Article

Towards Agrirobot Digital Twins: Agri-RO5—A Multi-Agent Architecture for Dynamic Fleet Simulation

Jorge Gutiérrez Cejudo, Francisco Enguix Andrés, Marin Lujak, Carlos Carrascosa Casamayor, Alberto Fernandez and Luís Hernández López

Article

Towards Agrirobot Digital Twins: Agri-RO5—A Multi-Agent Architecture for Dynamic Fleet Simulation [†]

Jorge Gutiérrez Cejudo [1], Francisco Enguix Andrés [2], Marin Lujak [1,*], Carlos Carrascosa Casamayor [2], Alberto Fernandez [1] and Luís Hernández López [2]

[1] Research Centre for Intelligent Information Technologies (CETINIA), University Rey Juan Carlos, 28933 Madrid, Spain; jorge.gutierrez@urjc.es (J.G.C.); alberto.fernandez@urjc.es (A.F.)
[2] Valencian Research Institute for Artificial Intelligence (VRAIN), Universitat Politècnica de València (UPV), 46022 Valencia, Spain; fraenan@upv.es (F.E.A.); carrasco@dsic.upv.es (C.C.C.); lhernand@upv.es (L.H.L.)
[*] Correspondence: marin.lujak@urjc.es
[†] This paper is an extended version of our paper and presented in Workshop on Adaptive Smart areaS and Intelligent Agents (ASSIA) at PAAMS'23, 22nd International Conference on Practical Applications of Agents and Multi-Agent Systems, Salamanca, Spain, 26–28 June 2024.

Abstract: In this paper, we propose a multi-agent-based architecture for a Unity3D simulation of dynamic agrirobot-fleet-coordination methods. The architecture is based on a Robot Operating System (ROS) and Agrobots-SIM package that extends the existing package Patrolling SIM made for multi-robot patrolling. The Agrobots-SIM package accommodates dynamic multi-robot task allocation and vehicle routing considering limited robot battery autonomy. Moreover, it accommodates the dynamic assignment of implements to robots for the execution of heterogeneous tasks. The system coordinates task assignment and vehicle routing in real time and responds to unforeseen contingencies during simulation considering dynamic updates of the data related to the environment, tasks, implements, and robots. Except for the ROS and Agrobots-SIM package, other crucial components of the architecture include SPADE3 middleware for developing and executing multi-agent decision making and the FIVE framework that allows us to seamlessly define the environment and incorporate the Agrobots-SIM algorithms to be validated into SPADE agents inhabiting such an environment. We compare the proposed simulation architecture with the conventional approach to 3D multi-robot simulation in Gazebo. The functioning of the simulation architecture is demonstrated in several use-case experiments. Even though resource consumption and community support are still an open challenge in Unity3D, the proposed Agri-RO5 architecture gives better results in terms of simulation realism and scalability.

Keywords: distributed MAS; Unity3D; SPADE3; FIVE; agrirobots; ROS; multi-robot task allocation; multi-robot routing; multi-robot simulation

1. Introduction

In response to the escalating global demand for sustainable and high-yield agriculture, the imperative to deploy autonomous, cost-efficient, and resilient agricultural robot (agrirobot) fleets is increasingly evident, particularly in rural areas marked by labor scarcity and hard working conditions. In such adverse settings, where reliable access to communication networks may also be limited, the integration of digital twins becomes imperative. A digital twin embodies a virtual surrogate of a physical object, system, or process that provides real-time data for monitoring, analysis, and simulation for improved performance and decision making. Digital twins play a pivotal role in enhancing the robustness and efficiency of agriculture robot fleets operating in the edge-cloud continuum.

The goal of the presented research is to propose a multi-agent-based architecture for a realistic simulation of dynamic agrirobot-fleet-coordination methods. This is a challenging issue in rural areas with low connectivity and varying weather-dependent geographically

dispersed farming operations or tasks. The tasks in this context include plowing, harvesting, and pesticide and herbicide spraying, among others. In the medium term run, we aim to develop agrirobot digital twins, which are virtual representations of the physical robots and their environment, to enhance the robustness and efficiency of agriculture robot fleets operating in real-world scenarios.

In this paper, which builds on [1], we propose Agri-RO5, a distributed multi-agent architecture for dynamic fleet simulation that integrates various cutting-edge technologies for a realistic simulation and testing of dynamic agrirobot-fleet-coordination approaches before their employment in the real world. The Agri-RO5 architecture includes advanced simulation tools to tackle the intricate agriculture fleet vehicle routing problem (AF-VRP) in rural areas with low connectivity. AF-VRP considers both dynamic task assignment and vehicle routing, as presented in [2].

Agri-RO5 builds upon the foundation of Agrobots-SIM [1], a powerful simulation package designed for mobile robot coordination in agriculture. Embracing the vision to develop agrirobot digital twins, our system prioritizes a highly distributed architecture, seamlessly facilitated by the incorporation of SPADE3 [3] and FIVE [4]. By harnessing the simulation capabilities of Unity3D (http://unity3d.com (accessed on 22 December 2022)), the proposed architecture offers a highly realistic simulation environment for testing and validating agricultural robot fleets.

SPADE3 (Smart Python Agent Development Environment) is a middleware for developing and executing scalable multi-agent systems (MAS), written in Python. It features a fully open, scalable, and extensible development and execution environment that makes full use of a standard, well-known communication protocol XMPP (eXtensible Messaging and Presence Protocol) (http://xmpp.org (accessed on 22 December 2022)), transparent integration of humans and agents, and a set of development mechanisms which facilitate the implementation of MAS applications.

The Agri-RO5 architecture incorporates the FIVE framework, whose last version includes GTG-CoL algorithms [5], which allows us to easily define the real-world environment and seamlessly incorporate the algorithms to be validated into SPADE agents inhabiting such an environment. It also allows for the creation of three-dimensional environments by using a built-in text-based map editor. In addition, it enables the rapid creation of custom agent avatars, such as, e.g., a mobile robot equipped with sensors (cameras, GPS, LIDAR (Light Detection and Ranging), soil sensors, and force and pressure sensors, among others) or a sensor fixed in the environment. This sensor-rich environment is crucial for developing accurate digital twins that closely mirror the capabilities and challenges faced by real-world agriculture robot fleets, even in harsh environments where communication infrastructure is limited.

The key features of the Agri-RO5 architecture include:

1. *The seamless simulation of dynamic agrirobot-fleet coordination*. Agri-RO5 addresses the critical challenges of dynamic multi-robot task allocation (MRTA), the vehicle routing problem (VRP), and battery autonomy management as well as the dynamic mounting of implements on the robots for heterogeneous task execution.
2. *Simulation realism*: Leveraging Unity3D's capabilities, the proposed architecture provides a realistic simulation environment that closely mimics real-world conditions. This includes a high image resolution and dynamic updates of environment data, implements', tasks', and robots' parameters to account for contingencies that may arise during operations.
3. *The scalable and easy creation of multi-agent systems and three-dimensional environments*. SPADE3 facilitates flexible multi-agent decision making including adaptive choices in dynamically changing situations (see, e.g., [3,6]), and the FIVE framework enables a seamless creation of three-dimensional environments and the incorporation of algorithms into SPADE agents.

Finally, due to the requirement of high simulation fidelity representing the real-world conditions, the implementation in Unity3D and ROS allows for the straightfor-

ward implementation of the coordination solution into the controllers of the real physical robotic systems.

This paper is organized as follows. In Section 2, we describe the state of the art in the simulation of agriculture robot fleets and present the requirements for the simulation architecture for dynamic agriculture robot fleet simulation. Section 3 presents the proposed Agri-RO5 architecture together with its state-of-the-art components: the Robot Operating System (ROS), the proposed Agrobots-SIM package, SPADE middleware, and the FIVE framework. We describe in detail the developments performed in the proposed Agrobots-SIM package that derives from the Patrolling_SIM package [7] in Section 4. In Section 5, we show the functioning of the Agri-RO5 architecture in several use-case experiments available in the GitHub repository (https://github.com/JorgeGutierrezCejudo/AgroRobotSimulator. git (accessed on 22 December 2022)) and compare it with the benchmark Gazebo architecture that we present in this section. Section 6 discusses the implementation choices considering real-world conditions and relevant challenges. The conclusions and the lines of future work in Section 7 close this paper.

2. State of the Art and Architecture Requirements

In this section, we first present the related state of the art in agriculture multi-robot fleet simulation and then list the requirements for a simulation architecture for agriculture multi-robot fleet applications.

