Ref. Ares(2024)4713794 - 30/06/2024

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: D9.3 – V1.0

Human-multi-robot task planning field validation

Dissemination Level	PUBLIC
Submission Date	2024-06-30
Work Package	WP9
Task	T9.6
Type	Report
Version	1.0
Authors	Victor Nan Fernandez-Ayala, Mani Hemanth Dhullipalla
Approved by	UNIROMA3

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 contained therein.

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

Executive Summary

This report describes the multi-robot task planning and control framework developed within CANOPIES project to enable agronomic collaboration between humans and logistic/farming robots that operate together in a shared physical workspace. Specifically, this document describes the temporal logic-based approach which allows various agronomically relevant tasks and high-level spatio-temporal robot behavior to be encoded in terms of human interpretable semantics. We then describe how such specifications can be translated into low-level planning and control actions through computationally efficient algorithms for real-time replanning. The validation of these algorithms is performed in a number of ways: through CANOPIES simulator, hardware experiments in indoor lab, and, finally, through experiments in field using CANOPIES robots. We describe the details of these field validation in this report.

Table of Contents

1 Introduction . . . 5
1.1 Purpose of document . . . 5
1.2 Objectives of T9.6 . . . 5
1.3 Related works . . . 5
1.4 Document overview . . . 6
2 Preliminaries . . . 7
2.1 Linear temporal logic (LTL) . . . 7
2.2 Robot tasks . . . 7
2.3 Software architecture . . . 7
2.4 LTL core & planner . . . 8
2.4.1 LTL to nondeterministic Büchi automaton (NBA) . . . 8
2.4.2 Agent model . . . 9
2.4.3 Product automaton & plan generation . . . 9
2.5 Agent-level software . . . 9
2.6 Low-level navigation stack . . . 10
2.7 Human-in-the-loop (LTL) . . . 10
3 Adapting to agronomic tasks . . . 11
3.1 Online replanning . . . 11
3.2 Designing state transitions . . . 11
3.3 Agronomic task specifications for CANOPIES . . . 11
4 System architecture . . . 13
4.1 Simulation . . . 13
4.2 Actual robot . . . 13
5 Code Implementation . . . 15
5.1 ROS LTL planner . . . 15
5.2 Agronomic task planning . . . 15
5.3 Integrating planner and navigation stack . . . 17
5.4 VR environment and simulations . . . 19
5.4.1 HIL simulations . . . 21
6 Hardware experiments in the field . . . 22
6.1 Setup (as of June 19th 2023) . . . 23
6.2 NUCs and communication network . . . 23
6.3 Demonstration of replanning after disturbance . . . 26
6.4 Acknowledgement . . . 28
7 Implementational challenges . . . 30
7.1 Difference in the orientation of reference frames . . . 30
7.2 Non-uniform poles in the field . . . 30
7.3 Communication challenges . . . 31
8 Benchmarks for validation . . . 32
9 Conclusions . . . 35

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

Abbreviations and Acronyms

WP	Work Package
LTL	Linear Temporal Logic
STL	Spatial Temporal Logic
CBF	Control Barrier Functions
ROS	Robot Operating System
AP	Atomic Proposition
wFTS	Weighted Finite Transition System
NBA	Nondeterministic Büchi Automaton
PBA	product Büchi Automaton
NUC	Next Unit of Computing
VR	Virtual Reality
HIL	Human-In-the-loop
MPC	Model Predictive Control
SLAM	Simultaneous Localization and Mapping
MIC	Mixed-Initiative Control
GPS	Global Positioning System
RTK	Real-Time Kinematic

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

1 Introduction

A key requirement for enabling natural and seamless human-robot collaboration in performing complex tasks within the agriculture sector is the ability to provide abstract real-time mission specifications to the robots in a manner that is easily interpretable to human collaborators, and yet unambiguous to the robots. Such requirements are particularly common in a precision agriculture setting, wherein robot tasks are dynamically determined by the human collaborators and communicated in real time.

Over the past decades, Linear Temporal Logic (LTL) has emerged as a conventional method for specifying the complex behavior of robotic systems. It is particularly effective since it provides rich syntax to encode complex tasks into LTL formulas that are close to natural language, facilitating their utilization, for instance, with speech-based commands.

1.1 Purpose of document

This document presents the theoretical and validation results of Task 9.6 ”Field validation of the task planning for human-multi-robot systems”.

1.2 Objectives of T9.6

A key objective of the CANOPIES project is to develop robot control design and planning framework that takes into consideration humans in the shared workspace and other dynamically changing set of mission requirements while ensuring that the farming and logistic robots perform the agronomic tasks pre-assigned to them. These tasks are often specified to robots in an abstract manner (LTL formulas). Such high-level specifications make them easily interpretable and natural to us as designers; one can systematically and formally extract robot plans in terms of a sequence of state-transitions that lead to satisfaction of the LTL formulas. The report WP8.3 describes the technical details and research contributions of temporal logic-based robotic task planning approach with human inputs/interference in the loop.

The primary objective of Task 9.6 is to demonstrate, through field validation, the operation and effectiveness of the temporal-logic based planning algorithms developed under Tasks 8.2 and 8.3. These experiments take place in environments where humans also play a dynamic role as supervisors through human-in-the-loop (HIL) techniques, and/or as dynamic obstacles in the field. The report presents details on i) adaptation and implementation of LTL planners (with HIL), ii) on the hardware setup for the experiments, and iii) the benchmarks considered for validation.

1.3 Related works

The agriculture sector, which traditionally has a lower technological level, is seeing a recent modernization with the rise of AI, Big Data and robotics [1]. The most widespread integration of these technologies nowadays is in the field of UAVs, which can be used to automate the process of inspection of crops [2] and to facilitate spraying of liquids like water, fertilizer, herbicides, and pesticides [3]. One of the main benefits of increasing automation is that large teams of robots can be used to cover a larger area in a more cost-efficient manner, e.g., through drone swarms [4]. In addition, the development of specific grippers for pruning [5] and vision algorithms for the detection of complex clusters of fruits [6], together with mobile platforms capable of traversing the challenging environment of agricultural fields [7], and the recent societal interest on

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

advancing the sector through different funding opportunities and competitions [8], has allowed researchers to start considering more complex tasks related to precision agriculture. In this context, some examples include possible interactions among humans and robots which need to happen in a safe manner [9], immersive teleoperations [10] and socially-aware robot navigations [11].

1.4 Document overview

The rest of this document is structured as follows: common preliminaries of this deliverable are detailed in Section 2, including the contextualization of the research lines presented here within the general scenario for WP8 and some common definitions.

Section 3 shows how the LTL formulation can be adapted for the specific agronomic tasks of the CANOPIES project.

Section 4 focused on the system architecture for both the actual robots in the field and the simulator environment.

Section 5 details the code implementation of our planner, the structure and key components of the core planner, along with software dependencies, as well as the simulations that have been performed in the VR environment.

Section 6 describes the hardware experiments on the field and the actual demonstration.

Section 7 focuses on the issues we found during the execution of the experiments and what actions were taken to fix them.

Lastly, Section 8 shows the validation of our field results with respect to the original benchmarks developed.

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

2 Preliminaries

2.1 Linear temporal logic (LTL)

An LTL formula over a set of atomic propositions AP is recursively defined as

$$\varphi ::= true \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc \varphi \mid \varphi_1 U \varphi_2,$$

where $a \in AP$ and $\bigcirc$ (next), U (until). The Boolean connector disjunction $\vee$, and temporal operators $\Diamond$ (eventually) and $\Box$ (always) can be defined as $\varphi_1 \vee \varphi_2 := \neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\Diamond\varphi := \top U \varphi$ and $\Box\phi := \neg\Diamond\neg\phi$. LTL formulas are interpreted over infinite words made of subsets of AP, i.e., over $(2^{AP})^\omega$. The full semantics and syntax of LTL can be found in [12].

The satisfaction of a formula φ at position $i \in \mathbb{N}$ of word ω, denoted by $\omega(i) \vDash \varphi$, is defined recursively as follows:

$$\begin{aligned}
\omega(i) \vDash a &\Leftrightarrow a \in \omega(i), \\
\omega(i) \vDash \neg\varphi &\Leftrightarrow \omega(i) \nvDash \varphi, \\
\omega(i) \vDash \varphi_1 \wedge \varphi_2 &\Leftrightarrow \omega(i) \vDash \varphi_1 \text{ and } \omega(i) \vDash \varphi_2, \\
\omega(i) \vDash \bigcirc\varphi &\Leftrightarrow \omega(i+1) \vDash \varphi, \\
\omega(i) \vDash \varphi_1 U \varphi_2 &\Leftrightarrow \exists j \geq i \text{ such that } \omega(j) \vDash \varphi_2 \text{ and} \\
&\quad\ \forall k \in [i, j), \omega(k) \vDash \varphi_1.
\end{aligned}$$

A word ω satisfies an LTL formula φ, written as $\omega \vDash \varphi$, if $\omega(1) \vDash \varphi$.

2.2 Robot tasks

The robot is assigned a task φ, which is expressed as an LTL formula and is specified with the following structure:

$$\varphi = \varphi^{hard} \wedge \varphi^{soft} \tag{1}$$

where φ^{hard} and φ^{soft} are respectively the hard task and soft task LTL formulas that define the high level task. The hard task φ^{hard} defines specifications that should strictly be satisfied, such as safety requirements. The soft task φ^{soft} could include optional tasks for which satisfaction is less stringent and could be violated if incompatible with the hard task.

2.3 Software architecture

ROS consists of a node-based architecture within which the proposed planner is integrated (see Fig. 1). The LTL (core) planner node computes the plans for each robot agent based on the LTL specification, the agent model, and its environment. The agent model is represented as a weighted finite transition system (wFTS), which is a tuple $\mathcal{T}_i = (Q_i, \Sigma_i, \rightarrow_i, q_0^i, AP_i, L_i, W_i)$ where $Q_i = \{q_1, ..., q_{N^t}\}$ is the finite set of states; Σ_i is the set of actions, $\rightarrow_i \subseteq Q_i \times \Sigma_i \times Q_i$ is the transition function; q_0^i is the set of initial states; AP_i is the set of atomic propositions for $\mathcal{T}_i$; $L_i : Q_i \rightarrow 2^{AP_i}$ is the labeling function; $W_i : Q_i \times \Sigma_i \times Q_i \rightarrow \mathbb{R}^+$ is the

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

weight function as cost of transition in $\rightarrow_i$. The planner node uses the wFTS state from the agent node to either update the plan or output the next action command to be executed. Each agent has a specific node to communicate with the planner node and convert the high level plan to agent-specific low-level commands.

