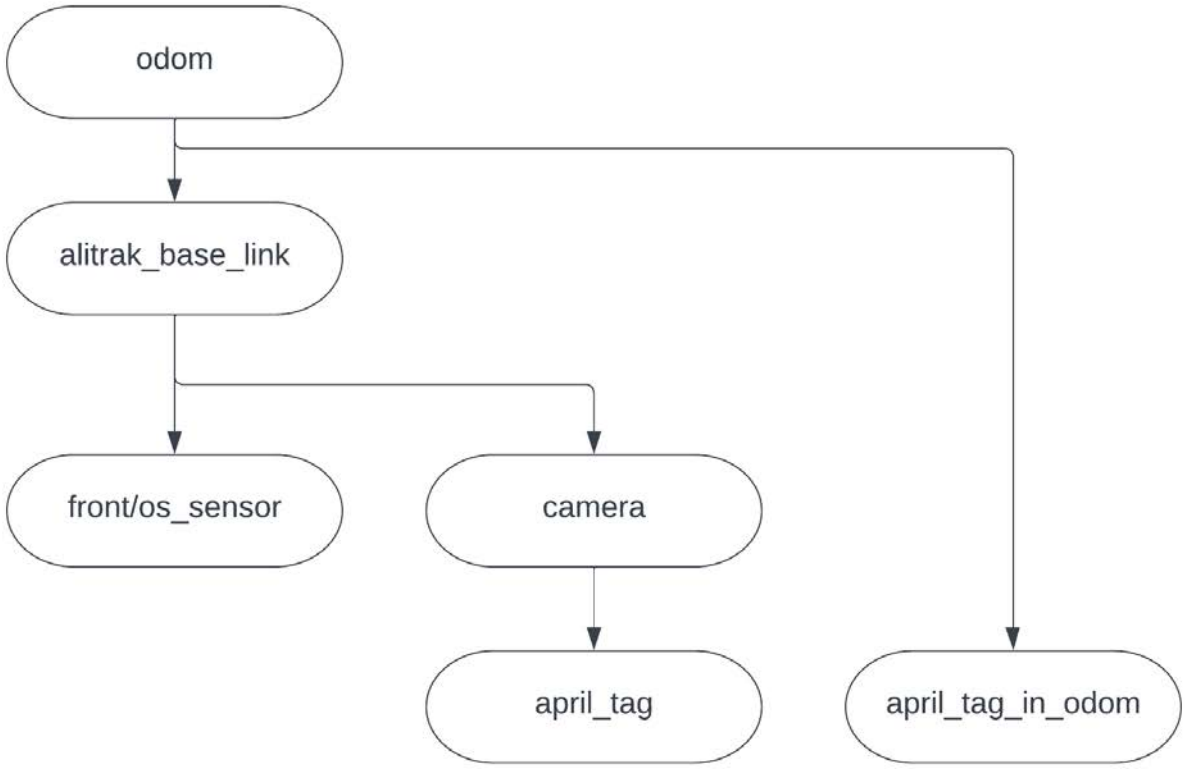


Figure 60 - Coordinate frame transformation tree with odometry.

After the pose information of the detected apriltags is transformed into the odometry frame of the robot, the measurements are fed to an Extended Kalman Filter (EKF) that is tuned for tracking a stationary target (that is the stationary farming robot). Indeed, using an EKF allows the system to handle sporadic data (e.g., when one of the apriltags is not visible) and outliers in a robust manner. The output of the EKF is the estimated pose of the farming robot that the logistics robot will use as the *docking target*.

