Towards Stigmergic Heterogenous Symbiotic Robot Teams

Enhancing RFID-based Stigmergic Robots for Mapless Environments

Author: Abdussalam A.Alajami

TESI DOCTORAL UPF / Year of the thesis: 2023

THESIS SUPERVISOR
Dr. Rafael Pous
Department of Information and Communication Technologies

The best way to imagine a better future for humans, is to invent it.

Acknowledgements

First and foremost, I am extremely grateful to my supervisor, Prof. Rafael Pous for his invaluable advice, continuous support, strong belief in my abilities, and patience during my Ph.D. study. His immense knowledge and plentiful experience have encouraged me in all the time of my academic research and daily life. I would also like to extend my thanks to the company Keonn-Technologies for allowing me to use some of its facilities for conducting experiments.

I'd like to acknowledge the assistance and help of all the members of the Ubi-CA Lab research group throughout my study process. Furthermore, I would like to express my gratitude to my parents. Without their tremendous understanding and encouragement over the past few years, it would be impossible for me to complete my study.

Finally, my warmest regards to all my friends and close ones to my heart especially, it is their kind help and support that have made my study and life in Spain, Catalonia a wonderful time.

Abstract

The study in this thesis presents new solutions for increasing the performance of RFID-based inventory robots.

The first novel solution is the design of an RFID-based inventory aerial robot for the problem of stock-counting in large warehouses with very high shelves, thus, reducing the risks of human injury that accompany performing such tasks, costs, and time of operation. This robot design uses a stigmergic-based navigation algorithm to enable full autonomy in mapless environments.

The second solution presented in this thesis is the development of a simulation tool that enables the robotic community to utilize RFID sensors with robots in simulation, for the goal of reducing time and costs, especially in the case of using operational-cost expensive aerial robots. The simulation tool is based on a simplified probabilistic model that considers statistical, geometrical, and some antenna-related parameters. The proposed tool is validated using various experiments in the laboratory. These validation experiments test and validate the robustness of the simulation tool with different environment layouts, the number of RFID tags in the environment, and different robot types.

The third solution presented in this thesis is a localization model that is designed for distributed heterogeneous multi-robots to extend their collaboration, for solving the problem of increasing the performance of task-oriented team robots in map-less environments, this is done by exploiting the heterogeneity feature in the team. The proposed model was tested in both laboratory environments and in simulation. The simulation experiments expose the use of this model to increase the performance of a heterogeneous team of robots performing an inventory task.

Finally, this thesis presents new innovative hybrid robot structure designs, that aim to amplify the abilities and features of an individual robot. The main proposed design adapts in hardware and software, the functionality of aerial and ground robots in one system, for the purpose of exploiting the beneficial characteristics that associate both robot types, at the same time mitigating the drawbacks of operating these robots individually.

Contents

List of Figures

List of Tables

Chapter 1

CONTEXTUALIZATION

1.1 Motivation and Contribution

The general motivation for this study was the use of robots to amplify human potential, increase productivity and accuracy, and moving from simple reasoning towards having superior cognitive abilities. The ability to design different forms of robots that can automate the manual process of human inventory in large retail stores and warehouses, with the goal of reducing the risks and labor costs of operating such tasks, is indeed a major focus and motivation of the study in this thesis. Industry 4.0 has paved the way for a world where smart factories will automate and upgrade many processes through the use of some of the latest emerging technologies. The two important technologies that are discussed are UAVs and RFID. The study is considered a continuity of several years of research done by my research team Ubica Lab, where they utilized and designed different models of Unmanned Ground Vehicles (UGVs) equipped with Radio Frequency Identification (RFID) technology to solve the problem of substituting humans performing an inventory task. The primary goal was to utilize these powerful machines to contribute to Industry 4.0, thus, exploiting robots to automatize the inventory management of a retail shop or a big warehouse, doing this accurately and continuously without getting tired or the need for changing shifts, which increases the overall performance, reduces the overall costs, and secures the quality of service (QoS), which means an accurate inventory at all time. However, as all newly explored technologies, some benefits and limitations are associated. A part of the study in this thesis discusses using robots with different abilities to tackle the problem of performing a full successful inventory of large warehouses, where products associated with RFID tags exist on low and very high shelves up-to 6-10 meters. Humans performing manual inventories at large warehouses and reading products at very high shelves are prone to injuries aside from the incremental

costs of performing risky tasks. UAVs and RFID technology, are becoming very popular in the era of Industry 4.0, especially for retail, logistics, and warehouse management. However, autonomous UAVs navigating in indoor map-less environments while performing an inventory mission is, to this day, an open issue for researchers. Therefore, the focus of this study began with leveraging the technology of Unmanned Aired Vehicles UAVs for automatizing the inventory of large retail stores and warehouses with products placed on very high shelves. Exploiting UAVs adds various and diverse benefits to the inventory process compared to UGVs. A few examples of these benefits are: the agility of these machines enables them to navigate in hard-to-reach areas for humans or UGVs, they do not depend on the characteristics of the ground surface, they can read products located at heights that are considered risky for humans and impossible to reach for UGVs and many more. However, the process of designing these aerial machines to operate in GPS-denied environments and to perform an inventory mission is costly and accompanied by high risks of possible crashes that could cause human injuries. This, however, leads to a strong motive to find a solution for reducing these risks and costs that accompanies the design and testing of such machines for such tasks. The midsection of this study discussed the design of a simulator that enables the simulation of RFID technology with robotics. The goal of this simulator is to enable the wide community of robotics to simulate environments with robots and RFID sensors before the real deployment process, which reduces a lot of time and cost and is risk-free.

1.2 Contribution

This thesis presents not just one but, multiple contributions in leveraging robots toward Industry 4.0. The first contribution is the design of an RFID-based inventory UAV that uses a custom-designed exploration navigation method to take inventory of any space without the need for any prior knowledge or the use of a pre-constructed map of the environment. The designed robot and method only rely on the existing items to be inventoried to be tagged with RFID labels.

During this thesis, the design of the autonomous UAV took several iterations to reach the goal of an autonomous independent navigation and inventory task-enabled UAV that is able to operate in indoor, unexplored spaces, and without a pre-installed map as a reference.

The second contribution that this thesis presents is the design of a simulator tool that enables the use of RFID technology with robots. The designed simulator tool not only contributes to the robotics community for research purposes but can also be used by retail companies to test inventory robots in environments where large quantities of RFID tags exist. Moreover, the thesis presents the adaptability

of the simulator tool to operate on different forms of robots, multi-robots, and with different quantities of RFID tags installed in the environment.

The third contribution that this thesis presents is the development of a multi-robot localization method for the goal of increasing the performance of a group of heterogeneous robots performing a task. The thesis presents how by using the designed method, a group of heterogeneous robots are able to increase their cognitive abilities in the environment. This method aims to exploit the heterogeneity feature of robots in a group. Various scenarios of inventory-based aerial and ground robots are shown, which collaborate together to increase the performance towards the completion of an inventory mission.

The fourth and final contribution that this thesis presents is the complete design of a self-heterogeneous hybrid aerial and ground robot. The goal of this design is to serve the hypothesis of aggregating the beneficial properties of an aerial and ground robot in one machine, for the purpose of increasing the performance of an inventory robot in future work. Various designs were presented articulating the development steps it took to reach the final structure design and functionality of the robot.

The chapters of this thesis are organized as the following, section 1.2 introduces the technologies of RFID and Robotics, which are considered the main pillars that the research on which this thesis is based, it also introduces previous work on retail robots where a joint contribution published article[1] is briefly covered. Chapter 2, illustrates the design evolution of the robot used in the experiments conducted for this research. In chapter 3, the first contribution study is introduced, which reproduces a published article[2] under the title of "**Design of a UAV for Autonomous RFID based inventories using Stigmergy**". In chapter 4, the second contribution study is introduced, which reproduces a published article and a conference paper [3,4] under the title of "**A Design Platform To Simulate RFID Systems For Robots**", which also explains the design of an open source simulation tool that is already published in the official website for ROS[5]. In chapter 5, the third contribution study is introduced, which reproduces a published

[1] Alajami, A.A.; Santa-Cruz, L.; Pous, R. A Design of an Energy-efficient Self-Heterogeneous Aerial-Ground Vehicle. 9th International Conference on Automation, Robotics and Applications (ICARA 2023), Abudhabi, 2023

[2] Alajami, Abdussalam A., Guillem Moreno, and Rafael Pous. 2022. "Design of a UAV for Autonomous RFID-Based Dynamic Inventories Using Stigmergy for Mapless Indoor Environments" Drones 6, no. 8: 208. https://doi.org/10.3390/drones6080208

[3] A. A. Alajami, G. Moreno and R. Pous, "A ROS Gazebo Plugin Design to Simulate RFID Systems," in IEEE Access, vol. 10, pp. 93921-93932, 2022, doi: 10.1109/ACCESS.2022.3204122.

[4] A. A. Alajami, R. Pous and G. Moreno, "Simulation of RFID Systems in ROS-Gazebo," 2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA), Cagliari, Italy, 2022, pp. 113-116, doi: 10.1109/RFID-TA54958.2022.9924062.

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

article under the title of "**A ROS-based Distributed Multi-robot localization and orientation Strategy for Heterogeneous Robots**" [6]. In chapter 6, the fourth contribution is introduced, which reproduces a published conference paper with some minor dents under the title of "**Unmanned Hybrid Aerial-Ground Vehicle Designs**",[7] and other hybrid robot designs are introduced. Finally, the conclusions and list of publications are introduced in chapter 7 and 8.

[6] Alajami, A.A.; Palau, N.; Lopez, S; Pous, R. A ROS-based Distributed Multi-robot localization and orientation Strategy for Heterogeneous Robots. Intelligent Service Robotics, 2023

[7] Alajami, A.A.; Santa-Cruz, L.; Pous, R. A Design of an Energy-efficient Self-Heterogeneous Aerial-Ground Vehicle. ICARA-2023 Conference, Abu Dhabi, 2023

Introduction

1.3 RFID Technology

1.3.1 Overview

Radio Frequency Identification (RFID) refers to a wireless system that is usually composed of three components, RFID tags, RFID readers, and adequate antennas. RFID belongs to a group of technologies referred to as Automatic Identification and Data Capture (AIDC). AIDC methods automatically identify objects, gather related data, and input the data directly into computer systems with almost non-human intervention. RFID methods utilize radio waves to accomplish this. An RFID tag is composed of an integrated circuit and an antenna, which are used to receive the radio wave signal from the reader, and use the received radio wave signal to transmit the data stored in the tag to the RFID reader (also called an interrogator). The reader then decodes the radio waves into a more usable form of data. Information collected from the tags is then transferred through a communications interface to a host computer system, where the data can be stored in a database and analyzed at a later time. Passive RFID tags are powered by the reader and do not have a battery. Active RFID tags are powered by batteries. In this study, we only use passive RFID tags in our experiments.

RFID tags can store a range of information from one serial number to several pages of data. Readers can be mobile so that they can be carried by hand, or they can be mounted on a post or on mobile robots as we introduce in this study. Reader systems can also be built into the architecture of a cabinet, room, or building.

RFID technology is employed in many industries to perform such tasks as:

(i) Inventory management.

(ii) Asset tracking.

(iii) Personnel tracking.

(iv) Controlling access to restricted areas.

(v) ID Badging.

(vi) Supply chain management.

Ultra high frequency (UHF-RFID) technology has substantially a lot more benefits than using barcode identification methods. The fact that this technology does not require direct visual contact to be able to extract the data from the subject. It can also subcutaneously read or identify a very long number of codes known as Electronic Product Code (EPC) due to the 96 bits code-length, whereas barcodes only encode the Stock Keeping Unit (SKU), which represents a set of unique items. An example in retail can be given, a T-shirt is usually encoded as 13 digits using bar codes, whereas all T-shirts for a given model, color, and size would share the same SKU code. The RFID protocol used in this study is GEN2 defined by GS[8]. This protocol uses Ultra High Frequency (UHF) at around 900MHz. This standard allows the use of three sessions (S0, S1, and S2) that define the protocol communication between the reader and RFID tags. The specifications of these sessions are as the following:

1. Session S0: The RFID tags in this session, will constantly keep responding to the readers' requests as long as they receive one from the reader. This session is used when the objective is to read as many tags as possible in a continuous manner.

2. Session S1: In this session, the RFID tags sleep for a few seconds to up to 1 minute after detection.

3. Session S2: Session S2 is usually used in dense environments when it comes to stock counting. It is known to have the best results in these environments. This is because it sleeps the read RFID tags, giving a higher opportunity to read hard-to-read tags by lowering the interference between RF waves.

All the RFID systems in this study are Commercial Off the Shelf (COTS) systems that are commercialized by Keonn[9]. The RFID reader used in this study is of model AdvanReader- 160[10].

1.3.2 Using RFID for Inventory Management

Using UHF-RFID technology offers several benefits for inventory management, such as:

[8]https://www.gs1.org/epc-rfid
[9]https://keonn.com
[10]https://www.keonn.com/rfid-components/readers/advanreader-160.html

1. *Improved visibility and faster scanning.* Since using RFID technology for reading an RFID tag does not require a "line-of-sight" scan like bar-codes, this makes it possible to detect and scan the RFID tags from a considerable distance. This, therefore, leads to faster inventory missions. RFID tags can also be detected and scanned in any orientation and still give improved visibility into your inventory with the potential for more frequent updates and scanning locations.

2. *Reduced labor costs.* Labor costs in taking an inventory are considered as much as 50-80% of distribution center costs, RFID offers potential benefits and reduction of labor costs. Inventory missions and product counting can be done much faster and automatically with just a small number of scans, it also does not require a huge workforce to process this data. Although the RFID labor compared to bar-code is low, the improvement of this feature will be a key goal of this study, thus replacing the human worker with an autonomous robot.

While there are some benefits of using RFID tags for inventory management, the technology also comes with several disadvantages that hinder usability and introduce other concerns such as:

1. *Security concerns*. RFID technology as any wireless communication technique tends to face security vulnerabilities. RFID tags can be scanned by a compatible wireless device and copy data from the tags. This could later be used to create a cloned tag or copy the information to another tag, a risk of particular concern in the retail industry.

2. *Demanding infrastructure needs*. Setup for these systems requires the integration of the inventory management system, network, hanging antennas, or special shelves with lifts for workers to reach the heights where products are stored which can take a significant amount of time and resources to set up. This drawback is also further focused on in this study. Having ground and aerial robots performing an inventory mission requires less or almost non-environment change or special infrastructures.

While the use of UHF-RFID technology in inventory management offers some compelling and tangible benefits, there is still room for improving RFID inventory management and mitigating its drawbacks. One of the focuses of this study is to improve autonomous inventory management. This is done by developing robots to perform inventory missions while doing that with the least human involvement possible.

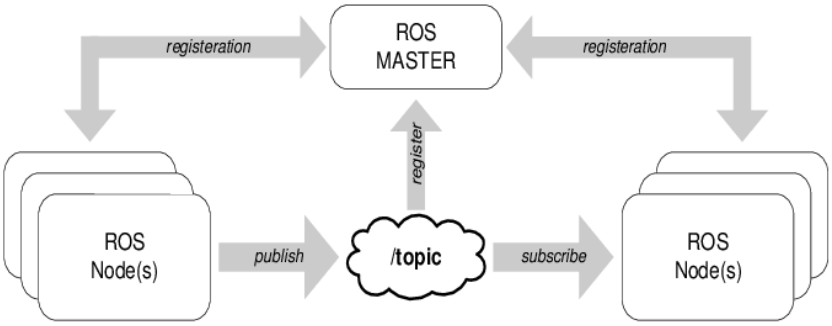
1.4 Robotics General Overview

1.4.1 ROS

The Robot Operating System known as (ROS) is not an actual operating system, but an open-source framework and set of tools that provide the functionality of an operating system on a heterogeneous computer cluster. Its usefulness is not limited to robots, as we will see in this study. ROS could be used with different technologies alone or in combination. However, the majority of tools provided are focused on working with peripheral hardware.

The key feature of ROS is that it allows the design of complex software without knowing how certain hardware works. ROS provides a communication network between different software that represents different hardware. It connects a network of processes (nodes) with a central hub as for ROS version 1. However, in ROS 2 the framework is decentralized, nodes can be run on multiple devices, and they connect to that hub in various ways.

The main ways that ROS allows these created nodes to communicate are by providing requestable services, or defining publisher/subscriber connections with other nodes as shown in Fig. 1.1a. Both methods communicate via specified message types. Some messages are provided by the core packages and others are user-defined depending on the desired purpose, but message types can be defined by individual packages.



(a) ROS nodes and topics.

As we will see in further sections, we use ROS to assemble a complex system by connecting existing solutions for small problems. The way the system is implemented allows us to:

1. Replace in real-time components with similar interfaces, removing the need of stopping the system for various changes. This helps a lot in debugging, especially when in the process of designing a new robot as in our case for designing an unmanned aerial vehicle.

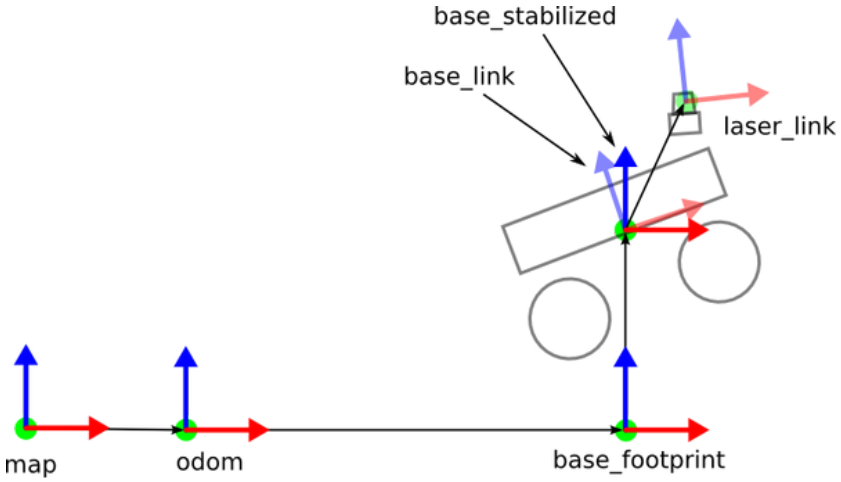
2. Connecting nodes using programming languages such as C++ and python. This is done by just implementing the appropriate messaging system. This made it easy for us to reuse pre-created nodes and connect them to ours.

3. Multiplexing outputs of multiple nodes or software into one input for another, allows parallel solving of various problems.

4. Using ROS enabled us to not just connect nodes representing components of a single robot, but it enabled the connection between all nodes of all robots that are connected over a wireless network

5. The fact that ROS runs on a LINUX operating system allowed us to use and integrate software or deep learning models that are able to run on LINUX to be integrated to the ROS framework.

6. The use of transforms and frames, in robotics, the positions and orientations of objects (e.g., robots, sensors, and obstacles), are often expressed as transformations of one coordinate frame into another. As time advances, these transformations change (e.g., if the robot moves). ROS provides an API for tracking these frames as they evolve in time using quaternion transformations basics. Quaternions are four-dimensional numbers (x, y, z, w). A rotation by an angle θ about the axis (a, b, c) is represented by the quaternion ($a \cdot \sin\frac{\theta}{2}$, $b \cdot \sin\frac{\theta}{2}$, $c \cdot \sin\frac{\theta}{2}$, $\cos\frac{\theta}{2}$). ROS allows the attachment of hundreds if not thousands of frames with each other while maintaining the relative positions and information of each frame. An illustrative example of how multiple frames and transforms are connected from a project within this study thesis is shown in Figs. 1.2a and 1.2b.



(a) An illustration example of how frames are connected for a simple ground robot with a sensor.

(b) An illustration example of multiple frames connected in ROS.

Figure 1.2: Examples of frames and transforms in ROS.

1.4.2 Simulation

Simulators aid robotics research in a multitude of ways. The benefits include a reduction in cost, better management of time, and an added level of safety when dealing with complex environments. The use of simulators in robotics research is widespread, underpinning the majority of recent advances in the field [T1].

Autonomous UGVs research – including legged, wheeled, and tracked robots are one of the largest studied domains in robotics. There are many fields that are incorporated into this sub-domain, including navigation, locomotion, cognition, control, perception, Simultaneous Localization and Mapping (SLAM), and many others. Various simulators that are specialized to simulate UGVs exist, such as Airsim[11], CARLA[12], CoppeliaSim[13], Gazebo[14], MujoCo[15], PyBullet[16], SOFA[17], UWSim[18], Chrono[19], and webots[20].

A comparison between the capabilities of the discussed robotics simulators in areas that are identified to be critical to the domain of mobile ground robots

[11] https://microsoft.github.io/AirSim/
[12] https://carla.org/
[13] https://www.coppeliarobotics.com/
[14] https://gazebosim.org/home
[15] https://mujoco.org/
[16] https://pybullet.org/wordpress/
[17] https://www.sofa-framework.org/
[18] http://www.irs.uji.es/uwsim/
[19] https://projectchrono.org/
[20] https://cyberbotics.com/

is shown in Tables 1.1 and 1.2. Critical features identified include the ability to model sensors commonly used in the field, as well as common forms of locomotion, the ability to import various environments, and inbuilt support for ROS.

Simulator	Point-clouds	Force sensor	Actuators	Multi-Body Import	Soft-Body Contacts	DEM Simulation
Airsim	✓	✗	✗	✗	✗	✗
CARLA	✓	✗	✗	✗	✗	✗
CoppeliaSim	✓	✓	Linear	✓	✗	✓
Gazebo	✓	✓	Linear	✓	✗	Fluidix[21]
MujoCo	✓	✓	✓	✓	✓	✓
PyBullet	✓	✓	Linear	✓	✓	✓
SOFA	✗	✗	✓	✓	✓	✓
UWSIM	RGBD	✓	✗	✓	✗	✗
Chrono	✓	✓	✓	✗	✓	✓
Webots	✓	✓	Linear	✓	✗	✗

Table 1.1: First comparison of different Simulation capabilities between different UGV Simulators.

Simulator	Fluid Mechanics	ROS support	HITL[22]	Teleportation	Realistic rendering	Inverse kinematics
Airsim	✗	✓	✓	✓	✓	✗
CARLA	✗	✓	✗	✓	✓	✗
CoppeliaSim	✗	✓	✓	✓	✗	✓
Gazebo	Fluidix	✓	✓	✓	✗	✓
MujoCo	Limited	✗	HAPTIX[23]	HAPTIX	✗	✗
PyBullet	✗	✗	✗	✓	✗	✓
SOFA	✓	✓	✓	✓	✓	✗
UWSIM	✗	✓	✓	✓	✓	✗
Chrono	✓	✗	✗	✓	✓	✓
Webots	Limited	✓	✗	✓	✗	✗

Table 1.2: Further comparison of different Simulation capabilities between different UGV Simulators.

[17]https://classic.gazebosim.org/tutorials?tut=fluids&cat=physics
[18]https://en.wikipedia.org/wiki/Human-in-the-loop
[19]https://doi: 10.1109/HUMANOIDS.2015.7363441

Modern UAV simulators allow researchers to replicate complex real-world environments by modeling turbulence, air density, wind shear, clouds, precipitation, and other fluid mechanics constraints. They also support various sensors such as Lidars, GPS, cameras, etc. Digital elevation models, or height maps, are also used to simulate the terrain underneath the UAV. Simulators that are able to simulate aerial robots are fewer compared to UGV simulators. The most important UAV simulators are Gazebo, AirSim, Flightmare, jMAVSim, and Webots. Tables 1.3 and 1.4 shows a comparison of important features required for aerial robotic research.

Simulator	Realistic rendering	GPS	Barometer	Sonar	Radar	PX4[24]
Airsim	✓	✓	✓	✗	✗	✓
Flightmare	✓	✗	✗	✗	✗	✗
Gazebo	✗	✓	✓	✓	✓	✓
Webots	✗	✓	✗	✓	✓	✓

Table 1.3: First comparison of different Simulation capabilities between different UAV Simulators.

Simulator	ArduPilot[25]	HITL	Point-Clouds	ROS support	VR Support
Airsim	✓	✓	✓	✓	✓
Flightmare	✗	✗	RGBD only	✓	✓
Gazebo	✗	✓	✓	✓	✓
Webots	✗	✓	✗	✓	✓

Table 1.4: Second comparison of different Simulation capabilities between different UAV Simulators.

Microsoft's AirSim, is based on Unreal-Engine[26], and supports IMU, magnetometer, GPS, barometer, and camera sensors. AirSim, provides a built-in controller called simple_flight, and also supports open-source controllers such as PX4. AirSim, is resource-intensive and hence requires large computing power

[20]https://px4.io/
[21]https://ardupilot.org/
[26]https://www.unrealengine.com/en-US

to run when compared with other simulators. It has been used in drone racing, wildlife conservation, and depth perception from visual images.

Flightmare[27] combines a flexible physics engine with the Unity rendering engine into a powerful simulator. Flightmare simulates high-fidelity environments including warehouses and forests. Sensor models are available for IMU and RGB cameras with ground-truth depth and semantic segmentation. The simulator is well suited for applications in deep/reinforcement learning.

jMAVSim[28] is another widely used simulator, mainly due to its tight coupling with the open-source PX4 controller owing to the initial goal of testing PX4 firmware and devices. jMavSim supports basic sensing and rendering.

Webots is an open-source simulator with an extensive set of supported sensors, including cameras, Lidars, GPS, etc. It allows the addition of custom physics to simulate things such as wind and integrate data from OpenStreetMap to create more realistic environments. Integration with the Ardupilot flight controller is supported. Webots have been used in multi-agent simulations, mitigation of bird strikes, and landing applications.

Gazebo is a popular simulator for both indoor and outdoor applications not only for UAVs, but also for UGVs. The ROS interface provided by Gazebo contributes to the simulator's popularity and simplifies the process of testing control software in simulation and transferring it onto the physical system. Gazebo provides the capability to import environments from digital elevation models, SDF meshes, and OpenStreetMap. It is also possible to import robot models from their Universal Robot Description Format (URDF) files. Being a rigid body simulator, the simulator runs quickly and can simulate multiple robots in real-time. Although Gazebo itself doesn't provide motion planning functionality, its tight integration with ROS allows ROS path planners to be used. Gazebo relies on the LiftDrag Plugin to simulate aerodynamic properties and supports many common sensors such as stereo-cameras and Lidar. Gazebo uses a Hector plugin, which adds UAV-specific sensors such as barometers, GPS receivers, and sonar rangers. Gazebo supports a comprehensive list of UAV models, and open-source hardware controllers such as Ardupilot and PX4 which can be integrated for hardware-in-the-loop simulations. Gazebo, however, features limited rendering capability compared to Unity and Unreal Engine. Gazebo has found applications in, e.g., autonomous navigation, landing on moving platforms, multi-UAV simulation, and visual servoing. The fine comprehension of simulating capabilities for UAVs and UGVs makes Gazebo the simulator used in all parts of the studies within this thesis.

[27] https://uzh-rpg.github.io/flightmare/

[28] https://dev.px4.io/v1.10_noredirect/en/simulation/jmavsim.html

1.5 Robot Types

Robots in general can be designed in different forms, sizes, and types. Each corresponds to the required application it is required to perform. The six most common types of robots are autonomous mobile robots (AMRs), automated guided/ground vehicles (AGVs), articulated robots, humanoids, Cobots, and hybrids. Robots are used to drive efficiency, expedite processes, improve safety, and enhance experiences across many industries. In this study, we will briefly explain the types and forms of robots that were used in the conduction of the experiments that are illustrated in this thesis.

1.5.1 Ground Robots (UGR)

Unmanned Ground Vehicles (UGV) or Unmanned Ground Robots (UGR), are referred to as robots that keep close proximity to the ground. From quadrupeds to rovers, and even humanoids, ground robots are being designed for different tasks in harsh, dangerous, and remote environments. The major task of these mobile robots is to replace humans in performing tasks with great risks. In recent years, the evolution in power efficient single board computers (SBC), also known as companion computers (CC) that carry powerful central processing units enabled the use of automation algorithms to function on ground robots. This increased the performance of these robots, therefore their applications as well. The most general form of ground conventional robots is wheeled robots. This thesis will cover in detail the ground robot used in the construction of some experiments in the following chapters. A UGV's hardware structure normally consists of 3 hardware blocks, each containing different hardware components and having a specific use. These blocks are:

1. *The main drive block.* This block consists of the hardware parts required to able the robot to move on a given surface. This block also includes the chassis or body frame that the hardware parts are attached to. The hardware components of this block are:

    (a) Wheels/motors Odometry source: motors and wheels are essential hardware for the movement of the UGV. They do not only provide the force, momentum, or torque required for the UGV body to move, but also can supply localization information to the central processing unit with relative coordinates to the map. In other words, they are used to translate the exact linear and rotary movements of the UGV to able it to sense its position in the environment. Wheel localization is done by hall sensors that exist within the structure of the wheels. These

sensors sense the number of revolutions the wheel spins with respect to time, which is used for distance calculations.

(b) Power source: for mobile UGVs, some sort of power storage system is needed to be carried on board such as lithium-ion batteries (Li-Ion), lithium polymer batteries (Li-Poly), or nickel–metal hydride batteries (NiMH). For the UGV used in these experiments, a 29v (dual 2200mAh Li-Poly 14.4v batteries connected in parallel) is used to power all 2 blocks in the UGV.

(c) Power converters: DC-DC step-down power converters or known as Buck converters[29], are mounted and used on the UGV. The main purpose of these devices is to supply adequate power to the peripherals and sensors from the battery, each according to their functional input allowed voltage.

2. *The Sensor block.* The Sensor block consists of sensor hardware that defines the UGV's capabilities. Sensors on this block allow the UGV to obtain a model of the environment. The UGV can be equipped with one to multiple types of sensors depending on the application and navigation requirements. Proximity sensors are considered one of the important type of sensors that can be used to sense obstacles in the environment or build maps. Localization sensors are another group of sensors that are essential for a UGV. For example, GPS is a localization sensor that uses satellites to provide coordinates to the UGV. Other sensors such as hall sensors in wheels, tracking cameras, or triangulation-based sensors can be used for localization. Visual sensors like cameras that are able to relay imagery information back to the user are also commonly used for controlling or influencing the navigation behavior of the UGV. The data extracted by a camera mounted on a UGV can be used as data to be processed or is processed in real-time by using models based on Artificial neural networks[30]. For the case of the specific UGV used for conducting the experiments in this thesis, the UGV is equipped with two proximity sensors, which are a 2D/3D Lidar sensor and a bumper sensor. The onboard sensors will be used to help construct a local costmap[31], for the purpose of detecting and avoiding obstacles in the environment. Furthermore, this block also contains a central processing unit usually in the form of a programmable CC. This CC will be responsible for analyzing, processing, and outputting necessary data that governs the behavior of the robot.

[29] https://en.wikipedia.org/wiki/Buck_converter
[30] https://en.wikipedia.org/wiki/Artificial_neural_network
[31] http://wiki.ros.org/costmap_2d

3. *The Payload block.* This third and final block contains the hardware components that define the application of the UGV. The capability of the UGV to carry additional hardware for application purposes is somehow limited in form, size, and motor power. For the UGV used for conducting the experiments in this thesis, the payload consists of 4 RFID antennas and a reader for RFID inventory and RFID-based navigation, which will be further explained in the next following chapters.

   The used UGV in this thesis, the so-called "Robin 50" shown in Fig. 1.7, was designed and developed by Keonn technology cooperating with the Ubica Lab team from Pompeu Fabra University.

1.5.2 Unmanned Aerial Vehicles (UAVs)

The second important type of robots are aerial robots, in other words (robots that can fly). A UAV is defined as a powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a payload. UAVs, are used more and more in many applications because of their rapid and cost-effective deployment.

UAVs, can be used in many scenarios and have formed various types. As shown in Fig. 1.3, there are many classification criteria. A typical classification criterion is to categorize UAVs into fixed-wings, rotary-wings, flapping-wings, which are based on the wing type. Another classification would be based on the weight of a UAV, miniature UAVs (between 1 & 25 kg), and heavier UAVs (more than 25 kg) [T2]. Arjomandi et al. in [T3] summarized some other classification methods: According to the flight endurance/range, they can be classified into short (less than 5 h, less than 100 km), medium (between 5 & 24 h, between 100 & 400 km), and long/range (more than 24 h, more than 1500 km). Also, they can be classified in terms of the maximum flying height of the UAV, as low altitude (less than 1 km), medium altitude (between 1 & 10 km), or high altitude UAVs (more than 10 km).

The overlap of these classification criteria will produce different effects. For example, the widely used miniature short-range quad-rotor UAVs can be used to complete aerial photography tasks. However, some UAVs in multiple categories are difficult to design and implement, such as high-altitude long-endurance UAVs designed for performing some patrol tasks, and small long-range drones for reconnaissance. One of the drawbacks of these kinds of UAVs is the contradiction between their payload and energy/power supply weight. These factors, however, will be discussed more in detail in different chapters in this thesis.

The UAV used in this study has been designed to operate in indoor areas. This

increases the complexity of the designed UAV in hardware, safety, and control. More depth of the design structure and hardware/software used for the UAV utilized to conduct experiments in the thesis will be further explained in Chapter 2.

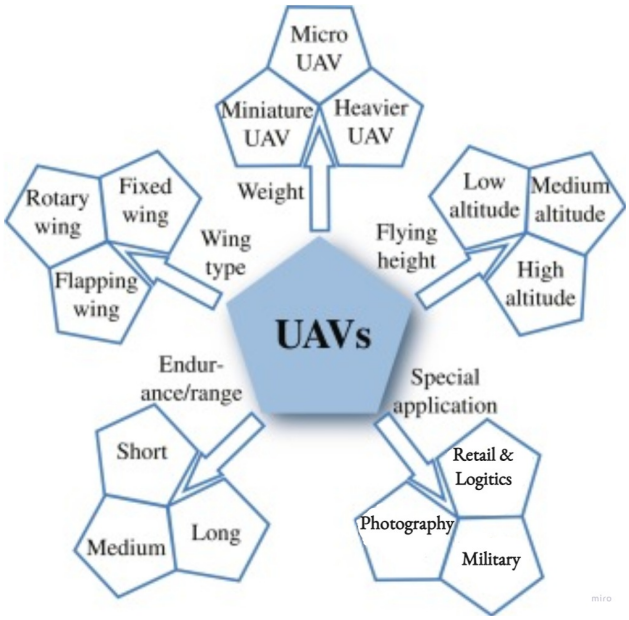


Figure 1.3: Classification of the UAV system.

1.5.3 Other Types of Robots

As mentioned before, 6 types of robots exist. Besides the already mentioned UGVs and UAVs, articulated robots, humanoids, Cobots, and hybrids are focused on by robotics researchers to exploit these diverse machines for different purposes and applications.

Articulated robots also known as robotic arms, are meant to emulate the functions of a human arm. The functionality of this robot highly depends on the degree of freedom DOF or joints the robot has. These can feature anywhere from two to 10 rotary joints. Each additional joint or axis allows for a greater degree of motion which can be ideal for various applications such as welding, material handling, machine tending, grasping, and packaging.

The other type of robots are called humanoids. While many mobile humanoid robots may technically fall under the domain of an AMR, the term is used to identify robots that perform human-oriented functions and often take human-like forms. Humanoids, use many of the same technology components as AMRs to sense their environment, plan their trajectory, and react as they carry out tasks,

such as providing customer help in hotels or airports or offering concierge services.

Cobots or so known as collaborative robots are designed to function alongside or directly with humans. While most other types of robots perform their tasks independently, or in strictly isolated work areas, Cobots, are designed to share spaces with workers in order to help them accomplish more. Cobots, are often used to eliminate manual, dangerous, or strenuous tasks from a daily based workflow. Furthermore, Cobots can operate by responding to and learning from human movements.