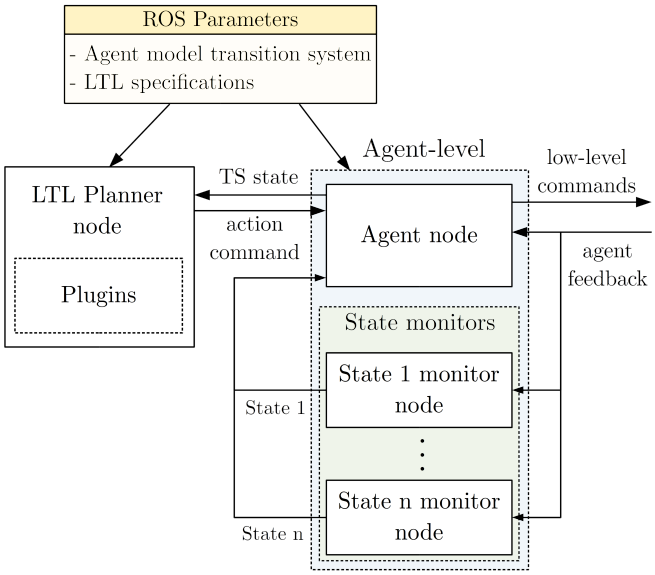

Figure 1: LTL automaton stack

Separating the LTL core planner from the agent-specific node facilitates integration on a wide range of robot platforms, including mobile robots and multi-DOF robot manipulators. The planner relies on a finite Büchi automaton translation of the LTL specification and outputs high-level commands (`string` type) to the agent node. These high-level commands are converted into low-level inputs for the specific agent. State monitors then map the agent physical state to the abstract wFTS state. The associated documentation provides information on how the LTL formula and the agent model wFTS are defined with example code [13].

The wFTS is shared as a ROS parameter and accessible by any ROS node (see Figure 1) [14]. A plugin functionality allows additional code to run within the planner node, providing a modular way of implementing new features, such as HIL features (see [15] for more details).

2.4 LTL core & planner

The core package contains the general planner that is not agent-specific. The planner node takes as input an LTL task φ in the form of (1), as well as an agent model wFTS.

2.4.1 LTL to nondeterministic Büchi automaton (NBA)

First, the hard and soft LTL formulas φ^{hard} and φ^{soft} are used to generate the respective NBAs [16] $\mathcal{B}_{\varphi^{hard}}$ and $\mathcal{B}_{\varphi^{soft}}$ via the LTL2BA software[17]. Then, the planner generates a combined Büchi automaton $\mathcal{B}_{\varphi} := \mathcal{B}_{\varphi^{hard}} \times \mathcal{B}_{\varphi^{soft}} = (S, S_0, 2^{AP}, \delta, F)$ using the safety-ensured product automaton developed in [18]. As part

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

of this safety-ensured product automaton, the design parameter $\beta \in \mathbb{R}_{\geq 0}$ is implemented, which adds a penalty for violating the soft task. Setting a higher value will help enforcing the soft task. Note that β is dynamic and can be updated online. The plan is published on a ROS topic, which is further described in the documentation [13].

2.4.2 Agent model

The agent model consists of a discretization of the agent's workspace into regions along with allowable transitions between regions, and other types of actions that can be performed (e.g., pick/drop). The agent modelling is based on the wFTS.

Multi-dimensional motion models and complex action models are handled by combining $N \in \mathbb{N}$ motion/action models together via the intersection of wFTSs. Guard functions G_i are used to identify allowable transitions between each wFTS, allowing for region-specific actions. We will refer to the ith motion /action model as the wFTS $\mathcal{T}_i$. The agent model is thus constructed from the intersection of all wFTS $\mathcal{T}_i, i \in \{1, .., N\}$.

The planner can be used with any type of agent as long as the state-space can be described as a finite transition system. Whatever the combination of $\mathcal{T}_i$, the LTL planner yields a sequence of actions such that each action is performed one-at-a-time. Only one dimensional state changes occurs by executing one action.

2.4.3 Product automaton & plan generation

The planner node uses the previously obtained NBA $\mathcal{B}_\varphi$ and the wFTS intersection $\mathcal{D}$ to generate a discrete plan. The code for plan generation is based on [19]. A product Büchi automaton (PBA) $\mathcal{A}_\mathcal{P}$ is defined as a tuple:

$$\mathcal{A}_\mathcal{P} = \mathcal{B}_\varphi \otimes \mathcal{D} = (\mathcal{S}_\mathcal{P}, \delta_\mathcal{P}, \mathcal{S}_{\mathcal{P},0}, \mathcal{F}_\mathcal{P}, \mathcal{W}_\mathcal{P})$$

where $\mathcal{S}_\mathcal{P} = S \times Q^P$; transition relation $(\langle s, q\rangle, \langle s', q'\rangle) \in \delta_\mathcal{P}$ iff $\exists \sigma \in \delta, s' \in \delta(s, \sigma)$ and $\exists \sigma_a \in \Sigma^P, (q, \sigma_a, q') \in \rightarrow_P$; the set of initial states $\mathcal{S}_{\mathcal{P},0} = S_0 \times Q_0$; the set of accepting states $\mathcal{F}_\mathcal{P} = (F \times Q^P)$; the weight function $W_\mathcal{P} : \delta_\mathcal{P} \rightarrow \mathbb{R}^+, W_\mathcal{P}(\langle s, q\rangle, \langle s', q'\rangle) = W_P(q, q')$. From this PBA, one can find an optimal run and project it back to the wFTS intersection $\mathcal{D}$ using model-checking methods [20].

Accepting runs have the following *prefix-suffix* structure: $r_\mathcal{P} = p_0, p_1 \cdots p_k(p_{k+1} \cdot \cdots \cdot p_n p_k)^\omega$, where $p_0 \in \mathcal{S}_{\mathcal{P},0}$ and $p_k \in \mathcal{F}_\mathcal{P}$. The output word has two distinct parts: a finite *prefix* part that is executed only once from the initial state p_0 to an accepting state p_k and a *suffix* part that is repeated infinitely from the accepting state p_k to itself[20]. Additionally, the optimal accepting run minimizes a cost function based on transition weights. A design parameter $\gamma \in \mathbb{R}_{\geq 0}$ adds weight on the cost of the suffix. A higher value for γ will therefore favor accepting runs with shorter suffixes.

The optimal accepting run has an associated input word, i.e., an action sequence, for the agent to execute to satisfy its specifications. This plan is also in the *prefix-suffix* structure. The planner node keeps track of the execution of the plan and outputs the next action command to the agent node.

2.5 Agent-level software

The agent-level nodes (see Fig. 1) are agent-specific (or even scenario-specific). The agent node is in charge of interpreting the action command (of `string` type) and sending the appropriate low-level commands to

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

the physical system. We denote by the low-level control $u_{r_i} \in \mathbb{R}^{m_i}$ the action to be implemented in the transition $\rightarrow_i$ for $m_i \in \mathbb{N}$ and $i \in \{1, ..., N\}$. Additionally, state monitors (working as independent nodes or integrated within the agent node) track the physical system configurations and relate them to states in the respective $\mathcal{T}_i$. The agent node then aggregates these states to a TS state and publishes it to the planner node.

2.6 Low-level navigation stack

The navigation and localization software stack developed in T3.5 by UNIROMA3 was used to implement the high-level actions related to motion provided by the LTL planner. The stack provides localization capabilities using the onboard Lidars with landmark-based SLAM, and navigation with a Model Predictive Control (MPC) motion planner for reaching positions in the field while avoiding collisions. The navigation stack also provides control subroutines for special cases, such as docking with another robot to exchange boxes.

A map was generated by the navigation and localization stack that contains information about static obstacles and relevant landmarks, and was later used by the LTL planner. The algorithm also returns the location of the robot with respect to the field reference frame during runtime for the LTL planner to use.

The MPC controller takes into account additional low-level locomotion-related constraints, such as minimizing rotations to avoid marking the terrain excessively and keeping a safe distance from the plants. The controller then calculates the trajectory for a given goal position, and returns status information. A detailed description of the navigation stack is available in Deliverable 9.1 – Field Validation of the Robot Navigation Functionalities.

2.7 Human-in-the-loop (LTL)

For the Human-In-the-Loop (HIL) context, we consider that the humans can 1) specify new tasks to the robot in real time and 2) take over the control of the robot. The former is achieved by assigning appropriate arguments/options when bringing up the system and choosing among a set of pre-specified formulas. The latter is achieved through a low-level scheme based on mixed-initiative control (MIC) formulation [21], such that

$$u \triangleq u_r(x) + k(x)u_h(t), \tag{2}$$

where $u_r(x)$ is the planner input at state x, $u_h(t)$ is the human input at time t, and $k \in [0, 1]$ is a smooth function indicating how close x is to a trap state. A $trap\,state$ is defined as a PBA state from which the Büchi acceptance condition cannot be fulfilled. The set of all trap states for an agent i is denoted by $\mathcal{O}_t^i$, which has an associated distance metric to track the proximity of the current state to $\mathcal{O}_t^i$ for $\mathcal{T}_i$. To ensure that the MIC does not violate any hard task φ_{hard}, the new input must guarantee that the agent never reaches $\mathcal{O}_t^i$ no matter the human input provided. The function $k(x)$ approaches 0 (and the autonomous control takes over) when the state x is in proximity to $\mathcal{O}_t^i$, and $k(x) \rightarrow 1$ (and the human input is also applied) otherwise. To implement these inputs, the navigation stack also includes a joystick functionality that allows a human to drive the robot through direct commands.

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

3 Adapting to agronomic tasks

3.1 Online replanning

Note that the initial plan for each robot i does not account for the motion of other robots and humans in the environment. Thus, online replanning is necessary during the implementation. Each robot i can identify conflicts using the perception algorithm for navigation detailed in Deliverable 9.1. Whenever a possible collision is detected, the LTL planner first conducts replanning, which finds a new accepting run for the PBA $\mathcal{A}_{\mathcal{P}}^{i}$ that is collision-free. Then the MPC controller inside the navigation stack performs a collision avoidance maneuver which brings the robot to the new planned trajectory.

3.2 Designing state transitions

To generate the workspace W shown in Fig. 2, the robots must first explore the vineyard and create a static map using the localization software stack. Afterwards, the terrain is discretized in a grid of a size specified by the user and the allowed transitions between regions are calculated accounting for obstacles that the static map provides. These transitions are $goto\,r$ actions, which allow the robot to go from its current grid state to grid state r.