7.2 Methodology

As outlined in Deliverable 7.1 “Coordinated Maneuvering for Box-Exchange Operation”, at a high level, the coordinated maneuvering is formulated as a navigation problem for the logistic robot. To elaborate further, the logistic robot task involves *i)* localizing the farming robot, *ii)* moving to a

This project is funded by the European Union G.A No 101016906

position in close proximity (shown in Figure 61-left), and *iii)* ultimately executing the docking procedure (shown in Figure 61-right). More in detail, the first pose which is reached by the logistic robot, denoted as $\boldsymbol{q}_1^D$, is aligned with the sagittal axis of the farming robot, with the front pointing towards the latter and positioned at a distance d_1^D. Similarly, the second desired pose of the logistic robot, denoted as $\boldsymbol{q}_2^D$, is aligned with the farming robot, also with the front facing the latter, but at a closer distance d_2^D, with $d_2^D < d_1^D$, to enable the box exchange between the two platforms. We define the phase when the first pose $\boldsymbol{q}_1^D$ is reached as the *navigation phase*, and the subsequent phase, upon reaching the second pose $\boldsymbol{q}_2^D$, as the *docking phase*.



Figure 61 - Depiction of the coordinated maneuvering relevant phases

To localize the farming robot within the vineyard, we rely on two approaches, used in the two different phases of the maneuvering:

1. During the navigation phase, the map of the logistic robot is initially shared with the farming robot. This ensures both robots operate in the same coordinate system. Then, the estimated pose of the farming robot, as determined by its SLAM algorithm, is continuously communicated to the logistic robot via the Wifi Mesh network.
2. During the docking phase, the farming robot is in the field of view of the logistic one and its pose is estimated based on the specialized perception system described in the previous section to achieve greater localization accuracy.

Given the farming robot pose, the desired poses of the logistic robot $\boldsymbol{q}_1^D$ and $\boldsymbol{q}_2^D$ are computed according to transformation matrices as detailed in Deliverable 7.1.

To realize the entire coordination strategy for the box exchange a Finite State Machine (FSM), illustrated in Figure 62, has been designed.