Finally, different type of robots, called Hybrid robots, are usually composed of a mixture of the different types of robots previously mentioned. These robots normally are designed for the purpose of sharing the features of different types of robots in one machine. An example design of this type of robot will be presented in chapter 6. The various types of robots are often combined to create hybrid solutions that are capable of more complex tasks. For example, an AMR might be combined with a robotic arm to create a robot for handling packages inside a warehouse. As more functionality is combined into single solutions, compute capabilities are also consolidated.

1.6 Previous Work on Retail Robots

There are many challenges faced by large retail chains today [T4]. Some of these challenges include frequent out-of-stock, product misplacement, organized retail crime (OCR) including theft, and lower profit margins due to stiff competition, especially in online retails. High manpower costs make it difficult to deploy more people required for efficiently managing the stores. This has prompted researchers to look for technologies that can be utilized to improve the current store management practices. The use of RFID-based smart shelves [T5], is one such example. Using mobile robots for retail monitoring is a new concept that has been exploited recently by researchers. Nowadays, inventory robots differ from each other by the different technologies used for the stock count. Some inventory robots are RFID-based, some are vision-based, and some are a combination of both technologies.

RFID-based inventory robots generally rely on a map with a preset set of waypoints. The goal is to navigate through all the areas in the given map, hence covering all possible areas that contain stocks. These types of robots aim to read all RFID labels for a full inventory of store products.

Vision-based robots, however, will navigate through the aisles of a store, looking for missing products, and empty spaces and checking the prices of the products. Both types of technologies have advantages and drawbacks.

Vision-based inventory robots have more perspective of their environment,

they can achieve more by analyzing the characteristics and content of the vision frame, however, they are considered highly complex machines that require high power consumption to operate the computationally expensive processes on their processing unit, in which to process all the data from the visual sensors or cameras. The drawbacks of using vision as a primary technology for retail make it very hard to adapt to power-limited robots such as UAVs. RFID technology, however, does not require powerful onboard computers or expensive processing units to process the data from these sensors, making it adaptable for power-limited robots such as UAVs, which opens up a much wider angle of properties and advantages in retail.

The autonomy of retail robots is another important dimension that has been always a focus and challenge for researchers and retail companies using these robots. Retail robots, based on their navigation autonomy, can be categorized as the following.

1.6.1 SLAM navigation-based Inventory Robots

Most of the reliable robots used in the market nowadays navigate based on a semi-autonomous system called Simultaneous Localization and Mapping (SLAM). The objective of SLAM is to concurrently build a map of an environment and allow the robot to localize itself within that environment. Although SLAM is not only devoted to mobile robots, it was first thought of as a tool for mobile robot autonomous navigation. Since its early beginnings, the SLAM scheme has been developed and optimized in different ways. Autonomous navigation in SLAM requires that the robot is able to decide on its own the destination within the environment being mapped. For retail robots, the mobile robot has some knowledge about its state but it cannot take any action according to that. An implementation of a semi-autonomous navigation strategy in SLAM is required to navigate in the environment.

Sensors on the mobile robot play an important role in the quality of localization and mapping in SLAM. Different manufacturers of inventory robots designed their robots with different combinations of sensors, all aiming to increase the performance of these robots by having robust localization of themselves within a map with dynamic objects.

Some examples of the leading robots in the field are:

1. Bossa Nova Robots[32]: Bossa Nova's inventory robot is a vision-based inventory robot, that is currently deployed in more than 50 stores across the United States and in Europe, making it the largest deployment of this kind of technology in any retailer. Bossa Nova creates service robots for the global

[32]https://www.bossanova.com/solutions/#retail

retail industry. The robot's mission is to make large-scale stores run efficiently by automating the collection and analysis of on-shelf inventory data by collecting terabytes of data that retailers use to increase on-shelf availability. Bossa Nova's robot navigates autonomously through aisles using preset waypoints, navigating safely among customers and store associates. Bossa Nova's robot automates the collection of on-shelf product data, extracts actionable information in real-time, and delivers prioritized tasks to team members. Retailers can also leverage this data to inform the supply chain and build mobile experiences like Walmart's Smart Assistant. The robot's navigation system relies on a 2D front view Lidar mounted on the base and 2 depth cameras in the front and rear bottom of the robot as shown in Fig. 1.4.

Figure 1.4: Bossa Nova inventory 2020 robot.

2. Stockbot[33] from PAL Robotics: Stockbot inventory robot is an RFID-based robot. Its main objective is to take a full inventory of a retail store and provide the location of the products while navigating using a semi-autonomous technique with SLAM. This robot now is running in Decathlon Singapore Labs Store, it has also been tested in MediaMarkt in the past. The navigation system of this robot relies on 2 depth cameras mounted on the top side and a 2D Lidar on the bottom as shown in Fig. 1.5.

[33]https://pal-robotics.com/robots/stockbot/

Figure 1.5: Stockbot inventory robot.

3. Robin200[34] from Keonn: This inventory robot was originally designed with significant contributions by previous members of our research group. The first commercial design was Robin200 [T6], whose first prototype was presented in 2013. Robin200 was granted the first patent for RFID autonomous robots in 2018 [T7], describing the architecture and algorithms on which the robot is based, which are also presented in this work. A more detailed description of this robot is in [T6]. The research involved in the design of this robot can be considered a starting point for the research of our research group. This thesis is considered a continuation of that first step in the milestone.

   The key feature of Robin200 is its capacity of linking SLAM-based navigation with RFID detections. The navigation strategy of Robin200 is composed of two thresholds that governed the behavior of the robot. The robot will tend to stop the navigation process when it is able to detect a number of RFID tags higher than the set threshold in a set period of time. The robot will then rotate around its verticle axis a pre-set number of times, this orienting the RFID antennas mounted onboard for maximizing the detection of RFID tags. The robot will continue on its path until it reads a higher number of tags than the threshold which will trigger the whole process again.

[34] https://keonn.com/systems-product/robin-200/

The operative procedure of this robot is divided into two phases, the mapping phase, and the inventory phase.

The mapping phase consists of manually navigating the robot in the environment in order to create a map of the area to inventory and define the path that the robot should follow during the process.

The inventory phase is the phase that is consecutive to the mapping phase. This phase relies on the mapping phase as the robot would not be able to perform an inventory mission in areas that are not mapped. In the inventory phase, the robot tries to navigate following the path created in the mapping phase, doing so autonomously and avoiding dynamic obstacles on the path. The navigation system of Robin200 relies on 2 depth cameras, both placed on the bottom of the robot facing the front and rear of the robot. The latest version of this robot is also equipped with a 3D Lidar Fig. 1.6 shows an illustration of Robin200.

Figure 1.6: Robin200 inventory robot.

Nevertheless, the current design of RFID-based inventory robots is not completely autonomous. The reason is that they require the manual creation of a map of the

store before they can operate in it. Moreover, when the layout of the environment suffers significant changes, these robots require that the map of the store is done again. In very large stores these robots suffer from scalability issues due to the size of the map. For full autonomy, while performing inventory missions, robots need to navigate autonomously in an environment without the need for human intervention. Therefore, robots would need to explore their environment and make optimum decisions on navigation toward completing a full inventory mission of a retail store.

1.6.2 Stigmergic Navigation-based Inventory Robot (Robin 50)

Towards the goal of a completely autonomous inventory-based mobile robot platform, research that was accomplished by my former research group members, which proposes a novel navigation strategy composed of a stigmergic-based algorithm for multi-robot systems, to solve the problem of autonomous stock counting in stores using RFID technology.
The proposed solution is completely autonomous since it does not require a map, is scalable to any environment size, requires less computational resources, and significantly reduces the cost since the solution can be implemented in simpler and cheaper robots. This improvement is achieved by a paradigm shift in the way that the robot completes the stock counting task. The robot, instead of navigating based on previously recorded inventory goals, uses live RFID identifications from the RFID labels present in the store as a reference, in order to complete the stock counting.
Moreover, several robots can collaborate by means of detecting RFID labels and completing the task of navigating in the same store. Hence, we can conclude that robots use environmental information as a communication channel and to store information. This behavior is called stigmergy.
This type of algorithm is also very well suited for robotics due to its intrinsic ability to simplify the robot's hardware and software and to scale to multi-robot systems.
The robot hardware design is focused to be simple and cost-effective. Its navigation system relied on having one 2D Lidar and a low-resolution depth camera, the navigation also depended on the RFID payload system. The RFID payload consists of a Keonn AdvanReader 160 [T8] RFID reader having four RF ports, each port connected to a Keonn Advantenna SP11 [T9] RFID antenna placed each in a different orientation as shown in Fig. 1.7. More insight into this autonomous model can be seen in [T10].

Figure 1.7: Robin 50 autonomous inventory robot.

1.6.3 Conclusions and lessons learned

UGVs have proven by practice to be useful in terms of automatizing inventory of retail stores or small-sized warehouses, due to many characteristics that incorporate the body form. UGVs have proven to be secure in the presence of humans, as they are programmed to operate with slow velocities. The fact that they operate in a 2D plane (on a solid surface), makes them less prone to drift caused by wind or other physical properties. Their body form, allows them to carry large energy storage batteries, high computational power processing units, and more power-hungry sensors. This gives these robots the ability to operate for a longer time, making them very good to operate in larger retail stores. However, the natural form of inventory UGVs brings also drawbacks. Large warehouses usually contain products at heights that reach up-to 10 meters or more. Due to the fact that UGVs only operate in the 2D plane, products at high heights are not reachable for detection for the inventory UGVs. Another limitation is that UGVs are not able to navigate on any given surface, the inventory mission is prone to fail if the robot mistakenly falls in a small hole, steps over thick matrices, its wheels get tangled with cables, etc. UGVs would not be able to maneuver over small obstacles or in narrow spaces. Analyzing the drawbacks that faced the designed robots from previous studies done by the research team, has oriented the course of this study to investigate the exploitation of miniature-sized UAVs for large inventory warehouses. Since UAVs suffer from weight and power constraints due to their physical characteristics, we have decided to design a UAV following the navigation paradigm of Robin 50.
The UAV designed for this thesis would be intended to operate fully autonomously in map-less environments, it is also designed to operate indoors for performing inventory missions. The next chapter 2, discusses in detail the design evolution of

the RFID-based inventory UAV during the thesis.

Chapter 2

UAV DESIGN

Throughout the research presented in this thesis, the design of an autonomous UAV for the task of autonomous inventory in mapless environments went through several stages and design versions. The UAV underwent major improvements in hardware and software as the research progressed. The goal was to design a model with stable indoor flights capabilities, able to carry a payload (around 4kg in weight and $25cm^3$ in size), able to carry an energy source that can supply all the sensors and electronics onboard with sufficient power, able to process all the data for obtaining a stable fight with accurate navigation and to run all the custom designed algorithms for detecting and avoiding obstacles alongside with the RFID related algorithms for navigation and inventory as described in further sections of this thesis. No UAV models that could fulfill these requirements existed in the market at the time of writing this thesis.

2.1 UAV V1

2.1.1 Design Structure

The initial version of the designed UAV started with adequately choosing and integrating the essential parts of a UAV that enables the robot to have a steady and stable flight with open-source features. The main hardware components of this version are shown in Table 2.1. The hardware used to design the first version of the UAV in this study will be explained using 2 parts or, as we call, blocks as shown in Fig. 2.1:

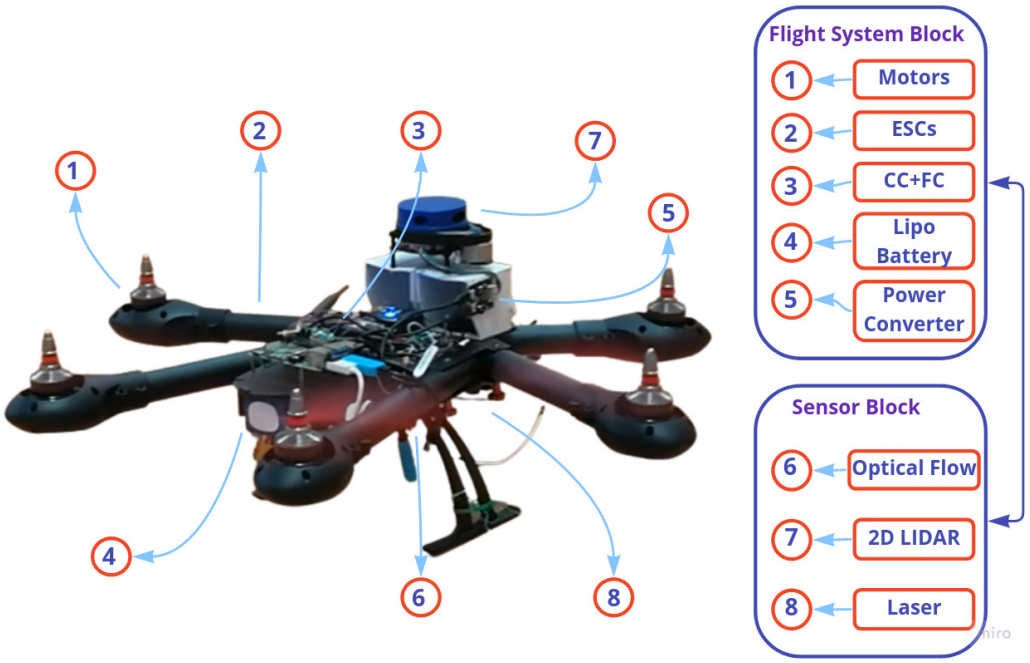


Figure 2.1: Illustration of the hardware configuration of UAV v1.

Hardware Component	Type	Quantity
Autopilot or Flight controller (FC)	Pixhawk 2.4.8	1
Electronic Speed Controllers (ESC)	60A SimonK	6
Motors	Brushless T-motors 780kv	6
Companion Computer (CC)	Asus jetson nano	1
Proximity Sensor	2D Lidar + Laser Pointer (Lidar lite v3)	2
Energy Source	4cell 6,000mAh Lipo battery	1
Position Hold Sensor	Optical Flow (PX4FLOW)	1

Table 2.1: The Hardware description of the UAV v1.

1. *The flight system block*: this block consists of the essential hardware needed for the UAV to fly or hover with some degree of stability and programmatic control.

   (a) *Flight controller (FC)*: the flight controller, or so-called autopilot, controls the flight and motion of the UAV. The UAV can rotate its body

and accelerate by creating speed differences between each of its motors. The FC uses the data gathered by the internal and external sensors to calculate the desired speed for each of the motors. The FC sends this desired speed to the Electronic Speed Controllers (ESCs), which translate this desired speed into a signal that the motors can understand.

Calculations for having precise movements, fusing and filtering sensory information, and estimating the safety and durability of a flight are all done by an algorithm installed onboard. The autopilot uses the most commonly used flight control algorithm called PID control which stands for Proportional, Integral, and Derivative control. This algorithm reads out and reacts to incoming information from the sensors at a fast rate, therefore making the drone flight more stable.

(b) *Frame and motors*: the number of motors and the design of the frame has a significant impact on the characteristics of a UAV. They have a direct impact on stability, flight time, speed, payload weight, and almost every other aspect of a UAV. In this study, corresponding to the desired payload weight needed for this UAV to carry and the importance of stability as it will fly indoors, a hexacopter (a UAV with 6 motors and propellers) will be used. The selection of kv[1] ratings of the motors was carefully considered in order to achieve a stable reliable flight.

(c) *Electronic speed controllers (ESCs)*: the electronic speed controller's main job is to follow a speed reference signal received from the FC and vary the switching rate of a network of field effect transistors (FETs). By adjusting the duty cycle, or switching frequency of the transistors, the speed of the motor is changed. The rapid switching of the current flowing through the motor is what causes the motor itself to emit its characteristic high-pitched whine, especially noticeable at lower speeds.

(d) *Power source*: there are various types of batteries (depending on their chemical structure) that can be used for UAVs. However, a lithium-ion polymer (LiPo) battery will be used for the designed robots explained in this thesis. A LiPo battery (also known as Li-poly, lithium-poly, PLiON, and other names) is a rechargeable Li-ion battery with a polymer electrolyte in the liquid electrolyte used in conventional Li-ion batteries. There are a variety of LiPo chemistries available. The reason we chose a LiPo battery in the design of the UAV used to conduct

[1] https://www.rotordronepro.com/understanding-kv-ratings/

experiments in this study is the weight versus power characteristics and trade-off. Lipo batteries can store a high capacity of energy considering their size and weight. A 4-cell (each 4.2V) 15.8V, 6,000 mAh LiPo battery is used for the designed UAV. This battery has a high discharge rate of 70c[2] ratings, which is the rate at which a battery is discharged relative to its maximum capacity (approx. 6A/h).

(e) *Power converter*: as the main power source has a high voltage output rating, DC to DC down voltage converters are necessary to be used. This is for supplying the electronic components and sensors onboard with adequate power rating input. For this UAV design, 2 DC to DC power converters were used: one to down transform 25V to 5V at 4A output capacity, which will supply the CC onboard and the other will transform 25V to 5v 2A for the sensors on board.

2. *The sensor block*: the UAV carries the necessary sensors onboard, defined by the requirements of its application. However, unlike a ground robot, the weight, computation cost, and power consumption must be considered in choosing these sensors for the UAV, as these parameters directly impact the characteristics of the UAV in terms of flight time, payload weight, and stability. These reasons among many, make designing a UAV more complex compared to UGV. For the UAV used in this study, the sensor hardware block diagram contains:

   (a) *2D Lidar Sensor*: light Detection and Ranging (Lidar), is a remote sensing method that uses light in the form of a pulsed laser to measure distance. The Lidar sensor used provides proximity sensing of the environment and is used for navigation purposes.

   (b) *Laser sensors (Lidar light v3)*: this device measures distance by calculating the time delay between the transmission of a Near-Infrared laser signal and its reception after reflecting off of a target, which then is translated into distance. Proprietary signal processing techniques are used to achieve high sensitivity, speed, and accuracy in a small, low-power, and low-cost system. This sensor could not directly be connected to the CC, therefore a low-power programmable board (ARDUINO) was needed to act as a mediator between the main CC and the sensor, the connection of the sensor. The laser has a detection range of 0-40m and an accuracy of +/- 2.5cm at distances greater than 1m.[3]

[2]http://mit.edu/evt/summary_battery_specifications.pdf

[3]https://static.garmin.com/pumac/LIDAR_Lite_v3_Operation_Manual_and_Technical_Specifications.pdf

(c) *Optical flow sensor*: the optical flow sensor used is a vision sensor capable of measuring optical flow or visual motion and outputting a motion measurement based on optical flow. Its configuration consists of an image sensor chip connected to a processor programmed to run an optical flow algorithm. It operates by using the ground texture and visible features to determine the UAV's ground velocity. The setup required this sensor to be faced downwards to the ground, plus a distance sensor which is the laser pointer previously mentioned. The sensor is connected via the I2C communication protocol directly to the autopilot.

2.1.2 Operating Mechanism of the UAV v1

The flight and navigation of the first designed version of the UAV relied mainly on the IMU sensors that existed inside the autopilot and partially on the proximity sensors that were connected to the CC. The IMU contains a gyroscope to measure and maintain the orientation and angular velocity of the UAV, it also contains an accelerometer for measuring linear velocity and finally, a barometer that measures altitude by sensing the air pressure. Although the output data of these sensors are used to control and stabilize the UAV, however, gyroscopes are more advanced than accelerometers, as they can measure the tilt and lateral orientation of the object whereas accelerometers can only measure linear motion.

The designed flight algorithm for this version relied mainly on the output of the gyroscope to constantly try to maintain the UAV at a fixed horizontal level to maintain a stable flight in the air. The algorithm tries to mitigate the drift of the UAV which is detected by measuring the linear acceleration with the accelerometer. The barometer, however, is prawn to have noisy altitude measurements due to its high sensitivity to temperature changes and to different environmental conditions. Therefore, the designed algorithm will not be fully dependent on the barometer for sensing and adjusting the height of the UAV, however, it will mainly rely on the precise laser sensor for measuring its height from the ground.

The algorithm uses the horizontal motion measurements from the optical flow sensor to try to counter any drift forces and stabilize the UAV in a fixed position while hovering.

2.1.3 Conclusions

The first version of the UAV relied on low-level sensors for flight and navigation. This led it to not require the high processing power of the data, which resulted in lower power consumption and higher flight time of up to 25min. However, this version of the designed UAV did not achieve the main goal of having a stable

and robust flight. The UAV suffered from small linear drifts that were not always detectable by the motion sensors. This was due to the fact that the ground in some cases lacked different features or characteristics which made it hard to detect the motion by the optical flow sensor. The UAV lacked a precise positioning system and a source of self-localization data or positioning system, which made it hard to precisely navigate the UAV to a fixed position.

2.2 UAV v2

2.2.1 Design Structure

The second version of the UAV design called UAV v2, is an improved version of the first model UAV v1. The hardware used to design UAV v2 is explained in 3 blocks as illustrated in Fig. 2.2 and in Table 2.2:

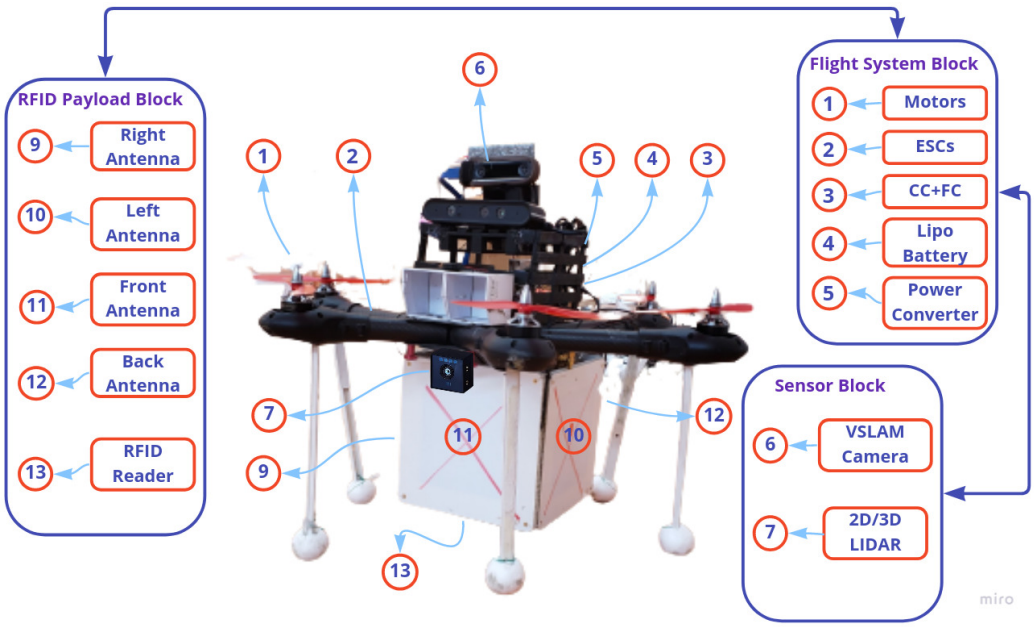


Figure 2.2: Illustration of the Hardware configuration of UAV v2.

Hardware Component	Type	Number
Autopilot or Flight controller (FC)	Pixhawk 2.4.8	1
Electronic Speed Controllers (ESC)	60A SimonK	6
Motors	Brushless T-Motors 780kv	6
Companion Computer (CC)	Asus jetson nano	1
Proximity Sensor	2D+3D Lidar	1
Energy Source	6cell 6,000mAh LIPO battery	1
Position Hold Sensor	VSLAM Camera	1

Table 2.2: The Hardware description of the UAV v2.

1. *The flight system block*: the same flight system block used in UAV v1 is used in UAV v2 of the UAV design.

2. *The Sensor block*: the UAV v2 model was designed to increase the precision of the stability of the UAV from version 1. This implies adding a sensor that is able to provide self-localization sensing in GPS-denied environments while maintaining its autonomy. In order to keep the UAV autonomous and scalable to work in any indoor environment no modification of the environment could be done to help guide the UAV or increase its stability in the air while hovering. Therefore, a high-level visual sensor (Tracking Camera) was added to achieve this goal. The sensor hardware block of the designed UAV v2 model contains:

   (i) *Visual Simultaneous Localization and Mapping VSLAM camera*: in recent years, SLAM using cameras only has been actively discussed because the sensor configuration is simple but the technical difficulties are higher than others. Since the input of such SLAM is visual information only, the technique is specifically referred to as visual SLAM (VSLAM).
   VSLAM algorithms have been proposed in the field of computer vision, robotics, and Augmented Reality (AR) [T11]. Using CMOS sensors to act as eyes to see the surrounding environment, an IMU that acts as the inner ear to sense balance and orientation, and a computer to act as the brain that fuses the information into instantaneous location and mapping. One major potential goal for VSLAM systems is to replace GPS tracking and navigation in certain applications.
   GPS systems are not usable indoors in many cases, or in big cities where the view of the sky is obstructed, and sometimes in bad weather

conditions they are low in accuracy. VSLAM systems solve each of these problems as they're not dependent on satellite information and they are taking accurate measurements of the physical world around them. For the UAV used in this study, a Realsense intel 365 VSLAM stereo tracking camera will be used, shown in Fig. 2.3a.

The algorithm in this camera will extract features from the surrounding environment and represent them as nodes for each frame as shown in Fig. 2.3b, it will therefore connect these nodes with the consecutive frames. Translating the displacements of these nodes from the consecutive frames to relative motion and pose messages is done by infusing the IMU information.

(a) The Intel RealSense Tracking Camera T265 is a complete embedded SLAM solution that uses Visual Inertial Odometry (VIO) to track its own orientation and location (6DoF) in 3D space.

(b) Example of "Harris Corner Features" detected in an image.

Figure 2.3: An illustration of INTEL tracking camera.

(ii) 2D/3D Lidar: this sensor provided the UAV with proximity sensing of the environment in the 2D plane with an 8m distance, and in the 3D plane with a 2m distance, by using light and imagery sensing.

3. The RFID payload block: the UAV will carry the same payload as the UGV of Robin 50 which was introduced in section 1.2, but on the bottom of its structure rather than on top. It is composed of a lightweight structure skeleton embedding the RFID reader, the power converter/distributor circuit board, the RFID reader, and the 4 RFID antennas. Each antenna will be facing a different direction. This block provides all the RFID-related data to other blocks when needed.

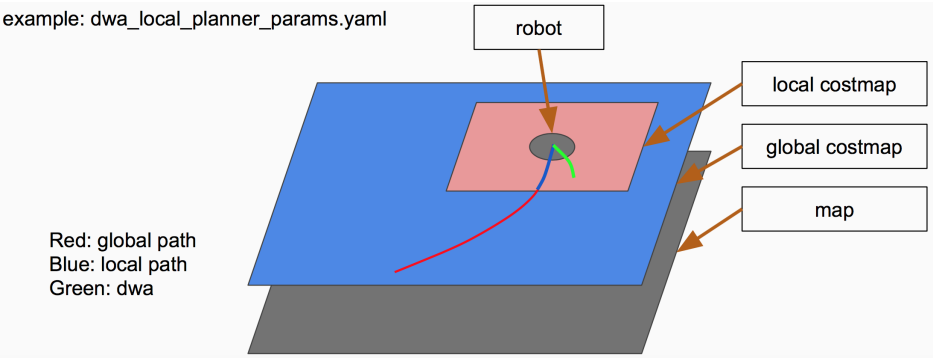
2.2.2 Operating Mechanism of UAV v2

The flight and navigation of UAV v2 use the VSLAM localization camera as an indoor positioning system. Having an indoor odometry source enabled precise navigation and a stable flight. Unlike UAV v1, the designed flight algorithm uses the output data of the VSLAM sensor for 3D localization, therefore, replacing the laser and the optical flow sensors that were used in UAV v1.

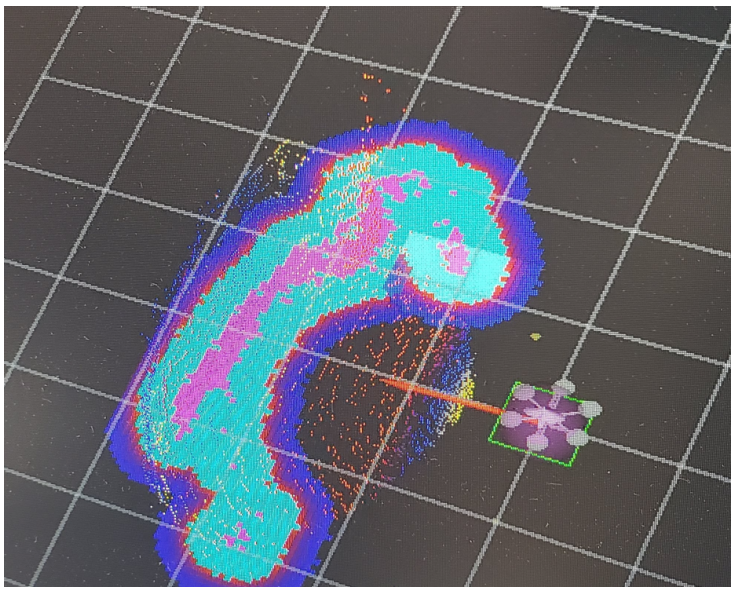
As mentioned in the hardware configuration, UAV v2 uses a 3D depth camera for proximity sensing of the environment.

The UAV utilizes the mentioned proximity and localization sensors, to construct a local costmap using a ROS-based mapping software called Move_base [T12].
The ROS Move_base node creates a cost map that is based on an occupancy grid. The created cost map, is a grid in which every cell has an assigned value (cell-cost). The cell-cost refers to the probability of that cell having an obstacle. With the use of the costmap, trajectories passing cells with the lowest cost are generated.
The Move_base node uses two cost maps, local for determining current motion and global for a trajectory with a longer range. The UAV uses a path planning algorithm that uses the Trajectory Rollout and Dynamic Window approaches (DWA) to local robot navigation on a plane. Given a plan to follow and a costmap as shown in Figs. 2.4a and 2.4b, the controller produces velocity commands to send to a mobile base. The algorithm uses a holonomic[4] based control system to move the UAV towards the path to the goal.

[4]https://www.mecharithm.com/holonomic-nonholonomic-constraints-robots/



(a) An illustration of the DWA Path planner functionality.



(b) An illustration of a constructed costmap of the UAV.

Figure 2.4: An illustration of the construction of the costmaps and paths created by the UAV.

2.2.3 Conclusions

The UAV v2 model, used 2 high-level visual sensors, the "VSLAM camera" and the "2D/3D Lidar" in order to be able to construct a costmap. However, having 2 high-level sensors demands high processing power and capacity from the CC. Supplying power to a capable CC and the sensors on board consumes a lot of power which drastically reduces the flight time.
The flight time of this version of the UAV was around 45s, which made the UAV

unfit for inventory tasks as more time is required. The flight time was not the only issue that faced this design, the computation power required by the path planning algorithm onboard required, varied corresponding to the complexity of the environment. In environments where a lot of obstacles were present, the UAV faced down throttling due to the high computation power required to process all the data and generate a path towards the goal while avoiding obstacles.
All of the conclusions above led to the third design of the UAV which will be explained in the following section.

2.3 UAV v3

2.3.1 Design Structure

The UAV v3 design utilizes the same hardware blocks as UAV v2, but with the replacement of the 2D/3D Lidar with an RGBD 3D point cloud depth camera.

The hardware used to design the third version of the UAV in this study is illustrated in Fig. 2.5 and in Table 2.3:

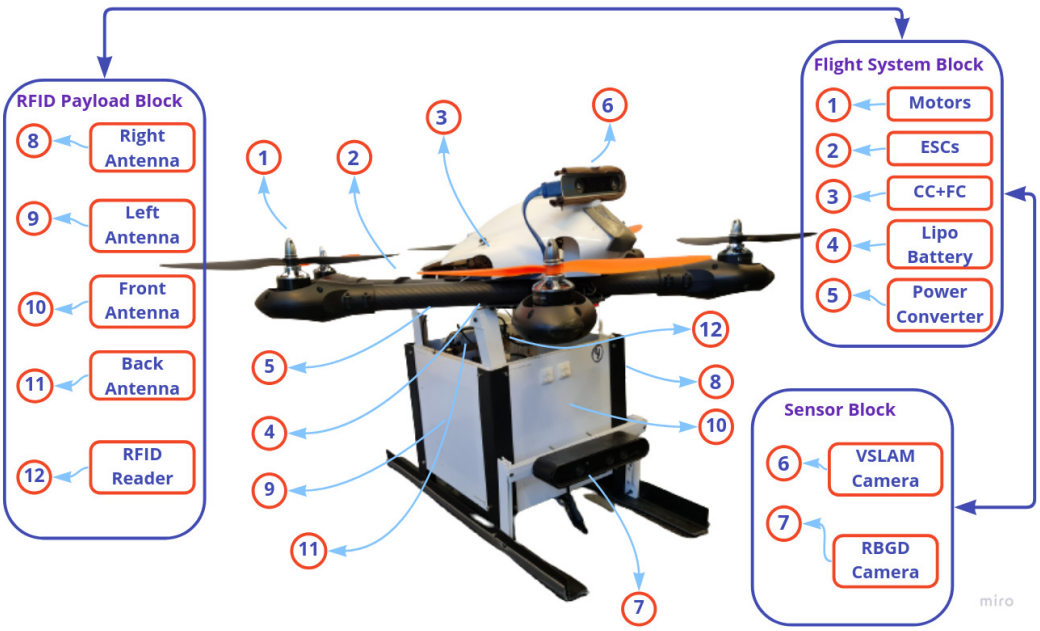


Figure 2.5: Illustration of the Hardware configuration of UAV v3.

Hardware Component	Type	Number
Autopilot or Flight controller(FC)	Pixhawk 2.4.8	1
Electronic Speed Controllers (ESC)	60A SimonK	6
Motors	Brushless T-Motors 780kv	6
Companion Computer (CC)	Asus jetson nano	1
Proximity Sensor	RGBD Camera	1
Energy Source	6cell 6,000mAh LIPO battery	1
Position Hold Sensor	VSLAM Camera Flow	1

Table 2.3: The Hardware description of the UAV v3.

2.3.2 Operating Mechanism of UAV v3

The UAV v3 model was designed to maintain a degree of automation and the stability of the UAV from the previous designs while reducing power consumption and obtaining a task-sufficient flight time. This implies reducing the high computation demand from the path planning and mapping algorithms. Therefore, a custom-designed navigation algorithm was designed for autonomous stigmergic navigation and obstacle avoidance. A detailed explanation of this algorithm will be shown in chapter 3.

2.3.3 Conclusions

This final version of the UAV design in this study was able to reach both goals, which consist of achieving precise positioning flight navigation and having task-sufficient flight time while carrying the same heavy payload in previous designs. However, this came at the cost of deducting the ability to create a costmap of the observed surrounding environment and planning more complex paths and routes around objects indoors.

Chapter 3

DESIGN OF A UAV FOR AUTONOMOUS RFID BASED INVENTORIES USING STIGMERGY

3.1 Abstract

Unmanned aerial vehicles (UAVs) and radio frequency identification (RFID) technology are becoming very popular in the era of Industry 4.0, especially for retail, logistics, and warehouse management. However, the autonomous navigation for UAVs in indoor map-less environments while performing an inventory mission is, to this day, an open issue for researchers. This article examines the method of leveraging RFID technology with UAVs for the problem of the design of a fully autonomous UAV used for inventory in indoor spaces. This work also proposes a solution for increasing the performance of the autonomous exploration of inventory zones using a UAV in unexplored warehouse spaces. The main idea is to design an indoor UAV equipped with an onboard autonomous navigation system called RFID-based stigmergic and obstacle avoidance navigation system (RFID-SOAN). RFID-SOAN is composed of a computationally low cost obstacle avoidance (OA) algorithm and a stigmergy-based path planning and navigation algorithm. It uses the same RFID tags that retailers add to their products in a warehouse for navigation purposes by using them as digital pheromones or environmental clues. Using RFID-SOAN, the UAV computes its new path and direction of movement based on an RFID density-oriented attraction function, which estimates the optimal path through sensing the density of previously unread RFID tags in various directions relative to the pose of the UAV. We present the results

of the tests of the proposed RFID-SOAN system in various scenarios. In these scenarios, we replicate different typical warehouse layouts with different tag densities, and we illustrate the performance of the RFID-SOAN by comparing it with a dead reckoning navigation technique while taking inventory. We prove by the experiments results that the proposed UAV manages to adequately estimate the amount of time it needs to read up-to 99.33% of the RFID tags on its path while exploring and navigating toward new zones of high populations of tags. We also illustrate how the UAV manages to cover only the areas where RFID tags exist, not the whole map, making it very efficient, compared to the traditional map/waypoints-based navigation.