Additionally, a transition system for the robot load state is created (see Fig. 3). In order to encode the relationship between the different transition systems, a guard G_i can be specified for the actions, i.e. the action can only be performed in a specific state of W. For example, the action *get grapes* can only be used in regions where vines with grapes are located. The total transition system that is then fed into the planner is the product transition system of both the robot load state and the workspace grid state.

3.3 Agronomic task specifications for CANOPIES

These agronomic tasks are specified to manipulation endowed mobile robots through temporal logic syntax, which accordingly plan their action sequences in real time while accounting for dynamic entities such as humans and other robots. Besides acting as moving obstacles, the humans also collaborate with the robots in sub-tasks such as loading/unloading boxes from the robots. Furthermore, humans can specify new tasks to the robots in real time, such as “move to loading station” or “perform box exchange” or “pick grapes” in a sequence that satisfies task specifications given by human user.

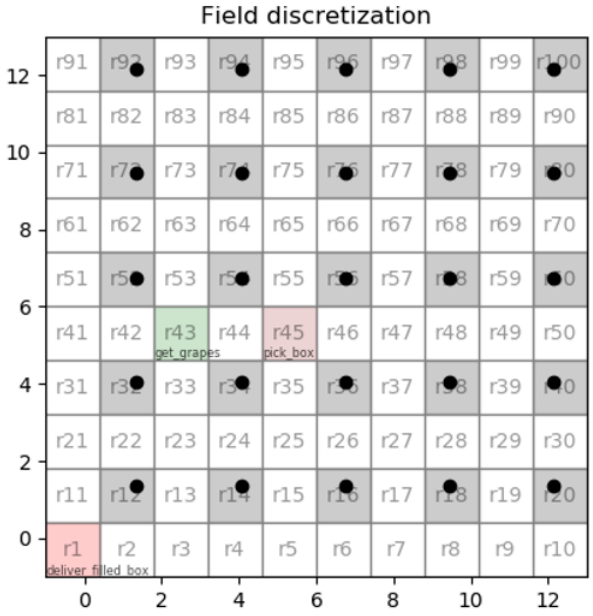

Figure 2: The workspace discretized into a 10×10 grid. Each white cell is considered as a state. The grey cells contain static obstacles marked by black dots (i.e. pergola poles that support the vines as seen in the bottom right panel) and are excluded from the state transition map. The colored cells (r1, r43, r45) represent the only regions in which the specified load action can be performed.

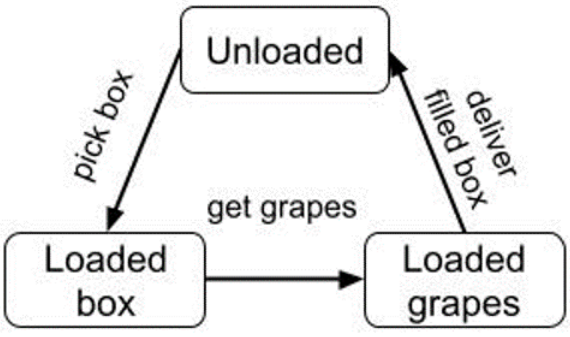

Figure 3: Diagram of the states and actions for the robot load transition system in the vineyard.

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

4 System architecture

In this system, there are two robots that are used to perform the experiment: farming and logistics robots. Each of them follows its own plan to achieve a given task. To conduct the experiment, the system is first tested and verified in the simulation. Then, the system is deployed in the actual field. There are three main parts in the system: planner stack, navigation stack, and robot. The planner stack is used to generate the plan from the LTL planner for the robot. The navigation stack is used to navigate the robot to follow the given pose from the planner. The robot here represents the simulator or the actual hardware used to perform the experiment. ROS is used as a main communication between the system.

4.1 Simulation

The simulation is used to test and verify the software and communication before deploying it in the actual field. To allow for quick testing of the algorithms, a Gazebo scene was implemented which simulates the actual field and the robot. ROS is used to communicate between the robot and the planner. The system is designed to be as similar to the actual robot as possible. As shown in Fig. 4, there are two NUCs for the planner, each for each robot. The navigation stack and simulator are running on the same laptop. These NUCs communicate with the rest of the system through the same Wi-Fi network. With this structure, we can conduct the same experiment in the simulation as in the actual field. This will ensure that the system is working properly before implementing it in the actual field.

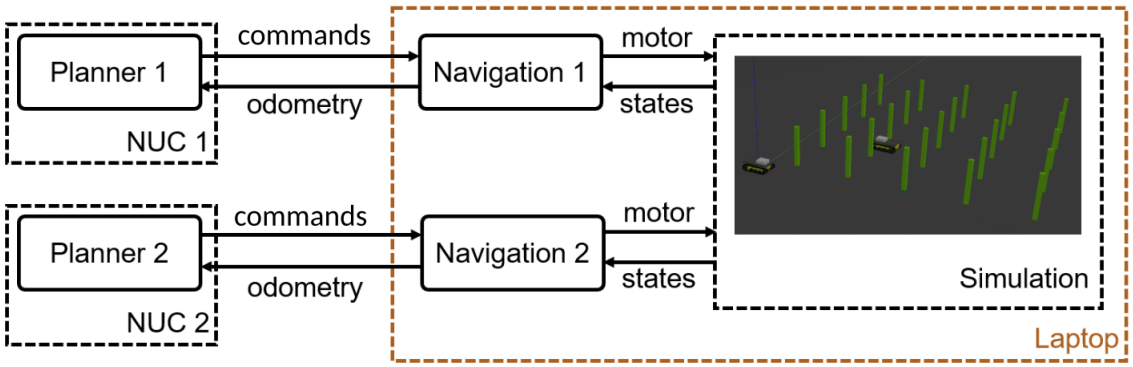

Figure 4: System architecture for the simulation

4.2 Actual robot

In the actual robot experiments, each robot is equipped with a planner NUC and a navigation NUC as shown in Fig. 5. The planner NUC is used to run the planner stack and the navigation NUC is used to run the navigation stack. These NUCs communicate with each other via ethernet on the robot. The navigation NUC is also connected to the Wi-Fi network to receive RTK corrections from the GNSS base station as well as communicate with the other robots. In order to access NUCs on the robot, a user needs to connect with the same Wi-Fi network as that of the robot.

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

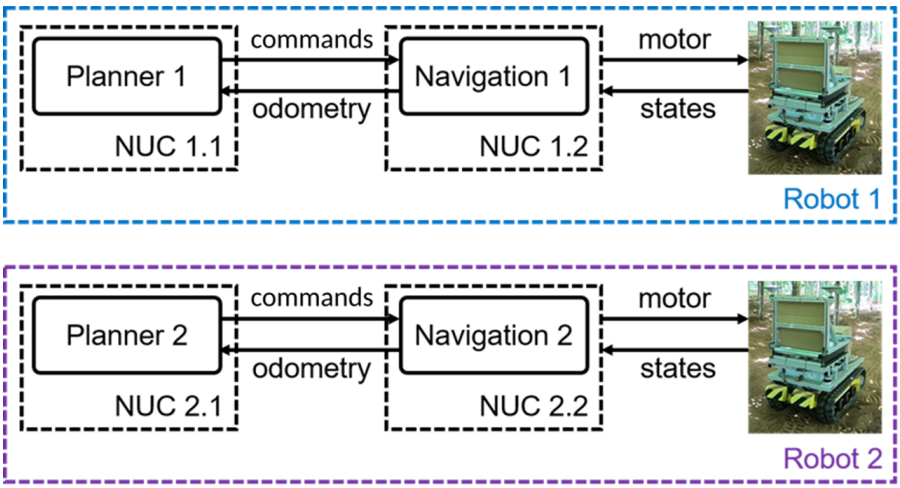

Figure 5: System architecture for the actual robot

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

5 Code Implementation

5.1 ROS LTL planner

The LTL_automaton_core package provides the underlying LTL planner. On startup, the planner is configured with an LTL formula and a transition system. While running, the planner takes as input the current transition state and outputs the next action to take.

5.2 Agronomic task planning

The transition state of the robot includes

- Current residing region (one of many regions)
- Current loading status (one of “unloaded”, “loaded_box”, “loaded_grapes”)

The transition system is generated by discretizing the map into a 10x10 grid with cell size 1.4m x 1.4m (for example, see Fig. 2). Each grid cell is represented by a region. The actions for each region are such that the robot can only move to all adjacent obstacle-free regions. These will be referred to as “move actions”. “Move actions” change the current residing region of the robot.

Besides “move actions”, certain regions have additional possible transitions which belong to one of “pick box”, ”get grapes” or “deliver filled box”. These actions change the current loading status of the robot. They are detailed in the table below:

State Actions	Initial State	Final State
pick boxes	unloaded	loaded_box
get grapes	loaded_box	loaded_grapes
deliver filled box	loaded_grapes	unloaded

The LTL formula for the farming robot is specified as follows:

$$\text{Robot 1}: \quad (\square\lozenge \text{loaded grapes}) \wedge (\square\lozenge \text{unloaded})$$

This translates to the hard task “always eventually load grapes and always eventually unload grapes”.

```
1 [INFO] [1687517352.082805]: LTL Planner: --- Planning in progress (static) ---
2 [INFO] [1687517352.083565]: LTL Planner: Hard task is: ([]<> loaded_grapes) && ([]<> unloaded)
3 [INFO] [1687517352.084273]: LTL Planner: Soft task is:
4 [INFO] [1687517352.203439]: LTL Planner: full Buchi constructed with 3 states and 8 transitions
5 [INFO] [1687517352.313995]: LTL Planner: full model constructed with 309 states and 1152 transitions
6 [INFO] [1687517352.315040]: LTL Planner: initial state in TS is {('r23', 'unloaded')}
7 [INFO] [1687517352.341893]: LTL Planner: full product constructed with 927 states and 4608 transitions
8 [INFO] [1687517352.659428]: LTL Planner: Prefix plan: ['goto_r24', 'goto_r25', 'goto_r35', 'goto_r45', 'pick_box', 'goto_r44', 'goto_r43',
  'get_grapes', 'goto_r33', 'goto_r23', 'goto_r13', 'goto_r3', 'goto_r2', 'goto_r1', 'deliver_filled_box', 'goto_r2']
9 [INFO] [1687517352.660399]: LTL Planner: Suffix plan: ['goto_r3', 'goto_r13', 'goto_r23', 'goto_r24', 'goto_r25', 'goto_r35', 'goto_r45',
  'pick_box', 'goto_r44', 'goto_r43', 'get_grapes', 'goto_r33', 'goto_r23', 'goto_r13', 'goto_r3', 'goto_r2', 'goto_r1',
  'deliver_filled_box', 'goto_r2']
10 [INFO] [1687517352.661091]: LTL Planner: --- Planning successful! ---
```