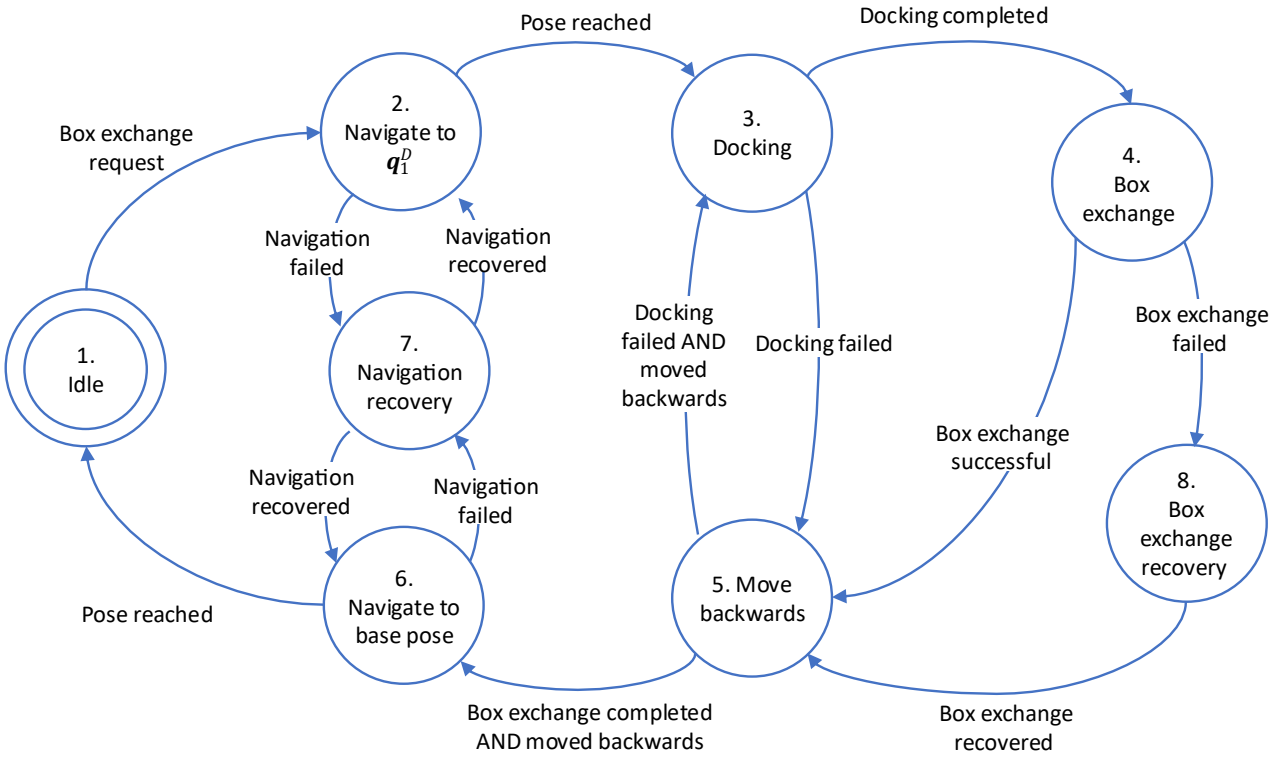


Figure 62 - FSM for the coordinated maneuvering and box exchange

The possible system states are the following:

1. *Idle,* representing the starting system state where the logistic robot is inactive, and the local planner state is set to "Idle".
2. *Navigate to* $\boldsymbol{q}_1^D$, where the logistic is tasked with reaching the first pose for the docking. The local planner module is configured in the "Navigation" state, while the global planner generates waypoints for the local planner. Additionally, the pose of the farming robot is acquired through direct communication with the Wifi mesh network.
3. *Docking,* where the logistic is tasked with reaching the second docking pose $\boldsymbol{q}_2^D$ (after having reached $\boldsymbol{q}_1^D$). In this case, the local planner is set to the "Docking" state to ensure a cautious maneuvering, and the desired pose is directly provided as input to the local planner, as the proximity of the logistic robot to the goal configuration does not require the use of the global planner. The pose of the farming robot is estimated using the perception system, with continuous updates. This continuous updating ensures that as the logistic robot approaches the farming robot, the perception system provides increasingly accurate information, allowing for real-time adjustments to the docking pose as needed.
4. *Box exchange,* where the actual exchange of the boxes from one platform to the other is realized activating the BEM conveyors. In this state, no motion of the mobile base is foreseen.
5. *Move backwards,* where the logistic moves backwards from pose $\boldsymbol{q}_2^D$ to $\boldsymbol{q}_1^D$. The local planner is set to the "Out-of-docking" state, and the desired pose is directly provided as input to the local planner.

This project is funded by the European Union G.A No 101016906

6. *Navigate to base position,* where, after the completion of the box-exchange, the logistic robot returns to a predefined base pose. The local planner module is configured in the "Navigation" state, with waypoints provided by the global planner.
7. *Navigation recovery,* this state is reached in case the navigation fails. An additional attempt to reach the desired pose is performed and, in case of failure, the system enters a paused state, awaiting human intervention.
8. *Box exchange recovery,* this state is reached in case the BEM fails. Three attempts are made to exchange the box. In case the box exchange process does not succeed after three tries, human intervention is required. Figure 63 shows an instance of a box exchange operation that initially failed but was successfully recovered through a retry. In this scenario, the box became stuck in an intermediate position between the two conveyors. Restarting the farming conveyor enabled the successful transfer of the box.



Figure 63 - Example of failed box-exchange operation that has been recovered by re-attempting the exchange

The flow among the states is the following. The system starts in the idle state. Upon receiving a box exchange request, it transitions into the navigation state to reach the pre-docking pose, i.e., $\boldsymbol{q}_1^D$**.** In case of failure, it enters the navigation recovery state until recovery is accomplished. Upon successful navigation, the subsequent docking state is entered. If docking fails, the robot moves backwards (and enters the respective state) and then restarts the docking phase. This backward-forward motion aids in leveling the ground, thereby enhancing the chances of achieving accurate positioning. When docking is completed, i.e., the pose $\boldsymbol{q}_2^D$ is reached, the system is ready to initiate the box exchange. By exploiting the BEM sensors, the status of the box exchange operation is recognized. In case of failure, the box exchange state is entered until recovery is accomplished. In case of success, the robot moves backwards to reach again pose $\boldsymbol{q}_1^D$ in a safe manner. Afterwards, the robot is tasked with navigating to the predefined base pose. If any failure during the navigation occurs, the navigation recovery state is entered until recovery is achieved.