The inception of the digital twin (DT) concept, attributed to M. Grieves in a white paper [8], involves the integration of virtual and physical assets within the realm of product lifecycle management. While the utilization of DT has started to grow up in agriculture since 2017 [9], the potential for their pervasive application exists across diverse spatial and temporal scales, accompanied by varying degrees of complexity. A suggested roadmap for the incorporation of digital twins in agriculture, grounded in specific applications, is presented in [10]. An open challenge lies in the large amount of resources they require to be developed and the high complexity of the physical twins [11].

2.1. State of the Art in Multi-Robot System Simulation

Vehicle fleet simulation is a well-researched topic necessary for the efficient implementation of fleet-coordination solutions in the real world for, e.g., UAV fleets [12,13], car fleets [14,15], or train fleets [16]. Contrary to these, service robotics have specific requirements, such as hardware abstraction, device drivers, and communication between processes over multiple machines.

Significant advancements have been made in recent years in the dynamic simulation of mobile multi-robot systems (see, e.g., [17]). One prominent approach is the utilization of Robot Operating System (ROS)-compatible, physics-based simulation engines, such as the open-source simulators MORSE (Modular OpenRobotic Simulation Engine) [18] and Gazebo [19,20], and commercial simulators CoppeliaSim (formerly known as V-REP [21]) and Webots [22]. While several simulators can be integrated with the ROS, Gazebo stands out by offering advanced and complex 3D simulations for both robots and environments. It is often used as the default simulator for the ROS since the two systems work together seamlessly. Gazebo supports the simulation of various robot models and enables the development and testing of algorithms for multi-robot coordination and collaboration.

In the context of agrirobotics, a spectrum of specialized robots exists, each tailored for a specific task (e.g., irrigation, plowing, and harvesting, among others), as well as generic robotic platforms that use detachable heterogeneous implements to perform different tasks [23,24]. However, these robots are very complex, requiring a high degree of autonomy and very elaborate control systems. The availability of simulation tools that represent the physical actuation of robots with high fidelity allows for the experimentation and evaluation of different approaches before their deployment. This way, the cost of deployment in the real world can be reduced. In [25], different agricultural robot-simulation tools were compared. The mobile robot-simulation environment in [26] allows for the analysis of the

performance, cooperation, and interaction of a set of autonomous robots moving in a three-dimensional (3D) world. The FroboMind platform [27] evaluates the task performance in precision agriculture. Nebot et al. [28] present an architecture to control a group of robots in charge of maintenance tasks in agriculture. AgROS [29] is a farm emulation tool that introduces advanced technologies such as autonomous ground vehicles (UGVs). FarmBot [30] is an open-source precision agriculture simulator designed to serve as a cost-effective test bench for exploring and validating precision agriculture strategies before physical implementation.

Teslya et al. [31] propose an architecture based on the smart-space concept for ontology-based information exchange, the ROS for robot control, and Gazebo for 3D modeling and visualization of interaction processes including coalition formation, task decomposition, distribution, and winnings sharing with a functional example in a precision agriculture scenario.

2.2. Simulation Architecture Requirements

An architecture for a dynamic fleet simulation should effectively and efficiently simulate different scenarios in rural areas with varying imperfect connectivity for the agriculture fleet vehicle routing problem (AF-VRP). The AF-VRP [2] considers both the dynamic multi-robot task allocation (MRTA) with and without implements (see, e.g., [32]) and the vehicle routing problem (VRP) (see, e.g., [33]). Overall, a simulation architecture for the AF-VRP that comprises the dynamic MRTA and VRP in the agriculture fleet context should be able to simulate a wide range of scenarios as well as unforeseen contingencies. Its main features should include:

- *Autonomous agent support:* the architecture should be able to simulate autonomous agents, each one being able to make their own decisions without external intervention.
- *Multi-robot task execution:* The architecture should support MRTA and execution as well as robot routing throughout the assigned (multiple) tasks. Inter-agent communication support is fundamental for distributed and decentralized multi-robot collaboration and coordination in task execution.
- *Limited battery life:* Robots are supplied by a limited energy supply through batteries. To support battery recharge in continuous robot missions, the architecture should consider battery recharging points when calculating routes.
- *Efficiency and fairness measures:* The simulation architecture should be able to evaluate the efficiency and fairness of the robots in completing their tasks. This includes measuring the time and cost taken to complete a task, the amount of energy used, and the resources consumed, among others. With this aim, the architecture should support graph-theory approaches with weighted arcs, nodes, or both for optimization purposes.
- *Dynamic task performance:* The architecture should be capable of taking into account the dynamic nature of the scenarios and contingencies related to the execution in real time. This means that the tasks may change over time, and the robots should be able to react dynamically to these changes.
- *Implement-based task performance:* The simulator should be able to simulate tasks that require the use of an implement tool (e.g., a plow or a harvester) that may be attached to and detached from a robot.

3. Agri-RO5: Proposed MAS Architecture for Dynamic Fleet Simulation

In this section, we present the Agri-RO5 architecture, illustrated in Figure 1, as a comprehensive solution for the dynamic agriculture robot fleet simulation. We integrate the Agrobot-SIM package in FIVE that includes Unity3D. FIVE necessitates a nuanced integration with Agrobots-SIM due to its multi-faceted structure.
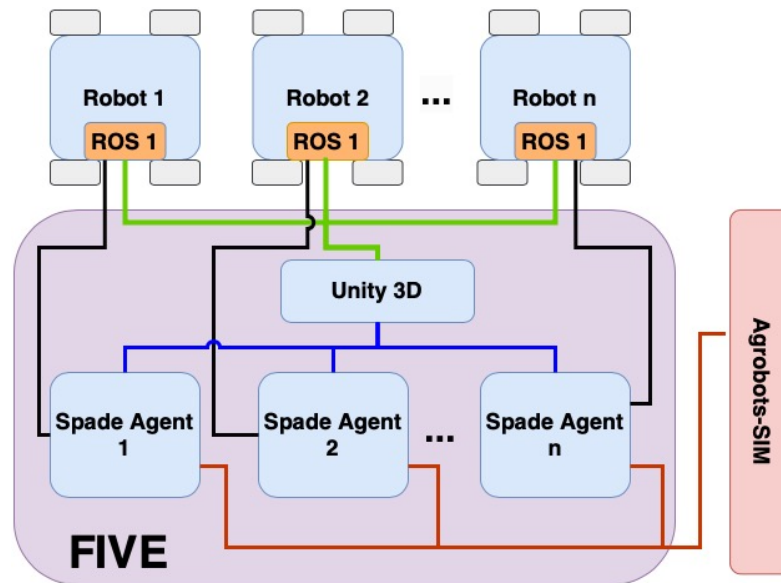
Figure 1. Agri-RO5: proposed multi-agent architecture for dynamic fleet simulation.

To achieve this, we leverage a state-of-the-art solution for ROS and Unity3D integration, thus yielding a powerful simulation platform (see, e.g., [34,35]). This integration is reinforced by specialized plugins and packages like ROS#, which is a set of open-source software libraries and tools in C# for communicating with the ROS from unity (see [36]). Additionally, the Unity Robotics Hub (URH) ([37]) provides a set of tools, libraries, or APIs that help in creating robotics simulations or applications in Unity3D that interact with the ROS and includes standard interfaces like messages and service types. If a ROS package is using some custom message type, the Message Generator Repository must be used, which is a part of the ROS TCP Connector, for sending/receiving messages from the ROS ([38]).

There is an extra layer of SPADE agents, each one in charge of modulating the behavior of a robot. A robot may be seen as a combination of software, "the *mind*", and hardware, (its physical *body*). The ROS controls the reactive movement of the *body* in the environment and its physical dynamics. On the other hand, the *mind* is associated with SPADE agents, one for each robot. These are in charge of inter-robot communication thanks to the XMPP communication protocol and the proactive task assignment for the movement of the *body*: it instructs the body to move towards the following task. The SPADE agents are interacting with the Agrobots-SIM package. Since SPADE agents are written in Python, we can easily create a communication bridge between the robots and the agents through the Rospy API [39].

In the following, we present the components of the proposed architecture: Robot Operating System (ROS), SPADE, FIVE, and Agrobots-SIM.

3.1. Robot Operating System (ROS)

The ROS is a flexible and open-source middleware framework that operates on top of a conventional operating system (such as Linux) and provides a set of tools and libraries for robotics software development [40]. It offers a standardized platform for communication and the integration of diverse hardware and software components [41].

The ROS allows for messaging between different devices (see, e.g., [42,43]) while operating by using a network of nodes (the fundamental processes that perform computation in the ROS) and various methods to exchange data, coordinate actions, and manage and coordinate functionalities (e.g., [44]). A node in the ROS refers to an executable entity that performs a computation. Each node represents a single specific task or function within the robotic system, such as controlling a sensor, actuator, or algorithm. Nodes can reside on the same computer or be distributed across multiple devices or robots.