Figure 6: Plan to “always eventually load grapes and always eventually unload grapes”

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

This transition system combined with the LTL formula allows the LTL planner to generate a region-by-region plan that causes the farming robot to execute the following sequence (assuming the robot starts with the "unloaded" status)

1. Move to a region where it can do the "pick box" action
2. Execute the "pick box" action
3. Wait for "pick box" acknowledgement
4. Move to a region where it can do the "get grapes" action
5. Execute the "get grapes" action
6. Wait for "get grapes" acknowledgement
7. Move to a region where it can do the "deliver filled box" action
8. Execute "deliver filled box" action
9. Wait for "deliver filled box" acknowledgement
10. Repeat

Similarly, the LTL formula for the logistics robot is specified as follows:

$$\text{Robot 2}: \quad (\square\lozenge \text{r11}) \wedge (\square\lozenge \text{r51})$$

Which translates to the hard task "always eventually go to region r11 and region r51" and the planner generates a plan that constantly moves the robot between region r11 and region r51.

```
1 [INFO] [1687451944.791729]: LTL Planner: --- Planning in progress (static) ---
2 [INFO] [1687451944.792356]: LTL Planner: Hard task is: ([]<> r11) && ([]<> r51)
3 [INFO] [1687451944.792946]: LTL Planner: Soft task is: []!r45
4 [INFO] [1687451945.019275]: LTL Planner: full Buchi constructed with 6 states and 16 transitions
5 [INFO] [1687451945.129987]: LTL Planner: full model constructed with 309 states and 1152 transitions
6 [INFO] [1687451945.130894]: LTL Planner: initial state in TS is {('r31', 'unloaded')}
7 [INFO] [1687451945.180938]: LTL Planner: full product constructed with 1854 states and 6966 transitions
8 [INFO] [1687451945.448818]: LTL Planner: Prefix plan: ['goto_r21', 'goto_r11', 'goto_r21', 'goto_r31', 'goto_r41', 'goto_r51', 'goto_r41']
9 [INFO] [1687451945.449724]: LTL Planner: Suffix plan: ['goto_r31', 'goto_r21', 'goto_r11', 'goto_r21', 'goto_r31', 'goto_r41', 'goto_r51',
  'goto_r41']
10 [INFO] [1687451945.450337]: LTL Planner: --- Planning successful! ---
```

Figure 7: Plan for the robot to constantly move between r11 and r51.

Both plans implicitly avoid obstacle occupied regions as the planner never generates move actions that target obstacle occupied regions.

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

5.3 Integrating planner and navigation stack

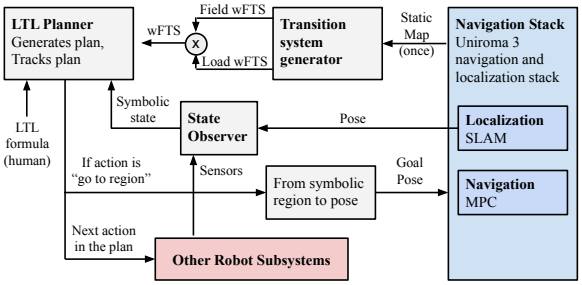

Figure 8: Flowchart showing high-level module dependencies

As shown in Fig. 8, the position output (of type /nav_msgs/Odometry) by the navigation stack is converted to a region (of type /std_msgs/String) compatible with the LTL planner. This conversion is done by checking the position against the predefined region positions and finding the region in which the robot position is within the region's radius. The action-to-take output (of type /nav_msgs/PoseStamped) by the LTL planner is converted by a node called simple_waypoint_chaser to a desired position (of type /nav_msgs/Odometry) before sending to the navigation stack. Since the action-to-take output may not explicitly define the desired orientation, the desired orientation is set such that the robot faces the direction it is travelling, assuming the robot travels from the current region to the next region in a straight line.

```
# This represents an estimate of a position and velocity in free space.
# The pose in this message should be specified in the coordinate frame given by header.frame_id.
# The twist in this message should be specified in the coordinate frame given by the child_frame_id
Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

Figure 9: /nav_msgs/Odometry