Throughout all phases, the state of the bumpers is constantly monitored. As soon as the activation of a bumper is detected (indicating a collision), the local planner is switched to the "Idle" state, and a system failure is signaled.

7.3 Experimental validation

The experimental analysis of the coordinated maneuvering presented below is divided into three parts: validation of the perception system, validation of the maneuvering process without a box exchange, and validation of the complete pipeline, which includes the box exchange.

7.3.1 Perception validation

The perception validation process consisted of two stages: laboratory validation at the DTI premises and experimental field validation in May 2023. This deliverable will primarily focus on the field validation, as the laboratory validation was already described in Deliverable 7.1.

During the field validation, the docking perception module was integrated with the rest of the navigation stack. However, this integration posed challenges in terms of system resources, as both applications were resource intensive. To address this, several data sequences were collected from the field with different combinations of framerates and image resolutions, while simulating the motion of a docking operation. The analysis conducted shed light on the impact of changes in image resolution and sensor framerate on system performance. This evaluation resulted in identifying a combination that achieves a favorable balance between overall system accuracy and resource allocation.

Criteria were assessed by aiming to reduce computational resources while minimizing inconsistencies in april_tag pose detections. The outcomes of the experiments are presented in Table 10. Based on the experimental validation, it was noticed that the use of limited image resolution can decrease the detected tag distance and the accuracy of absolute detected distance. However, the overall disparities are negligible because higher image resolution is only necessary for tag detection at distances beyond what is needed for the docking process as a whole. Additionally, the EKF filter effectively handles the minor detection inconsistencies, resulting in no impact on the accuracy of docking.

Table 10 - Outcomes of the experimental evaluation pertaining to the stability and performance of the overall perception system.

Framerate	Image resolution	
	1280 x 720	1920 x 1080
6	62.5%	88,7%
15	102%	175%
30	172%	181%

This project is funded by the European Union G.A No 101016906

Therefore, an effective choice of the parameters (6fps, 1280 x 720 image resolution) was made to ensure a smooth operation of the entire system, without compromising its overall performance.

Due to the reduced frequency of detections from the AprilTag detector, the implemented extended Kalman filter used for tracking individual AprilTags in the odometry coordinate frame needed to be adjusted, as well as the calculated offset between the robot center and the physical AprilTags mounted on the robot. In particular, these modifications were necessary to compensate for the decreased detection frequency and maintain accurate tracking. The recommended parameters were then tested and validated in the field thus concluding the implementation of the docking perception system.

7.3.2 Maneuvering validation

The FSM depicted in Figure 62 has been implemented using Python and resorting to the library SMACH[5], which allows to create hierarchical state machines fully integrated with ROS middleware. The action-and-topic ROS interface of the planner was utilized in all phases to obtain acknowledgments and facilitate goal updates.

Based on experimental trials, we established the desired distances between the two representative points of the bases as follows: $d_1^D = 3.5$ m and $d_2^D = 1.6$ m. The latter distance ensures that the front aluminum bars of the robots are positioned at a distance of approximately 0.33 m. In this regard, a distance less than 0.24 m would result in bumper collisions, as depicted in Figure 64. Conversely, a distance of roughly 0.36 m, shown in Figure 65, has been confirmed to allow successful box exchange when the platforms are properly aligned. Hence, we opted for a distance that is slightly smaller than 0.36 m but still greater than the activation threshold for the bumpers.