Keywords: *robotics; stigmergy; ROS; UAV; digital pheromones; navigation; exploration; autonomous; inventory; warehouses; aerial navigation*

3.2 Introduction

Industry 4.0 has paved the way for a world where smart factories will automate and upgrade many processes through the use of some of the latest emerging technologies [A1]. Two of these important technologies are UAVs and RFID. UAVs have evolved a great deal in the last several years in terms of technology (e.g., autopilots, sensors, power efficient motors, battery capacities, and sizes) [A2, A3], and have reduced significantly in their cost. UAVs can help the automobile industry and in performing tedious tasks [A4, A5]. One of the important tasks is performing inventory missions in warehouses. Another is exploring new inventory zones in dynamic stock warehouses, which is an important factor for dynamic inventory management (DIM). DIM allows the detection of the risk of stock-outs and overstocks and therefore, respond to it by adjusting targets and sales projections [A6]. DIM helps reduce unnecessary shortages, improve sales, margins, and inventory turns. Manual inventory needs the mobility of a lot of resources, especially in big warehouses belonging to big supply chains, where shelves can easily reach heights of 10 m. This causes delays in inventory management due to the time consumption of the task. Workers are also prone to injuries when performing inventory management where products represented as RFID tags are located at great heights. Furthermore, high costs are associated with manual inventory due to the costs of the inventory workers and blocking warehouse activities during the process. Many efforts have been made for automating the inventory process. We will focus in this paper on a solution that exploits the mobility and the ability of a UAV to carry RFID sensors. This enables the UAV to be a great candidate to replace humans for the tasks of dynamic inventory in warehouses. Moreover, they represent a great economic solution for the task, as well as the diminution of the

time consumption and injury risks of the task. This paper also aims to mitigate the human intervention that associates most conventional map-based inventory robots which is needed to help create a reference map of the warehouse environment, this is achieved by increasing the autonomy and ability of the UAV to navigate in map-less indoor environments, exploring new inventory zones, and performing an inventory mission. The study in this paper addresses the design of a UAV that uses a navigation strategy based on strigmergy.

The basic idea of stigmergy is that traces left within an environment trigger an action that stimulates the performance of a future action [A7, A8]. Robots can benefit from using the concept of stigmergy. The conventional computational paradigm of robotics typically involves sensing the surrounding environment, analyzing features, building or modifying the world model/map, processing this information to find some sequence of actions that might lead to the success of a given mission, then executing that action sequence one step at a time while updating the world model/map and re-planning, if necessary, at any stage. The fit between stigmergy and behavior-based [A9] robots is excellent. It is the essence of stigmergy that the consequences of behavior affect subsequent behavior. Behavior-based robots cope well with unstructured dynamic environments and are inexpensive [A10]. UAVs, if designed as behavior-based robots, can benefit tremendously from stigmergy. In this paper, we introduce a UAV design that uses an RFID-oriented stigmergy algorithm for navigation, which provides a solution for full autonomy toward completing an inventory mission, doing so in unknown and unexplored areas. The ability of the UAV to decide and plan its consecutive path simply by following evidence and signs from the environment enables the UAV to be more autonomous throughout the whole inventory mission. The proposed RFID-SOAN only uses the existing RFID tags in a warehouse as input for navigating and path planning. This paper is structured as follows. Section 3.3 presents the state of the art. Section 3.4 is dedicated to the hardware design architecture. Section 3.5 presents the RFID-SOAN algorithm. The results of the experiments and simulations are described in Sections 3.6 and 3.7. Section 3.8 summarizes the conclusions of this work, and Section 3.9 presents the future work.

3.3 Related Work

In the past, there has been a variety of research studying the exploitation of UGVs with RFID technology for performing stock counting missions and taking a full inventory of retail shops [A11, A12]. However, the fact that these inventory robots are ground surface robots, means that they would perform poorly in big warehouses where products are placed on very high shelves. For this reason, the method of combining both technologies, RFID and UAVs, is gaining interest in

logistics, retail, and other sectors. Research in [A13, A14], shows different applications on UAVs carrying RFID systems. Recently, research has shown a rising interest in inventory task-oriented UAVs. This is due to the possibility of exploiting the 3D plane and the UAVs 6 degrees of freedom. However, this technology is still at its early stage, although some companies advertise solutions for the problem of an aerial inventory [A15, A16, A17]. A related study in [A18], discusses the application of UAVs for the purpose of product scanning in inventory and stock management using RFID in warehouses. Researchers from Samsung electronics [A19] have tried to solve the problems of robust navigation, stability, and robust aerial inventory for warehouses. In [A20], researchers from GyeongSang National University used UAVs on an inventory checking system based on RFID technology in open storage yard. These platforms either still lack the autonomy of the UAV in an indoor environment, or lack the ability to operate in unknown or dynamic environments, where the UAV navigates without a reference map of the environment.

Some research has been applied toward using stigmergy for increasing the autonomy of a robot; this is explored in [A21, A22, A23]. In Ref. [A24], the authors propose a solution for efficient supply chain management (SCM) by maintaining an accurate and close to real-time inventory of items using an unmanned ground vehicle (UGV). Their proposed algorithm uses the same RFID tags that retailers add to their products, so they can guide the robot to navigate through a complete stock counting task in the case of a dynamic inventory mission. This increases the autonomy of the UGV using stigmergy. This study is used as a starting point for our proposed navigation technique. Although inventory UGVs perform well in retail shops with uniformed ground surfaces or operating in small sized warehouses, these robots have constraints when it comes to operating in big warehouses where shelves are too high to be read from the ground or in non-uniformed ground surface warehouses. This is where we exploit the superiority of UAVs in such cases. The proposed solution in this paper addresses big warehouses and environments with tall shelves, where continuous inventory in dynamic environments is needed and no prior map is given to the UAV for navigation. Integrating the algorithm and the navigation system that is used in [A24] is too computationally expensive for UAVs, due to the limited computational capacity of these small machines. However, the design we propose can be considered an extension of the RFID stigmergic navigation paradigm, but more specialized for small-scale UAVs or behavioral-based robots.

3.4 Hardware Design and Functionality

The UAV is expected to operate in indoor (GNSS-denied) environments, where no prior information about the environment nor a map is provided to the UAV. Therefore, the design of a UAV that can self-localize indoors without the help of a reference map nor external sensors was a primary goal. The UAV is also expected to carry a relatively big and heavy payload while being able to fly for a task sufficient flight time duration. No such model that meets the intended requirements was accessible in the market. Therefore, the design of a relatively cost-effective custom UAV for such a task was needed. The UAV design, shown in Figure 3.1, is composed of three hardware blocks: B1, B2, and B3.

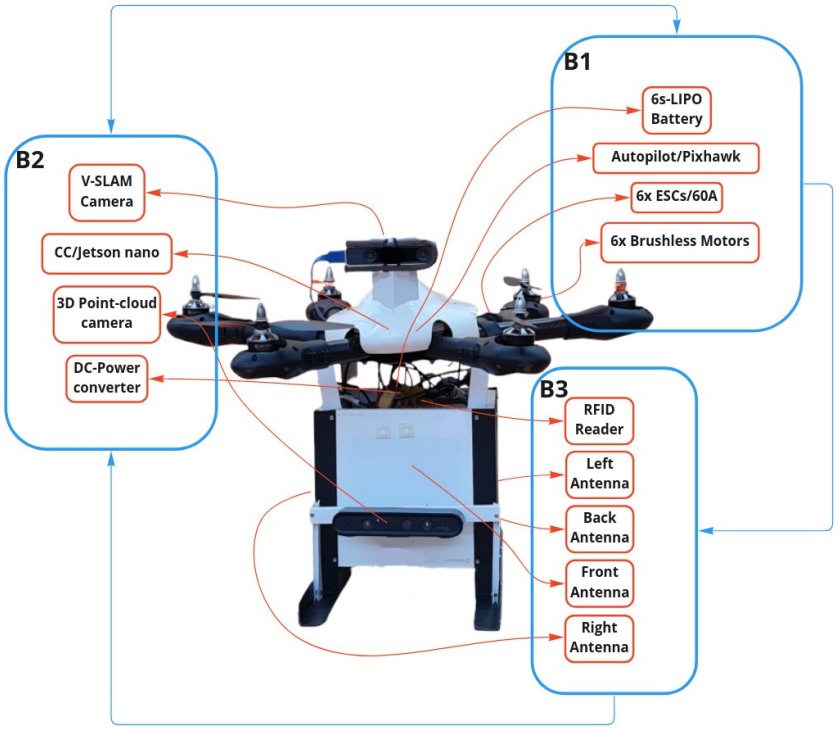


Figure 3.1: Hardware block diagram of the UAV.

3.4.1 Main Flight System, B1

The first block is the main flight system, which consists of the hardware needed to enable the UAV to fly and carry a heavy payload while maintaining a stable flight for an amount of time that is necessary to complete the inventory task. Therefore, a hexacopter frame layout with an efficient open-source autopilot (Pixhawk

2.4.8) that is compatible to operate with a companion computer was chosen for this model. B1 block is responsible for enabling the UAV to take off and hover with some degree of stability, which means that most of the flight dynamics are managed by the hardware in this block. The hardware includes an efficient high speed and high torque brush-less motors (T-motors 750 kv), large pitch propellers ($15'' \times 4.5''$), and electronic speed controllers (ESCs) with 45 A of output current capacity. The ESCs translate the received signal messages from the autopilot to sufficient energy that is supplied to each motor.

3.4.2 Sensors and Processing Units, B2

This block is responsible for adding intelligence to the contiguous flight system block. Most inventory warehouses or retail shops are indoor spaces or GPS-denied spaces. This led us to install a visual simultaneous localization and mapping (VSLAM) based camera (Realsense intel t265) to supply the autopilot with self localization messages and act as a source of the odometry. Special adaptation was made to infuse these messages into the autopilot, such as lowering the frequency rate of input pose messages to the autopilot to avoid overflow and filtering outlier input pose messages, resulting in an indoor guidance system for the UAV. This block also enables the possibility to sense the environment of obstacles nearby through a 3D-point-cloud depth proximity camera. All these sensor data, including the data received from the RFID payload block (B3), are processed by the companion computer (CC), which is a (Jetson Nano) single-board computer (SBC) that exists on this block. The CC runs the RFID-SOAN algorithm. The output of this CC is mainly the control signals in the form of the pose-goal or movement commands to the autopilot.

3.4.3 RFID-Payload, B3

The third block consists of the RFID payload. It is composed of a light-weight structure skeleton embedding the RFID-reader, a power converter/distributor circuit board (PDB), and a Keonn AdvanReader 160 [A25] RFID reader with four output ports, each port connected to a Keonn Advantenna SP11 [A26] RFID antenna placed each in a different orientation. This block provides all the RFID-related data to other blocks, as needed. Figure 3.1 shows how these blocks integrate with each other.

3.5 RFID-SOAN Workflow

The RFID-SOAN algorithm can be explained in two parts.

3.5.1 Part 1: Passive OA System:

The designed UAV is expected to detect and avoid obstacles of the 3D plane in its direction of movement. The UAV is equipped with a behavioral-based, low computation-cost, costume-designed passive OA system. Its main objective is to block any movement decisions toward the direction of an obstacle and allow movements toward the free-of-obstacles directions. An appropriate threshold value, which refers to the relative distance d of the obstacle, is set to ensure the safety of the UAV.

The input of the OA system is a stream of point cloud messages, which is received directly from the 3D point cloud depth camera. This point cloud is represented as the matrix in Equation (3.1)

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{d1} & a_{d2} & a_{d3} & \dots & a_{dn} \end{bmatrix} \tag{3.1}$$

In this matrix, the value of each element a_{ij} is equal to the depth of a point of space to the maximum depth that the sensor is able to read, d_{max}. Each point has a position represented by the coordinates i and j (relative to the position of the camera). If no point is detected at those coordinates, the value is equal to the maximum possible value. From this matrix, we transform the 3D space point cloud into a 2D plane to reduce the complexity and computational cost of processing the data, while maintaining the quality of the information extracted from those data. This is achieved by taking the minimum value of each column, resulting in a vector as shown in Equation (3.2).

$$L = \begin{bmatrix} \min\limits_i(a_{i1}) & \min\limits_i(a_{i2}) & \cdots & \min\limits_i(a_{in}) \end{bmatrix} = \begin{bmatrix} l_1 & l_2 & \cdots & l_n \end{bmatrix} \tag{3.2}$$

Then, we find the minimum value in L, as shown in Equation (3.3),

$$p = \min_i(l_i) \tag{3.3}$$

If $p < d$, the OA system considers that an obstacle is in front of the drone as shown in Figure 3.2.

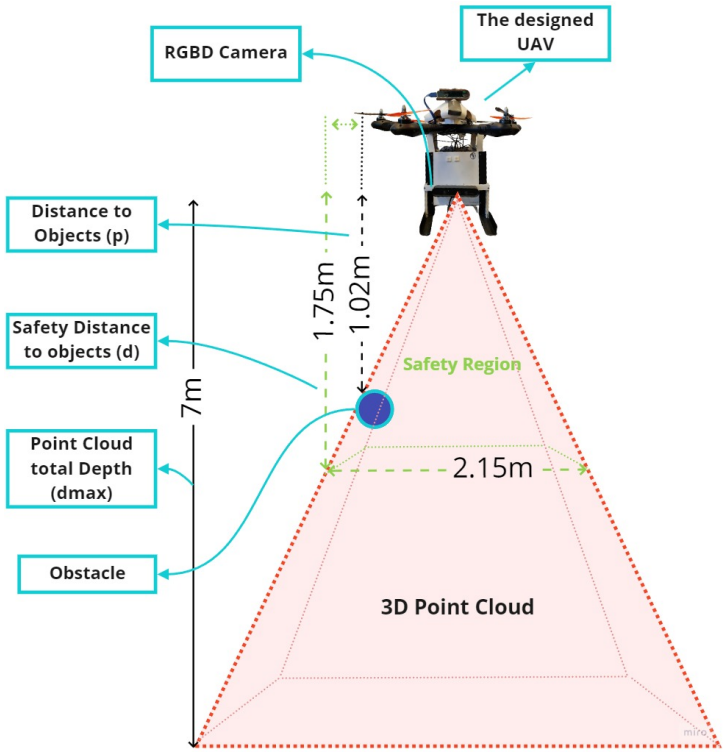


Figure 3.2: An illustration of the OA system parameters.

It is important to note that the UAV is designed to move only in the forward x-axis direction (using +Pitch angle around the y-axis) and not in sideways y-axis movements (using +-Roll angles around the x-axis), but it is able to rotate CW and CCW (using +-Yaw angles around the z-axis). This designed navigation scheme assures that the UAV covers the region ahead of it for obstacles at all times. The OA system will continuously operate as a passive system and prevent any forward movements in the presence of an object within d distance on the path to the designated goal; however, the yaw movements around the z-axis (CW or CCW rotations) are allowed to further scan for any obstacles in front of the UAV.

3.5.2 Part 2: The RFID Stigmergic Navigation Algorithm

The RFID protocol used by the reader is GEN2 session S0 [A27], where the RFID tags in this session will constantly keep responding to the readers request signals as long as they receive one from the reader. This indicates that all of the RFID tags within the coverage range of the antennas will be continuously scanned while the UAV is moving thus, increasing the chance of detection for all of the RFID tags, especially for the tags that are in close range of the antenna.

The algorithm's inputs are the 4 sets of data after every finite period of time Δt from each RFID antenna. Each set of data is composed of the sequence of EPC codes from the RFID tags detected for the first time during this time interval, from each antenna. It is worth mentioning that the RFID reader uses a technique called time domain multiplexing (TDM). TDM is a method of transmitting and receiving independent signals over a common signal path by means of synchronized switches at each end of the transmission line so that each signal appears on the line only a fraction of the time in an alternating pattern. This reduces the possibility of cross interference or reading the same RFID tag at the same exact discrete time by two different antennas.

The algorithm will compute a goal message for the UAV, which consists of a target pose message, using the direction of maximum radiation and the number of newly detected RFID tags of each antenna in the time interval, so that the UAV attempts to move to the specified position/orientation in the world following the maximum gradient of newly detected RFID tags.

After computing a goal pose message, the algorithm will request the status of the path toward that goal using the passive OA system. If no obstacle within d distance in the goal direction is detected, the CC will output command signals for the UAV to move in that direction. However, it will still detect if dynamic obstacles are in its path while moving. After reaching the goal, the same process is repeated again. The block diagram in Figure 3.3 illustrates the RFID-SOAN algorithm used by the UAV.

The new goal pose message/decision ψ_{new} is computed as follows. If $\vec{d_1}$, $\vec{d_2}$, $\vec{d_3}$, and $\vec{d_4}$ are the following 2D direction vectors shown in Equation (3.4):

$$\vec{d_1} = (1, 0)\ \vec{d_2} = (0, 1)\ \vec{d_3} = (-1, 0)\ \vec{d_4} = (0, -1) \tag{3.4}$$

and group into the direction matrix D shown in Equation (3.5):

$$D = \begin{bmatrix} \vec{d_1} \\ \vec{d_2} \\ \vec{d_3} \\ \vec{d_4} \end{bmatrix} \tag{3.5}$$

Each of these vectors corresponds to the x, y coordinates of the direction of maximum radiation of each antenna, relative to the front direction of the UAV. Note that $\vec{d_1}$ is the front direction. After a time Δt of reading RFID tags, we obtain a weights vector $\vec{w} = (w_1, w_2, w_3, w_4)$, which corresponds to the number of newly detected RFID tags read by each of the RFID antennas during this time interval, as shown in Table 3.1.

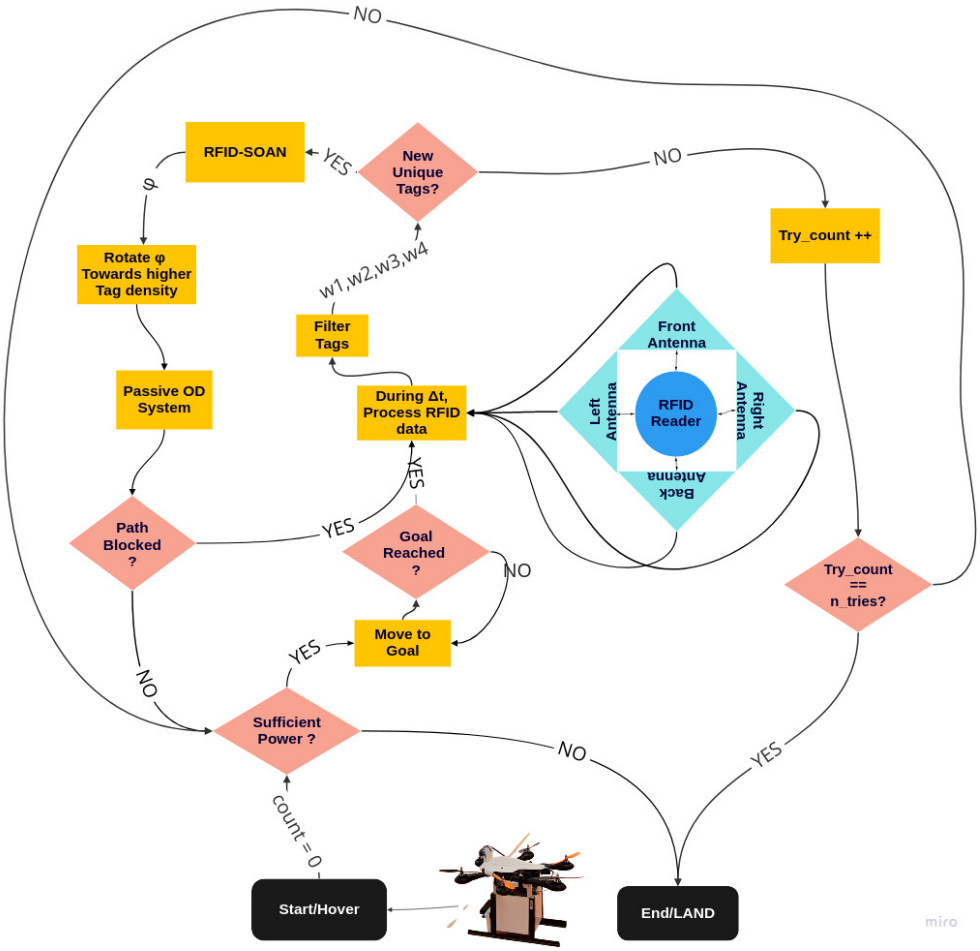


Figure 3.3: RFID-SOAN navigation algorithm workflow and block diagram.

Table 3.1: Orientation vector and weight of each RFID antenna.

Antenna Dir.	**Vector**	**Weight**
Front	$\vec{d_1}$	w_1
Right	$\vec{d_2}$	w_2
Back	$\vec{d_3}$	w_3
Left	$\vec{d_4}$	w_4

We obtain a new vector $\vec{V}$ in Equation (3.6) as the normalized weighted average of the vectors:

$$\vec{V} = \frac{\vec{w} \cdot D}{|\vec{w} \cdot D|} \tag{3.6}$$

To obtain the rotation angle θ, we compute the angle between V and the front

vector $\vec{d_1}$ shown in Equation (3.7):

$$\theta = \arccos \vec{V} \cdot \vec{d_1} \tag{3.7}$$

The new orientation of the drone ψ_{new} is the previous obtained angle plus the computed rotation angle θ shown in Equation (3.8):

$$\psi_{new} = \psi_{old} + \theta \tag{3.8}$$

In case the UAV does not read new tags in Δt, it will enter a loop of trying to read new tags in n number of tries $n_t ries$ set by the user. Once no new tags are read or no sufficient power is left, the UAV will decide to land.

3.6 Experiments

In logistics, the warehouse product flow determines the overall productivity and efficiency. The design of the warehouse layouts depends on various parameters, such as the available space, product throughput needs, and available resources. Some typical warehouse layouts considered include the U-shaped, I-shaped, and L-shaped patterns of shelves. The scenarios that we use to run our experiments use similar patterns of warehouse layouts.

In this section, we show the results of the experiments conducted to test the UAV model and its ability of it to autonomously navigate in an unknown environment toward completing an inventory mission using the RFID-SOAN system. We also compare the inventory results obtained with the results obtained using dead reckoning navigation on the same UAV, where the UAV will navigate with a pre-determined set of positions forming a route or path.

To validate the results obtained with both navigation schemes, we first measure, as a baseline, the total unique RFID tag readings with the UAV statically positioned and not moving throughout a fixed time duration. This time is equivalent to the duration needed to finish a mission while flying.

For all the experiments, the UAV is placed at a distance of 1.75 m from the shelves. We chose this distance carefully considering two important parameters: first, the necessary safety space margin required for the UAV to fly and maneuver as explained before and illustrated in Figure 3.2; secondly, at this distance, the radiation pattern of the RFID reader is at maximum, and therefore, it will cover a larger area of detection.

3.6.1 Scenario 1: One Side, One Aisle, 330 RFID Tags

In Scenario 1, we distribute 330 tags on a 15 m long shelf. Tags are placed in 11 boxes of 30 tags each. These boxes are uniformly horizontally placed throughout the shelf with a fixed altitude of almost 2 m as shown in Figure 3.4.

Figure 3.4: Lab setup of Scenario 1.

Experiment 1A: Scenario 1, UAV at a Static Position

In Experiment 1A, we want to establish a baseline that will be used to compare the effect of the two different navigation strategies (dead reckoning and stigmergy) on the inventory accuracy. For this, we first measure the total unique RFID tags in the environment with the UAV placed in a static position at the starting point for a fixed duration of time. This time should be equal to or greater than the total flight time required for the UAV to complete an inventory mission navigating through Scenario 1, making sure that no more tags can be read.

The results of the total tags read as a function of time is shown in Figure 3.5. Only 21.81% of the tags are read in this experiment.

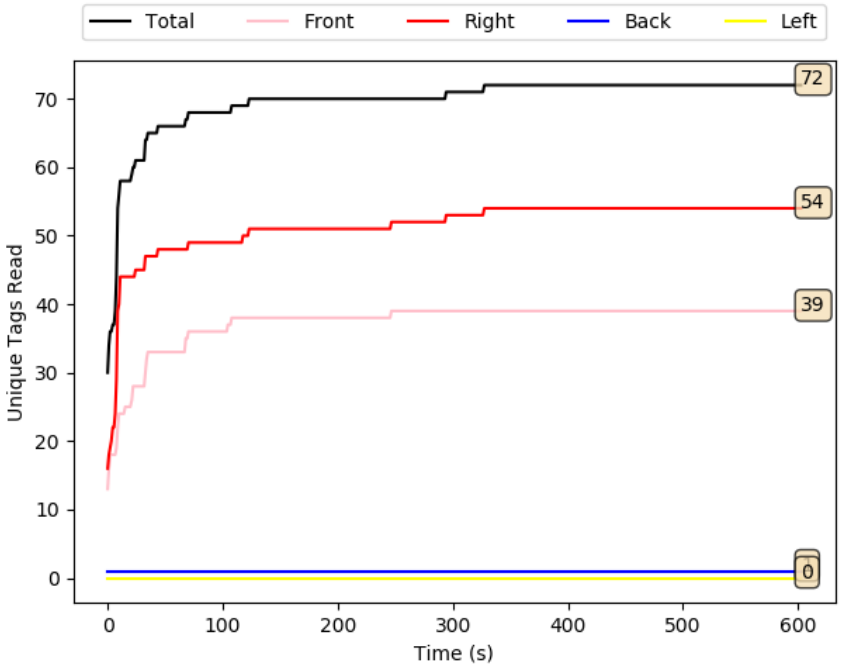
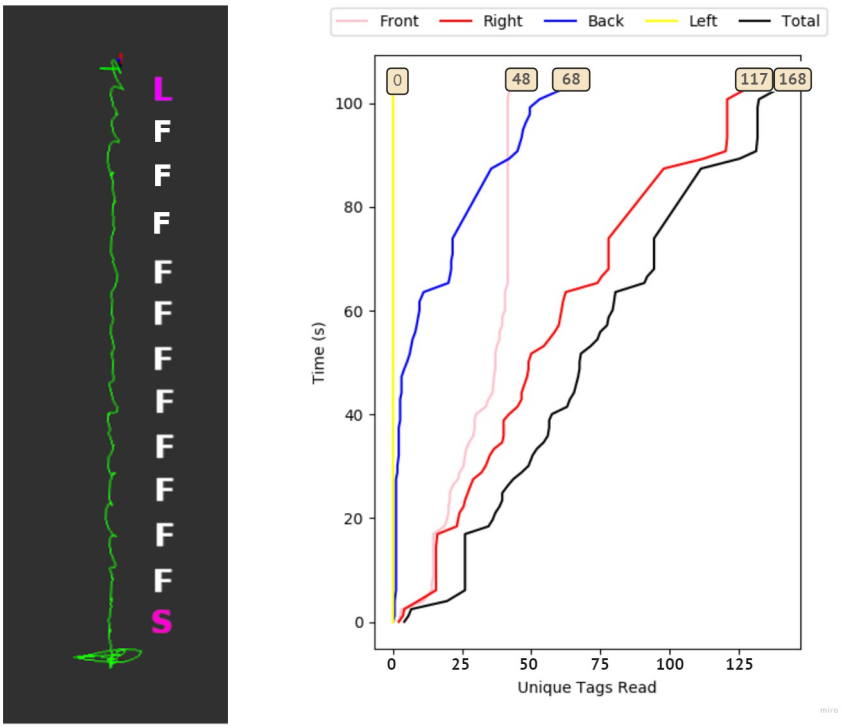


Figure 3.5: Scenario 1, Experiment 1A: total RFID tag readings vs. time from a static position equal to the starting point.

Experiment 1B: Scenario 1, UAV Using Dead Reckoning Navigation

In Experiment 1B, the UAV will navigate in Scenario 1 using dead reckoning in a fixed predefined path alongside the shelf containing tags.
This experiment represents the conventional way of pursuing an inventory mission for inventory robots, which requires creating a previous map and establishing a previous set of way-points. For conventional inventory robots, the RFID and the navigation systems operate completely independently of each other.
We show the resulting data from this experiment in Figure 3.6a,b. We can visualize the plots of RFID tag readings with time and the route of the UAV in Rviz. In this experiment, 50.90% of the tags are read, compared to the baseline value of 21.81% in Experiment 1A, showing the value of navigation, and hence the need for a UAV.

The symbols (TR, TL, F, TB, S, and L) in all the following Rviz illustrations refer to the pose positions, orientations, and actions (Turn Right, Turn Left, Forward, Turn Back, Start, and Land).



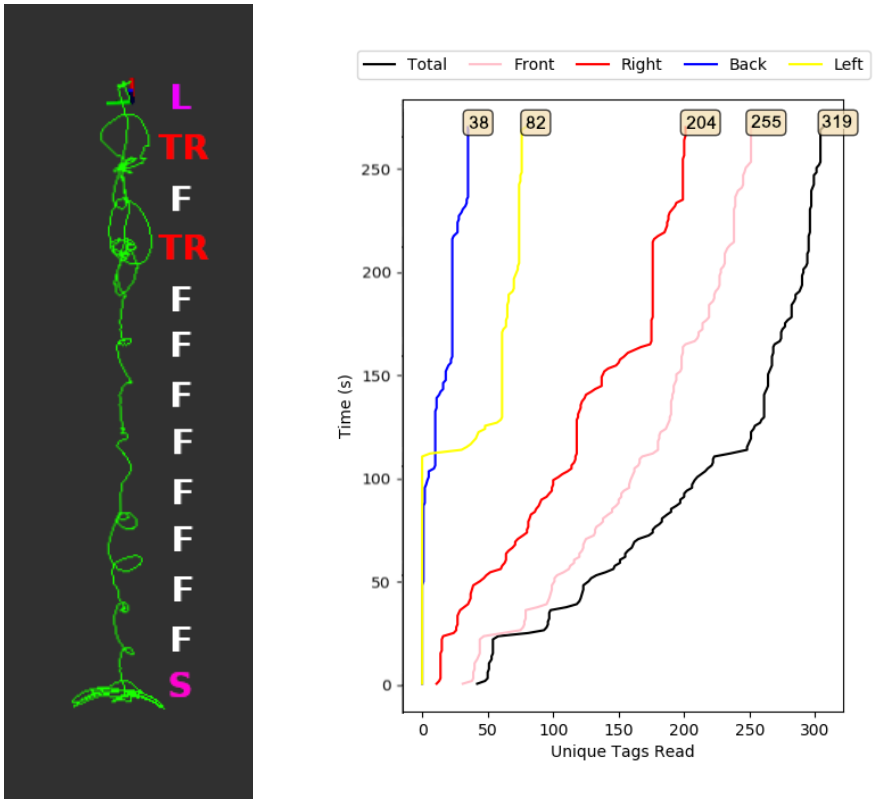
(a) (b)

Figure 3.6: Scenario 1, Experiment 1B: path shown on Rviz and total RFID tag readings vs. time when the UAV uses dead reckoning navigation. (**a**) UAV path in Rviz. (**b**) Unique RFID tags read vs. time.

Experiment 1C: Scenario 1, UAV Using RFID-SOAN Navigation

In Experiment 1C, the UAV will navigate using the proposed RFID-SOAN algorithm through Scenario 1. No prior information of the scenario, including a map or way-points, is supplied to the UAV to aid navigation.
We observe from Figure 3.7a that the UAV was successful at navigating autonomously through the path of new detected tags using the RFID-SOAN algorithm. We show the resulting tags read from this experiment in Figure 3.7b, where we can see that although the mission takes about twice as long as in Experiment 1B, 96.66% of the tags are read, compared to the 50.90% in Experiment 1B, showing the value of the stigmergy approach and the RFID-SOAN algorithm to increase the accuracy of the inventory.
The reason for this time difference is because the UAV decided to turn multiple times trying to explore and navigate toward the highest population density of tags, but was faced with an obstacle each time as shown in Figure 3.7a. The UAV re-

mains facing the direction of the obstacle if it is still able to read new tags. This ensures that all tags are read before leaving to a new target goal autonomously.



(a) (b)

Figure 3.7: Scenario 1, Experiment 1C: path shown on Rviz and total RFID tag readings vs. time when the UAV uses RFID-SOAN navigation. (**a**) UAV path in Rviz. (**b**) Unique RFID tags read vs. time.

3.6.2 Scenario 2: Two Sides, One Aisle, 330 Tags

In Scenario 2, we increase the complexity of the environment by placing fixtures on both sides of the aisle as shown in Figure 3.8a, so the UAV will have to navigate through this confined path between two fixtures simulating a close to a real warehouse environment.

(a) Top view of scenario 2.

Figure 3.8: Lab setup of Scenario 2.

Experiment 2A: Scenario 2, UAV at a static position

In Experiment 2A, the results of the total tags read as a function of time in Experiment 2A are shown in Figure 3.9. Only 33.33% of the tags are read in this experiment.

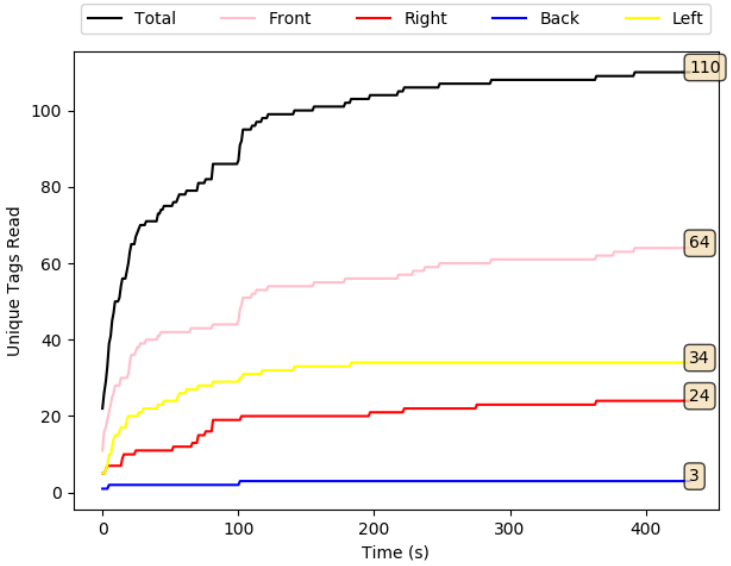
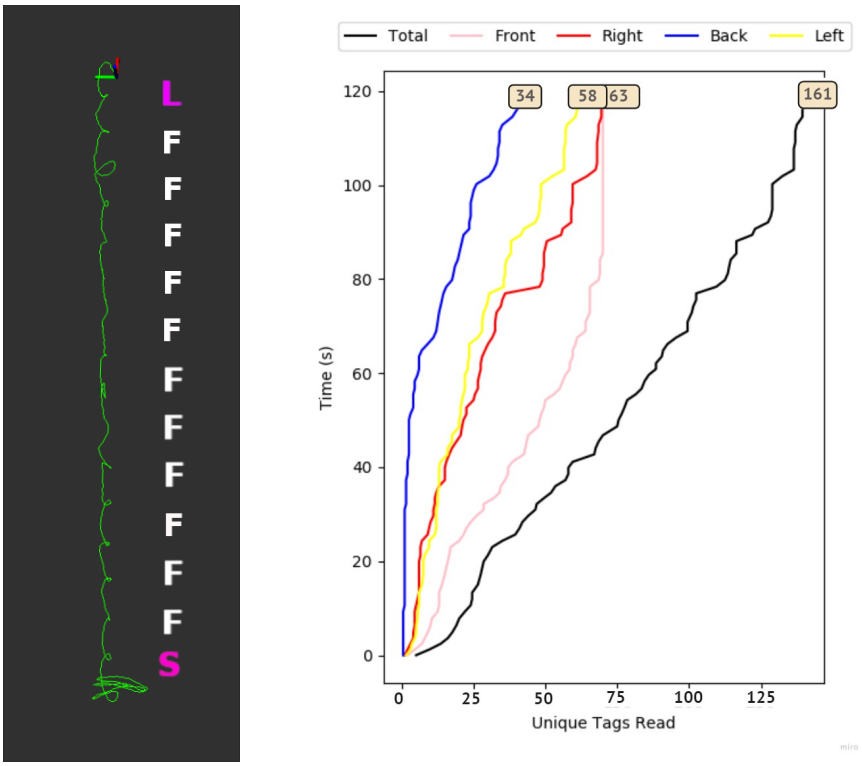


Figure 3.9: Scenario 2, Experiment 2A: total RFID tag readings vs. time from a static position equal to the starting point.