```
# A Pose with reference coordinate frame and timestamp
Header header
Pose pose
```

Figure 10: /nav_msgs/PoseStamped

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

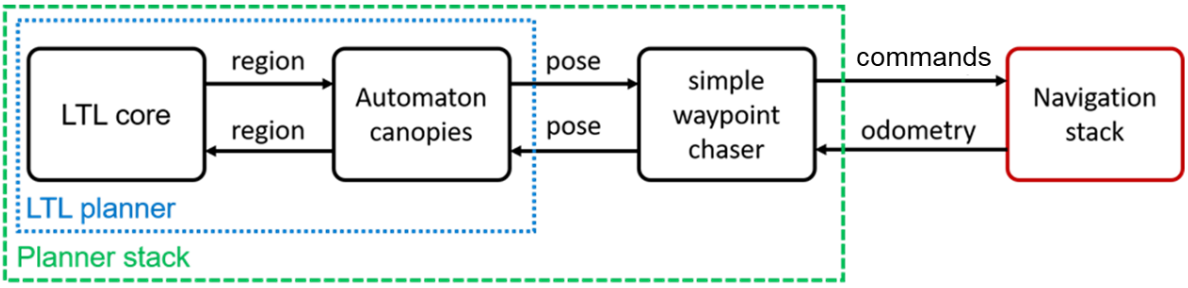

Figure 11: Planner stack

Our LTL package also has the possibility of being similarly integrated with the other modules such as BEM controller or Torso Controllers. In the same way that the planner request way-point tracking action from the Navigation stack while running its plan, it can also request high-level actions like “pick grape” or “exchange box” from other modules and await completion of these actions before moving forward with the rest of the planned task.

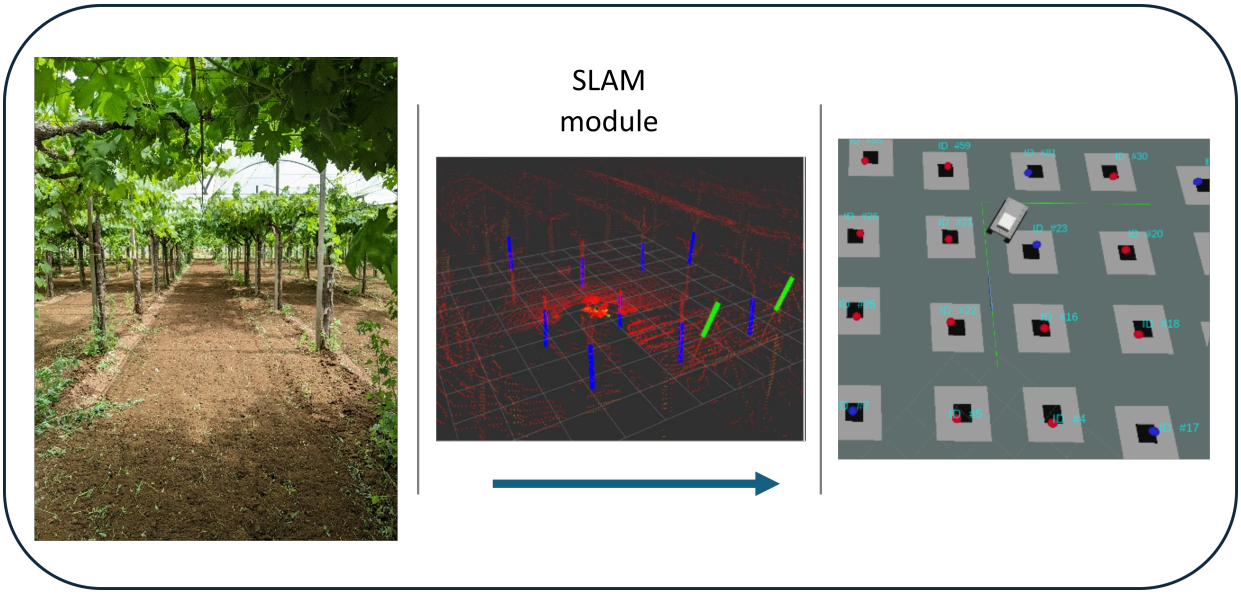

Figure 12: Offline input preprocessing for planner

The process of generating the transition map is as follows:

1. The robot is driven around the field and the SLAM algorithm (detailed in D9.1) generates the map of the environment with coordinates for the obstacles (referred to as pole).
2. The grid size (in terms of number of cells) and actual dimension of each cell (in terms of meters) is specified, and the grid is generated accordingly.

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

3. According to the detected pole coordinates, the containing grid is marked as occupied.
4. Special cells for the “pick box”, “get grapes” and “deliver filled box” actions are specified.
5. The entire grid is exported as a YAML dictionary to be loaded by the LTL planner.

5.4 VR environment and simulations

The system was tested in both Gazebo and Unity with the actual navigation and planning stack and mocked hardware. In this simulation setup, we use the same communication structure with the hardware experiment in the field. There are two planning NUCs to control two robots, one for each. These NUCs communicate with simulator via ROS on the same Wi-Fi network. The LTL planner is agnostic towards whether the system is running in simulation or in a physical environment as the LTL planner only communicates with the navigation stack which abstracts away the details of running on hardware or simulation.

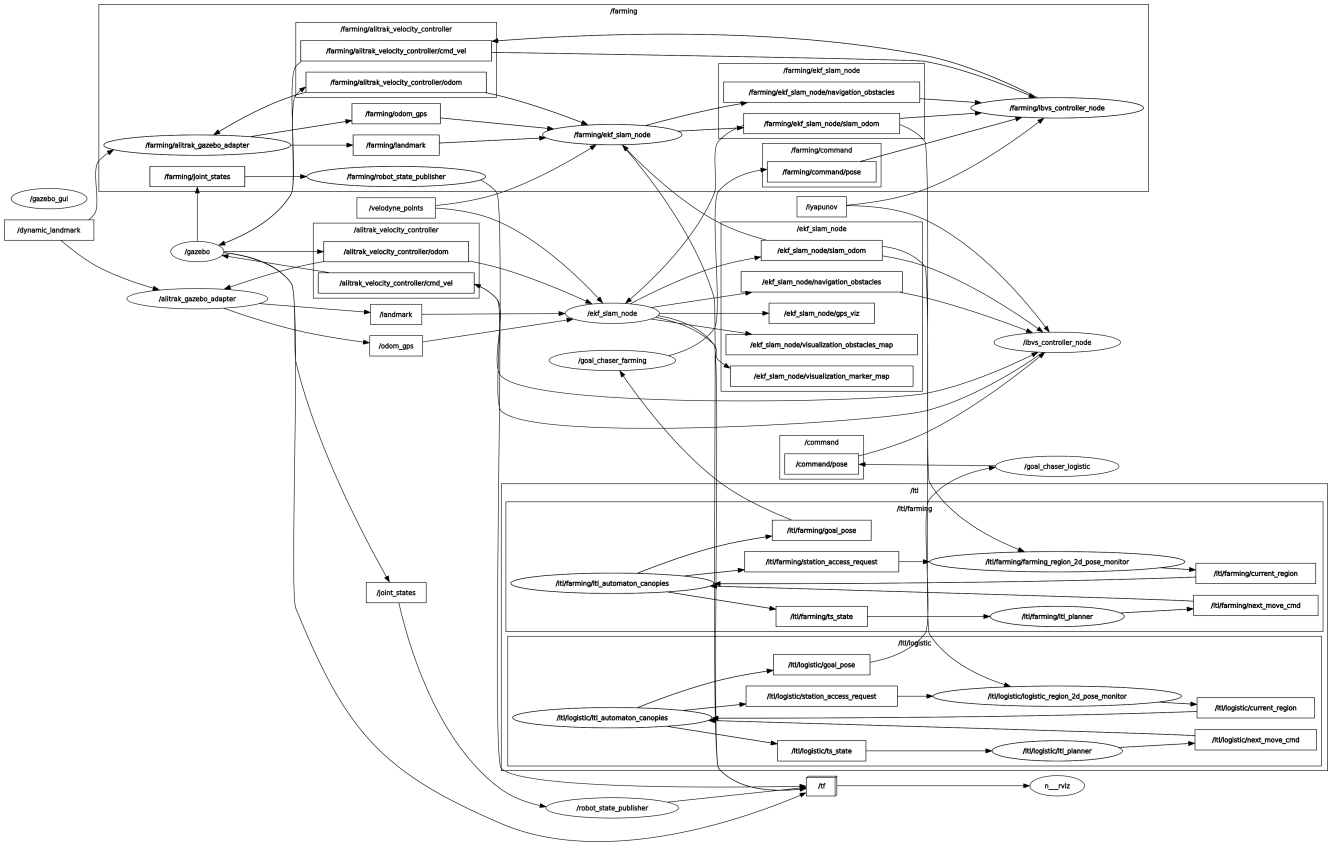

Figure 13: RQT graph of topics in simulation

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

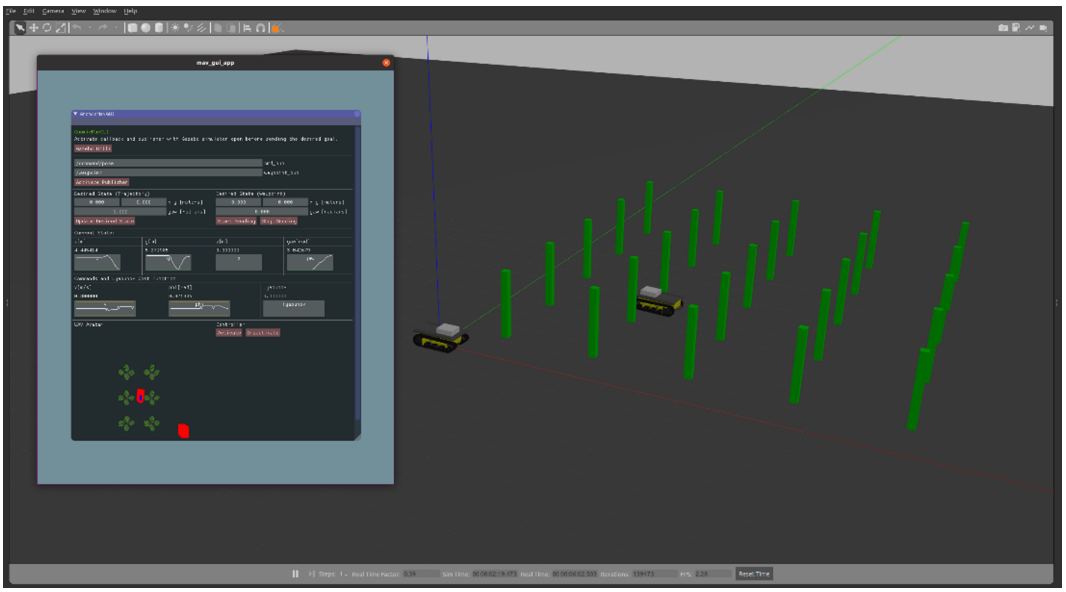

Figure 14: Gazebo environment

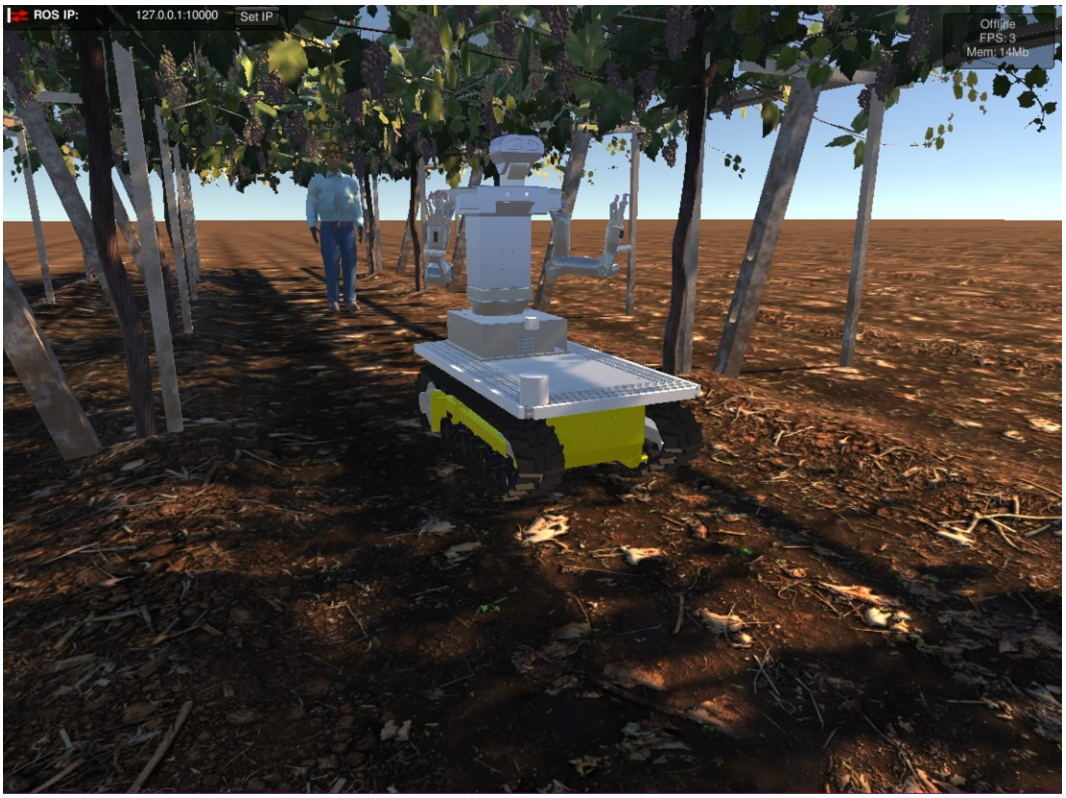

Figure 15: PALEBLUE VR simulation environment

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

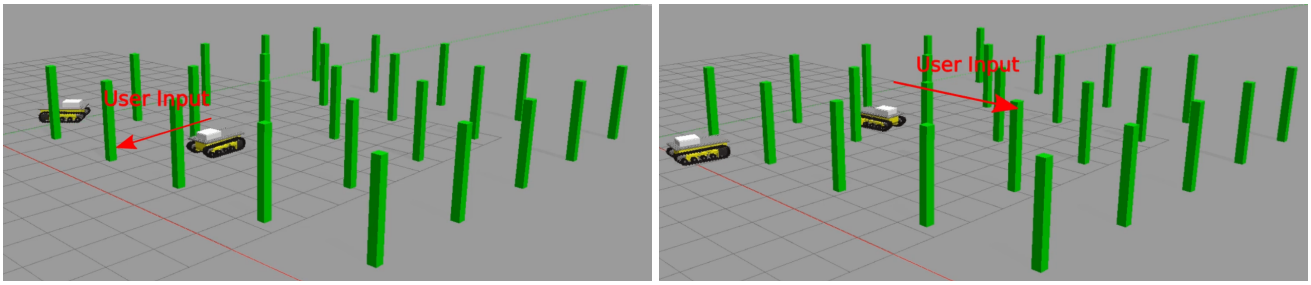

Figure 16: Two instances of human input provided through the MIC module developed in the Gazebo simulator. The human commands the low level controller to go in the direction of the red arrow while avoiding any $trap\ state$. (a) First instance. (b) Second instance.

5.4.1 HIL simulations

The Gazebo simulator was also used to test the HIL MIC module shown in (2) to ensure the system does not enter any of the $trap\ states$. Fig. 16 and the video [22] show two instances where a human low level command is inputted through the command velocity and the MIC module provides the desired motion while ensuring that no $trapstate$ is reached. The MIC module works in tandem with the navigation stack to prevent collisions which is also considered a $trap\ state$. When the human input stops, the LTL planner replans and the robots resume their original task.

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

6 Hardware experiments in the field

In this section, we describe hardware experiments in detail which took place between 20th – 23rd of June, 2023.



Figure 17: The two Alitrak robots used for the field experiments.

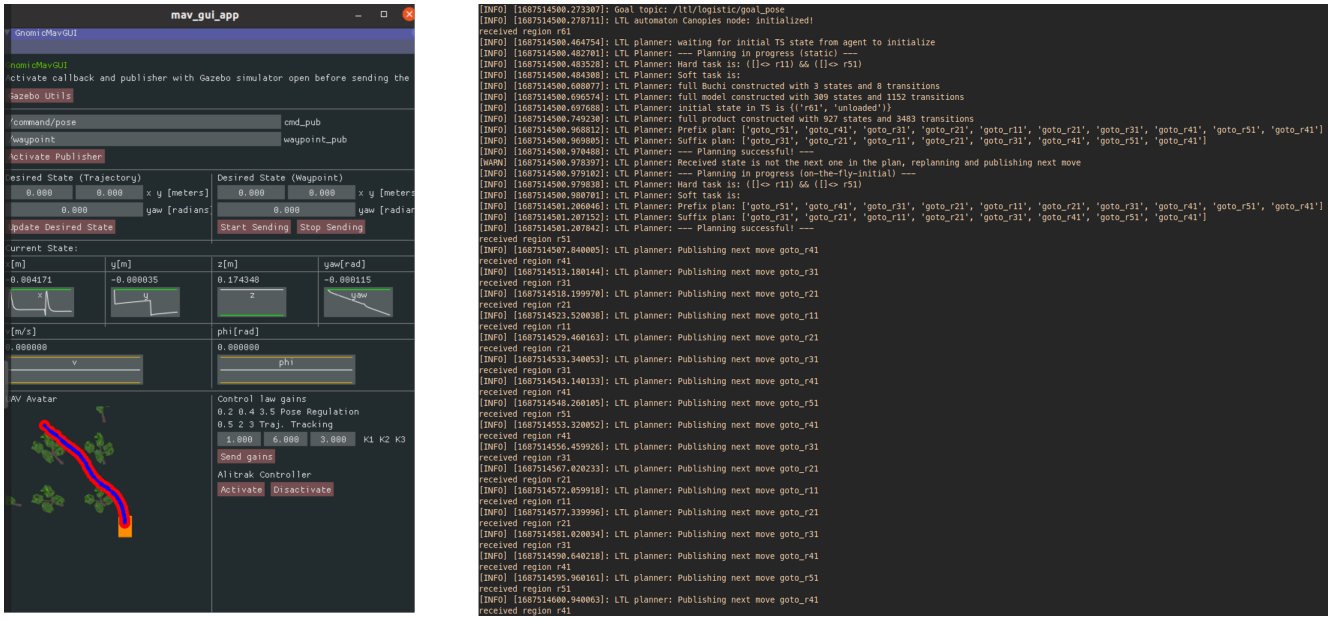

Figure 18: Monitoring and control setup for the laptops.

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



Figure 19: Setup to monitor the robots through laptop.

6.1 Setup (as of June 19th 2023)

Two mobile bases, shown in Fig. 17, were used to demonstrate multi-robot task planning with one human in the shared workspace. They were monitored through several laptops deployed on the field, Fig. 19 through the localization and navigation stack software, Fig. 18. The human can interact with the robots in three different ways in our experimental setup: firstly, as a dynamic obstacle as shown in Fig. 20; secondly, by directly controlling one agent through a remote controller device as shown Fig. 22; and thirdly, the human can collaborate with the robots towards agronomic subtasks such as loading box to the robots as shown in Fig. 21.

The field is setup such that there are poles at uniform intervals to support the vines, thus naturally dividing the vineyard into square region. We further refine these square grids and consider the occupancy of these refined grids into the LTL state space.

The robots are capable of communicating with each other at all times through a centralized network as well as they can communicate with the human users via a laptop, allowing the human to monitor and provide task specifications to the robots in real time. These communication setup is shown in Fig. 24. Besides the possibility to stop the execution of planning and navigation for the robots remotely using the laptops, for additional safety, the alitrak controller also has a "kill switch".

6.2 NUCs and communication network

The computing system of each robot involves two NUCs. One NUC runs the navigation stack which will be referred to as the navigation NUC. The other NUC runs the LTL planner which will be referred to as the

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



Figure 20: Physical interaction such as obstructing robots. The humans in the shared workspace can act as dynamic obstacles and activate safety controllers in the robots, forcing them to perform collision avoidance maneuvers.

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



Figure 21: Cooperative human-robot interaction.



Figure 22: Explicit interaction such as teleoperation.

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

planning NUC.

The planning NUC is an Intel NUC 12 Pro Mini PC NUC12WSKi70XC with a 12th Generation Intel® Core™ i7-1260P Processor (18M Cache, up to 4.70 GHz). It has 16GB RAM (2x 8GB DDR4-3200 SODIMMs) and 500GB Gen4 NVMe SSD (M.2 2280) storage. It runs the Ubuntu 20.04.5 LTS Focal Fossa OS with ROS Noetic Ninjemys.



Figure 23: Intel NUC 12 Pro Mini

Each NUC runs its own ROS core. To have the two ROS cores communicate with each other, a ROS bridge was implemented to forward topics and service calls between the two ROS cores. The ROS bridge physically runs on the planner NUC and works by creating a ROS node on both the planning ROS core and the navigation ROS core. A list of topics and services are specified for the bridge to forward. When the bridge ROS node of one ROS core receives a message/service call from the relevant topic/service, it sends the data via WebSocket to the bridge ROS node of the other ROS core, hence allowing the other ROS core to receive the same message/service call.

6.3 Demonstration of replanning after disturbance

When a disturbance is introduced, e.g., when a safety maneuver has to be performed, the motion of the robot stops following the LTL plan and a replanning needs to be executed. A log showing this replanning is shown in Fig. 25. To go further into detail:

1. After successful planning, the robot executes the plan.
2. At line 24, the robot attempts to move to region r31 but a human blocks its path causing the navigation stack to move the robot to avoid the human.
3. This causes the robot to end up in an unexpected region r42.
4. The planner detects that the robot is in an unexpected region and replans.

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

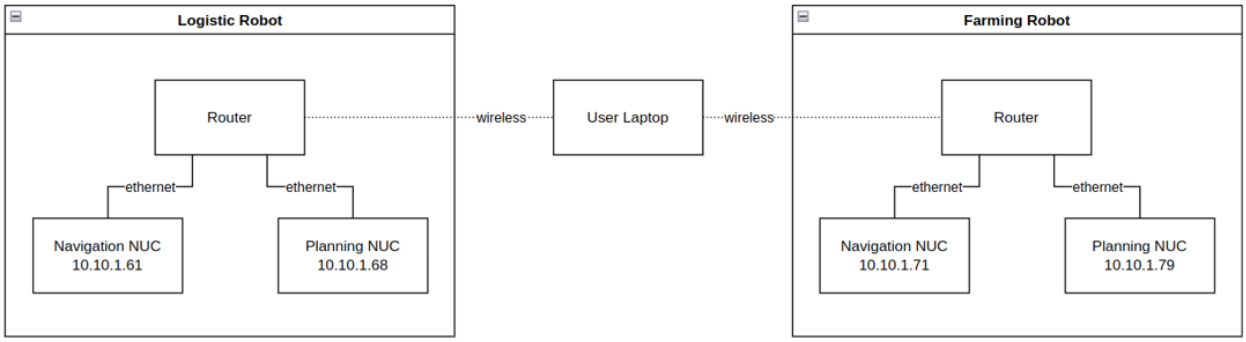

Figure 24: Communication setup among the robots.