A mobile robot is typically represented as a collection of nodes. These nodes might handle various functionalities such as sensing, actuation, planning, and control, among others. Each node communicates with others by using predefined protocols, exchanging messages through topics, services, or action servers to coordinate the robot's overall behavior.

Topics are predefined communication channels over which nodes may communicate with each other via asynchronous data streams by sending and receiving messages. Topics have names, and nodes can publish messages to a topic or subscribe to a topic to receive messages. Each topic is meant for a specific type of message, such as sensor data, control commands, or status updates. The communication can be achieved also through *services* by making requests and awaiting responses in a synchronous two-way communication, or by executing long-duration tasks with the capability for feedback and pre-emption by using *action servers* and *action clients*.

The ROS operates based on a networking architecture where a crucial component is the ROS master node. This master node acts as a central broker that manages the registration of various entities such as nodes, topics, services, parameters, and action names. It plays a vital role in facilitating communication between different nodes by allowing them to discover each other, thereby establishing the connections necessary for the exchange of information and functionalities within the robotic system.

The ROS has been applied to the agricultural domain both in low-budget robots with lower performance [45–47] as well as in more complex and high-performance robots [48] while facilitating autonomous navigation [49,50].

3.2. SPADE

SPADE [3] is a middleware for developing and executing behavior-based agents in Python that use instant messaging in XMPP (eXtensible Messaging and Presence Protocol) (https://xmpp.org/ (accessed on 22 December 2022)). This communication protocol allows for a transparent integration of humans and agents in conversations. It not only includes the common behavior types of any behavior-based agent platform (Cyclic, One-Shot, Periodic, Time-Out, and Finite State Machine), but it also has an extension allowing one to use a BDI behavior [6] expressed in AgentSpeak [51].

SPADE has been developed in systems where the integration of humans and agents is not only desired but is wanted to be as transparent as possible. For that reason, agents communicate through an XMPP Server, which is the communication devised for human chat applications, so that the same mechanism can be used for communicating either with humans or agents.

3.3. FIVE Framework

The FIVE framework [4] is a toolkit that has been built for developing systems where SPADE agents can be tested against Unity3D simulations. The main goal when developing FIVE was to obtain a toolkit allowing not only for the development of such simulated systems but also to easily change those simulations.

The FIVE framework is composed of three elements:

1. The FIVE Simulator Server, made with Unity3D, is not only the render engine allowing one to visualize the simulation, but it also manages the environment where agents are going to be situated, offering them the perceptions and actions for proper functioning.
2. A set of SPADE agents that populate the simulated environment. These agents are situated in the simulated environment managed by the FIVE Simulator Server.
3. The XMPP Server. Both the FIVE Simulator Server and the SPADE agents are registered in an XMPP Server to be able to locate each other and to mutually communicate. In fact, each one of them can be registered in a different XMPP Server, and even public XMPP Servers can be used; this part could not be running in the machines of the simulator owners.

Each component can transparently run on separate machines (including, of course, each SPADE agent being executed in a different host).

FIVE agents (based on SPADE) control the virtual avatar in the Intelligent Virtual Environment (IVE) generator managed by the FIVE Simulator Server. The framework grants network-failure toleration: if an agent is disconnected from the FIVE Simulator Server, it can be reconnected easily and resume its activity. These agents are designed as wrappers for the different algorithms to be tested in the generated simulations. Lately, they have been used to test Federated Learning algorithms [52] and the extension to Distributed Federated Learning based on Consensus to Geographical Threshold Graphs, called *GTG-CoL* [5].

3.4. Agrobots-SIM

The ROS libraries for task allocation and planning are the ROSPlan Library [53] and the Task and Motion Planning (TAMP) library (see, e.g., [54–56]). However, the ROSPlan uses the unscalable PDDL (Planning Domain Definition Language), while the TAMP library uses a hierarchical task network (HTN) planner for task decomposition. There are many ways of solving trajectory planning, e.g., [57]. However, none of these can efficiently and effectively model the VRP and the MRTA problem in agriculture, the subject of the Agri-RO5 architecture. However, Patrolling_SIM (https://github.com/davidbsp/patrolling_sim (accessed on 22 December 2022)) is a package for multi-robot patrolling that may be implemented in a multi-robot simulator based on the ROS, Stage [58]. Patrolling and vehicle routing are both problems that involve determining the most efficient and effective routes for a set of vehicles to follow to achieve a specific objective.

In patrolling, the objective is to monitor and secure a particular area or route while maximizing coverage of the area and possibly also the frequency of visits to certain high-priority zones while minimizing the total time or distance traveled, energy consumed, or other operational costs. In the realms of MRTA and the VRP, the objectives are distinct but related. MRTA focuses on allocating, in a one-on-one manner, a group of robots to a set of tasks, ensuring that the tasks are completed efficiently. In contrast, the VRP deals with optimizing the routes of a fleet of vehicles through a set of given (spatially distributed) tasks to ensure an efficient performance of a set of tasks by a vehicle fleet. While the MRTA problem focuses on the one-on-one allocation of tasks to robots and vice versa, in the VRP, one robot may perform multiple tasks. In both problems, the tasks could be as diverse as surveillance, item delivery or pickup, or search and rescue. For all the above reasons, we adapt and extend the Patrolling_SIM package and create Agrobots-SIM [1], explained in detail in the next section.

4. Agrobots-SIM Developments

In this section, we introduce the Agrobots-SIM package, which is a fundamental component of the Agri-RO5 architecture that we developed for the architecture. It is based on the Patrolling_SIM package. We outline the extensions made to the latter, specifically tailored to address the requirements for simulating agriculture fleet vehicle routing and task allocation as outlined earlier.

4.1. Patrolling_SIM

The Agrobots-SIM package builds upon the Patrolling_SIM package, designed for algorithm testing related to patrolling tasks executed by a team of robots within a predefined set of locations.

The scenario is modeled by using a graph representation, where nodes represent locations to be visited and arcs depict physical links between adjacent nodes. Each robot is iteratively assigned a task (next node to visit) based on a specific decision strategy. Eleven patrolling algorithms are integrated into the package, with node idleness (average time between robot visits) serving as a key performance indicator. Even though patrolling, task assignment, and vehicle routing are distinct problems, they share several similarities in the objectives, constraints, and optimization techniques involved, which we consider sufficient to adapt the Patrolling_SIM package to MRTA and the VRP in the agrirobot context. This is

why we study and extend this package. The Patrolling_SIM architecture (see Figure 2) is composed of the following:

1. Environment: If we compare the environment to a theater play, the simulation environment is the instance where the stage, the actors, and the play to be performed will be defined. In our case, the stage will be the *map*, the actors will be the *robots*, and the play will be the *graph*.

2. Algorithm: The role of decision-making algorithms is to calculate the next task for each robot based on the given problem definition. There are eleven patrolling algorithms included in the simulator.

3. Simulate engine: Patrolling_SIM follows an event-based approach as the system conditions change when an event occurs. In addition, this simulator allows for event creation.

4. Visualization: This module allows for the real-time visualization of robot behavior following a given routing or task-assignment algorithm. In addition, it can serve as a filter to evaluate whether the high-fidelity simulation considering robotic dynamics reflects the expected behavior of the algorithm. Among the options offered by the visualization module, one can visualize the range of the robot's sensors, follow a specific robot, or even observe the simulation from a robot's point of view.

5. Data analysis: Once a patrolling cycle is complete, a file with the results is created. A patrol cycle is completed when all points are visited twice. We need to remember that this package was initially created for patrolling, so the main feature to compare different algorithms is the time between each visit or *idleness*, i.e., the duration between two robot visits to a task.
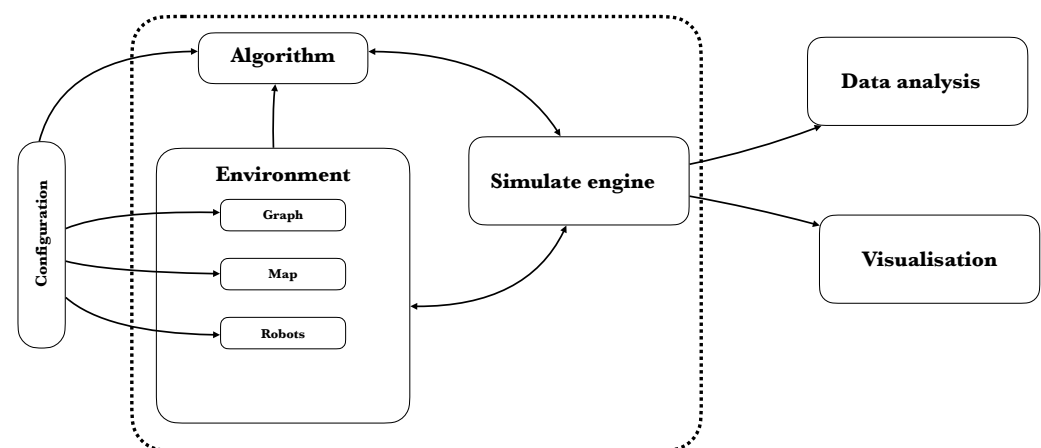

Figure 2. Patrolling_SIM architecture.