Experiment 2B: Scenario 2, UAV Using Dead Reckoning Navigation

In Experiment 2B, we show the resulting data from this experiment in Figure 3.10a,b, where we can visualize the plot of RFID tag readings with the time and route of the UAV in Rviz. In this experiment, 48.78% of the tags are read, compared to the baseline value of 33.33% in Experiment 2A.



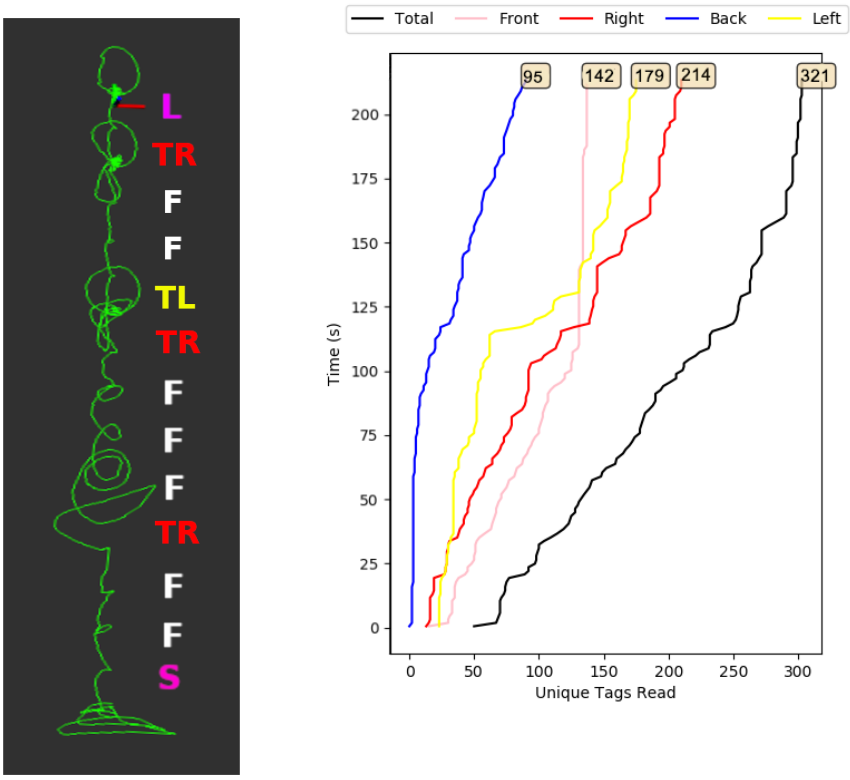
(a) (b)

Figure 3.10: Scenario 2, Experiment 2B: path shown on Rviz and total RFID tag readings vs. time when the UAV uses Dead Reckoning navigation. (**a**) UAV path in Rviz. (**b**) Unique RFID tags read vs. time.

Experiment 2C: Scenario 2, UAV Using RFID-SOAN Navigation

In Experiment 2C, we observe from Figure 3.11a, that the UAV successfully navigated and explored autonomously the path where the RFID tags were, using the RFID-SOAN algorithm without having any prior information of the environment nor human intervention. We show the RFID tag readings from this experiment in Figure 3.11b. In this case, the mission takes only 2/3 longer than in Experiment 2B. In Experiment 2C, 97.27% of the tags are read, compared to the 48.78% in Ex-

periment 2B, showing the value of the stigmergy approach and the RFID-SOAN algorithm to increase the accuracy of the inventory.



(a) (b)

Figure 3.11: Scenario 2, Experiment 2C: path shown on Rviz and total RFID tag readings vs. time when the UAV uses RFID-SOAN navigation. (**a**) UAV path in Rviz. (**b**) Unique RFID tags read vs. time.

3.6.3 Scenario 3: Two Sides, One Aisle, 660 RFID Tags

For Scenario 3, we design a layout to simulate real warehouse aisles, where a robot would have to navigate through aisles surrounding both of its sides with higher RFID tag density. For a UAV, this increases considerably the complexity of the mission, making the UAV navigate autonomously while avoiding obstacles and performing an inventory mission. The goal also is to simulate and analyze the designed UAV behavior in real-life warehouse environments. We increase the tag density to 660 tags, as we can see in Figure 3.12.

Figure 3.12: Lab setup of Scenario 3.

Experiment 3A: Scenario 3, UAV at a Static Position

In Experiment 3A, the results of the total tags read as a function of time are shown in Figure 3.13. Only 23.78% of the RFID tags are detected in this experiment.

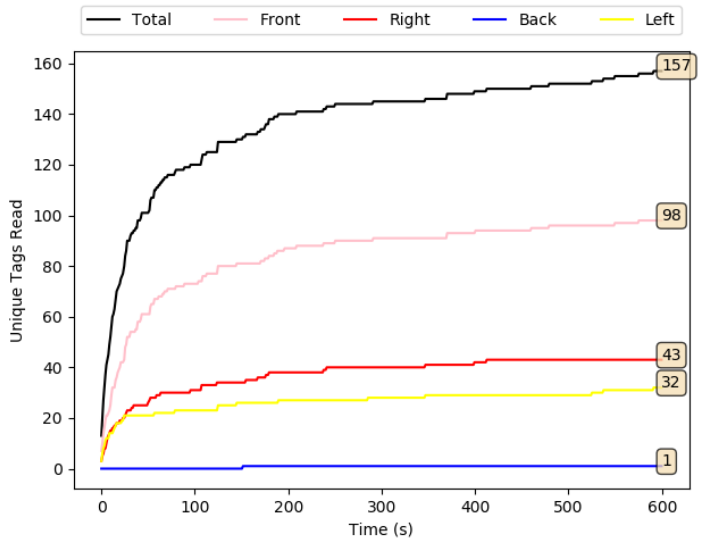


Figure 3.13: Scenario 3, Experiment 3A: total RFID tag readings vs. time from a static position equal to the starting point.

Experiment 3B: Scenario 3, UAV Using Dead Reckoning Navigation

In Experiment 3B, we show the resulting data from this experiment in Figure 3.14, where we can visualize the plot of RFID tag readings with time. In this experiment, 61.81% of the tags are detected, compared to the baseline value of 23.78% in Experiment 3A.

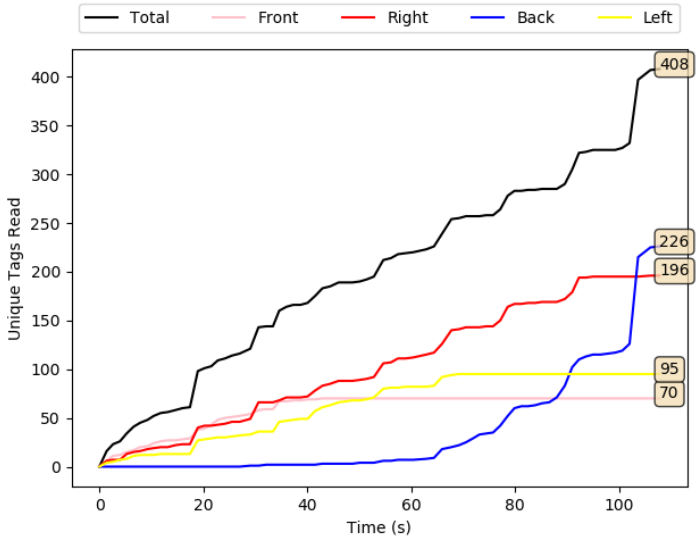


Figure 3.14: Scenario 3, Experiment 3B: total RFID tag readings vs. time when the UAV uses dead reckoning navigation.

Experiment 3C: Scenario 3 UAV Using RFID-SOAN Navigation

In Experiment 3C, we show the resulting tags read from this experiment in Figure 3.15, where we can see that in this case, the mission takes about four times as long as in Experiment 3B, which is because the UAV stayed hovering, oriented toward the highest population of tags, and not leaving until a higher density of new tags was detected in another direction and no obstacles lay on the way towards the goal targeted position, which was done autonomously. We can also see that the designed UAV managed to autonomously navigate and explore the map layout where RFID tags exist while performing an inventory mission with no need for a map/way-points nor human intervention. In Experiment 3C, 96.81% of the tags are read, compared to 61.81% in Experiment 3B, showing the value of the stigmergy approach and the RFID-SOAN algorithm to increase the accuracy of the inventory.

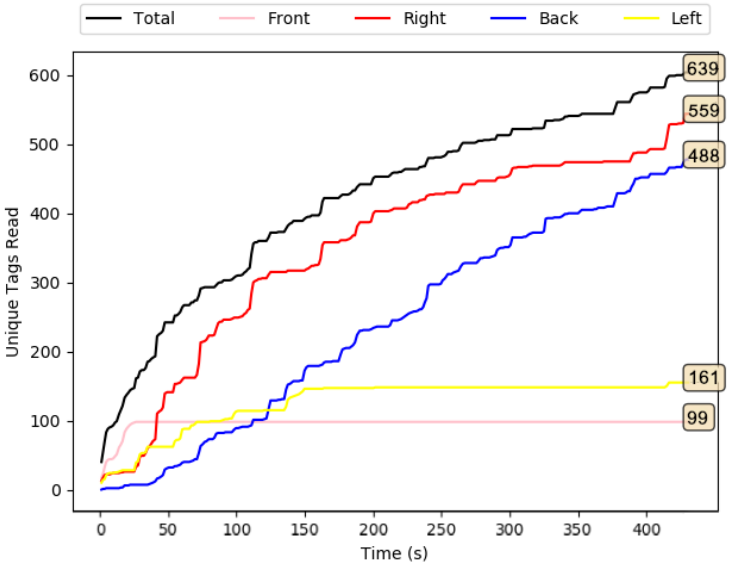


Figure 3.15: Scenario 3, Experiment 3C: total RFID tag readings vs. time when the UAV uses RFID-SOAN navigation.

3.6.4 Scenario 4: Fixtures Forming Two Aisles with a T Shape, Varying Number of Tags

For this final scenario, we try to prove that the RFID-SOAN algorithm can navigate successfully between shelves and around them, meaning that it will navigate through more complex paths, by using the RFID-SOAN algorithm. We design a T-shaped layout, thus, simulating one of the typical warehouse environments used, as we can see in Figure 3.16, where we show an image of the layout in the lab. For this layout, we decided to perform three different experiments, each with a different tag density (300, 480, 960). The tags are placed in the central aisle, and in only one of the lateral aisles. This information is not known to the RFID-SOAN algorithm, which must use the stigmergic approach to navigate and explore the path of the lateral aisle that has RFID tags placed on its shelves, while not choosing the path of the other lateral aisle with no RFID tags.

Experiment 4A: Scenario 4, RFID-SOAN Navigation, 300 RFID Tags

In Experiment 4A, 300 tags are uniformly distributed in a horizontal manner throughout the T-shaped layout. Figure 3.17a shows how the UAV chose only the lateral aisle where the tags were placed, ignoring the lateral aisle with no tags in it. Figure 3.17b, shows how the UAV was able to read 97.33% of all the tags in

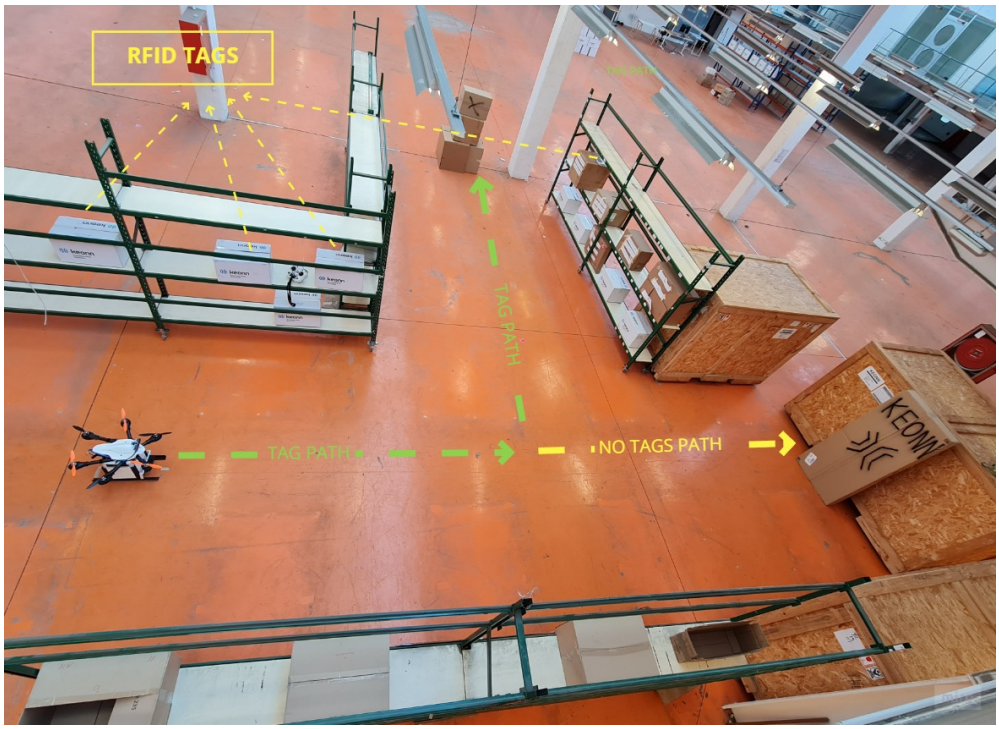


Figure 3.16: Laboratory Chapter Strigmergy of Scenario 4.

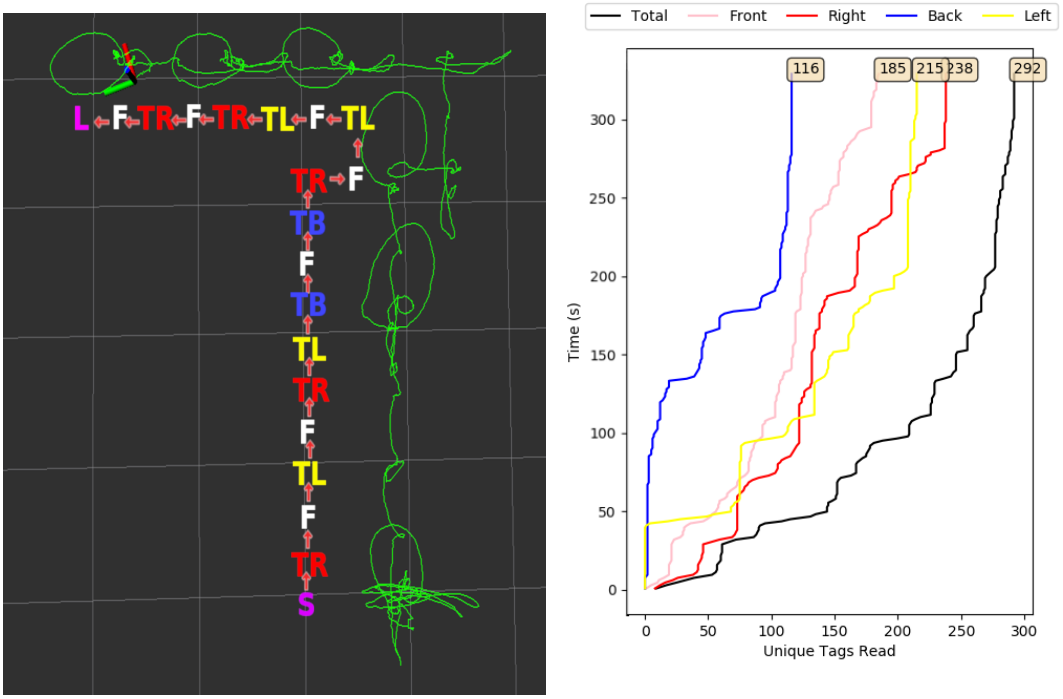
the environment in about 300 s.

Experiment 4B: Scenario 4, RFID-SOAN Navigation, 480 RFID Tags

In Experiment 4B, 480 tags are uniformly distributed in a horizontal manner throughout the T-shaped layout. Figure 3.18, shows how the UAV was able to read 97.29% of all the tags in the environment in about 400 s.

Experiment 4C: Scenario 4, RFID-SOAN Navigation, 960 RFID Tags

In Experiment 4C, 960 tags are uniformly distributed in a horizontal manner throughout the T-shaped layout. Figure 3.19, shows how the UAV was able to read 97.18% of all the tags in the environment in about 700 s.



(a) (b)

Figure 3.17: Scenario 4, Experiment 4A: path shown in Rviz and tags read vs. time, with the UAV using RFID-SOAN navigation in a T-shaped environment in which 300 RFID tags were placed. (**a**) UAV path in Rviz. (**b**) Unique RFID tag readings.

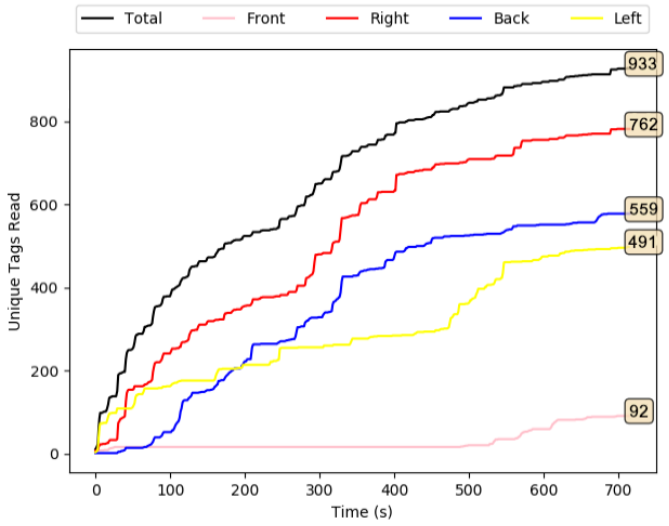


Figure 3.19: Scenario 4, Experiment 4C: tags read vs. time, with the UAV using RFID-SOAN navigation in a T-shaped environment in which 960 RFID tags were placed.

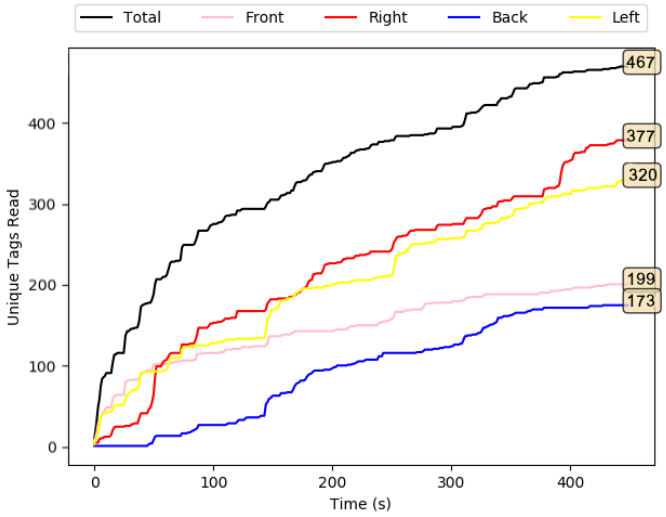


Figure 3.18: Scenario 4, Experiment 4B: tags read vs. time, with the UAV using RFID-SOAN navigation in a T-shaped environment in which 480 RFID tags were placed.

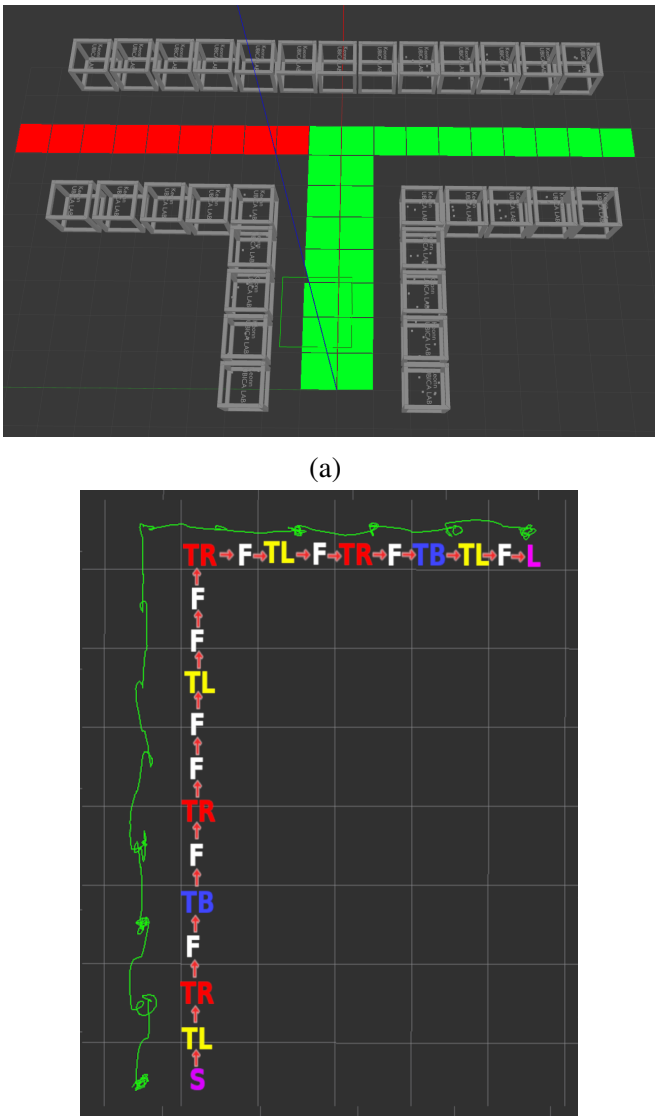
3.7 Scenario 5: Simulation

In order to test the RFID-SOAN navigation in longer and more complex map layouts, experiments may become too difficult for a particular lab space, and a particular maximum flight autonomy of the UAV. In these cases, using a simulator, such as the Gazebo [A28] ROS [A29] simulation environment, may be the best alternative. Gazebo can run the same ROS nodes that are run on the UAV, making the simulation very realistic.

3.7.1 Experiment 5A: T-Shaped Map Layout

In Experiment 5A, we use an RFID-system plugin [A30] that is composed of two sensor plugins: RFID-Tag plugin and RFID-Antenna plugin. The simulated UAV will have the same hardware architecture that we used for the UAV in the lab, and the simulated map layout will be exactly the same as the T-shaped layout that we used for the lab experiments of Scenario 4, in which 300 tags were distributed along the aisle that corresponds to the green line in the simulated layout in Gazebo, shown in Figure 3.20a. The red squares illustrate the path where no RFID tags are placed on the shelves. The green-colored squares on the grid represent the path that the UAV should take, as RFID-tags are placed and distributed along that path. Finally, a group of RFID-tags is associated with each big grey transparent box in the simulation. Figure 3.20b shows the Rviz path that the UAV took, which cor-

responds to the correct path following and considering the direction of the unique RFID tag density. Figure 3.21 shows the read tags during the whole mission. We observe that, in the simulation, the UAV read about 99.33% of the tags in about 600 s, compared to 97.33% of the tags in 400 s in Experiment 4A (Figure 3.18), which is in reasonable agreement given the complexity of the environment.



(a)

(b)

Figure 3.20: Scenario S1: T-shaped layout and path followed by the UAV. (**a**) Top view of the simulation layout. (**b**) UAV chosen path in Rviz.

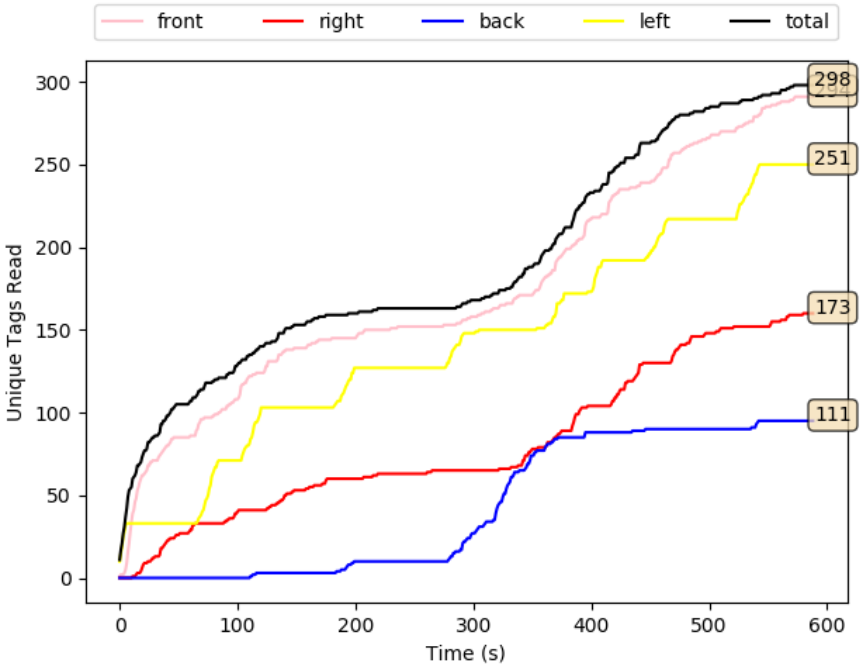


Figure 3.21: Scenario S1: tags read vs. time, with the UAV using RFID-SOAN navigation in a T-shaped environment in which 300 RFID tags were placed.

3.7.2 Experiment 5B: Square Shape Map Layout

In Experiment 5B, we test the map-less navigation and exploration feature of the RFID-SOAN algorithm and the inventory performance in a more complex environment. A square-shaped map layout is used with a much higher population density of tags placed on shelves with very high altitude (6 m) as shown in Figure 3.22, thus simulating close to real-life warehouse environments, where using a UAV can replace humans performing highly risky tasks. A total of 1700 tags are distributed along the aisle that corresponds to the green line in the simulated layout in Gazebo, shown in Figure 3.23a. Figure 3.23b shows the route that the UAV chose in Rviz. From Figure 3.24 we observe that the UAV read about 96.41% of the tags in about 1300 s, which shows the performance robustness the proposed algorithm, which was able to operate regardless of the tag density and complexity of the environment toward exploring unknown environments by using the existing RFID tags that represent products in warehouses as environmental clues.

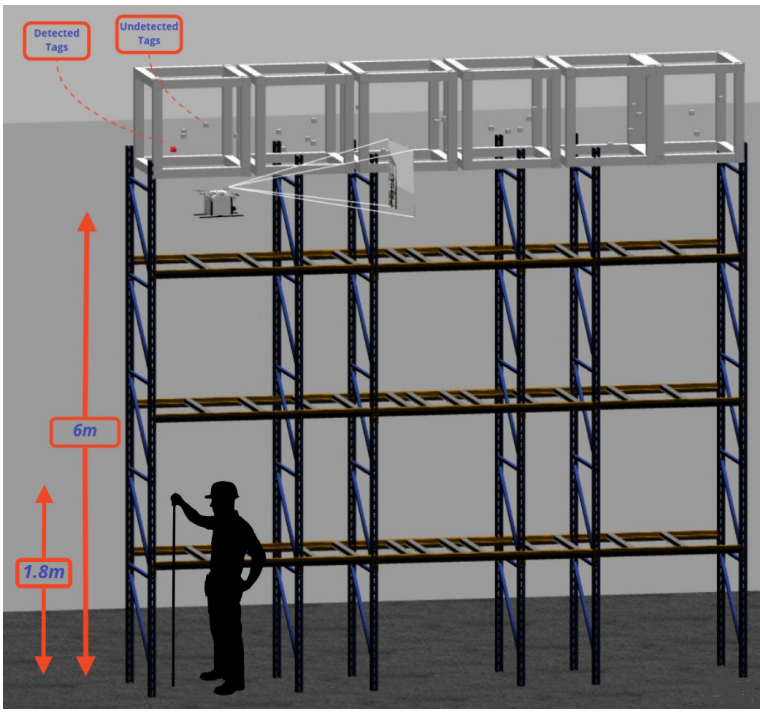
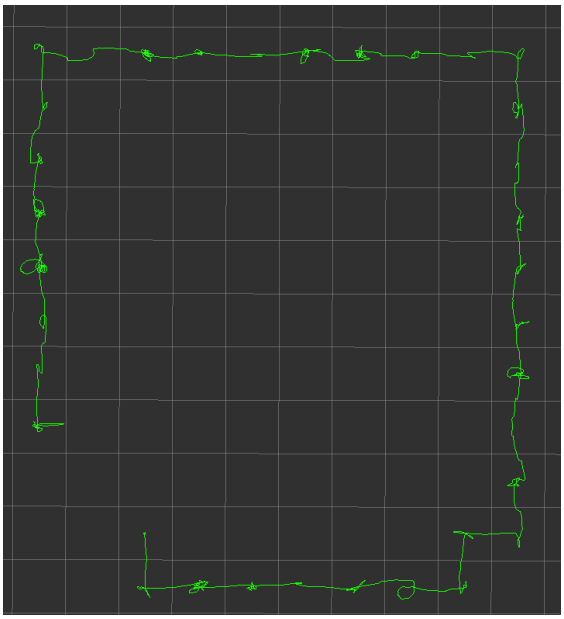


Figure 3.22: Experiment 5B: A gazebo illustration showing the height of a fixture shelve in the square shape map layout.

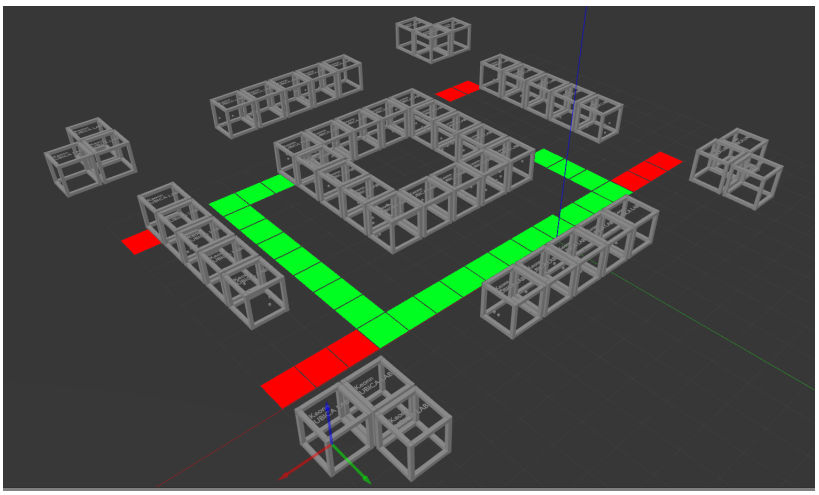
Table 3.2: Illustrates the results of all the experiments conducted with the UAV using the RFID-SOAN algorithm.

Experiment	Map-Layout	Num. Tags in Map	RFID Path Exploration	Read Tags
1C	1-side/1-isle	330	Successful	96.66%
2C	2-sides/1-isle	330	Successful	97.27%
3C	2-sides/1-isle	660	Successful	96.81%
4A	T-shape	300	Successful	97.33%
4B	T-shape	480	Successful	97.29%
4C	T-shape	960	Successful	97.18%
5A	T-shape	300	Successful	99.33%
5B	Square-shape	1700	Successful	96.41%

From Table 3.2, we observe that the designed UAV using the RFID-SOAN algorithm was successful in exploring the correct path where RFID tags exist and not navigating in paths where no RFID tags exist in all of the presented scenarios, doing so without using a reference map or having prior knowledge of the environment.



(a)



(b)

Figure 3.23: Experiment 5A: T-shaped layout and path followed by the UAV. (**a**) UAV chosen path in Rviz. (**b**) Side view of the simulation layout.

3.8 Conclusions

A solution design of a UAV performing dynamic inventory and warehouse exploration in map-less environments autonomously for Industry 4.0 was proposed in this paper. The future goal is to substitute humans performing risky inventory in

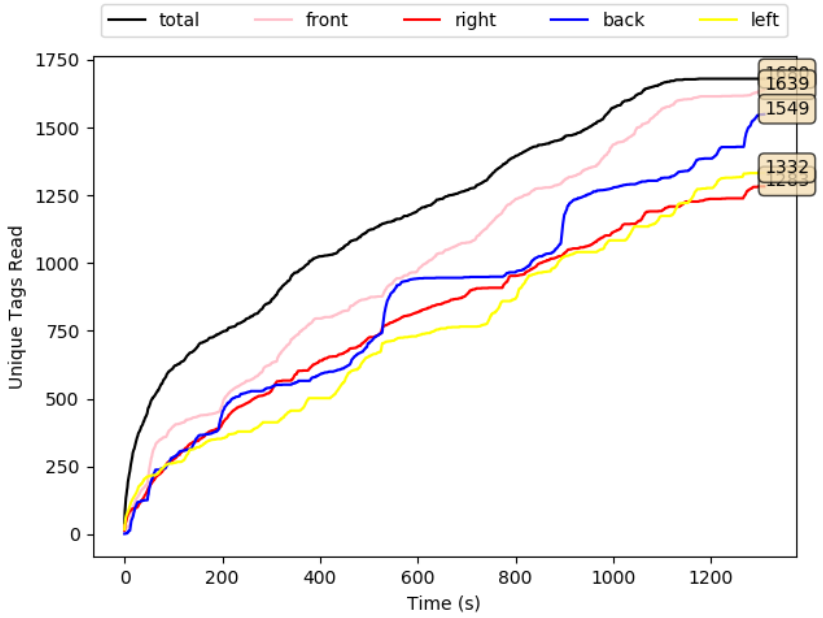


Figure 3.24: Experiment 5B: tags read vs. time, with the UAV using RFID-SOAN navigation in a square-shaped environment in which 1700 RFID tags were placed.

warehouses containing high shelves with an autonomous UAV. The system proposed can be very useful for exploring RFID-zones in un-mapped or dynamic storage type warehouses, where products are represented by RFID tags, doing so with almost no human intervention, which substantially reduces the operation cost.

Compared to dead reckoning navigation, RFID-SOAN has the distinctive advantage that it does not require any prior knowledge of the environment, nor the need for a map, or establishing a set of predefined way-points.

When left in an environment in which RFID tags are present, the UAV using the RFID-SOAN algorithm will navigate exploring the environment toward the direction in which more new tags are read, covering automatically the parts of the environment in which RFID tags are present. If the distribution of tags changes, the UAV will follow a different path, while dead reckoning would require redefining the way-points.

When compared to a baseline in which the UAV was static, and to dead reckoning navigation, RFID-SOAN consistently showed better inventory accuracy in a reasonable time, reaching an accuracy above 96.66% in all conducted experiments.

Compared to dead reckoning, RFID-SOAN will not waste scarce flight time exploring areas devoid of RFID tags.

The simplicity of use, the ability to inventory un-mapped environments, the adaptability with respect to different and changing layouts, and the accuracy in the core inventory task make the RFID-SOAN algorithm a much better navigation alternative than dead reckoning for autonomous inventory UAVs.

The proposed UAV using RFID-SOAN only relies on its own sensors, which means that it does not need modification of the environment or the placement of external sensors for it to navigate autonomously. The sensors used are very cost effective, making the UAV model relatively cheap compared to other proposed models (e.g. in [A31]).