```
1 [INFO] [1687452025.326934]: LTL Planner: Hard task is: ([]<> r11) && ([]<> r51)
2 [INFO] [1687452025.327759]: LTL Planner: Soft task is: []!r45
3 [INFO] [1687452025.617417]: LTL Planner: Prefix plan: ['goto_r41', 'goto_r31', 'goto_r21', 'goto_r11', 'goto_r21', 'goto_r31', 'goto_r41', 'goto_r51', 'goto_r41']
4 [INFO] [1687452025.618447]: LTL Planner: Suffix plan: ['goto_r31', 'goto_r21', 'goto_r11', 'goto_r21', 'goto_r31', 'goto_r41', 'goto_r51', 'goto_r41']
5 [INFO] [1687452025.619170]: LTL Planner: --- Planning successful! ---
6 received region r41
7 [INFO] [1687452028.144956]: LTL planner: Publishing next move goto_r31
8 received region r31
9 [INFO] [1687452035.684845]: LTL planner: Publishing next move goto_r21
10 received region r21
11 [INFO] [1687452040.844872]: LTL planner: Publishing next move goto_r11
12 received region r11
13 [INFO] [1687452045.965028]: LTL planner: Publishing next move goto_r21
14 received region r21
15 [INFO] [1687452049.204996]: LTL planner: Publishing next move goto_r31
16 received region r31
17 [INFO] [1687452060.045022]: LTL planner: Publishing next move goto_r41
18 received region r41
19 [INFO] [1687452065.124802]: LTL planner: Publishing next move goto_r51
20 received region r51
21 [INFO] [1687452070.244964]: LTL planner: Publishing next move goto_r41
22 received region r41
23 [INFO] [1687452073.604945]: LTL planner: Publishing next move goto_r31   Disturbance introduced here
24 received region r42
25 [WARN] [1687452078.224986]: LTL planner: Received state is not the next one in the plan, replanning and publishing next move
26 [INFO] [1687452078.226107]: LTL Planner: --- Planning in progress (on-the-fly-inittial) ---
27 [INFO] [1687452078.226953]: LTL Planner: Hard task is: ([]<> r11) && ([]<> r51)
28 [INFO] [1687452078.227785]: LTL Planner: Soft task is: []!r45
29 [INFO] [1687452078.510151]: LTL Planner: Prefix plan: ['goto_r41', 'goto_r31', 'goto_r21', 'goto_r11', 'goto_r21', 'goto_r31', 'goto_r41', 'goto_r51', 'goto_r41']
30 [INFO] [1687452078.511136]: LTL Planner: Suffix plan: ['goto_r31', 'goto_r21', 'goto_r11', 'goto_r21', 'goto_r31', 'goto_r41', 'goto_r51', 'goto_r41']
31 [INFO] [1687452078.511947]: LTL Planner: --- Planning successful! ---
32 received region r41
33 [INFO] [1687452081.605004]: LTL planner: Publishing next move goto_r31
34 received region r31
35 [INFO] [1687452091.044857]: LTL planner: Publishing next move goto_r21
36 received region r21
37 [INFO] [1687452096.065145]: LTL planner: Publishing next move goto_r11
38 received region r11
39 [INFO] [1687452100.944938]: LTL planner: Publishing next move goto_r21
40 received region r21
41 [INFO] [1687452104.345012]: LTL planner: Publishing next move goto_r31
42 received region r31
43 [INFO] [1687452114.765095]: LTL planner: Publishing next move goto_r41
44 received region r41
45 [INFO] [1687452119.784937]: LTL planner: Publishing next move goto_r51
46 received region r51
47 [INFO] [1687452125.045018]: LTL planner: Publishing next move goto_r41
48 received region r41
```

Figure 25: Log of the LTL planner after a disturbance is introduced.

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

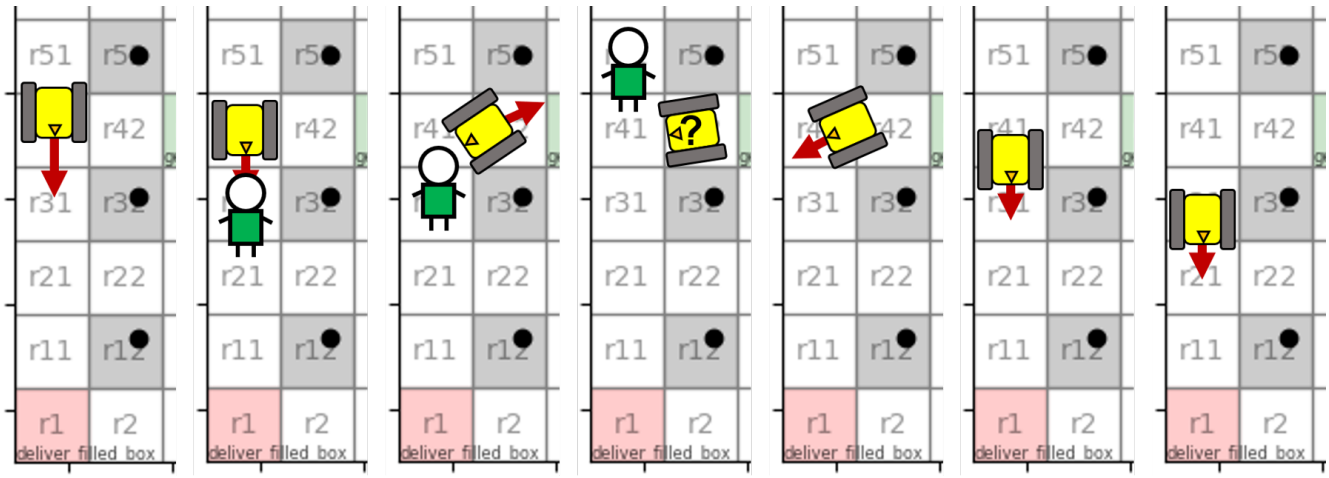

Figure 26: Diagram showcasing a specific instance of the replanning due to a human obstacle.

5. The robot then executes the new plan.
6. After the human obstacle is removed, the robot is able to continue towards its destination.

These specific replanning case is also shown pictorially in Fig. 26.

6.4 Acknowledgement

Once the robot reaches a station and performs an action, the robot waits for an acknowledgement which is provided by the user due to lack of manipulation capabilities at the time. This heuristic allows the LTL planner to be more adaptable since it is separated from the manipulation modules, which only need to send an acknowledgement when they are finished. Once this acknowledgement is received, the robot continues on its next task. And example of this behaviour is shown in Fig. 27.

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