Next, we discuss how we extended and modified the Patrolling_SIM package to accommodate for agricultural robot-fleet coordination.

4.2. Agrobots-SIM: A Modification of Patrolling_SIM

In this section, we bring the modifications performed in Patrolling_SIM necessary for the simulation of agriculture fleet vehicle-coordination methods.

4.2.1. Graph Representation of the Transportation Network

We use a directed weighted graph $G = (V, E)$ representing a rural area of interest, where V is a set of vertices corresponding to available robot locations and E is a set of arcs $(i, j) \in V$ connecting any two adjacent vertices. This graph is superposed on a 2D map of the region of interest, providing a visual representation of the robot-transportation network.

The 2D map serves as a spatial reference for the graph, where each vertex is associated with a 2D coordinate on the map, mapping the physical position of the robot in the real-world environment. The edges between vertices represent obstacle-free paths for robot movement between the corresponding locations.

Furthermore, the graph representation can be extended to incorporate additional information, such as travel times, energy consumption, or specific characteristics of the locations. This enriched graph representation enables a more comprehensive analysis of the transportation network, facilitating the development of efficient and effective strategies for robot deployment and navigation.

The tasks, as well as stations for battery charging and tools exchange, are defined on vertices. Since the robots move considering graph G, the granularity and precision in the graph design are essential for the seamless functioning of the robot fleet.

The movement of the robots is based on the action of the move-base ROS package. The move-base package provides an implementation of an action that, given a goal in the world, will attempt to reach it with a mobile base. The ROS move-base node links together a global and a robot's local planner to accomplish its global navigation task. Robots only rely on their local graph and map to avoid any obstacles. They must be dynamically updated for seamless and fluid robot operation.

Therefore, we have to take into account how we designate and generate the graph. An example of a final graph could be the one shown in Figure 3, where we divide the graph into different types of edges depending on their function.
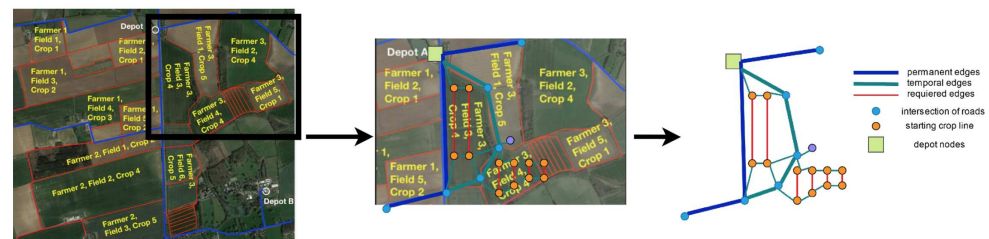

Figure 3. A graph example.

4.2.2. Route Cost as a Parameter of Comparison

In patrolling, the most relevant parameter is idleness (time between two consecutive visits to a task). However, in the agriculture robot-fleet coordination, one of the most important indicators to compare and analyze the routing efficiency is the route's cost. This may be, for example, the distance or time traveled, or the monetary cost, of each robot and the cost of the fleet as a whole. This is why we modified the data analysis module to include the route cost as a parameter of comparison.

This information may serve for the analysis of efficiency and fairness measures of the fleet. For this purpose, we used the previously built topic *position* in terms of the (x, y) coordinates of each robot in each period of time. By measuring the distance traveled by the same robot between two consecutive steps and accumulating it, we keep track of the accumulated cost of the route.

4.2.3. Robot Recharging Nodes

One of the biggest robots' limitations is their battery life. This limitation is taken into account in the development of Agrobots-SIM. We distinguish between *recharge vertices* $V_r \in V$ (where robots must go to recharge their battery) and *task vertices* $V_t \in V$ (with pending tasks to perform). We created a separate data structure for these vertices so that algorithms can take this constraint into consideration if needed. The structure of the charging vertices consists of the identifier, the coordinates of the vertex, the cost to reach the vertex, and an indicator of whether the vertex is momentarily in use or not (*idle or occupied*, respectively), and in this way, we dynamically consider the time necessary to charge the battery. A recharge vertex is only reachable if a robot's remaining autonomy required to reach it is not greater than its remaining battery autonomy.

4.2.4. Implements

In agriculture robotics, robots have to use different implements (tools) for different tasks (e.g., plow, harvester, etc.). Tasks may differ in the requirements, while robots may differ in implement compatibility. From this perspective, we distinguish two cases: (i) a robot with an inseparable implement that is specialized for a certain task, where compatibility is considered only between a robot and a task, and (ii) a robot requires a specific detachable implement to perform a task.

The exchange of the implements is defined on the set of vertices $v \in V_e$. A robot may detach its mounted implement and, if necessary, attach a new one available on the location of these vertices that have a sufficient infrastructure to deposit the implements. These vertices may overlap with the task vertices $v \in V_t$, where $V_t \subset V$.

Initially, we know the compatibility between implement and task. This means that we know which task a robot with a specific implement can perform. In the beginning, each robot is allocated an implement, whereas several robots may have the same implement. Each implement has a list of tasks that the robot can perform with this implement. In Figure 4, we can see an example of this, where each robot is assigned an implement and each implement has a series of tasks to perform defined with the same color as the implement. So, the robot with the pink implement can only realize the pink task.

The decision to allocate which implement is coupled to each robot corresponds to the resolution algorithm, which, depending on the benefit function, will indicate one or the other solution. Furthermore, the algorithm is aware of the task-implement compatibility information and will assign each robot a task that it can perform with the available implement.

Figure 4. Example of implements implementation.

In the future, we want to consider the fact that the robot can change the implement at any moment because they have dynamic behavior, so they can decouple the implement in any place. The information about the implement is considered in the graph, such as a priority task or goal to visit to realize the rest of the task.

4.2.5. Dynamic Graph Update

A dynamic task is a task that appears or its parameters change at a certain time during the simulation (e.g., task demand, resource requirements, duration, cost, etc.). The original Patrolling_SIM architecture does not permit this kind of update because the simulation engine loads the tasks at the start of the simulation from the graph file, and they do not

change throughout the simulation. To solve this problem, we used the creation of an event. Triggered by an event, we can load, at any moment, a new graph file. The "*all tasks completed*" event is triggered when a robot completes a given set of tasks. At this moment, the graph of the robot may be changed to a new one so it may continue pursuing new, previously unknown tasks. We must keep in mind that the graph change will only occur locally in the robot that has visited these tasks and not globally (each robot maintains its own local (compatible) version of the graph).

Another event that we propose is "*all tasks completed*". This event, which occurs when all tasks are executed, changes the graph in all robots at the same time. This can be performed on an ongoing basis, switching from one graph to another when a set of tasks is done. However, in order to guarantee a continuous task performance, the assignment of tasks to the robots must be balanced, thus avoiding some robots waiting for others to finish.

Moreover, we simulate a robot breakdown by the update of its graph to the one with only one isolated vertex representing its actual position.

4.2.6. RViz as a Debugging Tool

We incorporated RViz (an abbreviation for 'ROS visualization') as an additional visualization module, employing it as a tool for debugging our simulation. RViz is a 3D visualization software tool for robots, sensors, and algorithms. It allows for visualizing the robot's perception of its world (real or simulated) and displaying multiple pieces of information from the different types of messages of the ROS (topics, services, and actions). Furthermore, RViz is up to date with the latest ROS distributions. It allows for the use of markers, which are shapes that can be displayed at any moment. These markers open up new options for the study of algorithms since it is possible to simulate the appearance of dynamic obstacles and thus analyze the behavior of the dynamic algorithms.

5. Simulation Use Case

In this section, we look at some concrete simulations to show how Agri-RO5 works. We compare its performance with a benchmark architecture in Gazebo that we present in this section. The simulation experiments were performed on an MSI Prestige 16 A 12UD laptop with 32 Gb of memory, 16 Gb of RAM memory, Intel Core i7 of the 12th generation, and a GPU GeForce RTX 3050 Ti with 4 Gb of memory.