3.9 Future Work

i *Extending the RFID-SOAN algorithm for 3D Navigation.* Although the proposed algorithm enables the UAV to read above 96.66% of RFID tags in the scenarios presented, the tags in the ground truth were placed horizontally within a fixed height. This makes it easy for the UAV to read most tags by flying at a fixed altitude. The RFID-SOAN algorithm should be extended to three dimensions, enabling the UAV to inventory tags at different heights. Without this, inventories with tags at different heights must be approached as consecutive 2D inventories at increasing heights. This may require increasing the number of RFID antennas, with the consequent increase in cost and/or decrease in autonomy.

ii *Flight time* is considered a major parameter for UAVs, due to the limited size of the power source that they can carry. In order to increase this parameter for the designed UAV, lighter material for antennas and more power efficient RFID readers can be considered.

iii *Robust indoor positioning*. We are currently working on making the designed UAV more robust in indoor navigation, using extended sensor fusion to further assure accurate and stable obstacle avoidance while executing an inventory mission.

Chapter-3 References

[A1] Tiago M. Fernández-Caramés, Oscar Blanco-Novoa, Manuel Suárez-Albela, and Paula Fraga-Lamas. A uav and blockchain-based system for industry 4.0 inventory and traceability applications. *Proceedings*, 4(1), 2019.

[A2] Eric Sholes. Evolution of a uav autonomy classification taxonomy. In *2007 IEEE Aerospace Conference*, pages 1–16. IEEE, 2007.

[A3] Abdussalam AA Alajmi, Alexandru Vulpe, and Octavian Fratu. Uavs for wi-fi receiver mapping and packet sniffing with antenna radiation pattern diversity. *Wireless Personal Communications*, 92(1):297–313, 2017.

[A4] Audi Media Centre audi uses drones to locate vehicles at neckarsulm site. `https://www.audi-mediacenter.com/en/photos/detail/audi-uses-drones-to-locate-vehicles-at-neckarsulm\\-site-92519`. Accessed: 2022.

[A5] Mark C Tatum and Junshan Liu. Unmanned aerial vehicles in the construction industry. In *Proceedings of the Unmanned Aircraft System Applications in Construction, Creative Construction Conference, Primosten, Croatia*, pages 19–22, 2017.

[A6] Li Chen and Erica L Plambeck. Dynamic inventory management with learning about the demand distribution and substitution probability. *Manufacturing & Service Operations Management*, 10(2):236–256, 2008.

[A7] Francis Heylighen. Stigmergy as a universal coordination mechanism: components, varieties and applications. *Human Stigmergy: Theoretical Developments and New Applications; Springer: New York, NY, USA*, 2015.

[A8] Zachary Mason. Programming with stigmergy: using swarms for construction. In *ICAL 2003: Proceedings of the eighth international conference on Artificial life*, pages 371–374, 2003.

[A9] Istvan Karsai. Decentralized control of construction behavior in paper wasps: an overview of the stigmergy approach. *Artificial Life*, 5(2):117–136, 1999.

[A10] Alex M. Andrew. Behavior-based robotics by ronald c. arkin, with a foreword by michael arbib, intelligent robots and autonomous agents series, mit press, cambridge, mass., 1998, xiv 491 pp, isbn 0-262-01165-4 (£39.95; hbk). *Robotica*, 17(2):229–235, 1999.

[A11] Marc Morenza-Cinos, Victor Casamayor-Pujol, Jordi Soler-Busquets, Jos Luis Sanz, Roberto Guzm, and Rafael Pous. *Development of an RFID Inventory Robot (AdvanRobot)*, pages 387–417. Springer International Publishing, Cham, 2017.

[A12] Isaac Ehrenberg, Christian Floerkemeier, and Sanjay Sarma. Inventory management with an rfid-equipped mobile robot. In *2007 IEEE International Conference on Automation Science and Engineering*, pages 1020–1026, 2007.

[A13] G Greco, C Lucianaz, S Bertoldo, and M Allegretti. A solution for monitoring operations in harsh environment: A rfid reader for small uav. In *2015 international conference on electromagnetics in advanced applications (ICEAA)*, pages 859–862. IEEE, 2015.

[A14] Jian Zhang, Xiangyu Wang, Zhitao Yu, Yibo Lyu, Shiwen Mao, Senthilkumar CG Periaswamy, Justin Patton, and Xuyu Wang. Robust rfid based 6-dof localization for unmanned aerial vehicles. *IEEE Access*, 7:77348–77361, 2019.

[A15] Eyesee: the drone allowing to automate inventory in warehouses. `http://www.hardis-group.com`. Accessed: 2016.

[A16] Airborne data collection. `http://dronescan.co`. Accessed: 2016.

[A17] The flying inventory assistant. `http://www.fraunhofer.de`. Accessed: 2016.

[A18] Pranay Jhunjhunwala, M. Shriya, and Elizabeth Rufus. Development of hardware based inventory management system using uav and rfid. In *2019 International Conference on Vision Towards Emerging Trends in Communication and Networking (ViTECoN)*, pages 1–5, 2019.

[A19] Woong Kwon, Jun Ho Park, Minsu Lee, Jongbeom Her, Sang-Hyeon Kim, and Ja-Won Seo. Robust autonomous navigation of unmanned aerial vehicles (uavs) for warehouses' inventory application. *IEEE Robotics and Automation Letters*, 5(1):243–249, 2019.

[A20] Sung Moon Bae, Kwan Hee Han, Chun Nam Cha, and Hwa Yong Lee. Development of inventory checking system based on uav and rfid in open storage yard. In *2016 International Conference on Information Science and Security (ICISS)*, pages 1–2. IEEE, 2016.

[A21] Ioan Susnea, Grigore Vasiliu, Adrian Filipescu, Adriana Serbencu, and Adrian Radaschin. Virtual pheromones to control mobile robots. a neural network approach. In *2009 IEEE International Conference on Automation and Logistics*, pages 1962–1967, 2009.

[A22] Ali Abdul Khaliq. *From Ants to Service Robots: an Exploration in Stigmergy-Based Navigation Algorithms*. PhD thesis, Örebro University, 2018.

[A23] Ali Abdul Khaliq and Alessandro Saffiotti. Stigmergy at work: Planning and navigation for a service robot on an rfid floor. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1085–1092. IEEE, 2015.

[A24] Victor Casamayor-Pujol, Marc Morenza-Cinos, Bernat Gastón, and Rafael Pous. Autonomous stock counting based on a stigmergic algorithm for multi-robot systems. *Computers in Industry*, 122:103259, 2020.

[A25] Keonn's AdvanReader 160 keonn's advanreader 160. `https://keonn.com/components-product/advanreader-160/`. Accessed: 2022.

[A26] Keonn's AdvantennaSP11 keonn's advantennasp11. `https://keonn.com/components-product/advantenna-sp11/`. Accessed: 2022.

[A27] GEN. 2 rfid protocol gen. 2. `https://www.gs1.org/epc-rfid`. Accessed: 2022.

[A28] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[A29] Anis Koubâa et al. *Robot Operating System (ROS).*, volume 1. Springer, 2017.

[A30] RFID Plugin abdussalam alajami, a. a., rafael pous, and guillem moreno. (2022). rfid-sensor gazebo plugin. ros wiki. `http://wiki.ros.org/RFIDsensor\_Gazebo\_plugin`. Accessed: 2022.

[A31] Marius Beul, David Droeschel, Matthias Nieuwenhuisen, Jan Quenzel, Sebastian Houben, and Sven Behnke. Fast autonomous flight in warehouses for inventory applications. *IEEE Robotics and Automation Letters*, 3(4):3121–3128, 2018.

3.10 Overall Conclusion and Future Work

The study presented in Chapter 3, which consists of the design of a UAV that is able to perform an inventory task indoors and in map-less environments, took various phases to reach its goal and to obtain promising results. These phases started initially by designing a UAV that tended to perform the task of taking an RFID-based inventory, which meant various hardware needed to be tested to find the optimum combination of sensors for the design configuration. As mentioned in Chapter 2, various navigation algorithms were tested on the UAV measuring the cost-effectiveness, the power efficiency, the computational cost, the robustness and stability of navigation, the effect on flight time, the performance of taking an inventory, and other important parameters that affect the UAV. Throughout the optimization experiments, the UAV models suffered many crashes which made the UAV un-flyable and out of service for repair. These crashes resulted in time delays due to the shipping time of the parts and the repair time needed to repair the UAV. Nevertheless, the crashes during the first phase of the study in this thesis substantially increased the costs. This led us to find solutions to mitigate these costs using simulation. ROS allows the integration with some powerful simulation tools which allow to simulate robots and environment models with almost exact accuracy. Gazebo simulation tool as mentioned before allows to simulate robot models, sensors, and worlds graphically and even simulates the physical parameters and dynamics that incorporate these models. However, at the moment of designing the UAV, there were no available tools to simulate RFID technology with robotics. Therefore, a design of a plugin that simulates RFID sensors and can be used with Gazebo, and is ROS-based was a logical path to take at this point of the study. The motivation was to allow the vast robotics community to be able to simulate accurately RFID technology with robotics, doing so by creating an open-source tool and publishing it on the main wiki page for ROS. Chapter 4, discusses an article on the design of a simulator tool that simulates RFID technology with robotics.

Chapter 4

A DESIGN PLATFORM TO SIMULATE RFID SYSTEMS FOR ROBOTS

4.1 Abstract

Simulation, robotics, and Radio Frequency Identification (RFID) technology have significant roles in the new industrial revolution and their applications are key aspects of making Industry 4.0 a reality. Developing efficient use cases in Industry 4.0 almost always requires accurate simulation tools to be used in the digital world. The problem of simulating RFID readers for robotics in environments where high populations of RFID tags exist is addressed in this paper. This paper will discuss the design of an RFID system plugin based on Robot Operating System (ROS) and Gazebo simulator and the probability-based model on which the plugin is based. To assess the performance of the proposed system model, the simulation results of the designed plugin are compared with experiments. We also prove that the proposed simulator is flexible enough to be used on any robot platform, including aerial and ground robots. We show the initial results of the simulation of having an Unmanned Aerial Vehicle (UAV) and an Unmanned Ground Vehicle (UGV) equipped with an RFID reader, navigating in an environment in which RFID tags have been placed. The robots will be reading tags in different map layouts using RFID antennas, with different orientations. We compare the simulation and experimental results in terms of the total unique tag readings vs. time, for various map-layouts. Finally, we show how this plugin can be used in robotics research by using it to simulate a novel, RFID-based stigmergic navigation strategy. We illustrate, the accurate navigation of the UAV using the proposed plugin.

Keywords: *Gazebo, Industry 4.0, Inventory, KEONN, Retail, RFID Plugin, RFID Technology, Robotics, ROS, UAV*

4.2 Introduction

Robotics and automation are quickly becoming one of the main success factor in e-commerce and Industry 4.0. They have a very big impact on the world of logistics. The number of multipurpose industrial robots developed and designed in the Industry 4.0 for Europe alone has almost doubled since 2004 [B1]. An essential aspect of Industry 4.0 is autonomous production, warehouse, and inventory management methods powered by robots that can complete tasks intelligently, with the focus on safety, flexibility, versatility, cost effectiveness, and collaboration. Without the need to isolate its working area, its integration into human workspaces becomes more economical and productive, and opens up many possible applications in the industry [B2]. Industry 4.0 technology intends among other goals to revolutionize Inventory Management. New technologies are already transforming how businesses is approaching inventory management. AI algorithms, IoT-powered tracking systems and robots can optimize existing inventory management processes and streamline business planning [B3]. Simulations have an important role in Industry 4.0. It leverages real-time data to mirror the physical world in a virtual model, which can include robots, products, humans, and entire warehouses. This allows operators to test and optimize the environment and robot configuration in the virtual world before deployment, thereby driving down setup costs and increasing quality. It could also aid retail companies to evaluate the risks, costs, implementation barriers, impact on operational performance, and roadmap toward Industry 4.0 [B4]. For much of the robotics community, the Open-source Gazebo [B5] robot simulator is a fundamental tool in the development of ground and aerial robot applications for indoor and outdoor environments. The Gazebo simulator allows users to extend its functionality with user-defined plugins. By using an API, plugins have access to the simulation objects and data, can transmit information via topics by using Protocol Buffer [B6] messages, and apply torques and forces to objects in the simulation scenario. The plugins must be initialized with a robot, sensor, or world model. For applications using ROS [B7], the robot descriptions are specified in the SDF [B8] or URDF [B9] file formats.

Gazebo supports multiple interfaces, allowing users to interact programmatically with the simulation, including C++, a custom network transport, and ROS messaging.

The use of RFID technology with robotics has recently attracted a lot of attention both in the academic community and in the industry [B10], especially in logistics and retail ([B11], [B12], [B13], [B14]).

In this paper, we propose a solution for the problem of simulating RFID readers for robotics in environments where high populations of RFID tags exist. We propose a design of an RFID system plugin based on ROS and the Gazebo simulator and the probability-based model on which the plugin is based. This paper is structured as follows: Section 4.3, presents the state of the art. Section 4.4, presents the explanation of a simplified, effective, and fairly accurate Probability of Detection (PD) model on which the plugin is based. Section 4.5, presents the architecture and model integration of the designed plugin in ROS. Section 4.6, present the environment layouts and robots that were used to conduct the experiments illustrated in this paper. The results of the experiments in the laboratory and simulations are described in Sections 4.7 and 4.8. Section 4.9, summarizes the conclusions of this work, and Section 4.10, presents the future work.

4.3 Related work

Simulating environments before deploying them when large quantities of RFID sensors and robots are used saves a lot of time and cost. The simulation of these environments helps understand the weaknesses of the designed layout before its deployment in the physical world. Such environments can be warehouses, retail shops „etc. Having a simulation platform that allows to test different types of mobile robots not only helps to asses and optimize the environment, but also can be used to validate the performance of these robots, the planning/navigation algorithms/strategies, and the RFID-related algorithms in the presence of RFID technology. Very few RFID simulators have been designed in the past. There has been related work by other researchers in the RFID domain on the simulation of RFID systems. Han et al. in [B15], developed a system model of UHF RFID with a strong focus on the RF/analog design of the RFID reader. The model presented by Han et al. models the signal generation in the reader to verify whether the signal transmitted complies with the specified spectrum mask in the radio regulations. There is also a detailed model of the receiver part of the RFID reader that further analyses the effect of transmitter/receiver coupling. The wireless channel in their simulator is modeled as the vector addition of various multipaths. Authors in [B16], present an RFID simulation engine, called RFIDSim, which implements the ISO 18000-6C communication protocol [B17] and supports path loss, fading, backscatter, capture, and tag mobility models. They show that RFIDsim can be used to simulate large populations featuring thousands of RFID tags. Their model also simulates the deep fades that lead to frequent power losses of the passive RFID tags by modeling the multipath effects statistically. RFIDSim aimed to facilitate the relative comparison of different transmission control strategies. An approach in [B18], similar to RFIDsim, proposed a simulation platform that re-

lies on a discrete event simulator, designed also to implement a part of the ISO 18000-6C communication protocol supporting path loss, backscatter, capture, and tag mobility models. These models however are either too old, hard to adapt to robotics simulation platforms, or no longer available, and more importantly focused on the low-level communication issues between tags and reader antennas. Only a few of these models allow environment remodeling, design, or manipulation in real time. Most are not open source. Finally, none of the simulators can be easily adapted to work with robots or the tools that come with ROS. During the development of the proposed simulator in this paper, a study was published by the authors in [B19]. The authors propose a simulator that is implemented as a Gazebo plugin integrated with ROS. A tag localization algorithm that uses the phase unwrapping technique and hyperbolae intersection method employing a reader antenna mounted on a mobile robot is used to estimate the position of the tags deployed in the presented scenarios. The user needs to specify the frequency, the range, the phase noise, and the gain of the tag antenna, as well as physical parameters, like damping coefficient and friction. The outcomes of their experiments showed realistic results for environments with a low number of tags (up-to 10). However, it is not known how the model will behave when simulating real environments with large populations featuring thousands of RFID tags. The illustrated experiments comparing the simulator's performance with real life do not test the behavior of the simulator using different types of robots, asses the accuracy of RFID tag detection in different altitudes, nor was it tested in different environments with different tag densities. Having large populations of RFID tags in an environment introduce more parameters that may reduce the accuracy of the model. Cross interference is another type of interference that can degrade the performance of the simulator. It is most likely to occur between RFID systems and WIFI or personal area networks (WPAN) such as Bluetooth but only when devices share common or adjacent frequency bands within the environment. In [B20], authors propose an interference avoidance scheme that requires the knowledge of the theoretical maximum collision time and collision probability between RFID and WiFi/Bluetooth packets. This scheme generates an optimal channel based on the current usage of the adjacent frequency channels thereby reducing the interference. We conclude from the above, that it is extremely complex to consider all the parameters that affects the interrogated wave in a single algorithm, acquiring an exact estimation of whether a tag is detected or not by a reader, especially in environments where large populations of tags exist. In this paper, we introduce a simulation tool that simulates RFID systems in Gazebo. This plugin uses a different approach than the previously mentioned models. The model does not aim to estimate the detection of a single RFID tag. It rather estimates the probability that the placed RFID tag can be detected or not considering various environment and sensor parameters. This model unlike its counters is aimed to simulate dif-

ferent types of robots in environments where hundreds or thousands of tags are to be detected. This makes it ideal to simulate retail shops or big warehouses, where autonomous inventory robots would need to be tested in such environments before being deployed. The Gazebo environment is a very useful simulation tool for the broad audience in the robotics community, however, it is computationally costly and demands high system specification requirements. Complicated environments where large quantities of RFID tags (hundreds to thousands) are required to be detected, spawned, and managed can be very difficult to simulate. It may require special and expensive machines to be able to run the environment using the simulators discussed above. Due to the simplicity of the proposed plugin, it enables a large population of RFID tags to be simulated within the environment. In order for the plugin to work properly, a calibration phase to adapt the model to the environment is required, considering and encapsulating in a simplified manner most characteristics and interference parameters in the environment that are hard to consider. In this study, we run also multiple experiments comparing and validating the performance of the simulator in different laboratory environments. The proposed plugin is not designed to work for a specific sensor or a specific manufacturer. It is designed to be general for any RFID reader and with any robot in any environment. It is aimed to reach the broad audience of the robotics community; therefore, it is Open Source, and already published [B21] in the main ROS Wiki, website for ROS platform users, with about 370 downloads in the month of the submission of this paper.

4.4 The proposed RFID system model

4.4.1 Model Overview

The proposed model is based on estimating a *PD* for each tag by each RFID reader antenna. This probability depends on the relative position and orientation of the tag with respect to the antenna, so it must be recalculated every time this relative position changes, as the robot moves. The goal of the model is not to estimate whether a specific tag is read or not, but to estimate how many tags will be read from a given constellation of tags. When we compare with experiments, we will not compare tags on an individual basis, but we will compare whether the simulation and the experiment have read approximately the same number of tags, and at a similar rate (tags read per second), which is the way accuracy is calculated in RFID deployments.

The *PD* is defined for each tag-antenna pair as a function of 6 arguments. The first 3 are the distance and two angle coordinates of the tag with respect to antenna position and its direction of maximum radiation: R, θ_H, and θ_V, which in

Gazebo are the representation of the transforms between the sensor-frames [B22] "RFID-Antenna" and "RFID-Tag", which are automatically calculated by ROS.

The remaining 3 parameters are constants that depend on the particular RFID system used and its RFID settings. These parameters are R_0, the distance at which half of the tags can be read during a specific duration of time, and the antenna beam widths in the horizontal and vertical planes, $\Delta\theta_H$ and $\Delta\theta_V$. R_0 depends mostly on the Equivalent Isotropic Radiated Power ($EIRP$) which is defined as the product of the conducted power (P_{in}) and the antenna gain (G_t), $EIRP = P_{in} \cdot G_t$, as well as the sensitivity of the tags. $\Delta\theta_H$ and $\Delta\theta_V$ depend on the particular reader antenna used in the system. These 3 parameters must be supplied by the user of the plugin. The antenna beam-widths are normally found in the data sheet of the antenna used in the system, and R_0 must be adjusted by calibrating the simulation against some experiments.

R_0 is defined so that when $R = R_0$ and $\theta_H = \theta_V = 0$ the probability of detection is $PD = 0.5$. At other distances and angles, PD is calculated using the antenna pattern based on the beam-widths, and the $1/R^2$ decay of surface power density.

Increasing R_0 (by increasing the $EIRP$, the gain of the reader antenna, and/or the sensitivity of the tags), will allow the antenna to detect distant tags with higher probability while decreasing it will tend to allow only the detection of tags at shorter distances. On the other hand, using antennas with wider beam-widths will allow the antenna to detect tags at wider angles from the front direction of the antenna.

4.4.2 Model Definition

An antenna's radiation pattern describes how the antenna radiates/receives energy into/from all directions in space, and is three-dimensional. In the model, we approximate the normalized (maximum value of 1) radiation pattern as the function $D_0(\theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V)$ in Eq. 4.1.

$$D_0(\theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V) = \begin{cases} \cos^2\left(\frac{\pi}{2} \cdot \frac{\theta_H}{\Delta\theta_H}\right) \cdot \cos^2\left(\frac{\pi}{2} \cdot \frac{\theta_V}{\Delta\theta_V}\right) & \text{if } \theta_H \text{ and } \theta_V \leq \frac{\pi}{2} \\ 0 \text{ if } \theta_H \text{ or } \theta_V > \frac{\pi}{2} \end{cases} \tag{4.1}$$

Note that:

- $D_0(0, 0, \Delta\theta_H, \Delta\theta_V) = 1$
- $D_0(\pm\Delta\theta_H/2, 0, \Delta\theta_H, \Delta\theta_V) = \frac{1}{2}$

- $D_0(0, \pm\Delta\theta_V/2, \Delta\theta_H, \Delta\theta_V) = \frac{1}{2}$
- $D_0(\pm\Delta\theta_H/2, \pm\Delta\theta_V/2, \Delta\theta_H, \Delta\theta_V) = \frac{1}{4}$

This approximation is valid for antennas with a well-defined main lobe and considers any radiation in the back hemisphere as negligible.

Approximating the tag antenna as isotropic, and neglecting any multipath interference, the received power by the antenna is proportional to the reader antenna directivity and to $\frac{1}{R^2}$, as shown in Eq. 4.2.

$$P_{rec}(R, R_0, \theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V) \propto D_0(\theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V) \cdot \frac{R_0^2}{R^2} \quad (4.2)$$

Given that the probability of detection $pd(R, R_0, \theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V)$ must be defined so that:

- $pd(0, R_0, 0, 0, \Delta\theta_H, \Delta\theta_V) = 1$
- $pd(R_0, R_0, 0, 0, \Delta\theta_H, \Delta\theta_V) = \frac{1}{2}$
- $pd(\infty, R_0, 0, 0, \Delta\theta_H, \Delta\theta_V) = 0$

We arbitrarily define it as:

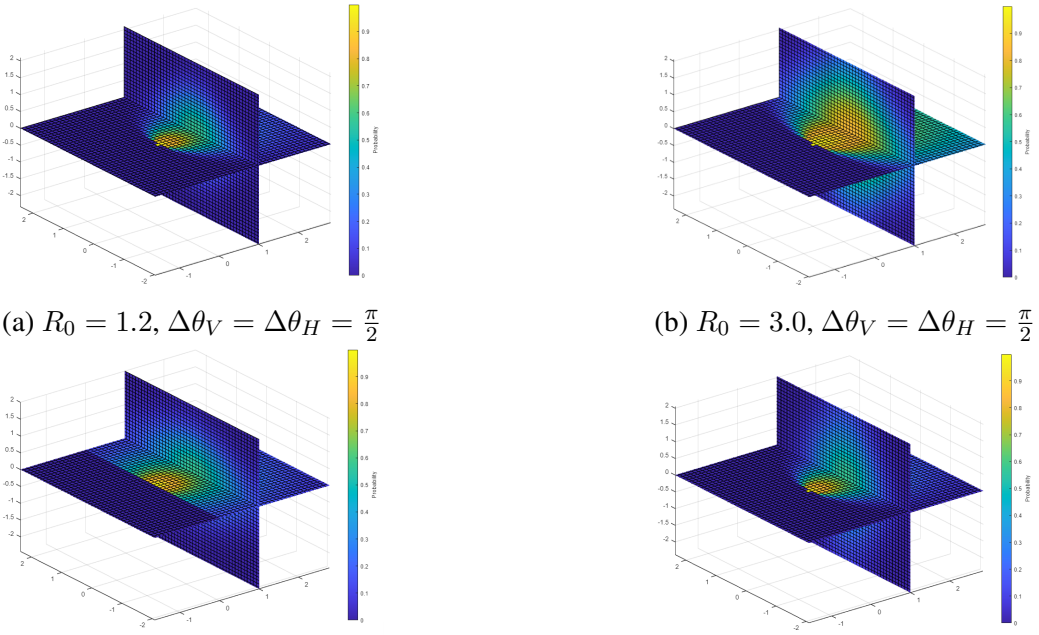
$$pd(R, R_0, \theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V) = B(x) = \frac{2}{1 + 3^{\sqrt{x}}} \quad (4.3)$$

where x is always positive, and is defined as:

$$x = \frac{P_{rec}(R_0, R_0, 0, 0, \Delta\theta_H, \Delta\theta_V)}{P_{rec}(R, R_0, \theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V)} = \frac{1}{D_0(\theta_H, \theta_V, \Delta\theta_H, \Delta\theta_V) \cdot \frac{R_0^2}{R^2}} \quad (4.4)$$

Any other smooth function $B(x)$ so that $B(0) = 1$, $B(1) = \frac{1}{2}$, and $B(\infty) = 0$ will give very similar results (e.g, $B(x) = \frac{2}{1+3^{x^2}}$, $B(x) = 2^{-\sqrt{x}}$, $B(x) = 2^{-x^2}$), but Eq. 4.4 has shown the best results when compared with experiment.

Fig. 4.1, shows a 3D visualization of the *PD* function using different values for each parameter.



(a) $R_0 = 1.2$, $\Delta\theta_V = \Delta\theta_H = \frac{\pi}{2}$

(b) $R_0 = 3.0$, $\Delta\theta_V = \Delta\theta_H = \frac{\pi}{2}$

(c) $R_0 = 1.2$, $\Delta\theta_H = \pi$, $\Delta\theta_V = \frac{\pi}{2}$

(d) $R_0 = 1.2$, $\Delta\theta_H = \frac{\pi}{2}$, $\Delta\theta_V = \pi$

Figure 4.1: *PD* for various parameter sets.

4.5 RFID System Plugin Architecture

The RFID system Gazebo plugin is composed of two independent plugins: the *RFID-Tag plugin* and the *RFID-Antenna plugin*. Both plugins are called from an SDF model file. Depending on the application scenario, they can be attached to the robot SDF model, or the world model, which represents the environment in Gazebo.

4.5.1 RFID Tag plugin

As in real life, RFID tags are placed in the environment attached to products, boxes, shelves, or any other objects. Each simulated RFID tag will be spawned and represented as very small cubes. RFID tags will be spawned in groups, associated with box-like transparent objects. This object is referred to as a fixture. The user will have the option to either load a costume map of fixtures or automatically generates a semi costume map by adjusting some parameters. The costume map is composed of a file that contains the fixture's id and coordinates in the map. The user can choose to load up to 10,000 fixtures and tags in an environment. The automatic map generator parameters are the number of fixtures, their

height, their associated number of RFID tags, and the spawn range, which can be accessed through the parameter list in the SDF file that Gazebo API permits. The RFID-Tag plugin allows the graphical change of the fixture and tag model designs. Real time manipulation of the fixtures and tags positions is allowed. The spawned RFID tags will be connected to the *odom_sim* frame which represents the map frame or the world in Gazebo.

4.5.2 RFID Antenna Plugin

Similar to the *RFID-Tag plugin*, the antenna model in the *RFID-Antenna plugin* is spawned by loading its SDF model file. A ROS frame will be generated at the same moment the antenna model is spawned in Gazebo, and the RFID antenna child frame will be connected to a parent frame as defined by the user. In the case of the robots used in our experiment, the antenna frames are directly connected to the robot frame *base_link*. The *RFID-Antenna plugin* inherits the probability model function, as explained in the *PD* model section. The *PD* requires the distance, azimuth, and co-elevation angles of each tag (R ,$\theta_H = \pi$, θ_V) to produce a probability of detection value. These parameters are provided by the direct processing of relative transforms between the RFID antenna pose and the RFID tag pose. This process is done in real time during the start of a mission. In Figs. 4.2, 4.3a, and 4.3b, we show the transform tree between the RFID tag, the RFID antennas, the robot frame, the sensor frames, the odom, and the map-world frame.

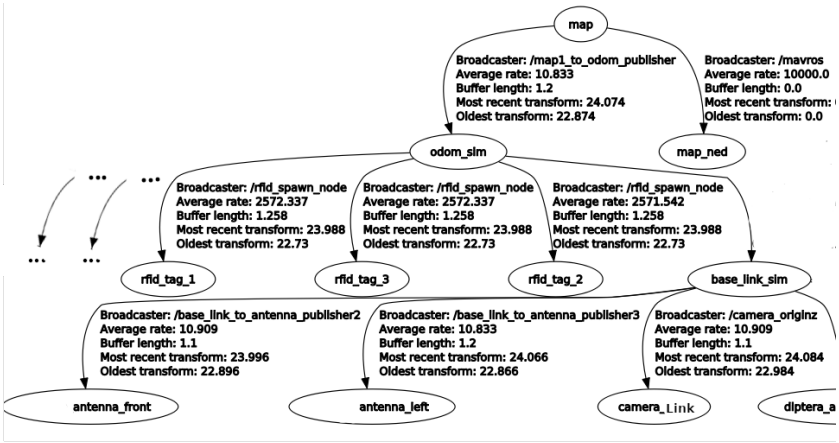
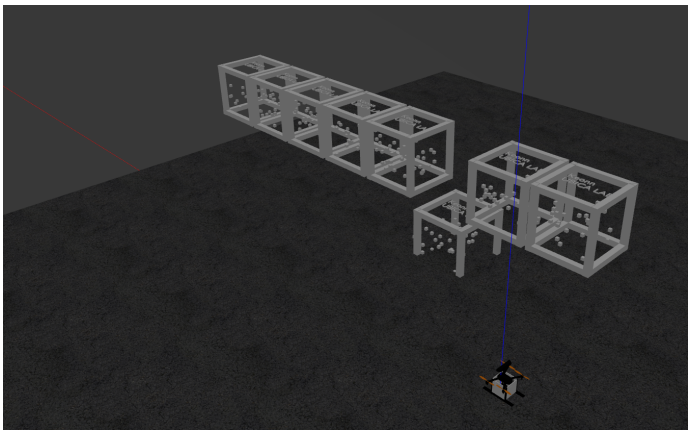
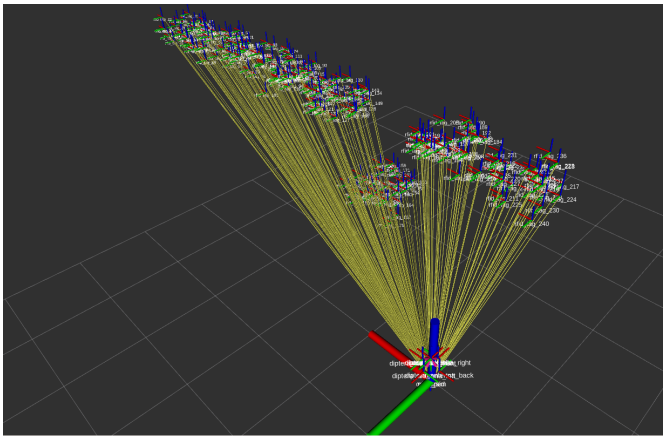


Figure 4.2: Frame tree from RQT-ROS.



(a) Gazebo illustration of an example scenario.



(b) Rviz illustration of an example scenario.

Figure 4.3: Example scenario shown in Gazebo and Rviz.

4.6 Environment Layouts and Robots Used IN The Experiments

4.6.1 Robots used in the experiments

Unmanned Ground vehicle (UGV)

The pre-designed and patent UGV in [B23] is used. This UGV is used to carry the RFID payload, and its hardware block diagram is shown in Fig. 4.4. The UGV is designed to autonomously navigate within the given path in the intended map layout. The RFID payload consists of a Keonn AdvanReader 160 [B24] RFID reader having four RF ports, each port connected to a Keonn Advantenna SP11 [B25] RFID antenna placed each in a different orientation.

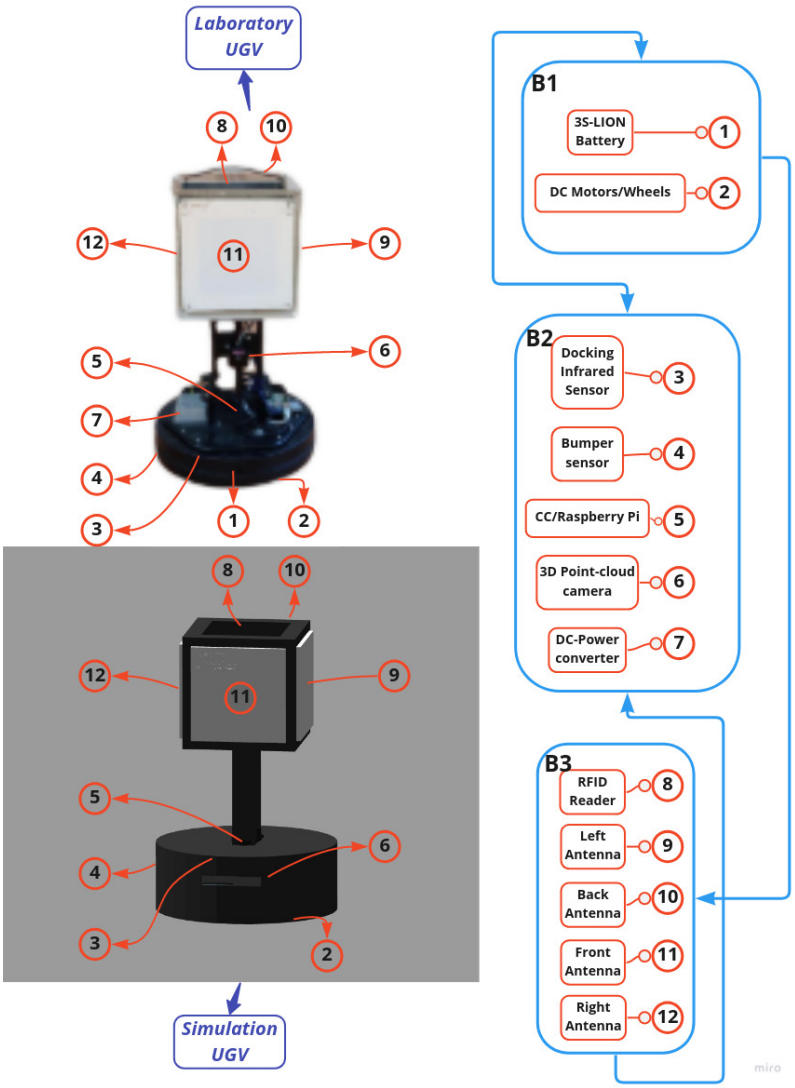


Figure 4.4: Hardware block diagram of the UGV.

Unmanned Aired vehicle (UAV)

UAVs have evolved a great deal in the last several years in terms of technology (e.g., autopilots, sensors, power efficient motors, battery capacities and sizes) [B26, B27], enabling them to be used for different purposes and applications including RFID-based inventory. In addition to the UGV, a custom-designed UAV was built especially for the purpose of aerial based inventory, and to evaluate the performance of the plugin compared to real laboratory environments. It is used to carry the RFID payload for laboratory tests. The UAV is composed of 3 main hardware blocks:

1. Block 1 (B1): The Main flight system: The UAV was designed to navigate indoors and to carry a heavy payload, therefore, a 6-motor UAV (hexacopter) frame was chosen. This chosen design layout enables a stable flight in indoor spaces during a task-sufficient flight time. An efficient open-source autopilot called (Pixhawk) that is compatible to operate with a companion computer was used. This block is responsible for most physical and mechanical properties required for hovering and contains 6 motors, 6 propellers, and 6 electronic speed controllers (ESCs). The ESC's main objective is to translate the signal received from the autopilot to energy from the energy source and supply it to the motors. The energy source is composed of a lithium polymer battery (LIPO) with the capacity of 8000mAh, operating voltage 26V, and with high energy discharge rate.