```
[INFO] [1687515588.064751]: LTL planner: Publishing next move goto_r25
received region r25
[INFO] [1687515593.024938]: LTL planner: Publishing next move goto_r35
received region r35
[INFO] [1687515600.524873]: LTL planner: Publishing next move goto_r45
received region r45
[INFO] [1687515605.624827]: LTL planner: Publishing next move pick_box
PICK ACTION OK, CHANGE TO LOADED BOX
[INFO] [1687515614.484860]: LTL planner: Publishing next move goto_r44
received region r44
[INFO] [1687515623.604922]: LTL planner: Publishing next move goto_r43
received region r43
[INFO] [1687515628.924834]: LTL planner: Publishing next move get_grapes
PICK GRAPES ACTION OK, CHANGE TO LOADED GRAPES
[INFO] [1687515640.044890]: LTL planner: Publishing next move goto_r33
received region r33
[INFO] [1687515649.384855]: LTL planner: Publishing next move goto_r23
received region r23
[INFO] [1687515654.324848]: LTL planner: Publishing next move goto_r13
received region r13
[INFO] [1687515659.344865]: LTL planner: Publishing next move goto_r3
received region r3
[INFO] [1687515664.404970]: LTL planner: Publishing next move goto_r2
received region r13
```

Figure 27: Acknowledgements for picking box and picking grapes shown in the log of the LTL planner.

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

7 Implementational challenges

In this section, we briefly discuss the main challenges faced while performing experiments for field validation of task planning algorithms and, subsequently, present solutions that were employed to successfully resolve these challenges. The challenges are described here to provide an understanding of the overall complexity of the scenario considered for field validation.

7.1 Difference in the orientation of reference frames

The orientation of the navigation map generated from geo-referenced data estimated by the EKF-SLAM algorithm differs from the orientation of the (discretized) map employed by the LTL planner. As a result, the odometry data from the robot will be converted into incorrect position information for the LTL planner. This is a concern for the LTL planner since it will process incorrect information for the region and sebexecute the wrong plan. During the experiment, this problem was effectively resolved by introducing a transformation matrix (as shown in Fig. 28) between the planner stack and the navigation stack to align the orientation of the map generated in LTL planner with that of the geo-referenced navigation map.

$$T(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

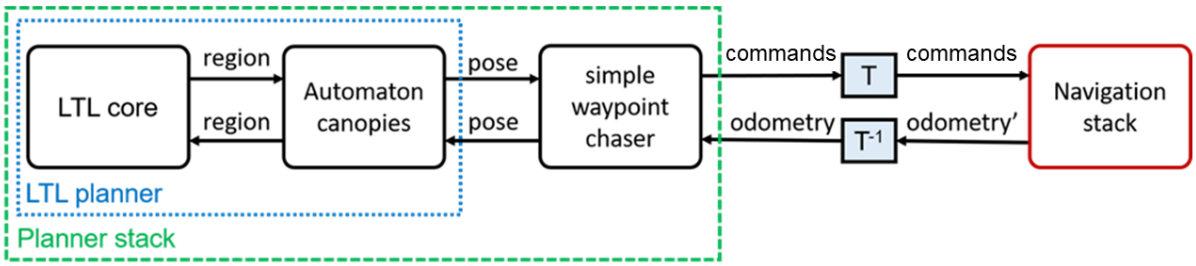

Figure 28: The new software schematic includes transformation matrix as shown in blue rectangular for the data between planner and navigation stack.

7.2 Non-uniform poles in the field

While generating the discretized workspace (see Fig. 2) for the LTL planner, for simplicity, we assumed that the pergola poles that supported vines were evenly spaced out. However, in reality, this assumption may not always hold. In other words, we found that not all the poles were uniformly distributed across the actual field. As a consequence, there are some regions in the field where the poles were closer to the edge of the grid in the planner. This means that the robot will not be able to pass through the region since the low-level object avoidance algorithm will try to avoid the pole. This concern was resolved by re-tuning the number of grids and the size of the grids in the planner; this ensured that the planner did not have a region where a pole was too close to the boundary.

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

7.3 Communication challenges

For the experiments, the robots are equipped with low-level obstacle avoidance algorithms; these are used to avoid obstacles (e.g., poles or humans) detected by the sensors (e.g. LIDAR) on the robot. In order to avoid inter-robot collisions, we use communication between the robots to be aware of each other's locations in the field. Occasionally, the available network resources were saturated owing to the frequent communication between the mobile bases and the laptops. This was temporarily addressed by i) switching-off odometry information exchange between the robots to free up network bandwidth, and ii) by employing the obstacle avoidance algorithm to detect and avoid other robots. More thoroughly, this concern was resolved by employing the inter-robot and intra-robot communication protocols developed by the consortium that do not exclusively rely on ROS-bridge architecture; these protocols were designed to prevent network overload.

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

8 Benchmarks for validation

At the beginning of the project, within D2.3, a set of benchmarks were developed for the validation of the experiments. The benchmarks related to D9.6 are shown in Table 1.

3.4 Multi-robot task planning with human operators	**1. Benchmarks for multi-robot task planning with human presence.** Test data will be collected to test the robustness and accuracy of the algorithms, which should have: - A spatial robustness of 5% of the distance between plants according to D2.1, Table 1, and a temporal robustness of 5 s where these are defined as the spatial and temporal distances between the behavior and the boundaries of the set of all behaviors satisfying the design property. **2. Benchmarks for multi-robot task planning with human in-the-loop.** Test data will be collected to test the robustness and accuracy of the algorithms, which should have: - A spatial robustness of 5% of the distance between plants according to D2.1, Table 1, and a temporal robustness of 5 s (see brief definition above). - An error threshold of 10% with respect to the robustness metrics for both global and local least violating solutions. - More than 60% success rate in reconfiguring the multi-robot plans according to the additional high-level human specifications under unforeseen environmental changes within the least violating solution margins (e.g., avoiding regions with misplaced tools or poor lighting conditions).

Table 1: Benchmarks for validation related to D9.6.

These benchmarks were developed with the thought in mind to use a specific subset of Temporal Logic called Signal Temporal Logic (STL), which has been used and elaborately explained in D8.3. STL has specific quantifiable spatial and temporal robustness metrics which are utilized to synthesize lower-level control for the robots involved. As a consequence, these metrics are easy to validate for our benchmarks and the works in D8.3, validated through experiments at our local lab facility in KTH, have proven to fulfill them. For this, we refer the readers to Sections 5 and 6 of D8.3, where it can be seen that our robustness metrics, through Control Barrier Functions (CBFs), are always fulfilled throughout the entire experiment.

However, for field validation we used another subset of Temporal Logic approaches, namely, Linear Temporal Logic (LTL). LTL only provides high-level paths for the robot to follow and satisfy the task specifications. LTL does not include quantifiable robustness metrics that STL approaches enjoy. However, the decision to use LTL as opposed to STL is primarily driven by the fact that the lower-level control for the robots is developed through external modules (namely, the navigation and localization software stack developed in T3.5) by UNIROMA3. These modules have been validated through their own benchmarks. For this reason, much of the 3.4 benchmarks listed in Table 1 and, particularly, the temporal ones do not apply anymore to our field experiments.

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

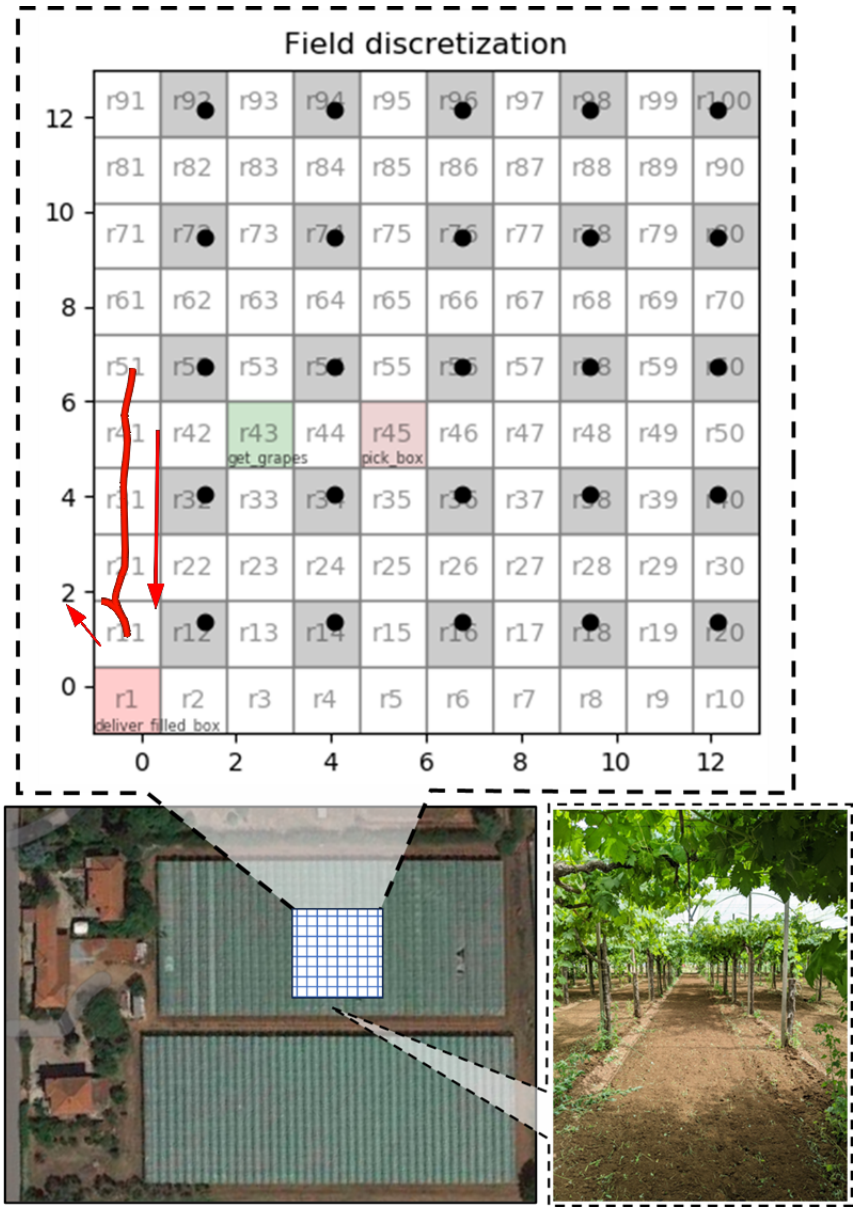

Figure 29: One of the runs for the field experiment validation.

Regarding the spatial robustness, the LTL planner works through the discretization of the workspace where some cells were reserved for the poles of the vineyard which act as the obstacles/plants referred to in the 3.4 benchmarks. Our planner ensures that those pole cells are always avoided and tries to ensure that the robot goes through the center of the empty cells as shown in Fig. 29. This means that the planner, by design, ensures that there is always a safety distance equal to at least half the side length of the square cell and ideally equal to the side length of the square cell. Since the lower-level navigation and localization stack takes care of following these waypoints, we refer to the benchmarks "1.1 Robots navigation" for more details.

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

Lastly, regarding the replanning success rate, during the field experiments all the replanning events were completed successfully with replanning times less than 300 ms in general. We provide one example of a replanning attempt in Fig. 30. In general, the logistic robots experienced an average of 12 replanning events per experiment execution (although some of them referred to the same event), and all the events were triggered by the lower-level collision avoidance between the robots or between the robots and the human.