[5] http://wiki.ros.org/smach

This project is funded by the European Union G.A No 101016906

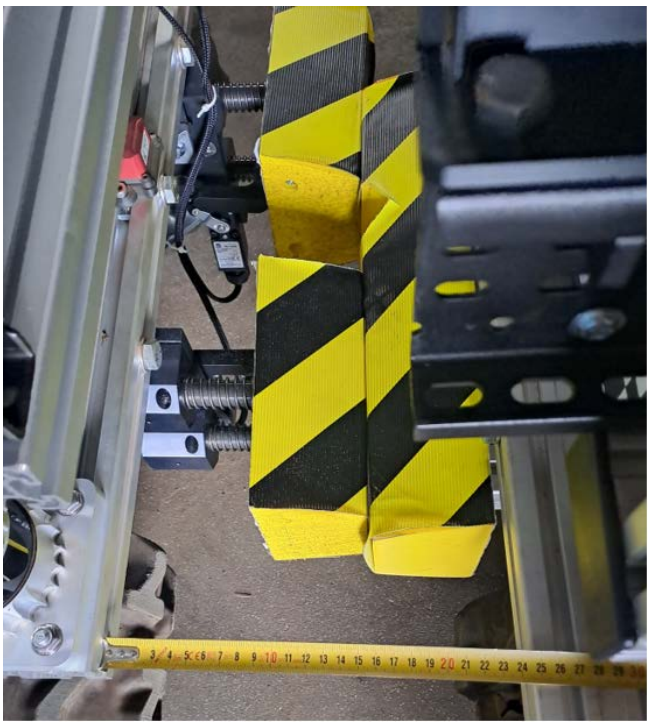


Figure 64 - Distance for bumper activation



Figure 65 - Distance for box exchange

To validate the maneuvering operation, we resorted to the respective benchmark in Deliverable 2.3 "Dimensioning of the Real-World Scenario for Final Demo" that is "Alignment of logistic and farming robots for box exchange with a success rate of at least 80%". Based on the experimental results of the BEM, we translated such requirement into reaching distance between the front aluminum bars of the robots in the range $[0.26,0.36]$ m and orientation misalignment in the range $[0,5]°$. We conducted 15 experiments where we measured the quantities highlighted in dotted blue lines in Figure 66, where A and B represent the two corner points of the front aluminum bar of the logistic

robot and C and D are of the farming robot. We denote the lengths of the segments connecting these points as P_1P_2 with $P_1, P_2 \in \{A, B, C, D\}$.

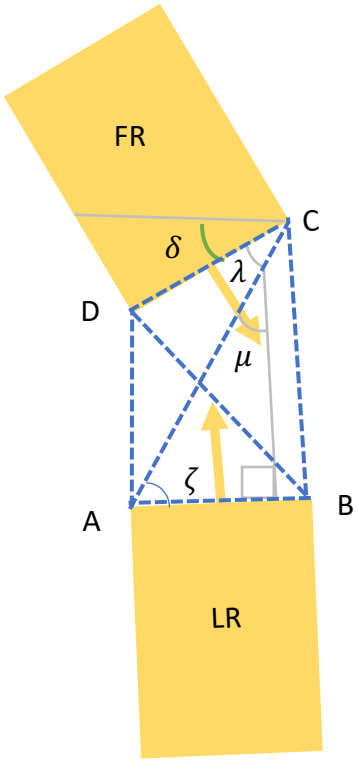


Figure 66 - Depiction of the quantities of interest for the evaluation of the coordinated maneuvering

The quantities of interest to evaluate the coordinated maneuvering accuracy are the orientation misalignment δ and the average distance $(\overline{AD} + \overline{BC})/2$. Auxiliary angles have been introduced to derive the angle δ given the measurements. Specifically, it holds

$$\zeta = \text{acos}\frac{\overline{AC}^2 + \overline{AB}^2 - \overline{CB}^2}{2\,\overline{AC}\,\overline{AB}},$$
$$\mu = \frac{\pi}{2} - \zeta,$$
$$\lambda = acos\frac{\overline{DC}^2 + \overline{AC}^2 - \overline{DA}^2}{2\,\overline{DC}\,\overline{AC}},$$
$$\delta = \frac{\pi}{2} - (\mu + \lambda).$$

Different starting configurations of the farming and logistic robots were tested as shown in Figure 67.