2. Block 2 (B2): Sensors and Processing Units: Most inventory warehouses or retail shops are indoor spaces or so-called GPS Denied Environments. This led us to select a visual Simultaneous Localization and Mapping (SLAM) based camera to supply the UAV with self-localization coordinates. Special adaptation was made to infuse these coordinates into the autopilot, resulting in an indoor guidance system for the UAV. This part is responsible for adding intelligence to the contiguous main flight system part. It also enables the possibility to sense the environment and obstacles nearby through a depth proximity camera. All these sensor data, including the data received from the RFID-Payload, are processed by a companion computer (CC). The CC will incorporate the navigation algorithm. This same algorithm will be used for the simulation and laboratory experiments. The output of this CC will be mainly the control signals in the form of pose-goal, or movement commands to the autopilot.

3. Block 3 (B3): Payload: The UAV will carry the same payload as the UGV but on the bottom of its structure rather than on top. It is composed of a light-weight structure skeleton embedding the RFID-reader, the power converter/distributor circuit board, the RFID-reader, and the 4 antennas. Each antenna will be facing a different direction. This block provides all the RFID-related data to other blocks if needed.

We can see its block diagram in Fig. 4.5.

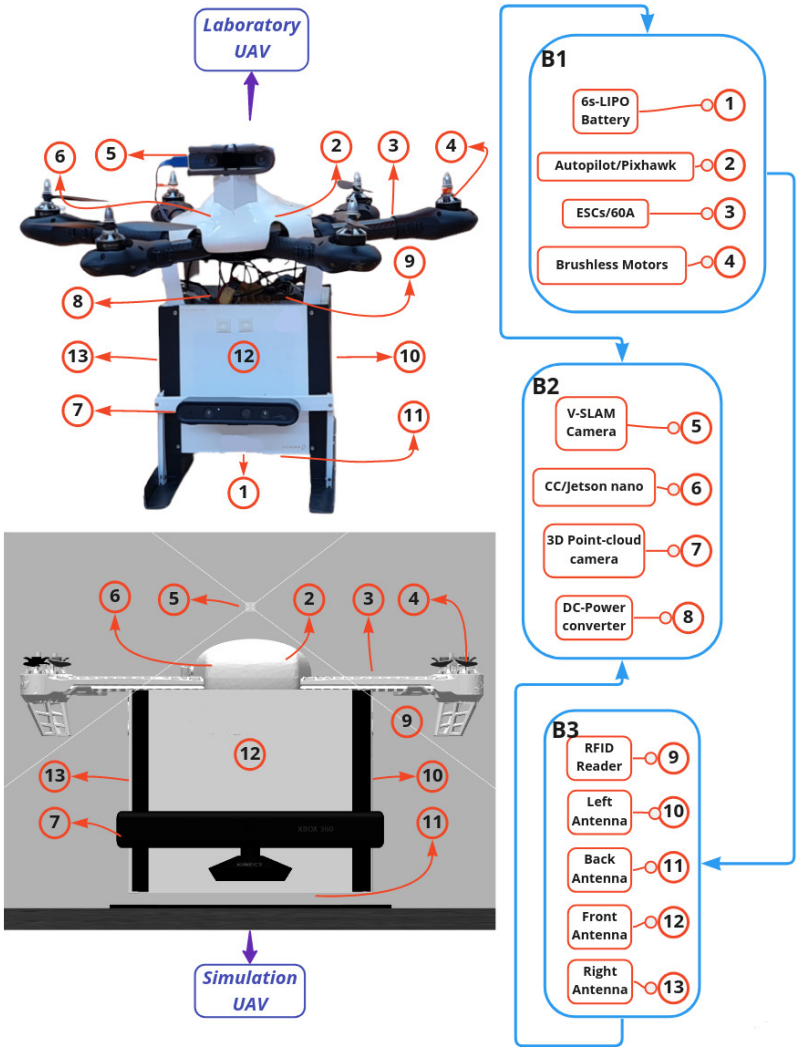


Figure 4.5: Hardware block diagram of the UAV.

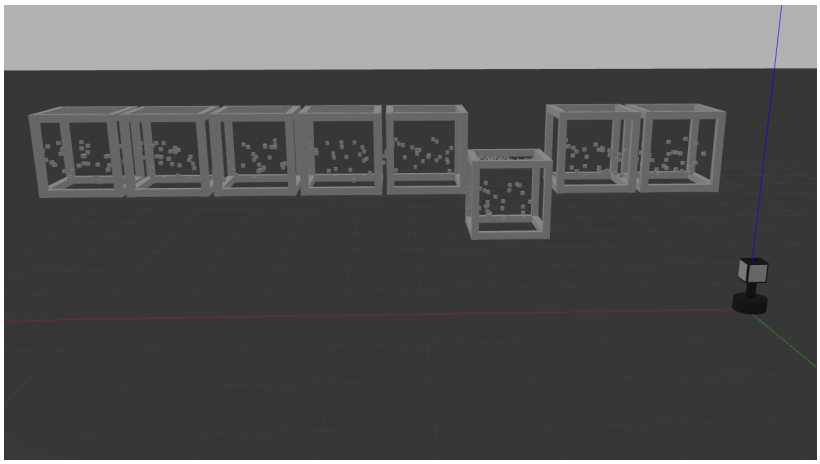
4.6.2 Laboratory and Simulated Environments Layouts

In this section we present the scenarios used for both robots in simulation and experiment:

Scenario 1: One-sided, uniformly-placed tags

In Scenario 1, 8 fixtures are placed at a fixed distance away from the robot's path. 30 tags are placed within each fixture. Figs. 4.6 and 4.7 show the scenario map layout in the laboratory and in simulation, each associated with both each robot: the UGV and the UAV.

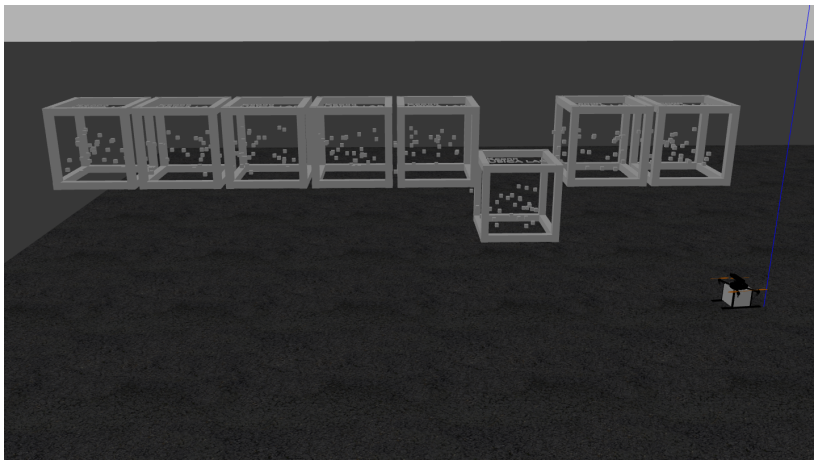
(a) Laboratory setup.



(b) Simulation setup.

Figure 4.6: Laboratory and simulation setups of Scenario 1 with the UGV.

(a) Laboratory setup.



(b) Simulation setup.

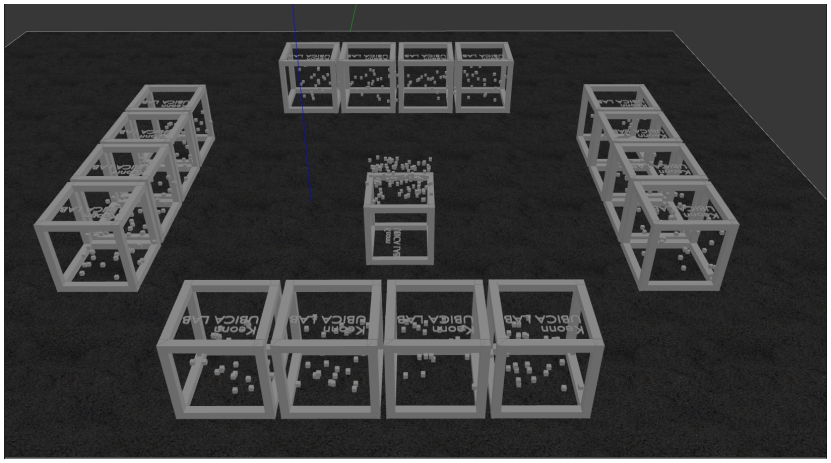
Figure 4.7: Laboratory and simulation setups of Scenario 1 with the UAV.

Scenario 2: Low-density uniformly-distributed tags in a square-shaped map layout

The RFID system plugin would need to prove its performance and accuracy in more complex environments, with different orientations of the tags with respect to the antennas. Therefore, we designed a square-shaped map layout with 600 RFID tags in Scenario 2. The tags were placed uniformly on each side of the square map, with some of the tags in the middle. The laboratory and Gazebo

simulation setups for Scenario 2 are shown in Fig. 4.8.

(a) Laboratory setup.



(b) Simulation setup.

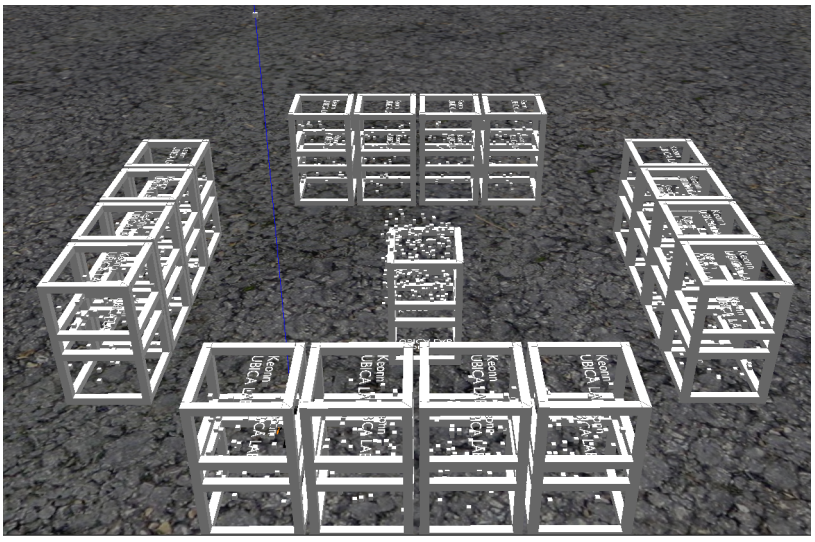
Figure 4.8: Laboratory and simulation setups of Scenario 2 with the UAV.

Scenario 3: Increased tag density, uniformly-distributed tags in a square-shaped map layout

Scenario 3 was used to prove the performance and robustness of the RFID system plugin in a higher-density situation. This was done by doubling the density of tags in Scenario 2 to 1,200 tags, as shown in Fig. 4.9. This scenario also tests the

performance of the plugin having to place tags in different positions on the z-axis.

(a) Laboratory setup.



(b) Simulation setup.

Figure 4.9: Laboratory and simulation setups of Scenario 3 with the UGV.

4.7 Comparison of Simulated and Experimental Results

For all the Experiments in these Scenarios, both robots will be tested with a constant velocity of 1m/s. However, lower or higher velocities could be used. It is important, to take into consideration the differences and the effects that the actual

physical environment has on both robots as compared to the simulation. These effects could be slight odometry or wheel misplacement errors due to the ground surface for a UGV, to small displacements in the UAV position, while constantly trying to stabilize itself in the air, this is due to the generated air turbulence from the propellers and the quality of the localization messages from the visual SLAM sensor.

Experiment 1. Scenario 1 With the UAV

In Experiment 1, we will use the UAV in Scenario 1 to test the performance of the RFID system plugin. We compare the results obtained from the simulation with the ones obtained from running the missions in the laboratory. We first place the UAV at a defined starting position at a distance d from the fixtures. The UAV will take off from that position and navigate in a straight line. We repeat these missions for different values of the distance d. We run these missions with only the right antenna active (the one facing the fixture). Fig. 4.10, illustrates the Experiment 1 setup. Figs. 4.11a and 4.11c, shows the simulation results at $d = 1.75m$ and $d = 3.5m$ respectively. Whereas Figs. 4.11b and 4.11d, show the Lab results at those distances. Figs. 4.12a and 4.12b, show plots that compare the RFID tag readings vs. time in Experiment 1 in simulation and the in the laboratory, showing a remarkable agreement both in the total number of tags read as well as in the rate at which they are read.

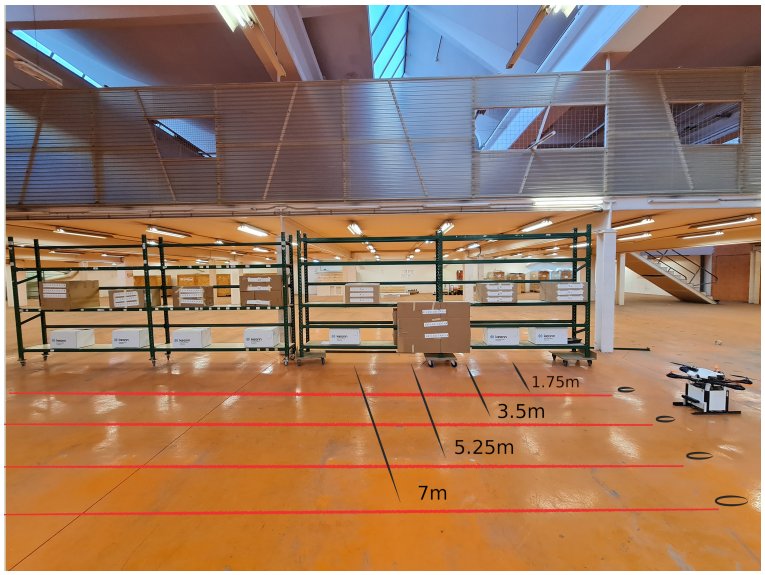
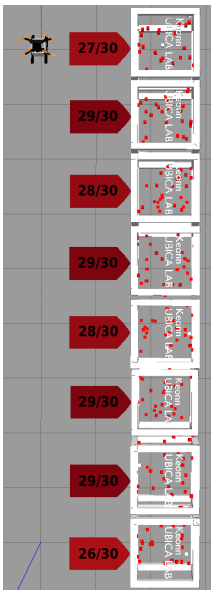
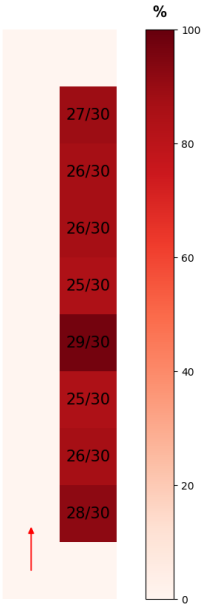


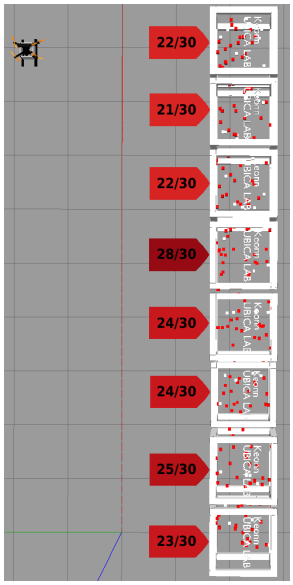
Figure 4.10: Experiment 1: Scenario 1 with UAV placed at different distances from the fixture with RFID tags.



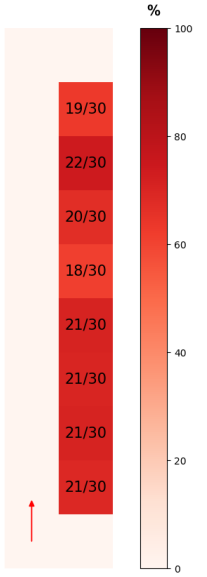
(a) Simulation: d = 1.75m.



(b) Laboratory: d = 1.75m.

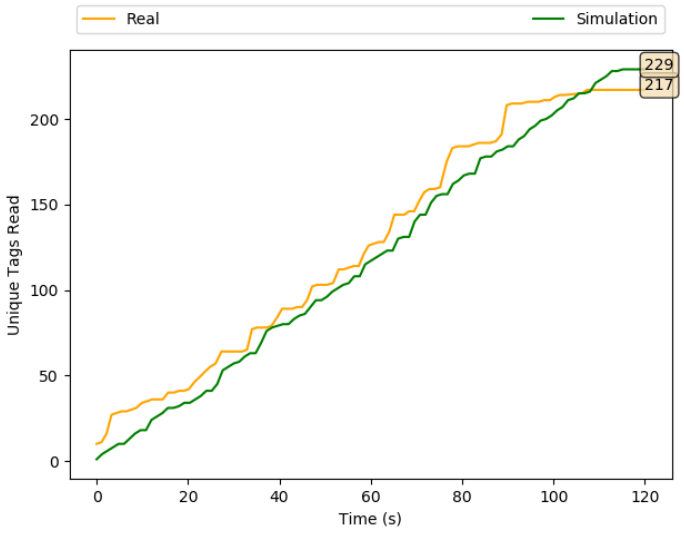


(c) Simulation: d = 3.5m.

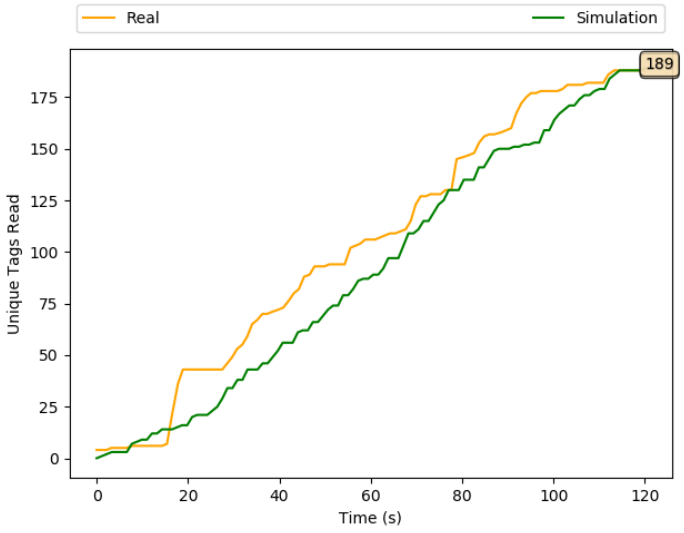


(d) Laboratory: d = 3.5m.

Figure 4.11: Experiment 1: Simulation and laboratory results.



(a) d = 1.75m



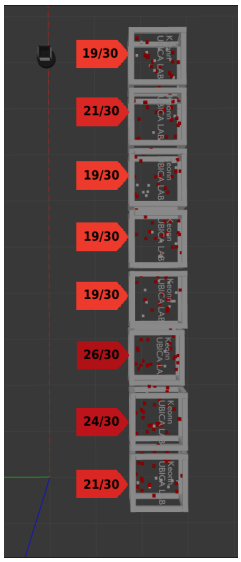
(b) d = 3.5m

Figure 4.12: Experiment 1: Simulation vs. laboratory unique RFID tag readings.

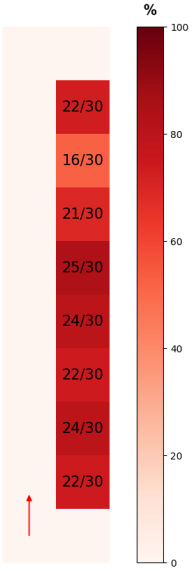
Experiment 2. Scenario 1 With the UGV

In Experiment 2, we repeated Experiment 1 but using a UGV. The Simulation results are shown in Figs. 4.13a and 4.13c, while Figs. 4.13b and 4.13d, shows the results of Experiment 2 in the laboratory. Figs. 4.14a and 4.14b show the RFID tag readings vs. time in Experiment 2 in simulation and the in the laboratory, again

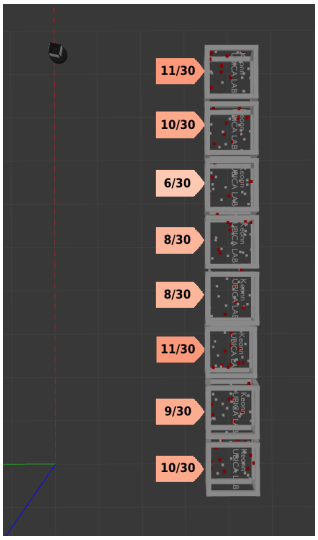
showing a remarkable agreement.



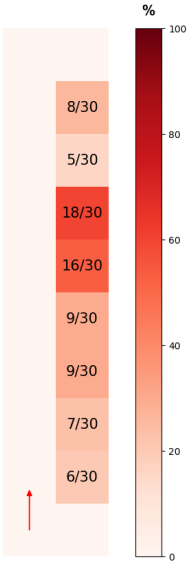
(a) Simulation: d = 1.75m.



(b) Laboratory: d = 1.75m.

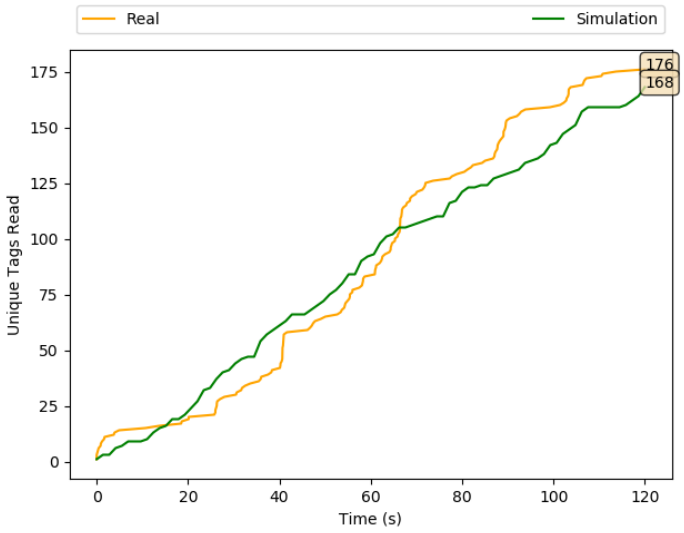


(c) Simulation: d = 3.5m.

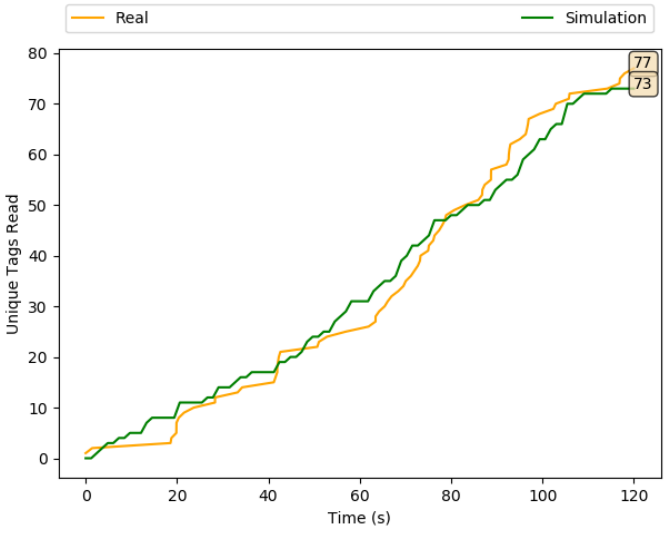


(d) Laboratory: d = 3.5m.

Figure 4.13: Experiment 2: Simulation and laboratory results.



(a) d = 1.75m



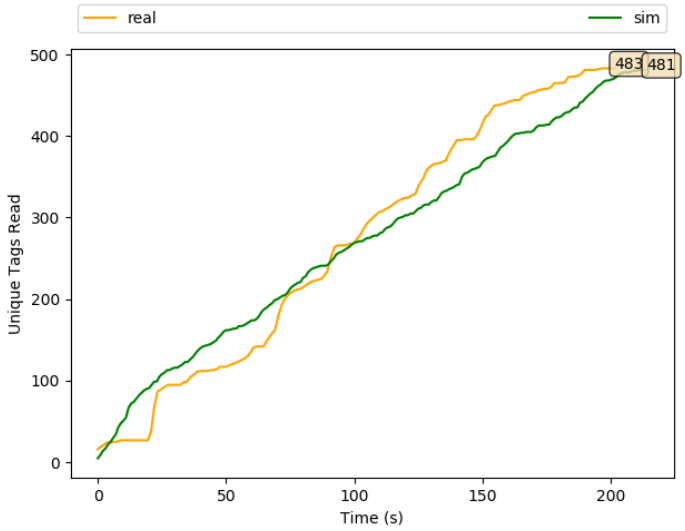
(b) d = 3.5m.

Figure 4.14: Experiment 2: Simulation vs. laboratory unique RFID tag readings.

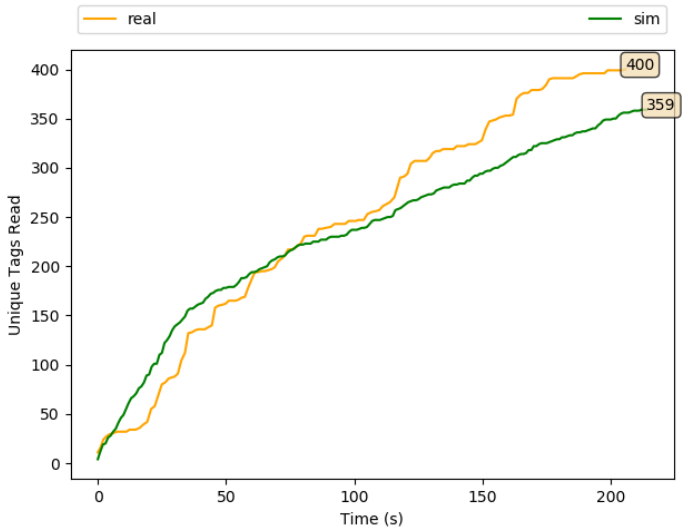
Experiment 3. Scenario 2 With the UAV

This experiment is designed to validate the plugin's performance in a slightly more complex environment. We compare the results obtained from the simulation and the laboratory when the UAV is navigating throughout Scenario 2. Since the scenario map layout is not a straight line as in Scenario 1, the effect of the

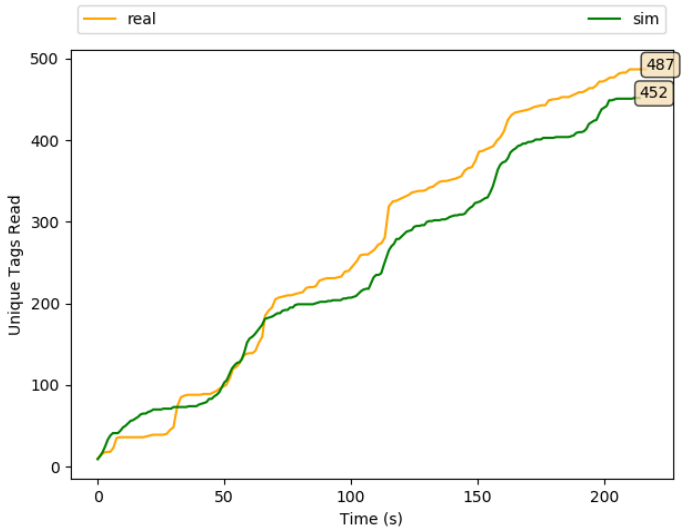
orientation of the RFID antenna on the total number of unique RFID tags read is more relevant. For this reason, we run Experiment 2 four times, and for each one we activate only one of the four RFID antennas mounted on the UAV with different orientations, as illustrated in Fig. 4.5. Figs. 4.15a, 4.15b, 4.15c, and 4.15d, shows the obtained simulation and laboratory results for each antenna orientation, validating the simulation model once again, in this more complex scenario.



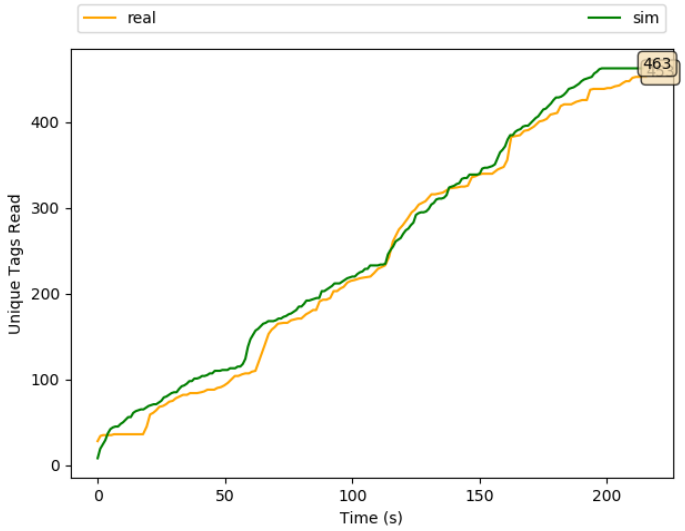
(a) Front Antenna.



(b) Right Antenna.



(c) Back Antenna.



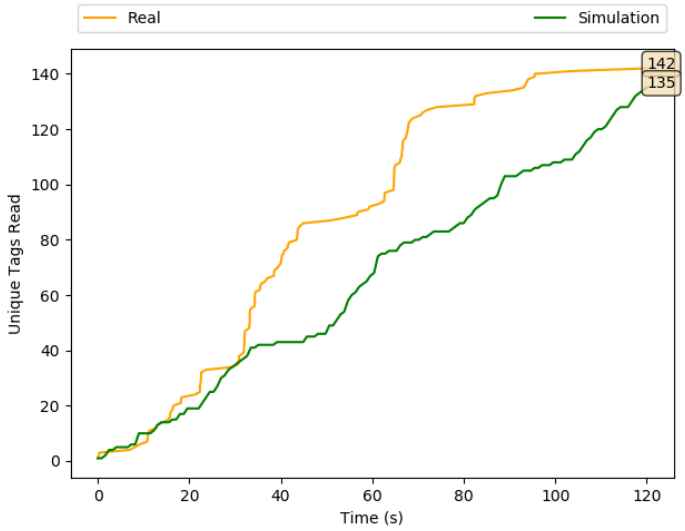
(d) Left Antenna.

Figure 4.15: Experiment 3: Simulation vs. laboratory unique RFID tag readings.

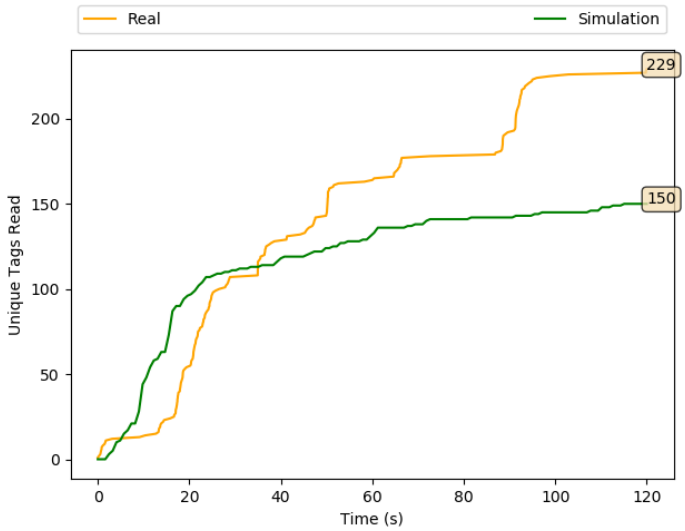
Experiment 4. Scenario 2 With the UGV

In Experiment 4, we repeated Experiment 3, but with replacing the UAV with the UGV. Figs. 4.16a, 4.16b, 4.16c, and 4.16d, illustrate the obtained simulation vs. laboratory results for each antenna orientation. In this experiment, the agreement between simulation and experiment is not as good, especially for the side

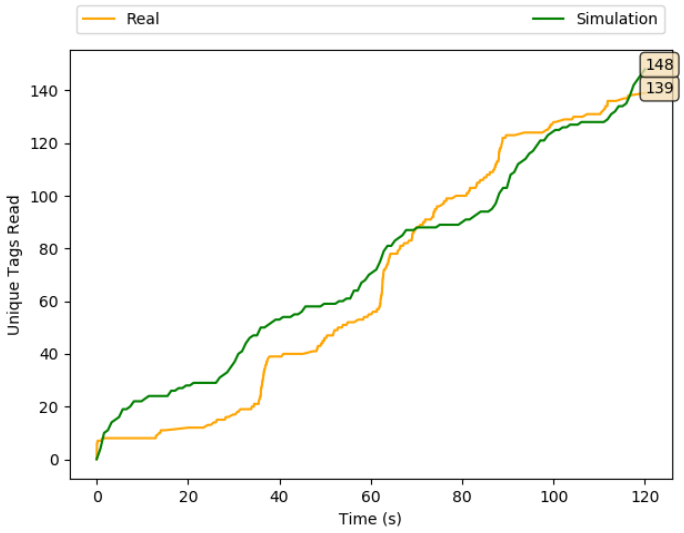
antennas. The reason could be due to the difficulty of precisely navigating the UGV within the path which relies only on the wheel odometry sensor for Dead Reckoning navigation. The UGV was slightly closer to the right tags than the left compared to the simulation. This can be noted from Figs. 4.16b and 4.16d, where more tags are read by the right antenna compared to the left in the laboratory.



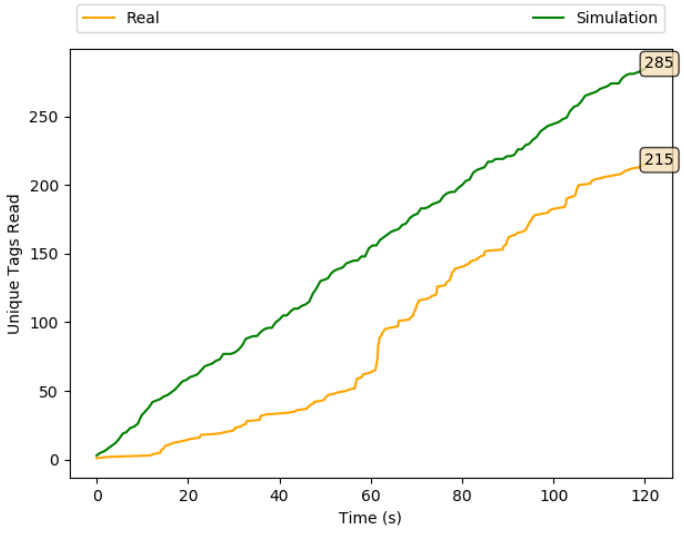
(a) Front Antenna.



(b) Right Antenna.



(c) Back Antenna.



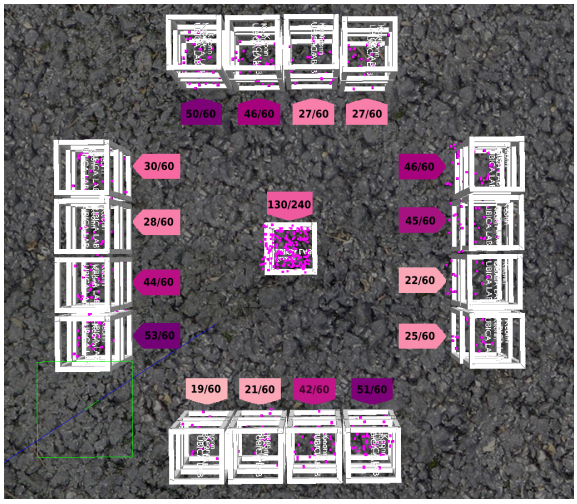
(d) Left Antenna.

Figure 4.16: Experiment 4: Simulation vs. laboratory unique RFID tag readings.