Both architectures were tested in an identical simulated vineyard setting defined as follows. A fleet of three autonomous robots (Warthog robot of Clearpath (https://clearpathrobotics.com/warthog-unmanned-ground-vehicle-robot/ (accessed on 22 December 2022))) modeled through the ROS are initially randomly located in a given three-dimensional space with wine trees distributed in 10 columns and six rows, as seen in Figure 5. We replicate the environment in FIVE (to be used by Agri-RO5) and Gazebo (benchmark) to facilitate a comparative analysis. The Agri-RO5 environment generated by Unity3D is illustrated in Figure 5, and the environment generated in Gazebo is illustrated in Figure 6.

For the simulation use-case experiments available in the GitHub repository (https://github.com/JorgeGutierrezCejudo/AgroRobotSimulator.git (accessed on 22 December 2022)), given is a set of tasks in the space (locations to visit). The robots should mutually assign the tasks and coordinate to avoid each other in their performance of the assigned tasks in the simulated vineyard.

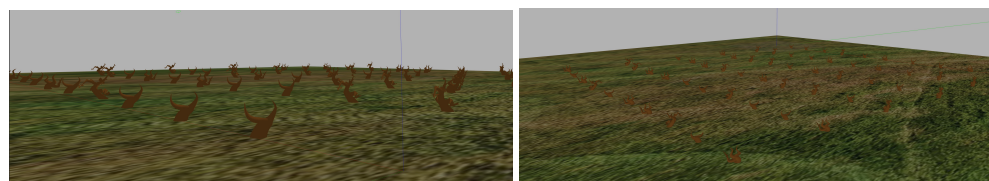**Figure 5.** Agri-RO5 environment generated by Unity3D.

Figure 6. Environment generated in Gazebo.

5.1. Benchmark: Agrobots-SIM with Gazebo

By the integration of the Agrobots-SIM package into Gazebo, we can generate 3D simulations of agriculture fleet vehicle-task allocation and routing with realistic robot physics and behavior. Gazebo specializes in the high-fidelity simulation of robots with their sensors and actuators as well as the environments they act in. It represents the physical aspect of different robot models through URDF (Unified Robot Description Format) and the physical behavior of the robot through the included plugins. Simulated sensors can publish data to the same topics used by real robots' sensors in the ROS. Similarly, simulated actuators subscribe to and operate on the topics used by real robot actuators.

Figure 7 shows the Gazebo integrated Agrobots-SIM architecture. Each robot is guided by the task assignment and routing solution provided by the algorithms included in Agrobots-SIM.
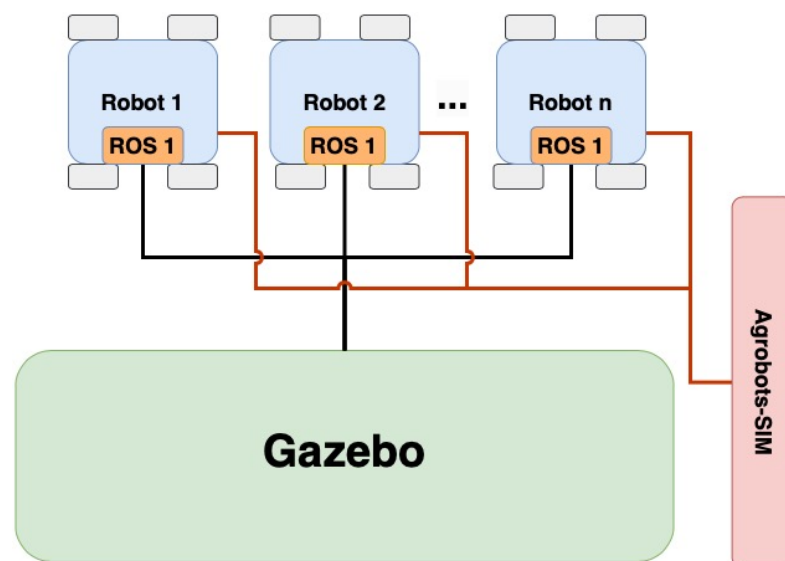

Figure 7. Agrobots-SIM + Gazebo.

Gazebo Setup

For this simulator, in the first instance, we used the ROS package *map2gazebo* that generated a bitmap file for the Gazebo environment. As shown in Figure 8, the gener-

ated world, although with proportions and measurements that corresponded to a real geographical map, was not representative of a vineyard.
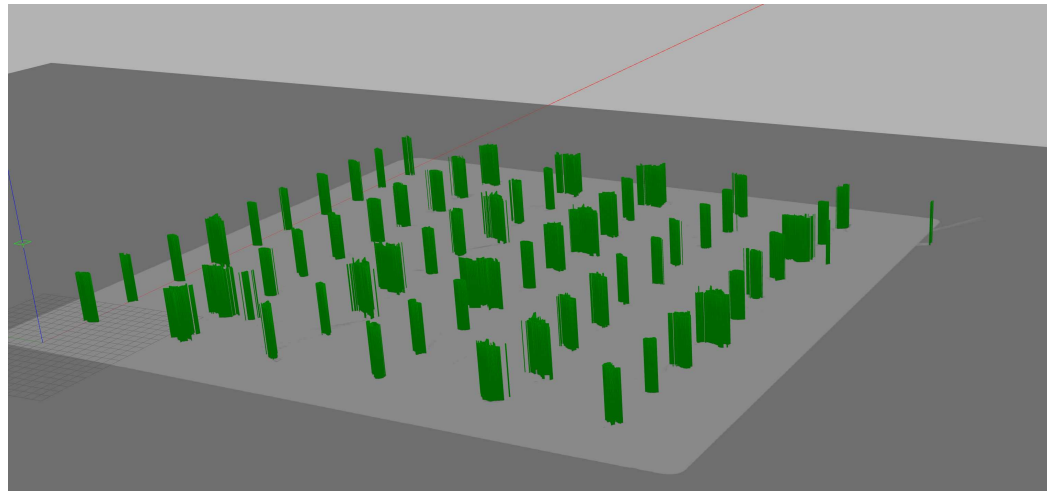

Figure 8. Gazebo environment created by the *map2gazebo* package.

Thus, we manually incorporated each of the vine trees, trying to follow the same topology as in Unity3D. In order for the environments to have the same measurements and proportions, we relied on the bitmap generated by the SLAM (Simultaneous Localization and Mapping) that we will see in the next section.

The incorporation of the robots in Gazebo is easy thanks to the available documentation and previous work on the use of robots in Gazebo. At the same time, there are already different plugins in Gazebo to simulate sensors, controllers, and actuators, which make their incorporation smooth and easy.

We assume an ideal behavior of sensors without errors in readings, giving an exact location of a robot. In this way, we do not need to rely on localization packages like *amcl* or *erkf_localization* to obtain robots' position in real time. We do so because we are comparing other features such as the simulator stability and other options. To obtain the exact position of each robot, we generate a Python script in order to configure the navigation. For the path and trajectory planner, we use move_base in both cases. However, we need to obtain the bitmap of the environment for the use of move_base. Therefore, we decided to apply the SLAM algorithm in the Unity3D environment and in this way dynamically generate the map of the environment during a multi-robot mission.

5.2. FIVE Setup for Agri-RO5

For the development of the environment in Unity3D, we employed FIVE, a tool that streamlined the creation of maps and environments. Through this tool, we successfully generated the environment depicted in Figure 5 by seamlessly integrating different vineyard models and selecting their structure patterns by configuring only a text file.

The virtual three-dimensional environment simulation is generated from a high-fidelity satellite image of the crop field based on a wine tree model database that recognizes the shapes and forms of the trees in a satellite image. However, it is also possible to create other orchards like an orange orchard manually without using the satellite image but modifying the configuration text files.

Once the environment is generated, the incorporation of the robots in Unity3D is not so simple and agile as in Gazebo. It is true that there are Unity3D tools to incorporate robots and work with the ROS, but there is not as much previous work as in the case of Gazebo. This is why we created, through C# scripts, each of the robots' sensors needed for navigation, as well as the robots' motion controllers. The aforementioned tools provide easy access to classes and other functions to recreate the realistic behavior of a robot.