This project is funded by the European Union
G.A No 101016906

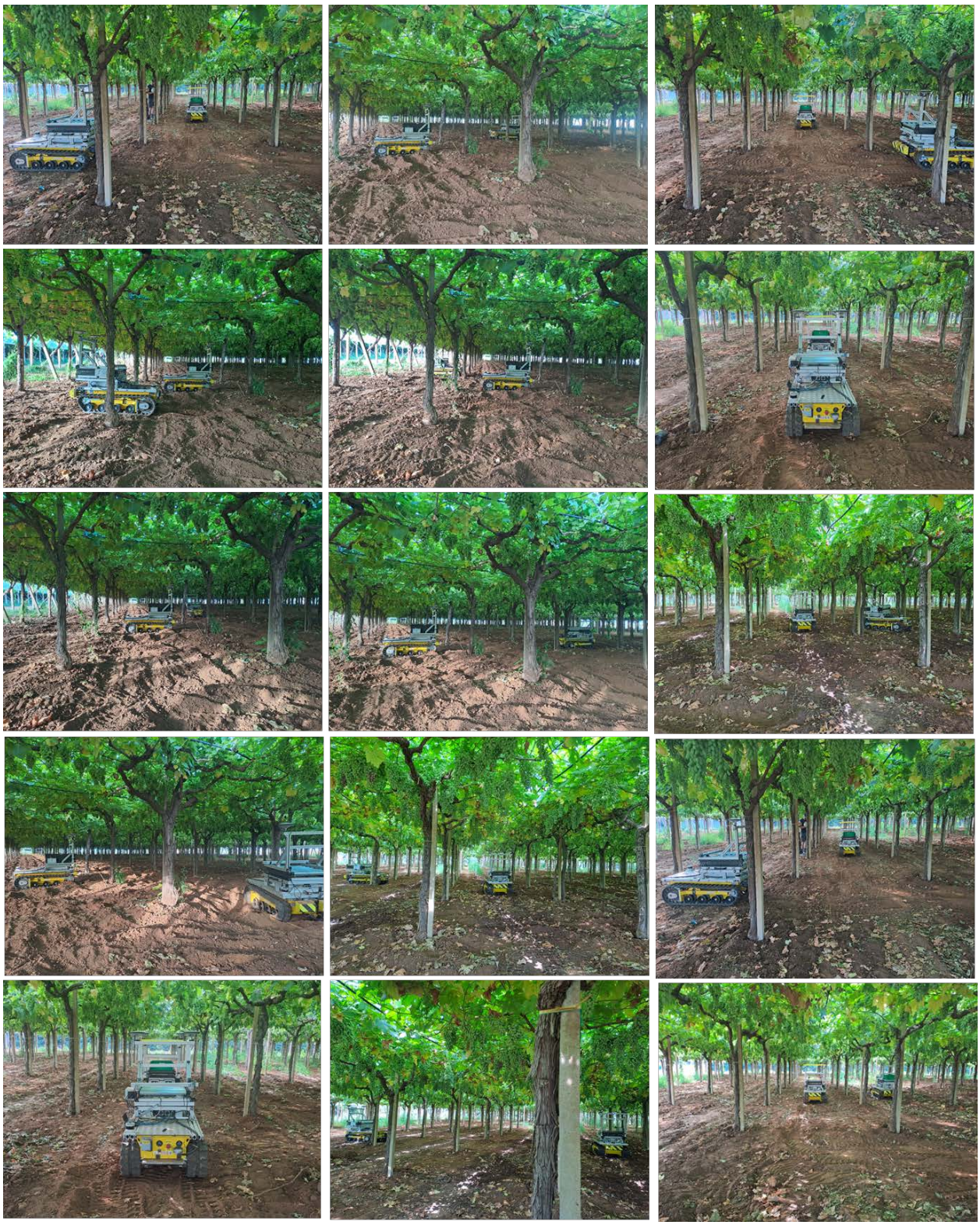


Figure 67 - Starting configurations for the validation of the coordinated maneuvering

Examples of final configurations reached during the experiments are provided in Figure 68.