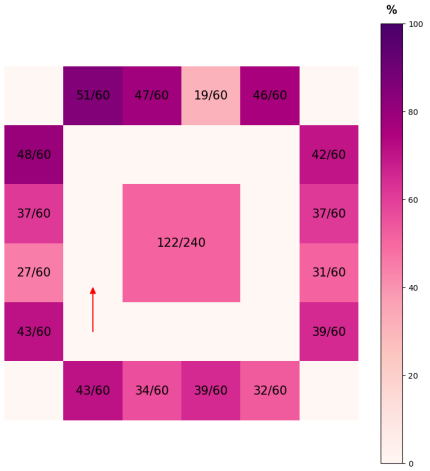
Experiment 5. Scenario 3 With the UAV

The goal of Experiment 5 is to test the performance of the RFID system plugin when used in an environment with a higher density of tags and having tags and fixtures placed in different positions on the z-axis. We compare the obtained results from the simulation and the laboratory, having the UAV navigating throughout

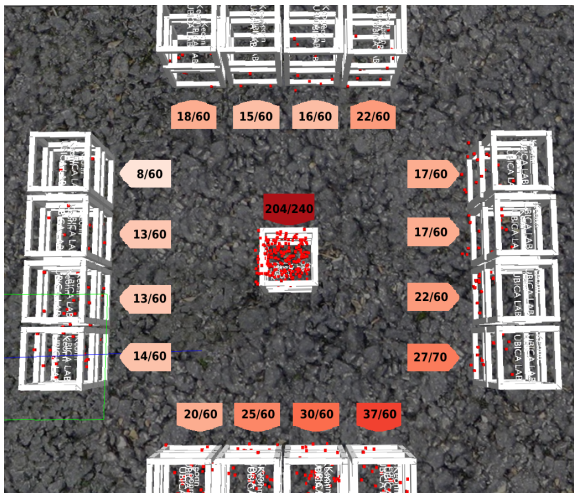
Scenario 3. As in Experiment 4, we repeated Experiment 5 four times, each with one of the four antennas active. We observe in Figs. 4.18a, 4.18b, 4.18c, and 4.18d, the results are in good agreement for all four antennas even though the tag density was doubled. Figs. 4.17a, 4.17c, 4.17e, and 4.17g, illustrate the position of the detected RFID tags in simulation, while Figs. 4.17b, 4.17d, 4.17f, and 4.17h, illustrate the position of the detected RFID tags in the laboratory, which are also in very good agreement.



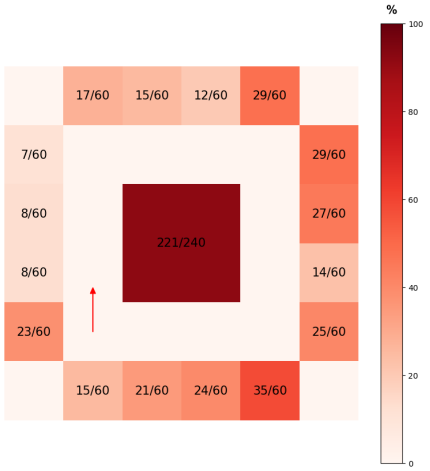
(a) Simulation: Front antenna.



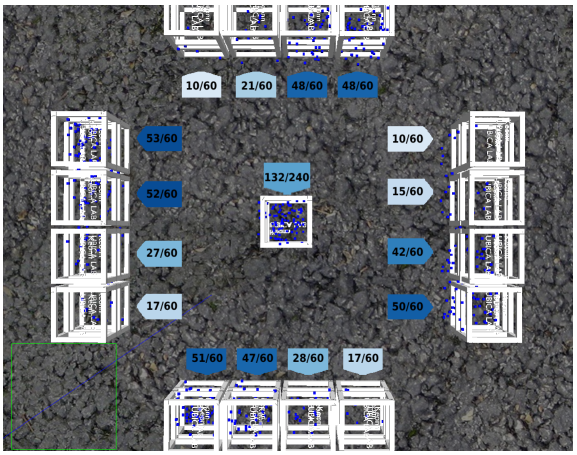
(b) Laboratory: Front antenna.



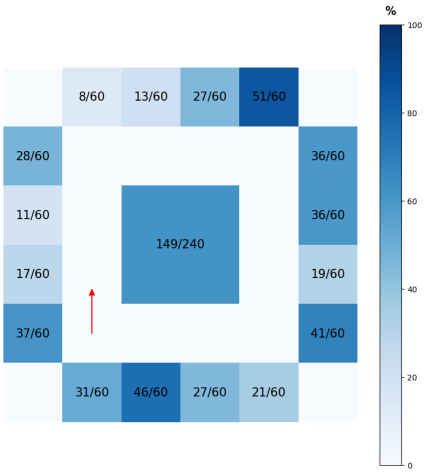
(c) Simulation: Right antenna.



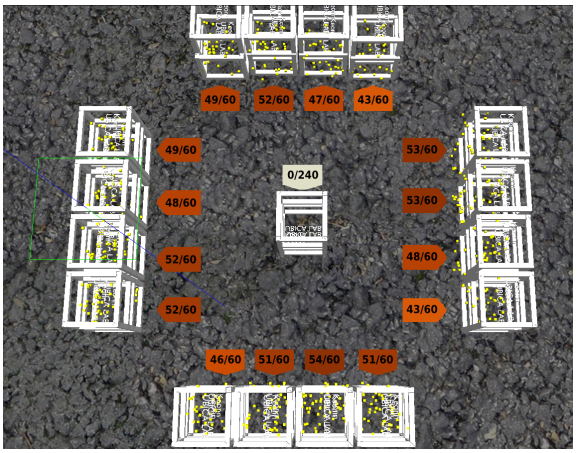
(d) Laboratory: Right antenna.



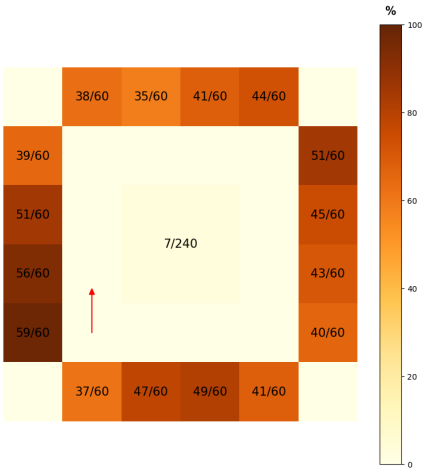
(e) Simulation: Back antenna.



(f) Laboratory: Back antenna.

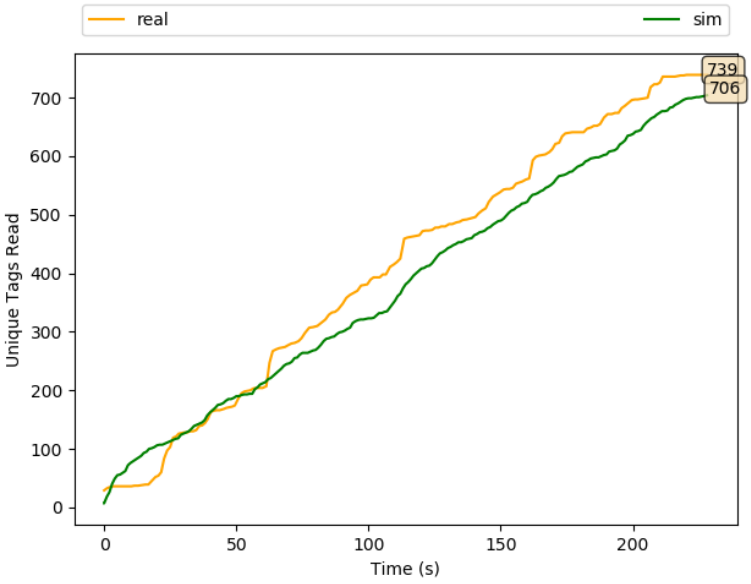


(g) Simulation: Left antenna.

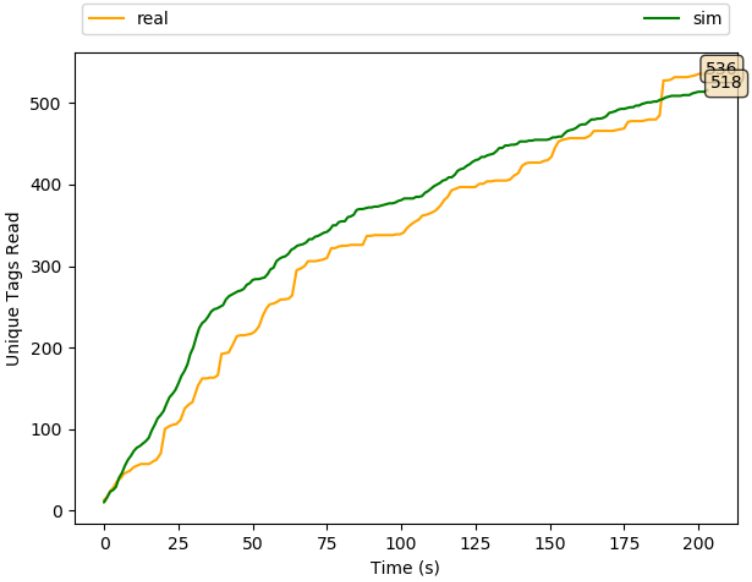


(h) Laboratory: Left antenna.

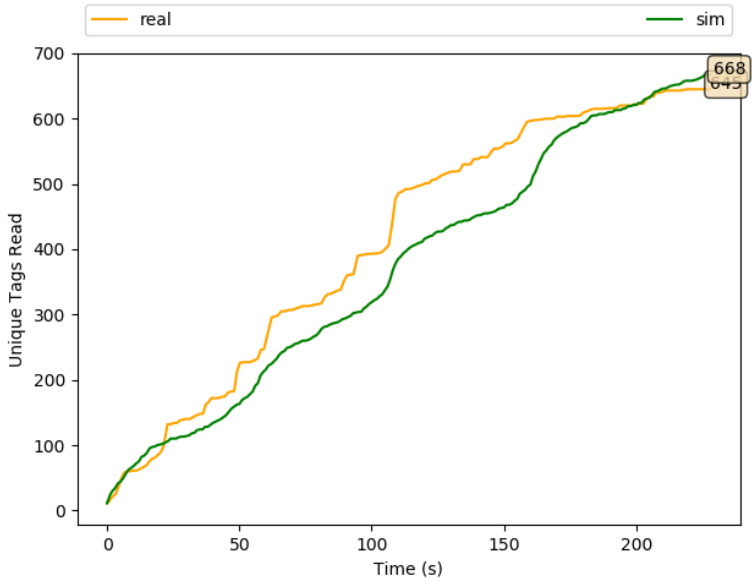
Figure 4.17: Experiment 5: Position of the detected RFID tags in simulation and in the laboratory.



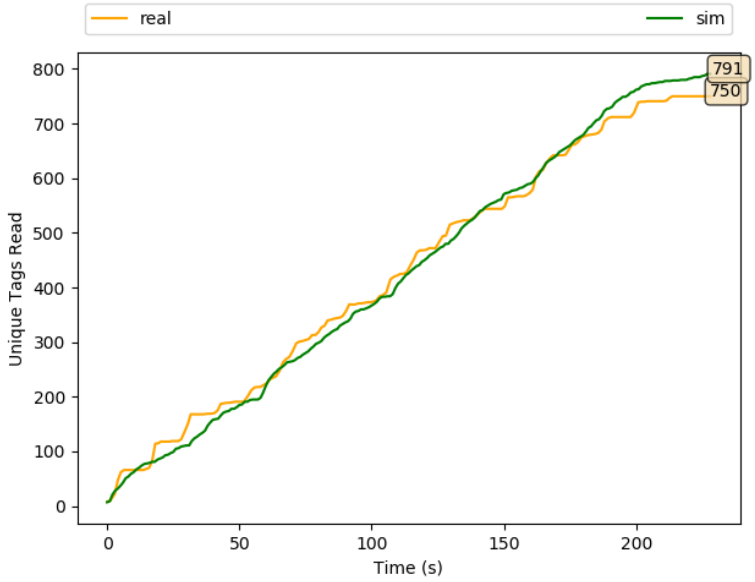
(a) Front Antenna.



(b) Right Antenna.



(c) Back Antenna.



(d) Left Antenna.

Figure 4.18: Experiment 5: Simulation vs. laboratory unique RFID tag readings.

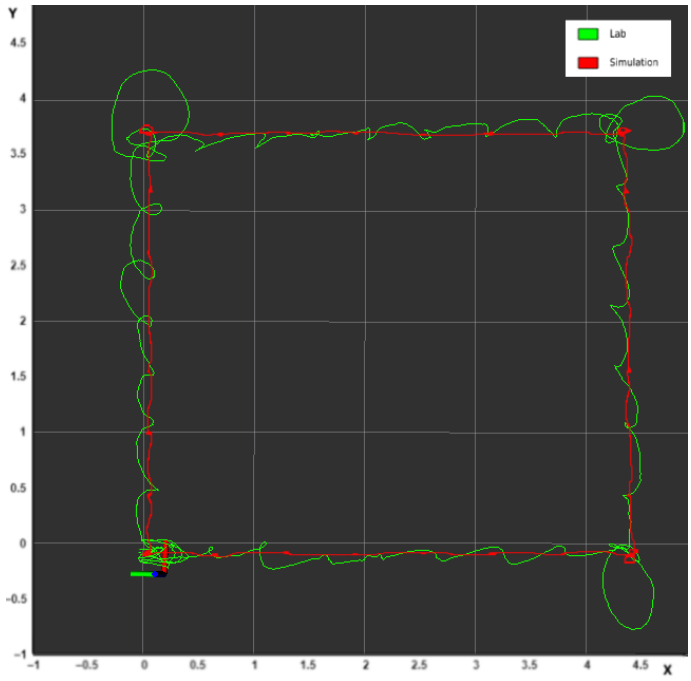
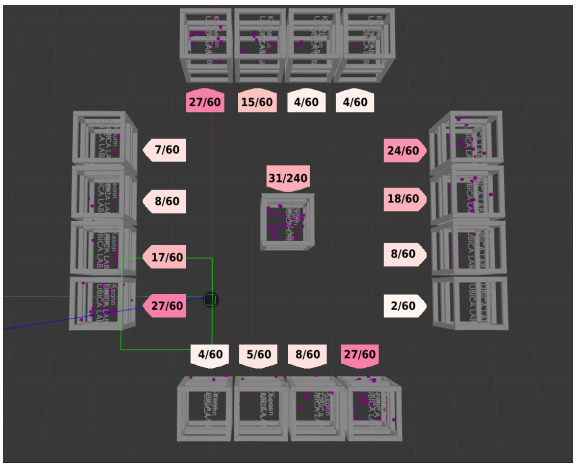


Figure 4.19: Comparing laboratory vs. simulation UAV paths

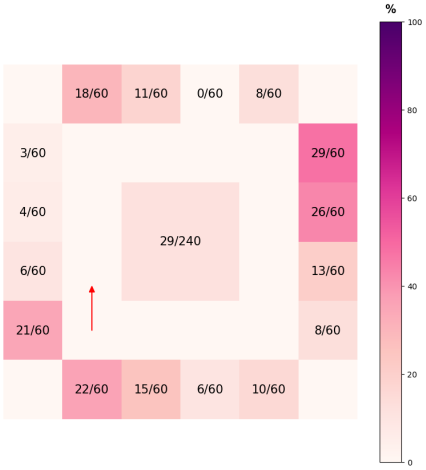
We can notice in Figs. 4.17g and 4.17h, that with only the left antenna active on the UAV, the majority of the detected unique RFID tags were located on the edges of the map environment. On the other hand, in Figs. 4.17c, and 4.17d, having only the right antenna active, we see that most of the detected tags were in the fixtures placed in the middle. Some differences can be observed in particular fixtures for other antennas, this is due to that the UAV navigation in the laboratory is not as smooth as in the simulation. As noted before, the UAV in the laboratory continuously tries to resist external drifting forces caused by turbulence generated by the propellers and localization accuracy, these forces could not be precisely simulated in Gazebo. The comparison of both paths of the UAV in the laboratory and simulation environments can be seen in Fig. 4.19.

Experiment 6. Scenario 3 With the UGV

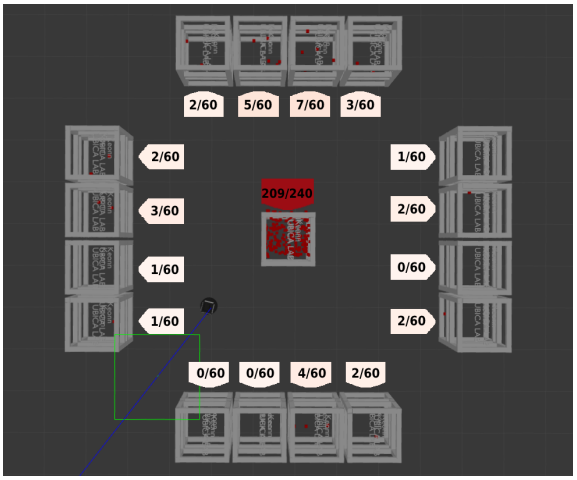
Finally, we repeated Experiment 5 but substituted the UAV with the UGV. The results from Experiment 6, shown in Figs. 4.21a, 4.21b, 4.21c, and 4.21d, show realistic and very good agreement with the exception of the right antenna. Figs. 4.20a, 4.20c, 4.20e, and 4.20g, illustrate the position of the detected RFID tags in Gazebo, while Figs. 4.20b, 4.20d, 4.20f, and 4.20h, show the detected RFID tags in the laboratory, again with realistic and good agreement.



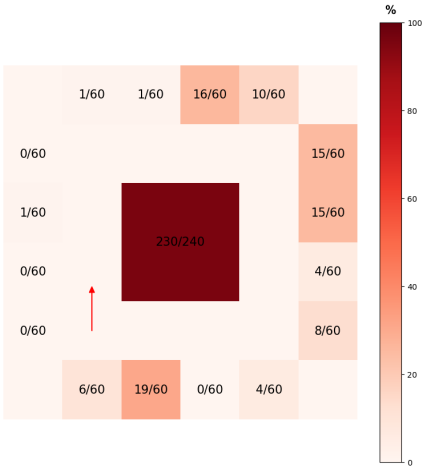
(a) Simulation: Front antenna.



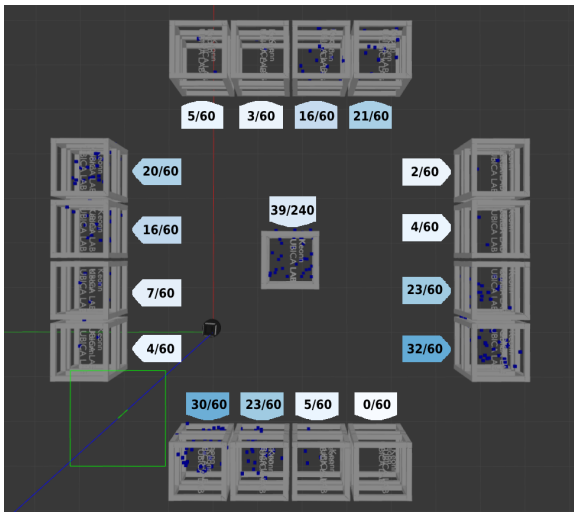
(b) Laboratory: Front antenna.



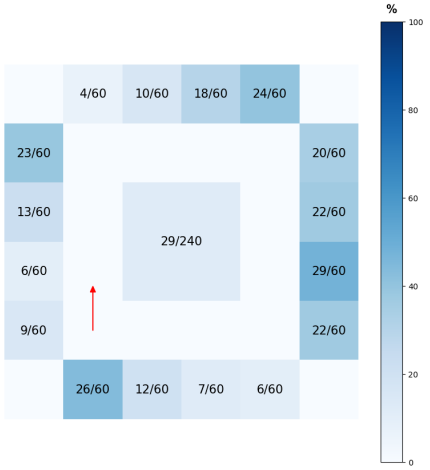
(c) Simulation: Right antenna.



(d) Laboratory: Right antenna.

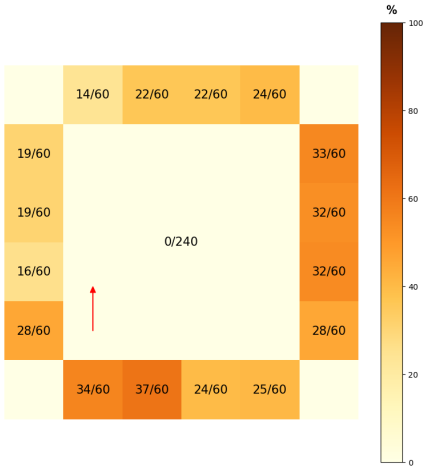


(e) Simulation: Back antenna.



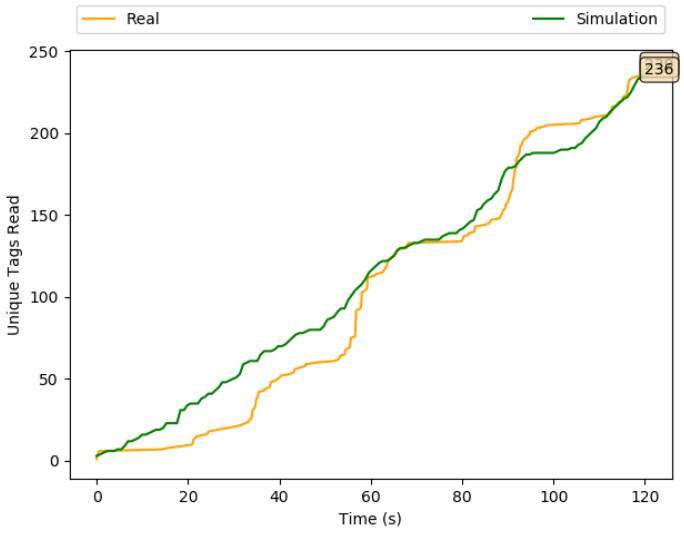
(f) Laboratory: Back antenna.

(g) Simulation: Left antenna.

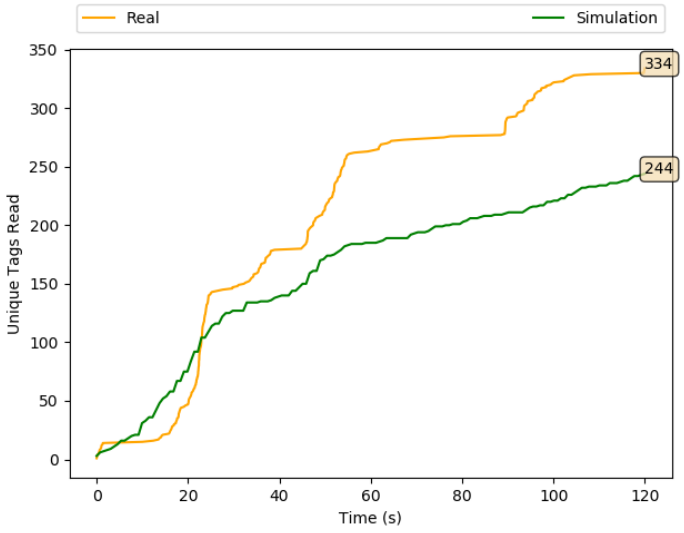


(h) Laboratory: Left antenna.

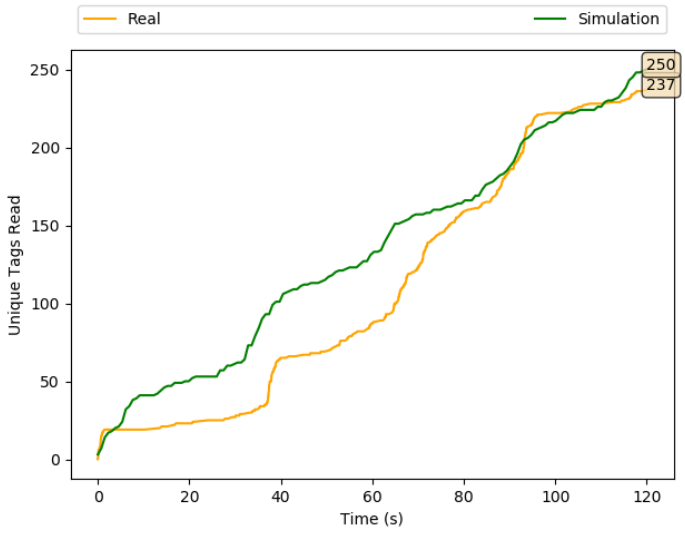
Figure 4.20: Experiment 6: Position of the detected RFID tags in simulation and in the laboratory.



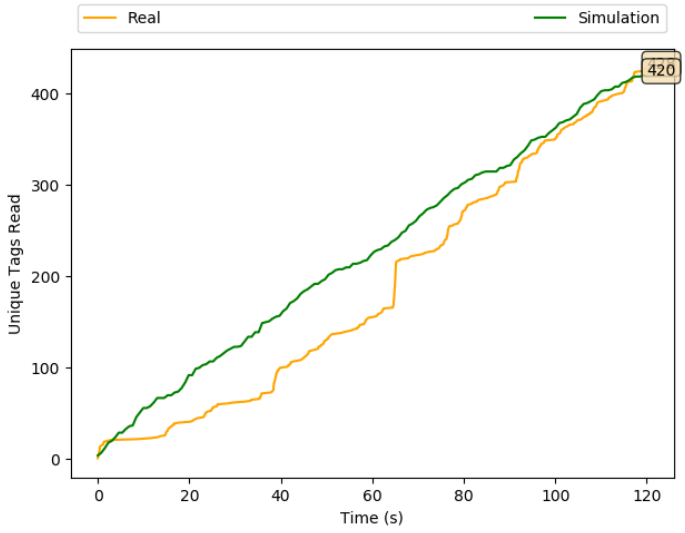
(a) Front Antenna.



(b) Right Antenna.



(c) Back Antenna.



(d) Left Antenna.

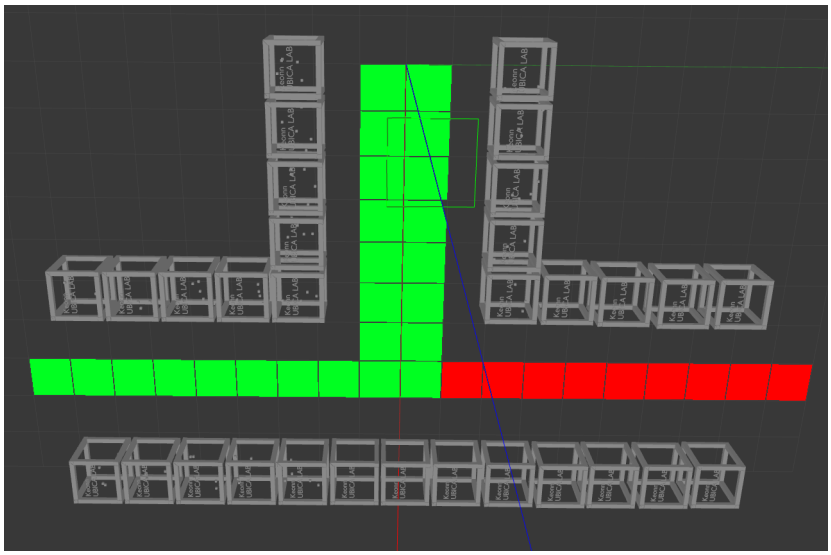
Figure 4.21: Experiment 6: Simulation vs. laboratory unique RFID tag readings.

4.8 An Example of the Use of the Plugin in Robotics Research: Stigmergic Navigation of a UAV

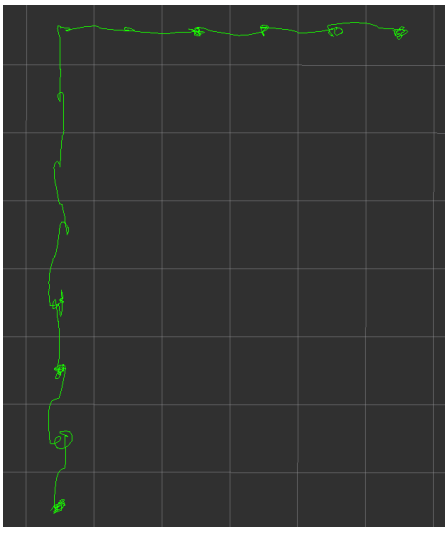
The ability of the RFID system plugin to simulate accurately the behavior of an RFID system payload (reader and antennas) on board an operating mobile robot in

an environment where RFID tags are present, enables it to be used in research that involves RFID and Robotics technologies working together. For this application scenario, we utilize the RFID system plugin to enable a UAV to navigate using stigmergy [B28] to inventory a space using RFID technology. In this scenario, we will have a UAV navigating autonomously through an environment where RFID tags are present, but whose quantity and position are unknown. The used stigmergic navigation technique consists of a robot navigating by choosing the direction in which a higher number of tags are detected for the first time. At every step, the UAV measures the number of new tags read by each of the four antennas and follows the direction of the antenna which reads more unique tags. Research on finding new UAV navigation strategies is expensive in time and money, and the possibility to run simulations to validate the algorithms can speed up the process considerably. But when RFID payloads are used, the simulation is only possible if a simulation tool is available for the RFID system, such as the RFID system plugin that we propose. For this simulation, 300 tags were uniformly placed in a horizontal manner, throughout the T-shaped layout. The simulation and laboratory setups are shown in Figs. 4.22a and 4.22b. Fig. 4.22c shows the path of the UAV in simulation, and Fig. 4.22d shows that the UAV was able to read 283 tags out of 300 overall tags in the environment (94.33%). Robotics researchers interested in using the RFID Gazebo plugin in their research can find it on the ROS wiki page and repository [B21].

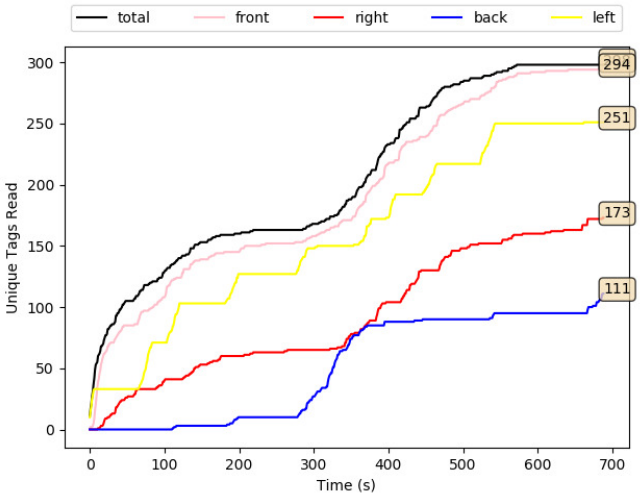
(a) Laboratory setup of the simulation Scenario of a T-shaped map layout.



(b) UAV's chosen path in Gazebo.



(c) UAV's chosen path in rviz.



(d) UAV's total unique tags readings.

Figure 4.22: A simulation of a UAV using a Stigmergic based navigation in a T-Shaped Map Layout.

4.9 Conclusions

In summary, this paper addresses the problem of simulating RFID systems, including readers, antennas, and tags, being used by robots to navigate and perform

other tasks, such as inventorying. The designed solution is composed of an easy to use ROS-Gazebo plugin based on a simple but accurate probability model that requires only 3 parameters from the user, only one of which must be calibrated (R_0) against measured results. The plugin was extensively tested by comparing simulation and experimental results in 3 different scenarios of increasing complexity and density of tags. In each of the three scenarios, an experiment was done with both a UGV and a UAV. In all six experiments, the simulation and experimental results were in enough agreement to use this simulation tool in robotics involving RFID sensors.

The plugin allows to simulate environments before deploying them especially when large quantities of RFID sensors and robots are needed to be used, which saves a lot of time and cost. The proposed plugin through simulation of these environments helps users understand the weaknesses of the designed layout before its deployment in the physical world. Such environments can be warehouses, retail shops „etc. The plugin allows robotics researchers to simulate environments in which robots and RFID technology interact for various applications. In summary, it is a powerful tool for researchers that use RFID technology to improve robots in any sort of manner. However, an extra layer of statistically measuring R_0 by the user is needed for the plugin to work as intended. The entire neglect of the back lobe and the consideration of the tag orientation is also considered a minor limitation for this version. The following section will briefly explain the future improvements to this plugin.

4.10 Future Work

There are several ways in which the accuracy of the Gazebo plugin could be improved:

1. Considering the pattern and orientation of the RFID tag antennas.
2. Using the actual radiation diagram of the reader antennas instead of approximating it based on the beam widths.
3. Automatizing the calibration of R_0 measurement, so that the users of the plugin do not have to define calibration procedures on their own.

These, together with improved documentation and examples of use will be the future work related to this Gazebo plugin.

Chapter-4 References

[B1] Roland Berger Strategy Consultants. New industrial revolution: How europe will succeed. international conference the next industrial revolution manufacturing and society in the xxi century. *Industry 4.0*, 01 2014.

[B2] Mohd Aiman Kamarul Bahrin, Mohd Fauzi Othman, Nor Hayati Nor Azli, and Muhamad Farihin Talib. Industry 4.0: A review on industrial automation and robotic. *Jurnal Teknologi*, 78(6-13), Jun. 2016.

[B3] Xue-Ming Yuan. Impact of industry 4.0 on inventory systems and optimization. In Tamás Bányai and Antonella Petrilloand Fabio De Felice, editors, *Industry 4.0*, chapter 3. IntechOpen, Rijeka, 2020.

[B4] William de Paula Ferreira, Fabiano Armellini, and Luis Antonio De Santa-Eulalia. Simulation in industry 4.0: A state-of-the-art review. *Computers & Industrial Engineering*, 149:106868, 2020.

[B5] Lenka Pitonakova, Manuel Giuliani, Anthony Pipe, and Alan Winfield. Feature and performance comparison of the v-rep, gazebo and argos robot simulators. In Manuel Giuliani, Tareq Assaf, and Maria Elena Giannaccini, editors, *Towards Autonomous Robotic Systems*, pages 357–368, Cham, 2018. Springer International Publishing.

[B6] Protocol buffer for topics, ros. `https://developers.google.com/protocol-buffers/docs/overview`. Accessed: 2021-10-29.

[B7] Ron Mittler. Ros are good. *Trends in plant science*, 22(1):11–19, 2017.

[B8] SDF sdformat base element description. `http://sdformat.org/spec`. Accessed: 2021.

[B9] Urdf format, gazebo ros wiki. `http://wiki.ros.org/urdf`. Accessed: 2021.

[B10] Paolo Mezzanotte, Valentina Palazzi, Federico Alimenti, and Luca Roselli. Innovative rfid sensors for internet of things applications. *IEEE Journal of Microwaves*, 1(1):55–65, 2021.

[B11] Marc Morenza-Cinos, Victor Casamayor-Pujol, and Rafael Pous. Stock visibility for retail using an rfid robot. *International Journal of Physical Distribution & Logistics Management*, 2019.

[B12] Marc Morenza-Cinos, Victor Casamayor-Pujol, Jordi Soler-Busquets, José Luis Sanz, Roberto Guzmán, and Rafael Pous. *Development of an RFID Inventory Robot (AdvanRobot)*, pages 387–417. Springer International Publishing, Cham, 2017.

[B13] Artur Khazetdinov, Andrey Aleksandrov, Aufar Zakiev, Evgeni Magid, and Kuo-Hsien Hsia. Rfid-based warehouse management system prototyping using a heterogeneous team of robots. 08 2020.

[B14] Riccardo Polvara, Manuel Fernandez-Carmona, Gerhard Neumann, and Marc Hanheide. Next-best-sense: A multi-criteria robotic exploration strategy for rfid tags discovery. *IEEE Robotics and Automation Letters*, 5(3):4477–4484, 2020.

[B15] Yifeng Han, Qiang Li, and Hao Min. System modeling and simulation of rfid. *Auto-ID Lab. White Paper*, 01 2005.

[B16] Christian Floerkemeier and Sanjay Sarma. Rfidsim - a physical and logical layer simulation engine for passive rfid. *Automation Science and Engineering, IEEE Transactions on*, 6:33 – 43, 02 2009.