Finally, we configure robot localization and navigation. Again, we chose to use a perfect localization and the ROS's package (node) *move_base* to calculate and plan the trajectory of each robot. As mentioned above, the SLAM algorithm was applied in the Unity3D environment. As seen in Figure 9, SLAM granted a dynamic creation of a map with high accuracy in Unity3D.
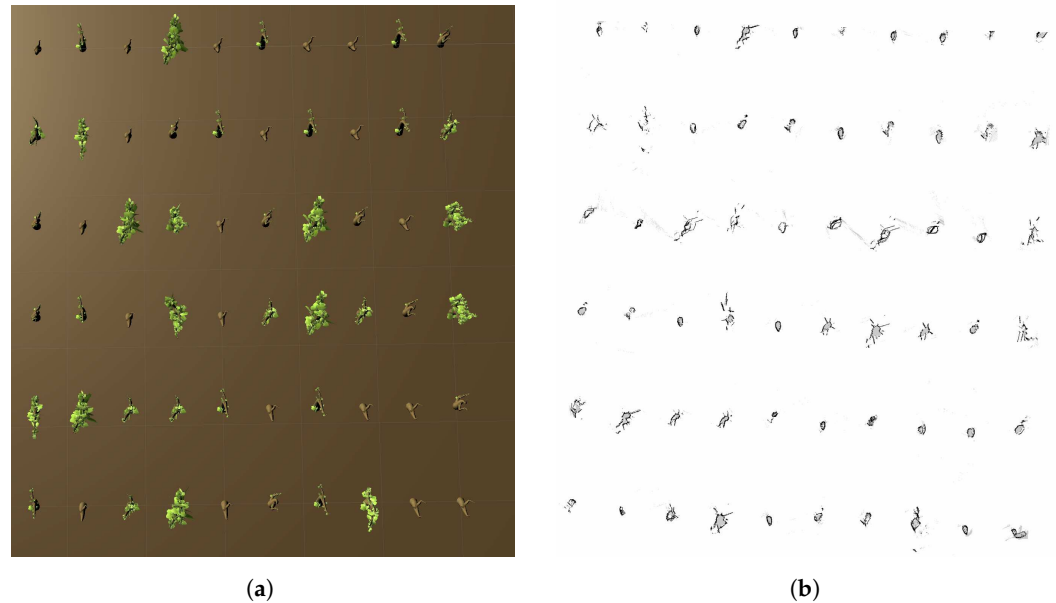

(**a**)   (**b**)

Figure 9. SLAM solution for Unity3D environment. (**a**) Unity3D environment. (**b**) Bitmap of the Unity3D environment.

6. Discussion

We simulated a fleet of three robots, as seen in Figure 10, and assigned them a set of tasks' coordinates through the graphical interface of RViz. We compare the scalability, ROS integration, development workflow, accuracy in terms of simulation realism, resource consumption, and customer support and documentation.

Figure 10. Three robots simulation.

Both simulations were carried out on the same computer to guarantee the equality of conditions between the two experiments. During the simulation, Unity3D's latency was very high, and the simulation showed intermittent (not fluid) behavior, so the Unity3D simulation was carried out on a different, stronger machine.

Because coordinates had to be sent through RViz, the run time varied depending on the time it took to modify the coordinates and the receiver robot of such coordinates.

Nevertheless, the same number of tasks was assigned to each robot, and in all cases, all robots successfully completed the tasks. Additionally, a good performance of the ROS obstacle-avoidance algorithms was observed in both simulators.

Next, we compare different features between Agri-RO5 and the Agrobots-SIM+Gazebo architecture, which are summarized in Table 1. These features highlight the most notable differences between these two architectures.

Other differences in the integrating components of Agri-RO5 and Agrobot-SIM+Gazebo have been compared in other articles, such as, e.g., [59], with an emphasis on the differences between Unity3D and Gazebo. It is important to note that in this case, we are not comparing Unity3D and Gazebo; rather, we are comparing Agri-RO5 with Agrobot-SIM+Gazebo.

Table 1. Comparison of Agrobots-SIM + Gazebo vs Agri-RO5.

Feature	Agrobots-SIM + Gazebo	Agri-RO5
Scalability	Time consuming	Simple and easy
ROS integration and development workflow	High integration	Low integration
Accuracy: simulation realism	Low fidelity	High fidelity
Resource consumption	Low resources needed	High resources needed
Support and documentation	Documentation, examples, and questions	Documentation

1. Scalability. Unity3D permits extending the map, modifying the models, and changing the environment just by changing a text file, or even by capturing a satellite image of a crop field, which makes these modifications very simple. On the other hand, in Gazebo, extending the environment modifying the models or changing them has to be performed manually, which is much more time consuming. Moreover, Agri-RO5 is a distributed system allowing one to execute each simulated robot in a different machine; allowing one to scale the number of robots in a simulation in ways that the Agrobots-SIM integrated in Gazebo does not permit as being centralized; and executing, in a single machine, the whole simulation.

2. ROS integration and development workflow. Following this discussion, let us explore the utilization options offered by each simulator. Gazebo is particularly associated with robotics, so it offers a greater variety because of numerous plugins that enable the integration of different sensors and controllers. However, these plugins are not 100% modular, so they have limitations in their use. To address this, you can always create scripts that utilize Gazebo's services and actions to generate fully customized sensors configured to one's preferences. On the other hand, as we have mentioned throughout this paper, Unity3D lacks these plugins or anything similar. Therefore, each sensor and necessary component for using the ROS must be created individually. To contextualize and exemplify this, to use move_base in the ROS, primarily, you need information from the map (handled by the map_server node external of any simulator), sensor information, robot odometry, and its transformations. In Gazebo, if precise localization is not necessary, you would only need to import the plugins for the robot controller and sensor to have the information required. However, in Unity3D, lacking these plugins, you would have to create modules through C# scripts to generate this information, which is more labor intensive.

3. Accuracy: simulation realism. It is obvious that Unity3D has more detailed graphic representations when compared to Gazebo, as can be seen in Figure 11. This is because Unity3D is widely used for the creation of video games or simulators. However, at this point, we also want to focus on physics. In Unity3D, we had some problems incorporating the robots and creating the controller for the wheel movements because the physics were so realistic that they made the robot skid and wheelie. This means that with Unity3D, you can

have a very accurate simulation with a high fidelity to reality. As for Gazebo, its physics are more controlled and closer to reality but less accurate.
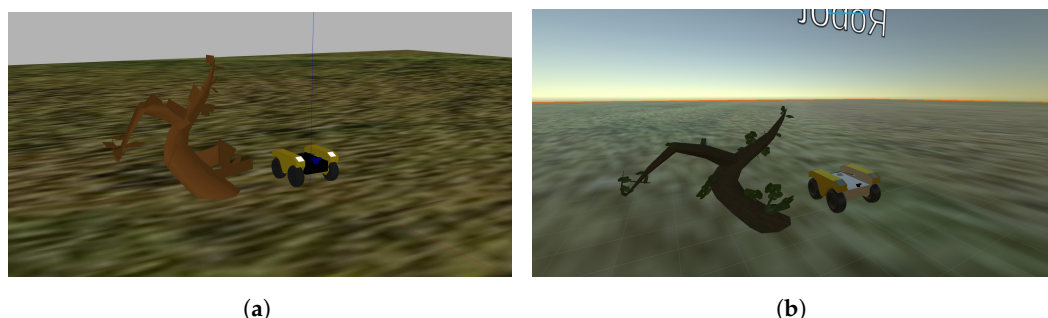

(**a**)   (**b**)

Figure 11. Comparison of models. (**a**) Gazebo models. (**b**) Unity3D models.

4. Resource consumption. Both simulators are stable. However, in Unity3D, the robot models and the environment are more realistic, requiring more computational resources. This means that on computers with low resources, the simulation can have large latency, resulting in intermittent simulation behavior, as was the case in the performed use-case simulations, where firstly the simulations were carried out on a computer with low resources, where it was observed that in Unity3D, there was latency, while in Gazebo, the simulation was fluid. However, in a second simulation with a computer with a graphics card and more resources, this problem was solved, and both simulators were fluid. This fact is well known in the literature (see, e.g., [59]).

5. Community support and documentation. Regarding the need to find a solution when faced with any issue, Gazebo provides documentation on its official website, explaining almost all the features of the simulator with various examples. Additionally, given its widespread use in robotics, there are many tutorials and forums with resolved questions that can be helpful when addressing problems. However, while there is documentation for each feature and functionality of the Unity3D simulator, robot integration is still not widely documented. Consequently, there are much less resources with answered questions or tutorials. Thus, it is more challenging to address issues that arise during multi-robot fleet simulation development in Unity3D.

7. Conclusions and Future Work

In this paper, we developed and integrated Agrobots-SIM with the FIVE framework to create Agri-RO5, a multi-agent architecture for agriculture fleet simulation. Agrobots-SIM is a ROS package based on Patrolling_SIM used for task assignment and vehicle routing in the coordination of agrirobot fleets. Agri-RO5 provides a more realistic tool for the development and evaluation of robot-fleet-coordination algorithms in agricultural environments than the traditional robot-simulation Agrobot-SIM+Gazebo benchmark architecture.