Figure 68 - Examples of final configurations

The experimental results are collected in Table 11. These confirm the fulfillment of the benchmark requirements, with the maneuvering succeeding 87% of the tests. Specifically, average orientation misalignment and distance equal to $1.43°$ and 0.3122 m have been recorded, respectively. Note that these results not only validate the coordinated motion, but also implicitly validate the effectiveness of the perception system which allows to accurately localize the farming robot.

Table 11 - Results for the coordinated maneuvering accuracy

Test	Orientation misalignment [deg]	Distance [m]
1	0.57	0.3185
2	0.16	0.3130
3	0.24	0.3435
4	0.81	0.2700
5	2.71	0.3115
6	1.79	0.2630
7	2.85	0.3685
8	2.42	0.3340
9	2.18	0.3395

10	2.1	0.2930
11	1.46	0.2510
12	1.21	0.3275
13	0.08	0.3235
14	0.81	0.3750
15	2.1	0.2520
Average:	**1.43**	**0.3122**
Std dev:	**0.95**	**0.0393**

7.3.3 Experiment with overall pipeline

The overall integrated system, including the box exchange operation, has been validated as reported in the accompanying video. In this experiment, two human operators work within the vineyard while the logistic robot is tasked with realizing the box exchange operation. This implies that the safety of the human operators must be guaranteed while they move within the vineyard. Relevant screenshots of the experiment are reported in Figure 69, where the FSM state in each is highlighted. Specifically, in the initial configuration illustrated in Figure 69.1, which shows the presence of a human operator denoted as H1, the request for box exchange is triggered. The system transitions into the state "Navigate to $\boldsymbol{q}_1^D$" and plans the trajectory to reach the desired pose. However, a second human operator, denoted as H2, moves within the vineyard, and obstructs the trajectory (Figure 69.2). This leads the robot to re-plan the trajectory in real-time to ensure human safety (Figure 69.3). Even after the human operator H2 moves away, a further online re-planning is observed due to the motion of the human H1 (Figure 69.4-Figure 69.6). When also the human H1 moves away from the robots working area (Figure 69.7), the logistic robot reaches the desired pose $\boldsymbol{q}_1^D$ (Figure 69.8) and the system transitions into the state "Docking", activating the respective cautious maneuvering. Upon reaching the docking desired pose $\boldsymbol{q}_2^D$ (Figure 69.9), the box exchange state is entered and the BEM conveyors activated accordingly, resulting in the box transitioning from the farming to the logistic robot (Figure 69.10). Once the box exchange is completed, the robot moves backwards (Figure 69.11) and finally returns to the base pose (Figure 69.12). Further re-plannings are observed in this last phase, which are not reported in this document for the sake of brevity. A further example of the overall box exchange pipeline in the absence of human operators is provided in the accompanying video.