```
[rosout][WARNING] 2023-06-23 12:03:29,740: LTL planner : Received state is not the next one in the plan , replanning and publishing next move
[rosout][INFO] 2023-06-23 12:03:29,741: LTL Planner : --- Planning in progress (on-the-fly-initial) - --
[rosout][INFO] 2023-06-23 12:03:29,742: LTL Planne r: Hard task is: ([]<> r11) && ([]<> r51)
[rosout][INFO] 2023-06-23 12:03:29,743: LTL Planne r: Soft task is:
[rosout][INFO] 2023-06-23 12:03:29,969: LTL Planne r: Prefix plan: ['goto_r41', 'goto_r31', 'goto_r21 ', 'goto_r11', 'goto_r21', 'goto_r31', ' goto_r41',  'goto_r51', 'goto_r41']
[rosout][INFO] 2023-06-23 12:03:29,970: LTL Planne r: Suffix plan: ['goto_r31', 'goto_r21', 'goto_r11 ', 'goto_r21', 'goto_r31', 'goto_r41', ' goto_r51',  'goto_r41']
[rosout][INFO] 2023-06-23 12:03:29,971: LTL Planne r: --- Planning successful! ---
```

Figure 30: An example of a replanning attempt.

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

9 Conclusions

This report highlights the field validation and experimental demonstrations conducted as part of the CANOPIES project to assess the effectiveness of the developed multi-robot task planning and control framework in agricultural settings. The validation focused on evaluating how well the proposed system supports agronomic collaboration between humans and robots within shared physical spaces.

In this report, we detail our field validation efforts from the following aspects:

- deploying the CANOPIES robots in real agricultural environments to test their ability to perform specified agronomic tasks under dynamic conditions, including the presence of humans and other robots in the workspace;
- demonstrating the practical application of our temporal logic-based approach in encoding and executing high-level tasks as low-level control actions in real-time, ensuring that the robots can adapt to changes and maintain operational efficiency;
- conducting comprehensive experiments to validate the system's robustness and reliability in the field, using a variety of scenarios and conditions to ensure that the robots can consistently meet the agronomic task requirements while coexisting with human operators.

In addition to these validation activities, we provide detailed accounts of the experimental setups, methodologies, and outcomes, showcasing how the CANOPIES robots navigate and perform in real-world agricultural tasks. These field experiments are crucial in proving the practical viability of our framework and in demonstrating its potential for widespread adoption in agricultural operations.

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

References

[1] Ye Liu, Xiaoyuan Ma, Lei Shu, Gerhard Petrus Hancke, and Adnan M. Abu-Mahfouz. From industry 4.0 to agriculture 4.0: Current status, enabling technologies, and research challenges. *IEEE Transactions on Industrial Informatics*, 17(6):4322–4334, 2021.

[2] Paolo Tripicchio, Massimo Satler, Giacomo Dabisias, Emanuele Ruffaldi, and Carlo Alberto Avizzano. Towards smart farming and sustainable agriculture with drones. In *2015 International Conference on Intelligent Environments*, pages 140–143, 2015.

[3] Mehran Shahrooz, Amin Talaeizadeh, and Aria Alasty. Agricultural spraying drones: Advantages and disadvantages. In *2020 Virtual Symposium in Plant Omics Sciences (OMICAS)*, pages 1–5, 2020.

[4] Man Liang and Daniel Delahaye. Drone fleet deployment strategy for large scale agriculture and forestry surveying. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 4495–4500, 2019.

[5] Eduardo Navas, Roemi Fernández, Delia Sepúlveda, Manuel Armada, and Pablo Gonzalez-de Santos. Soft gripper for robotic harvesting in precision agriculture applications. In *2021 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 167–172, 2021.

[6] Claus Smitt, Michael Halstead, Patrick Zimmer, Thomas Läbe, Esra Guclu, Cyrill Stachniss, and Chris McCool. Pag-nerf: Towards fast and efficient end-to-end panoptic 3d representations for agricultural robotics. *IEEE Robotics and Automation Letters*, 9(1):907–914, 2024.

[7] Sami Salama Hussen Hajjaj and Khairul Salleh Mohamed Sahari. Review of agriculture robotics: Practicality and feasibility. In *2016 IEEE International Symposium on Robotics and Intelligent Sensors (IRIS)*, pages 194–198, 2016.

[8] Sofia Matilde Luglio, Mino Sportelli, Christian Frasconi, Marco Fontanelli, Matteo Matteucci, Giulio Fontana, Enrico Piazza, and Davide Facchinetti. Field campaign and experimental design for robot performance evaluation (acre 2023). In *2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, pages 398–403, 2023.

[9] Paul Baxter, Grzegorz Cielniak, Marc Hanheide, and Pål From. Safe human-robot interaction in agriculture. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '18, page 59–60, New York, NY, USA, 2018.

[10] Yi Chen, Baohua Zhang, Jun Zhou, and Kai Wang. Real-time 3d unstructured environment reconstruction utilizing vr and kinect-based immersive teleoperation for agricultural field robots. *Computers and Electronics in Agriculture*, 175:105579, 2020.

[11] Athanasios Anagnostis, Lefteris Benos, Dimitrios Tsaopoulos, Aristotelis Tagarakis, Naoum Tsolakis, and Dionysis Bochtis. Human activity recognition through recurrent neural networks for human–robot interaction in agriculture. *Applied Sciences*, 11(5), 2021.

[12] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT press, 2008.

[13] Robin Baran, Meng Guo, Xiao Tan, Peter Varnai, and Wenceslao Shaw Cortez. LTL_automaton_core. `https://github.com/KTH-DHSG/ltl_automaton_core`.

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

[14] Stanford Artificial Intelligence Laboratory et al. Robotic operating system. https://www.ros.org.

[15] Robin Baran, Xiao Tan, Peter Varnai, Pian Yu, Sofie Ahlberg, Meng Guo, Wenceslao Shaw Cortez, and Dimos V Dimarogonas. A ros package for human-in-the-loop planning and control under linear temporal logic tasks. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 2182–2187. IEEE, 2021.

[16] J Richard Büchi. On a decision method in restricted second order arithmetic. In *The Collected Works of J. Richard Büchi*, pages 425–435. Springer, 1990.

[17] Denis Oddoux and Paul Gastin. LTL 2 ba : fast translation from LTL formulae to Büchi automata. http://www.lsv.fr/~gastin/ltl2ba/.

[18] Meng Guo and Dimos V. Dimarogonas. Multi-agent plan reconfiguration under local LTL specifications. *The International Journal of Robotics Research*, 34(2):218–235, 2015.

[19] Stephen L Smith, Jana Tůmová, Calin Belta, and Daniela Rus. Optimal path planning for surveillance with temporal-logic constraints. *The International Journal of Robotics Research*, 30(14):1695–1708, 2011.

[20] Calin Belta, Boyan Yordanov, and Ebru Aydin Gol. *Formal methods for discrete-time dynamical systems*, volume 15. Springer, 2017.

[21] Meng Guo, Sofie Andersson, and Dimos V Dimarogonas. Human-in-the-loop mixed-initiative control under temporal tasks. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, page 6395–6400. IEEE Press, 2018.

[22] Shankar A. Deka, Sujet Phodapol, Andreu Matoses Gimenez, Victor Nan Fernandez-Ayala, Rufus Wong, Pian Yu, Xiao Tan, and Dimos V Dimarogonas. LTL planning and control for multi-agent systems in real-world environments. https://youtu.be/4PO25VWl6y0, 2024.