The Agri-RO5 architecture allows for comparing different fleet-coordination algorithms for MRTA and the VRP in simulation experiments, where robots are modeled as intelligent agents. We focus on the limited robot battery life and the fleet's efficiency and fairness measures in dynamic environments. The simulator facilitates graph representations with dynamic graph updates and route cost as a parameter of comparison. Apart from being able to simulate the performance of the aforementioned algorithms, Agri-RO5 provides information both in real time (through the incorporation of RViz) and once the simulation is finished.

The Agri-RO5 developments are a part of a higher vision of developing agrirobot digital twins, where the coordination between the simulated and physical robot agents should be performed in the edge-cloud continuum. Thus, Agri-RO5 is the first step in a paradigm shift in the simulation and coordination of agrirobot fleets. The algorithms that require more computational resources may be run in the cloud, while edge computing is

used for reactive behaviors in real time due to unpredicted contingencies and a lack of communication networks.

Agri-RO5 offers researchers and practitioners a unified platform to develop, test, and refine their agricultural robot-fleet-coordination solutions, ultimately leading to more efficient and sustainable agricultural practices. However, the proposed platform is centralized since ROS1 works with a central master node. Furthermore, the drawback of the ROS1 structure is that it is centralized, relying on the ROS master node as a single point of failure. If the ROS1 master node faces a crash or malfunction, it hinders the establishment of new connections between nodes within the network. This aspect compromises the robustness and reliability of the system. We aim to take an additional step in process distribution by incorporating ROS2 to resolve this issue. ROS2 uses a middleware communication framework called the Data Distribution Service (DDS) that acts as a universal messenger for scalable and real-time data exchange between ROS nodes. DDS allows for the dynamic discovery of nodes, enabling nodes to communicate directly with each other without the ROS master node. Through DDS, the nodes can join or leave the network smoothly and can easily and automatically find and connect with each other. DDS offers benefits in terms of real-time communication, better security (the encryption of messages, authentication, and access control), and various Quality of Service policies (see, e.g., [60]).

In Agrobots-SIM, we added the following functionalities to the Patrolling_SIM package: a weighted directed graph representing the transportation network for the robots; the option of detachable implements that may be needed for the execution of agriculture tasks by robots; dynamic changes in the costs of a route for each vehicle during simulation; and adaptation from the patrolling problem to the multi-robot task-allocation problem and the vehicle routing problem, where the simulation stops when the last task is visited. Moreover, in the graph representation, we introduced stations for battery recharge and implement exchange, located at graph vertices.

Finally, in the long run, we envision agriculture robot digital twins that will enable the application of autonomous agriculture robot fleets in distant and harsh environments with limited communication. The idea is to explore how the proposed Agri-RO5 architecture matches up to the real environment, evaluating its level of accuracy. This process involves carrying out tests and simulations with real robots, allowing us to compare the performance and behavior of virtual systems with their physical counterparts. The goal is to gain a deeper understanding of the fidelity of the Agri-RO5 architecture and its ability to faithfully replicate real-world situations. This will lead us towards validation with real robots and will provide valuable information to refine and adjust our simulator. This step must also consider possible errors in the communication network. By considering imperfect communication networks prone to errors in the simulation, we will obtain digital twins of agriculture robot fleets with high applicability in practical and diverse harsh environments.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Gutiérrez-Cejudo, J.; Lujak, M.; Fernández, A. Agrobots Architecture and Agrobots-Sim Simulator for Dynamic Agri-Robot Coordination. In *Communications in Computer and Information Science*; Springer Nature: Cham, Switzerland, 2023; pp. 5–17. [CrossRef]
2. Lujak, M.; Sklar, E.; Semet, F. Agriculture fleet vehicle routing: A decentralised and dynamic problem. *AI Commun.* **2021**, *34*, 55–71. [CrossRef]
3. Palanca, J.; Terrasa, A.; Julian, V.; Carrascosa, C. Spade 3: Supporting the new generation of multi-agent systems. *IEEE Access* **2020**, *8*, 182537–182549. [CrossRef]
4. Carrascosa, C.; Enguix, F.; Rebollo, M.; Rincon, J. Consensus-Based Learning for MAS: Definition, Implementation and Integration in IVEs. *Int. J. Interact. Multimed. Artif. Intell.* **2023**, *8*, 21–32. [CrossRef]
5. Rebollo, M.; Rincon, J.; Hernández, L.; Enguix, F.; Carrascosa, C. GTG-CoL: A New Decentralized Federated Learning Based on Consensus for Dynamic Networks. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems, Guimarães, Portugal, 12–14 July 2023; Springer: Berlin/Heidelberg, Germany, 2023; pp. 284–295.
6. Palanca, J.; Rincon, J.A.; Carrascosa, C.; Julian, V.J.; Terrasa, A. Flexible Agent Architecture: Mixing Reactive and Deliberative Behaviors in SPADE. *Electronics* **2023**, *12*, 659. [CrossRef]
7. Portugal, D.; Iocchi, L.; Farinelli, A. A ROS-based framework for simulation and benchmarking of multi-robot patrolling algorithms. *Robot. Oper. Syst. (Ros) Complet. Ref.* **2019**, *3*, 3–28.
8. Grieves, M. Digital twin: Manufacturing excellence through virtual factory replication. *White Pap.* **2014**, *1*, 1–7.
9. Purcell, W.; Neubauer, T. Digital Twins in Agriculture: A State-of-the-art review. *Smart Agric. Technol.* **2023**, *3*, 100094. [CrossRef]
10. Pylianidis, C.; Osinga, S.; Athanasiadis, I.N. Introducing digital twins to agriculture. *Comput. Electron. Agric.* **2021**, *184*, 105942. [CrossRef]
11. West, T.D.; Blackburn, M. Is digital thread/digital twin affordable? A systemic assessment of the cost of DoD's latest manhattan project. *Procedia Comput. Sci.* **2017**, *114*, 47–56. [CrossRef]
12. Du, J.; Fan, Y.; Wang, K.; Feng, Y.; Yu, Y. AeroBotSim: A High-Photo-Fidelity Simulator for Heterogeneous Aerial Systems Under Physical Interaction. In *Communications in Computer and Information Science: Proceedings of the Cognitive Systems and Information Processing, Luoyang, China, 10–12 August 2023*; Sun, F., Cangelosi, A., Zhang, J., Yu, Y., Liu, H., Fang, B., Eds.; Springer Nature: Singapore, 2022; pp. 274–287.
13. De Rango, F.; Palmieri, N.; Santamaria, A.F.; Potrino, G. A simulator for UAVs management in agriculture domain. In Proceedings of the 2017 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS), Seattle, WA, USA, 9–12 July 2017; pp. 1–8.
14. Hörl, S.; Ruch, C.; Becker, F.; Frazzoli, E.; Axhausen, K. Fleet operational policies for automated mobility: A simulation assessment for Zurich. *Transp. Res. Part Emerg. Technol.* **2019**, *102*, 20–31. [CrossRef]
15. Bischoff, J.; Maciejewski, M. Simulation of City-wide Replacement of Private Cars with Autonomous Taxis in Berlin. *Procedia Comput. Sci.* **2016**, *83*, 237–244. [CrossRef]
16. Faccio, M.; Gamberi, M.; Persona, A.; Regattieri, A.; Sgarbossa, F. Design and simulation of assembly line feeding systems in the automotive sector using supermarket, kanbans and tow trains: A general framework. *J. Manag. Control* **2013**, *24*, 187–208. [CrossRef]
17. Farley, A.; Wang, J.; Marshall, J.A. How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion. *Simul. Model. Pract. Theory* **2022**, *120*, 102629. [CrossRef]
18. Echeverria, G.; Lassabe, N.; Degroote, A.; Lemaignan, S. Modular open robots simulation engine: Morse. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 46–51.
19. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2149–2154.
20. Noori, F.M.; Portugal, D.; Rocha, R.P.; Couceiro, M.S. On 3D simulators for multi-robot systems in ROS: MORSE or Gazebo? In Proceedings of the 2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), Shanghai, China, 11–13 October 2017; pp. 19–24.
21. Rohmer, E.; Singh, S.P.; Freese, M. V-REP: A versatile and scalable robot simulation framework. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1321–1326.
22. Michel, O. Cyberbotics ltd. webots™: Professional mobile robot simulation. *Int. J. Adv. Robot. Syst.* **2004**, *1*, 5. [CrossRef]
23. Cheng, C.; Fu, J.; Su, H.; Ren, L. Recent Advancements in Agriculture Robots: Benefits and Challenges. *Machines* **2023**, *11*, 48. [CrossRef]
24. Tiozzo Fasiolo, D.; Scalera, L.; Maset, E.; Gasparetto, A. Towards autonomous mapping in agriculture: A review of supportive technologies for ground robotics. *Robot. Auton. Syst.* **2023**, *169*, 104514. [CrossRef]
25. Shamshiri, R.; Hameed, I.; Pitonakova, L.; Weltzien, C.; Balasundram, S.K.; Yule, L.; Grift, T.E.; Chowdhary, G. Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison. *Int. J. Agric. Biol. Eng.* **2018**, *11*, 15–31. [CrossRef]