This project is funded by the European Union G.A No 101016906

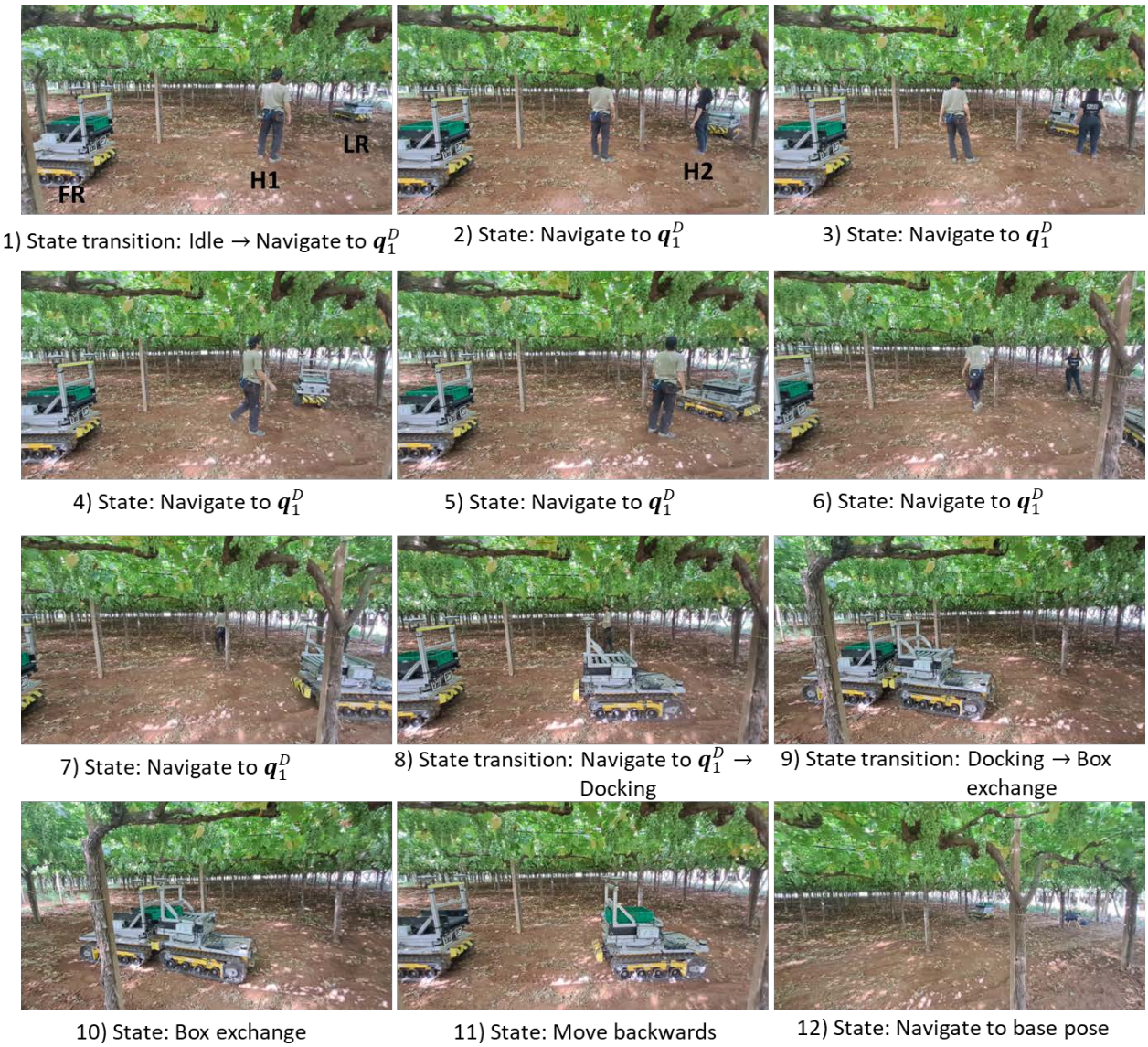


Figure 69 - Screenshots of the overall experiment

This project is funded by the European Union G.A No 101016906

8 References

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (p. 226--231).

Montemerlo, M., & Thrun, S. (2003). Simultaneous localization and mapping with unknown data association using FastSLAM. *IEEE International Conference on Robotics and Automation .*

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, 52-57.

Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *IEEE robotics & automation magazine, 13*(3), 108-117.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., . . . Leonard, J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics, 32*(6), 1309-1332.

Brink, K. M. (2017). Partial-update schmidt–kalman filter. *Journal of Guidance, Control, and Dynamics, 40*(9), 2214-2228.

Wang, J., & Olson, E. (2016). AprilTag 2: Efficient and robust fiducial detection. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*

Dalitz, C., Schramke, T., & Jeltsch., M. (2017). Iterative Hough transform for line detection in 3D point clouds. *Image Processing On Line, 7*, 184-196.