26. Emmi, L.; Paredes-Madrid, L.; Ribeiro, A.; Pajares, G.; de Santos, P.G. Fleets of robots for precision agriculture: A simulation environment. *Ind. Robot. Int. J.* **2013**, *40*, 41–58. [CrossRef]

27. Jensen, K.; Larsen, M.; Nielsen, S.; Larsen, L.; Olsen, K.; Jørgensen, R. Towards an Open Software Platform for Field Robots in Precision Agriculture. *Robotics* **2014**, *3*, 207–234. [CrossRef]

28. Nebot, P.; Torres-Sospedra, J.; Martínez, R. A New HLA-Based Distributed Control Architecture for Agricultural Teams of Robots in Hybrid Applications with Real and Simulated Devices or Environments. *Sensors* **2011**, *11*, 4385–4400. [CrossRef] [PubMed]

29. Tsolakis, N.; Bechtsis, D.; Bochtis, D. Agros: A robot operating system based emulation tool for agricultural robotics. *Agronomy* **2019**, *9*, 403. [CrossRef]

30. Murcia, V.A.; Palacios, J.F.; Barbieri, G. Farmbot simulator: Towards a virtual environment for scaled precision agriculture. In *Service Oriented, Holonic and Multi-Agent Manufacturing Systems for Industry of the Future: Proceedings of SOHOMA LATIN AMERICA 2021*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 234–246.

31. Teslya, N.; Smirnov, A.; Ionov, A.; Kudrov, A. Multi-robot coalition formation for precision agriculture scenario based on gazebo simulator. In Proceedings of the 15th International Conference on Electromechanics and Robotics "Zavalishin's Readings" ER (ZR) 2020, Ufa, Russia, 15–18 April 2020; Springer: Berlin/Heidelberg, Germany, 2021; pp. 329–341.

32. Lujak, M.; Salvatore, A.; Fernández, A.; Giordani, S.; Cousy, K. How to fairly and efficiently assign tasks in individually rational agents' coalitions? Models and fairness measures. *Comput. Sci. Inf. Syst.* **2023**, 75. [CrossRef]

33. López Sánchez, A.; Lujak, M.; Semet, F.; Billhardt, H. How to achieve fair and efficient cooperative vehicle routing? *AI Commun.* **2023**, 1–23. [CrossRef]

34. Andreasen, M.Z.; Holler, P.I.; Jensen, M.K.; Albano, M. MAES: A ROS 2-compatible simulation tool for exploration and coverage algorithms. *Artif. Life Robot.* **2023**, *28*, 757–770. [CrossRef]

35. López, D.D.L.P.; Orta, C.A.P.; Chávez, F.M.; Coronado, L.M.V. ROS2 and Unity based Simulation for telepresence robot. In Proceedings of the 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Male, Maldive, 16–18 November 2022; pp. 1–6.

36. Siemens. Ros-Sharp. 2023. Available online: https://github.com/siemens/ros-sharp (accessed on 22 December 2022).

37. Technologies, U. Unity Robotics Hub. 2023. Available online: https://github.com/Unity-Technologies/Unity-Robotics-Hub (accessed on 22 December 2022).

38. Technologies, U. ROS TCP Connector. 2023. Available online: https://github.com/Unity-Technologies/ROS-TCP-Connector (accessed on 22 December 2022).

39. ROS. Rospy. 2023. Available online: https:https://github.com/ros/ros_comm.git (accessed on 22 December 2022).

40. Joseph, L. *Robot Operating System (ROS) for Absolute Beginners*; Springer: Berlin/Heidelberg, Germany, 2018.

41. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source SW, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.

42. Quigley, M.; Gerkey, B.; Smart, W.D. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2015.

43. Joseph, L.; Cacace, J. *Mastering ROS for Robotics Programming: Design, Build, and Simulate Complex Robots Using the Robot Operating System*; Packt Publishing Ltd.: Birmingham, UK, 2018.

44. Koubaa, A. *Robot Operating System (ROS): The Complete Reference (Volume 3)*; Springer: Berlin/Heidelberg, Germany, 2019.

45. Wang, Z.; Gong, L.; Chen, Q.; Li, Y.; Liu, C.; Huang, Y. Rapid developing the simulation and control systems for a multifunctional autonomous agricultural robot with ROS. In *Intelligent Robotics and Applications: Proceedings of the 9th International Conference, ICIRA 2016, Tokyo, Japan, 22–24 August 2016*; Proceedings, Part I 9; Springer: Berlin/Heidelberg, Germany, 2016; pp. 26–39.

46. Patil, V.; Singhal, S.; Kshirsagar, D.; Rathod, T.; Sakaria, Y. AgriDoc: ROS integrated agricultural robot. In Proceedings of the 6th Smart Cities Symposium (SCS 2022), Hybrid Conference, 6–8 December 2022.

47. Jensen, K.; Nielsen, S.H.; Joergensen, R.; Boegild, A.; Jacobsen, N.; Joergensen, O.; Jaeger-Hansen, C. A low cost, modular robotics tool carrier for precision agriculture research. In Proceedings of the 11th International Conference on Precision Agriculture, Indianapolies, IN, USA, 15–18 July 2012.

48. Baek, E.T.; Im, D.Y. ROS-based unmanned mobile robot platform for agriculture. *Appl. Sci.* **2022**, *12*, 4335. [CrossRef]

49. Liu, Z.; Lü, Z.; Zheng, W.; Zhang, W.; Cheng, X. Design of obstacle avoidance controller for agricultural tractor based on ROS. *Int. J. Agric. Biol. Eng.* **2019**, *12*, 58–65. [CrossRef]

50. Post, M.A.; Bianco, A.; Yan, X.T. Autonomous navigation with ROS for a mobile robot in agricultural fields. In Proceedings of the 14th International Conference on Informatics in Control, Automation and Robotics (ICINCO), Madrid, Spain, 26–28 July 2017.

51. Bordini, R.H.; Hübner, J.F.; Wooldridge, M. *Programming Multi-Agent Systems in AgentSpeak Using Jason*; John Wiley & Sons: Hoboken, NJ, USA, 2007.

52. Rincon, J.; Julian, V.; Carrascosa, C. FlaMAS: Federated learning based on a spade mas. *Appl. Sci.* **2022**, *12*, 3701. [CrossRef]

53. Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtos, N.; Carreras, M. Rosplan: Planning in the robot operating system. In Proceedings of the International Conference on Automated Planning and Scheduling, Jerusalem, Israel, 7–11 June 2015; Volume 25, pp. 333–341.

54. Dantam, N.T.; Kingston, Z.K.; Chaudhuri, S.; Kavraki, L.E. An incremental constraint-based framework for task and motion planning. *Int. J. Robot. Res.* **2018**, *37*, 1134–1151. [CrossRef]

55. Garrett, C.R.; Chitnis, R.; Holladay, R.; Kim, B.; Silver, T.; Kaelbling, L.P.; Lozano-Pérez, T. Integrated task and motion planning. *Annu. Rev. Control Robot. Auton. Syst.* **2021**, *4*, 265–293. [CrossRef]

56. Mansouri, M.; Pecora, F.; Schüller, P. Combining task and motion planning: Challenges and guidelines. *Front. Robot. AI* **2021**, *8*, 637888. [CrossRef]

57. Yu, Y.; Shan, D.; Benderius, O.; Berger, C.; Kang, Y. Formally Robust and Safe Trajectory Planning and Tracking for Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 22971–22987. [CrossRef]

58. Vaughan, R. Massively multi-robot simulation in stage. *Swarm Intell.* **2008**, *2*, 189–208. [CrossRef]

59. Platt, J.; Ricks, K. Comparative Analysis of ROS-Unity3D and ROS-Gazebo for Mobile Ground Robot Simulation. *J. Intell. Robot. Syst.* **2022**, *106*. [CrossRef]

60. Zhang, J.; Keramat, F.; Yu, X.; Hernández, D.M.; Queralta, J.P.; Westerlund, T. Distributed robotic systems in the edge-cloud continuum with ros 2: A review on novel architectures and technology readiness. In Proceedings of the 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 12–15 December 2022; pp. 1–8.