[B17] Daniel Dobkin. *The rf in RFID: uhf RFID in practice*. Newnes, 2012.

[B18] Tiancheng Zhang, Yifang Yin, Dejun Yue, Qian Ma, and Ge Yu. A simulation platform for rfid application deployment supporting multiple scenarios. In *2012 Eighth International Conference on Computational Intelligence and Security*, pages 563–567. IEEE, 2012.

[B19] Salvatore D'Avella, Matteo Unetti, and Paolo Tripicchio. Rfid gazebo-based simulator with rssi and phase signals for uhf tags localization and tracking. *IEEE Access*, 10:22150–22160, 2022.

[B20] Chen Chi, T. M. a frequency hopping method for spatial rfid/wifi/bluetooth scheduling in agricultural iot. *Wireless Netw*, 25:805–817, 2019.

[B21] RFID Plugin abdussalam-alajami, rafael-pous, and guillem-moreno. rfid-sensor gazebo plugin description and repository in ros wiki. http://wiki.ros.org/RFIDsensor_Gazebo_plugin. Accessed: 2022.

[B22] Ros transforms and frames frames and transform description. http://sdformat.org/spec. Accessed: 2020.

[B23] Victor Casamayor-Pujol, Marc Morenza-Cinos, Bernat Gastón, and Rafael Pous. Autonomous stock counting based on a stigmergic algorithm for multi-robot systems. *Computers in Industry*, 122:103259, 2020.

[B24] Keonn's AdvanReader 160 keonn's advanreader 160. https://keonn.com/components-product/advanreader-160/. Accessed: 2022.

[B25] Keonn's AdvantennaSP11 keonn's advantennasp11. https://keonn.com/components-product/advantenna-sp11/. Accessed: 2022.

[B26] Eric Sholes. Evolution of a uav autonomy classification taxonomy. In *2007 IEEE Aerospace Conference*, pages 1–16. IEEE, 2007.

[B27] Abdussalam AA Alajmi, Alexandru Vulpe, and Octavian Fratu. Uavs for wi-fi receiver mapping and packet sniffing with antenna radiation pattern diversity. *Wireless Personal Communications*, 92(1):297–313, 2017.

[B28] Abdussalam A. Alajami, Guillem Moreno, and Rafael Pous. Design of a uav for autonomous rfid-based dynamic inventories using stigmergy for mapless indoor environments. *Drones*, 6(8), 2022.

4.11 Overall Conclusion and Future Work

The study presented in Chapter 3, exposes the benefits that a UAV has on performing autonomous inventory missions in indoor environments. We can clearly conclude the advantages that a UAV has over a UGV for such tasks. With the increasing requirements of UAVs performing inventory tasks for autonomy, intelligence, and multitasking, the efficiency and intelligence level of UAV single-machine operations are struggling to meet the requirements of such task applications. For example, when flying alone, the limited energy supply limits the flight distance and operation range. Table 4.1, exposes the differences between both types of robots comparing different aspects that affect the performance of performing an inventory task.

Feature	UAV	UGV
Task time capacity	up-to 1h	less than 10min
Payload size and weight capacity	+100kgs	-10kgs
Navigation	3D plane	2D plane
maneuverability and agility	superior	poor
Processing power limitation	limited	Not limited
Hardware Complexity	Complex	Simple
Software Complexity	Complex	Simple
Experimental Costs	very expensive	cheap
Security	Not secure	Very secure

Table 4.1: A Comparison Table between UGVs and UAVs illustrating differences in different aspects.

From Table 4.1, we note that due to the fact that a UGV is always placed on the ground plane, it is able to carry relatively heavy payloads compared to a UAV. A UAV is required to generate a force greater than 2 times the force of gravity applied on the payload and body mass. The other advantage of being able to carry a large size and heavy payload is the possibility of carrying large capacity power sources, which can provide sufficient energy to all parts of the robot for a long amount of time. Powerful processing units and high level sensors can be mounted on the UGV due to the ability to provide the energy required for these devices. The UAV however, can only carry a limited payload which means less power capacity sources and less functional time. However, we proved in Chapter 3, that using a UAV has many advantages in comparison with using a UGV for inventory missions, such as the ability to take inventory of items that are located on high

shelves, its mobility/locomotion is not surface dependent as UGVs, its ability to use the 3D plane for maneuverability and obstacle avoidance, the ability to create 3D maps, and the ability to explore the environment by exploiting the vertical plane. From the summary above, we concluded a hypothesis of the possibility of increasing the performance of taking an inventory, if both robots can collaborate to exploit the advantages of the heterogeneity feature in a team of robots. Therefore, a strategy that contributes to the collaboration feature between distributed heterogeneous robots for increasing their overall performance is presented and explained in Chapter 5.

Chapter 5

A ROS-BASED DISTRIBUTED MULTI-ROBOT LOCALIZATION AND ORIENTATION STRATEGY FOR HETEROGENEOUS ROBOTS

5.1 Abstract

The problem of estimating and tracking the location and orientation of a mobile robot by another in heterogeneous distributed multi-robots is studied in this paper. We propose a distributed multi-robot localization strategy (DMLS) that is Robotic Operating System (ROS) based. It consists of an algorithm that fuses data of diverse sensors from 2 heterogeneous robots that are not connected within their transform trees to localize and measure the relative position and orientation. The method exploits the robust detection of the Convolutional Neural Networks (CNN) and the accurate relative position measurements from the local costmap. The algorithm is composed of two parts: The localization part and the relative orientation measurement part. Localization is done by optimization and alignment calibration of the CNN output with the costmap in an individual robot. The relative orientation measurement is done by a collaborative multi-robot fusing diverse sensor data to align and synchronize the transform frames of both robots in their costmaps. To illustrate the performance of this strategy, the proposed method is compared with a conventional object localization and orientation measuring method that uses computer vision and QR codes. The results show that this proposed method is robust and accurate while maintaining a degree of simplicity

and efficiency in costs. The paper also presents various application experiments in the laboratory and simulation environments. By using the proposed method, distributed multi-robots collaborate to achieve collective intelligence from individuals, which increases team performance.

Keywords: *Multi-robots, Localization, Collaboration, Exploration*

5.2 Introduction

Localization is one of the main requirements for the autonomy of a mobile robot [C1]. In order to navigate autonomously in their work space, mobile robots must be able to localize themselves, indoors and outdoors. To successfully perform the tasks required of them, mobile robots need to know their exact position and orientation (pose). There have been numerous approaches to the localization problem for a single robot utilizing different types of sensors and techniques. In [C2], authors use a technique that combines position estimation from odometry with observations of the environment from a mobile camera for achieving accurate self localization. In [C3], authors present a Bayesian estimation and the Kalman filter for object tracking. In [C4], authors develop two algorithms to register a range scan to a previous scan in order to compute relative robot positions in an unknown environment.

The development of multiple robot systems that solve complex and dynamic problems in parallel is one of the key issues in robotics research. The multiple robot system, in comparison with the single robot system, has the advantage of collecting and integrating multiple sensor data from different robots for different purposes and applications. For this reason, many robotic applications require that robots are able to collaborate with each other within a team in order to perform a certain task [C5]. An interesting application for multiple robot' collaboration in [C6], where authors aim to maintain an accurate and close to real time inventory of items using Radio Frequency Identification technology (RFID), which is considered crucial for an efficient Supply Chain Management (SCM). They first define the problem of stock counting and then a solution based on a multi-robot system is proposed.

However, in order for robots to increase their collaboration performance, they will need to be able to localize each other in their own maps, which opens up a wide degree of new applications such as map sharing for exploratory robots in [C7], or map building and cooperative collaboration in [C8], and other [C9]. However, this is not a straight-forward process if they do not share a common map or no prior information is given to the robots.

The advantages that derive from the exchange of information among the mem-

bers of a team are more crucial in the case of heterogeneous team robots. A team of robots that is composed of different types or platforms carrying different proprioceptive and exteroceptive sensors, have different capabilities for sensing the environment and self-localization. The exploitation of the heterogeneity feature in distributed multi-robots allows researchers to find different strategies for cooperative multi-robot localization.

This paper proposes a Robotic operating system (ROS) based multi-robot localization and relative orientation measurement strategy for distributed heterogeneous robots. The proposed strategy or method helps increase the efficiency of the collaboration between heterogeneous multi-robots in a team by addressing the problem of accurately detecting and localizing multi-robots in unknown and unexplored environments autonomously and through cooperation. The proposed method exploits the capability of the heterogeneity feature that exists in a group of robots. It fuses data of diverse sensors from 2 heterogeneous robots that are not connected within their transform trees to mutually localize and measure their relative orientation. The method exploits the robust detection and tracking of a Convolutional Neural Networks (CNNs) and the accurate relative distance measurements from a local costmap. The exact localization and orientation measurement is done when two members of a robot team are in close range from each other, this is done by utilizing a stereo camera that exists on only one member of the team, and the 2D local cost-maps from both robots that are constructed using precise 2D-lidars.

5.3 Similar Work

Previous similar work has been done considering collaborative strategies for multi-robot localization. A number of authors have considered pragmatic multi-robot map-making. Some approaches use beacon type sensors. For example, in article [C10], authors propose an algorithm for model-based localization that relies on the concept of a geometric beacon. The algorithm is based on an extended Kalman filter that utilizes matches between observed geometric beacons and a priory map of beacon locations. The fact that a prior reference map was given and shared between the robots, meant that the robots transform trees were somehow connected. This made it relatively straightforward to transform observations from a given position to the frame of reference of the other observers. The algorithm exploited structural relationships in the data. In other work [C11], Rekleitis, Dudek, and Milios have demonstrated the utility of introducing a second robot to aid in the tracking of the exploratory robot's position and they introduced the concept of cooperative localization. Their approach is based on using pairs of robots that observe each other's behavior, acting in concert to reduce odometry errors. Their

approach improves the quality of the map by reducing the inaccuracies that occur over time from dead reckoning errors. However, prior knowledge of their initial location was also known, which reduced the autonomy of the system.
Vision-based systems have been widely used in robotic perception sensing. They are also useful for distinguishing an individual robot for handling the case of more than two robots. In the past few years, several authors have considered localizing team members of a group of robots using each other. They used various types of sensors for this objective including vision-based sensors. An example in [C12], proposes an algorithm that allows robots to identify and localize each other. The method requires all robots to be equipped with omnidirectional mono cameras. Jennings et al. in [C13], used stereo vision to build a grid map and localize robots by features in grid map such as corners. Their key idea is to find corners in a grid map and compare these corners with a-priory landmarks at known positions. They needed an exact reference of the map for their method to work, which makes it not useful for exploration based robots. Fox et al. proposed a multi-robot Monte Carlo Localization (MCL) approach in [C14], to localization with improved accuracy of MCL. Their research shows accurate localization results and can be used for multi-robot localization. However, it requires the transmission of a relatively large amount of information. Pereira et al. in [C15], proposed an approach to localize and track objects exploiting statistical operators and a simple graph searching algorithm. In [C16], distributed sensing was developed based on Kalman filtering to improve the target localization. However, these methods are only designed to recognize and localize a specific entity or target. In a multi-robot scenario where robot members collaborate, it is preferable and logical for each robot to localize and track all of its teammates, also important features in the environment.
Researchers have also used CNNs for high performance features and robot identification and tracking. In [C17], authors using CNNs present a robust multi-robot convoying approach that relies on visual detection of the leading agent, thus enabling target following in unstructured 3-D environments. However, although this method detects the other robot with high performance and measures the relative motions of that robot, it does not precisely localize the other robot nor measure its relative orientation.

The distributed multi-robot localization strategy (DMLS) strategy introduced in this paper, is designed to operate with the sensors or hardware that normally exists in an autonomous mobile robot that is able to construct a local costmap of its surrounding environment. However, within a team of two robots, one of the robots would require to be equipped with a camera. The camera can be the same proximity sensor (e.g., RGBD camera) or the self localization tracking camera that contributes to constructing a local costmap in the robot.
The DMLS strategy, exploits the strengths of the diverse sensors that exist in the

heterogeneous team robots. It Utilizes the robust detection and tracking in CNNs and the accurate proximity measurements extracted from a 2D local costmap that is constructed using a 2D lidar for the accurate localization and relative orientation estimations between the robots.

In section 5.6, the proposed localization method will be compared with a vision-based conventional QR-pose estimation method. The QR-pose estimation method was used due to its low computation and flexibility to operate on limited machines such as Unmanned Aerial Vehicles (UAVs) and simple distributed Unmanned Ground Vehicles (UGVs). It also operates in decentralized systems where both robots do not share prior information or a map. Finally, the fact that makes it is able to estimate position and relative orientation at the same time makes it a good comparison reference to be used. Some application scenarios in section 5.7 illustrate the contribution of the proposed method using a laboratory environment and a ROS-based simulation platform (Gazebo).

5.4 System Overview

For the experiments executed to implement and test the DMLS method introduced in this paper, two independent heterogeneous robots are used, called *Detector* and *Pawn*. These robots are designed to be able to cooperate on a shared task. The robots are able to communicate using a common network.

For the experiments run in the lab, the *Detector* robot will be a small sized UAV. The *Detector*'s hardware block diagram is shown in Fig. 5.1.
The UAV was designed to be able to execute stable navigation in indoor spaces therefore, a 6-motor UAV (hexacopter) frame was chosen. The frame design layout enables a stable flight in indoor spaces for a task-sufficient amount of time. An efficient open-source autopilot (Pixhawk 2.4.8) is used. A compatible companion computer (Jetson nano) is used as the main processing unit for the UAV. The UAV also contains 6 brush-less motors (750kv), 6 propellers (16"x4" size), and 6 electronic speed controllers (ESCs 40A of current discharge capacity). The ESCs main objective is to translate the signal received from the autopilot to energy from the energy source and supply it to the motors. The energy source used on the UAV is a lithium polymer battery (LIPO, 16V, 6cell), which capacity is 6000mAh, an operating voltage is 16.6V, and can supply a high current discharge rate.

Since the UAV was intended to operate in indoor spaces (GPS-denied spaces), a Visual Simultaneous Localization and Mapping (VI-SLAM) based camera was added to the system to supply the UAV with self localization messages. The VI-

SLAM camera used, has a built-in diverse suite of sensors which all feed into a VI-SLAM pipeline, which fuses them into a 6 DOF estimation of position and velocity of the camera relative to the environment at 200 Hz, therefore, precisely tracking and self-localize the UAV. Special adaptation was made to integrate these messages to the autopilot, resulting in an indoors guidance system for the UAV. For the Experiments, the *Pawn* robot is a UGV and its hardware block diagram is shown in Fig. 5.2.

The pre-designed and patent UGV in [C6] is used. The UGV is designed to autonomously navigate within the given path in the intended map layout. In order for the robots to be able to run the proposed localization method and successfully localize each other, the *Detector* must have a camera mounted for detection. This is not a requirement for the *Pawn*. However, sufficient sensors that enable the construction of a local cost-map will be necessary for both the *Detector* and the *Pawn*. For the laboratory Experiments, a UAV in [C18], has been chosen to be the *Detector* for the manifest advantage of utilizing 3D space. This increases the possibility of detection while hovering or flying. However, the UAV must land, or be in the same 2D plane as the *Pawn* after detection. This step is necessary for completing the sequential process of the proposed localization method, DMLS.

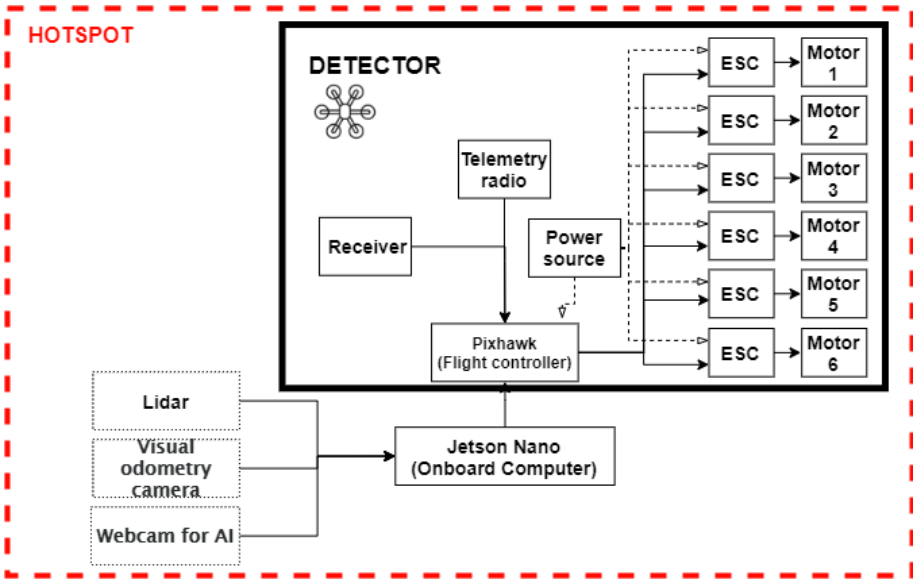


Figure 5.1: Hardware block diagram of the *Detector*.

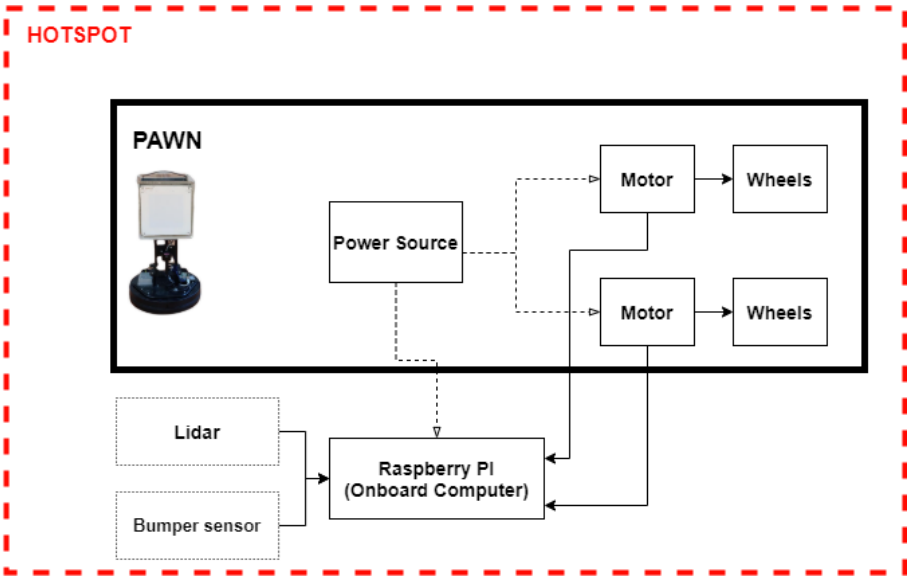


Figure 5.2: Hardware block diagram of the *Pawn*.

5.5 DMLS Framework

The proposed localization method consists of four parts.

5.5.1 First Part: Detecting and reducing the region of search (RoS) using CNN

Throughout a mission, the *Detector* would first detect the *Pawn* as soon as it is in the detection range of its camera and in line of sight (LoS). This is done by using a high-performance, cost-effective, and low-computation CNN model called "MobileNetV2" [C19]. The MobileNetV2 model developed by Google is designed to dramatically reduce the complexity cost and model size of the network. It is specially optimized to operate on mobile devices. The architecture delivers high accuracy results while keeping the parameters and mathematical operations as low as possible. This makes it adequate for resource limited mobile robots. The objective of this CNN is to identify and classify the class/object of the detected robot from a list of pre-trained classes/objects.

For our experiments, we trained the model with a dataset that is composed of around 2000 high resolution images per class/object that are required to be detected. The images for the dataset are taken from different angles, distances, and illumination environments to assure the accurate detection of the class/object at

any condition. Transfer learning was used on pre-trained existing models supported by the API. A total of 91 epochs (around 20h of training time), were used to train the model with 6 classes/objects which include the *Pawn* and some important features of the laboratory environment. A boundary box will be plotted on the object with the most likelihood (MLH) estimation to be the *Pawn*. The pixel positions of the boundary box in the x-axis and y-axis of the entire pixel-space PS of the image frame are provided by the CNN. The minimum and maximum pixel position values on the horizontal axis (PS_H) of the boundary box will be recorded as $xpix_{min}$ and $xpix_{max}$ as only the x-axis values will be used in the calculations of the RoS.

5.5.2 Second Part: Camera frame and cost-maps calibration and locating

The constructed local costmap of the robot is done by a ROS [C20] package, called Move_base [C21]. This is achieved using proximity sensors mounted on the robot. Move_base is a simple algorithm that constructs a matrix C, which represents a local cost-map. The element c_{ij} represents a cost function in every cell of the cost-map, or so-called cell-cost [C22]. This value represents the possibility of having an obstacle within that cell.

However, due to the fact that the size of the local costmap of a robot differs from one robot to another depending on the sensors type, detection range, and local costmap parameters configurations, this implies that there will be a misalignment of the size of the local costmap (width and height) with the horizontal field of view (HFOV) of a camera frame. In other words, the local costmap might include more obstacles in the environment than what the camera frame is able to visualize as shown in Fig. 5.3. As a result, a calibration/alignment step is needed to measure the effective local costmap that relates to the camera HFOV, which is represented as the pink highlighted area in Fig. 5.3.

Most local costmaps are usually bigger in size than the HFOV of a camera frame (if not, it can be set so by optimizing the local costmap size parameter). The effective local costmap will be a submatrix of the overall matrix that represents the entire local costmap. The elements $C\prime$ and $c\prime_{ij}$, represent the submatrix and the cost function in every cell of the effective cost-map.

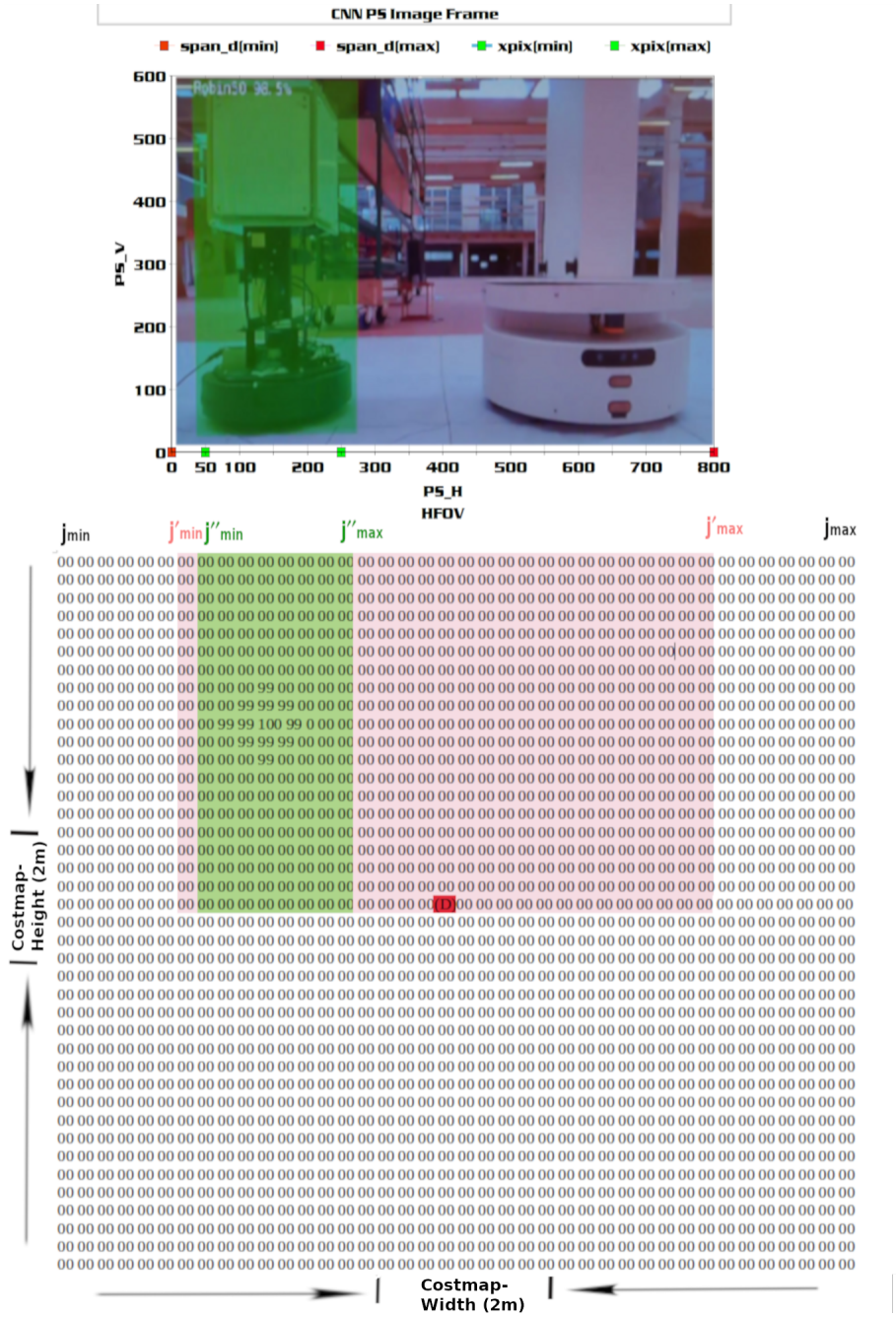


Figure 5.3: Pixel to Cost map RoS translation.

The calibration step consists of the measurement of the boundaries of this sub-matrix. By placing the robot in an empty free of obstacles position in space, and at a set distance $0 < d < costmap - height/2$ from the working distance [C23]

in-front of the robot and camera, an object is introduced so it can be visible at the edge of the left side of the image frame. From the local costmap a cell with a high cost value $c_{ij} = 100$ will be created, the column position of this cell will then be recorded $j\prime_{min}$ representing the left boundary or the starting column position of the submatrix. The same process will be repeated but this time introducing an object from the right side of the image frame to measure $j\prime_{max}$, which represents the column position of the right boundary of the submatrix marking the last column position of the submatrix. The submatrix representing the effective local costmap that matches the size of the camera frame span will only be considered as the region of search (RoS). In Fig. 5.3, an example that visualizes the submatrix and is highlighted in pink can be seen. The submatrix as shown is aligned with the image frame above.

From knowing $j\prime_{min}$ and $j\prime_{max}$, the width of the submatrix which is also equal to the HFOV span in meters represented as $span_d$ is calculated. Given the map resolution, $resol$ (m / cell), of the local costmap by the user that is used as an input parameter to construct the local costmap, it is possible to calculate the minimum span limit $span_{dmin}$, and maximum span limit $span_{dmax}$ in meters shown in Eqs. 5.1 and 5.2.

$$span_{dmin} = j\prime_{min} * resol \tag{5.1}$$

$$span_{dmax} = j\prime_{max} * resol \tag{5.2}$$

$$span_d = j\prime_{max} + j\prime_{min} \tag{5.3}$$

The RoS column range RoS_{CR} can be calculated in Eq. 5.4.

$$RoS_{CR} = j\prime_{max} - j\prime_{min} \tag{5.4}$$

Which represents the column size of the submatrix.

To align the horizontal pixels of the HFOV to the submatrix, in Eq. 5.5 the algorithm calculates the NPC, which represents how many pixels of the PS_H axis are associated per cell of the submatrix (pixels/cell).

$$NPC = \frac{PS\_H}{RoS_{CR}} \tag{5.5}$$

Using $xpix_{min}$ and $xpix_{max}$ and knowing that the robot is at the center of its local costmap $c_{\frac{imax}{2}\frac{jmax}{2}}$, which is marked as the red square in the middle of the matrix in Fig. 5.3, the algorithm calculates $j\prime_{min}$ and $j\prime_{max}$, which represents the left and right column position of the sub-submatrix as shown in Eqs. 5.6 and 5.7.

$$j\prime\prime_{min} = \frac{xpix_{min}}{NPC} + j\prime_{min} \tag{5.6}$$

$$j''_{max} = \frac{xpix_{max}}{NPC} + j'_{min} \tag{5.7}$$

This sub-submatrix C'', that contains the cells $C''_{ij''}$, represents the boundary box position in the effective costmap in the entire local costmap.

Fig. 5.3 shows an example of having a 2m x 2m local costmap C with elements $c[40][40]$. The upper section of the image shows the PS (PS_H = 800 pixels, PS_V = 600 pixels), which represents the output image frame from the CNN. The left and right boundaries of the detected boundary box are $xpix_{min}$ = 50 and $xpix_{max}$ = 250 as shown in the image. The entire local costmap C can be written as Eq. 5.8.

$$C = \begin{bmatrix} c_{00} & c_{01} & c_{02} & \dots & c_{0j_{max}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{i_{max}0} & c_{i_{max}1} & c_{i_{max}2} & \dots & c_{i_{max}j_{max}} \end{bmatrix} \tag{5.8}$$

Where i_{max} and j_{max} = 40. The *Detector* however, is at the position of $i = max/2$ and $j = max/2$. The range of the RoS for the rows is from 0 to $i = max/2$. Applying the equations above, it is possible to visualize the effective costmap in Fig. 5.3 as all the elements of the matrix colored in pink and green, which can be written as the submatrix c' in Eq. 5.9.

$$C' = \begin{bmatrix} c'_{0j'_{min}} & c'_{0(j'_{min}+1)} & \dots & c'_{0j'_{max}} \\ \vdots & \vdots & \ddots & \vdots \\ c'_{i\frac{max}{2}j'_{min}} & c'_{i\frac{max}{2}(j'_{min}+1)} & \dots & c'_{i\frac{max}{2}j'_{max}} \end{bmatrix} \tag{5.9}$$

The boundary box RoS which is shown in Fig 5.3 as all the elements of the array colored in green, which can be written as the sub-submatrix c'' in Eq 5.10.

$$C'' = \begin{bmatrix} c''_{0j''_{min}} & c''_{0(j''_{min}+1)} & \dots & c''_{0j''_{max}} \\ \vdots & \vdots & \ddots & \vdots \\ c''_{i\frac{max}{2}j''_{min}} & c''_{i\frac{max}{2}(j''_{min}+1)} & \dots & c''_{i\frac{max}{2}j''_{max}} \end{bmatrix} \tag{5.10}$$

The cell $c''_{ij''}$ with a cost value of 100 will represent the *Pawn* location in the *Detector* frame. The coordinates of the Pawn, therefore, can be directly extracted and recorded.

5.5.3 Third Part: Handshake

Up to this stage, only the coordinates of the *Pawn* in the *Detector* map-frame alone are known. However, to able the robots to collaborate and share important location information as we will discuss in the Section 5.7, the orientation of the

Pawn MUST be known. The relative orientation of both robots cannot be directly extracted using ROS transforms (TF packages [C24]). This is due to that the transform trees or the frames of both robots are not connected. More input is needed to compute the relative pose orientation of the *Pawn*. Therefore, a communication handshake between the *Detector* and the *Pawn* will take place. This handshake is done through a simple ROS service, "peer-to-peer communication", where an exchange of information is done between the robots. Since the coordinates of the *Pawn* $\vec{r}_P = (x_P, y_P)$ in the *Detector* map-frame are known, the distance between the *Detector* and the *Pawn* can be calculated using Eq. 5.11.

$$d = \|\vec{r}_P - \vec{r}'_D\| \tag{5.11}$$

where $\vec{r}'_D = (x'_D, y'_D)$ are the coordinates of the *Detector* in its own map-frame. This calculated distance will be then sent using the ROS service to the *Pawn*. In return, requesting an exchange of information from the *Pawn*. This information data will contain its own coordinates, hence, the *Detector*'s coordinates in the *Pawn* map-frame. Given the distance between the *Detector* and the *Pawn*, the algorithm within the *Pawn*, will search in its map-frame for a cell with a cost factor $c_{ij} = 100$, which represents an obstacle within approximately a distance d from itself. The coordinates of that obstacle will be obtained. At this point, the coordinates of that obstacle/object having the same footprint of the *Pawn*, represent the coordinates of the *Detector* in the *Pawn*'s map-frame. Since both robots are not connected within their frame tree in ROS, meaning they are independent, these coordinates must be translated and cannot be used directly. In subsection 5.5.4, the explication of how the *Pawn* pose orientation and the frame-transforms are calculated will be presented.

5.5.4 Fourth Part: Processing and Localizing

Once the *Detector* obtains the coordinates of the *Pawn* in the *Detector*'s frame, and the *Detector*'s coordinates in the *Pawn*'s frame, the algorithm, using rotation matrices as shown in the following equations, will be used to determine the relative pose. In ROS language, it would be the transform between the two frames, "$odom - pawn$" and "$odom - detector$". Using Rviz [C25], the illustration of the *Pawn* pose in x, y is shown, and the orientation in the z axis (yaw) in the local cost map. All experiments were conducted on a 3x3 meter scaled grid with 0.5-meter square-divisions as a ground truth in the laboratory and in Rviz. To further explain the algorithm that computes the relative pose we assume that:

$\vec{r}_D = (x_D, y_D)$ are the coordinates of the *Detector* with respect to the *Pawn*, (in the *Pawn*'s map-frame).

$\vec{r}_P = (x_P, y_P)$ are the coordinates of the *Pawn* with respect to the *Detector*, (in the *Detector*'s map-frame).

$\vec{r}_{D0} = (x_{D0}, y_{D0})$, are the coordinates of the *Detector* with respect to the *Pawn* having rotating the *Pawn* to equal orientation of the *Detector*, (in the *Pawn* robot map-frame). Considering the above declarations, Eq. 5.12 can be concluded.

$$\vec{r}_{D_0} = -\vec{r}_P \tag{5.12}$$

Having applied the rotation matrix formula between x_D, y_D and x_{D_0}, y_{D_0} to compute ϕ, which is the relative angle between the *Pawn* and the *Detector*, shown in Eq. 5.15. In Fig. 5.4, the illustration of the corresponding parameters and the frame transformation relation between the *Detector* and the *Pawn* can be seen.

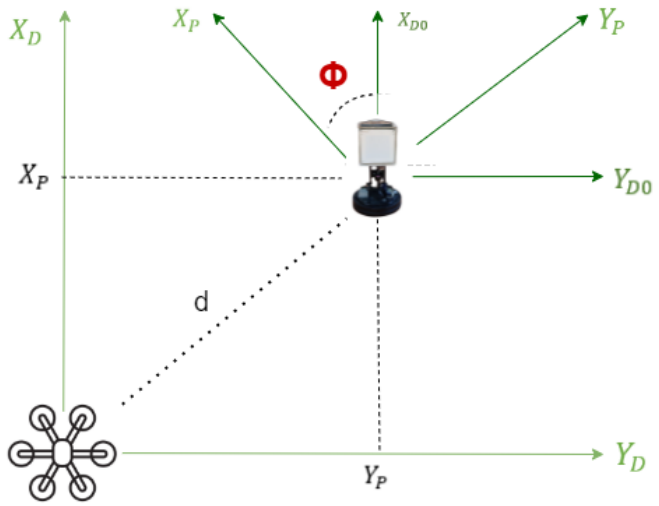


Figure 5.4: The *Detector* and the *Pawn* frame-transformation relation.

$$\begin{bmatrix} x_D \\ y_D \end{bmatrix} = \begin{bmatrix} \cos\phi & \sin\phi \\ -\sin\phi & \cos\phi \end{bmatrix} \cdot \begin{bmatrix} x_{D_0} \\ y_{D_0} \end{bmatrix} \tag{5.13}$$

Which yields to Eq.5.14,

$$\begin{bmatrix} x_D \\ y_D \end{bmatrix} = \begin{bmatrix} x_{D_0} & y_{D_0} \\ y_{D_0} & -x_{D_0} \end{bmatrix} \cdot \begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} \tag{5.14}$$

and Eq.5.15.

$$\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} = \begin{bmatrix} x_{D_0} & y_{D_0} \\ y_{D_0} & -x_{D_0} \end{bmatrix}^{-1} \cdot \begin{bmatrix} x_D \\ y_D \end{bmatrix} \tag{5.15}$$

Resulting in Eq. 5.16.

$$\begin{bmatrix} \cos\phi \\ \sin\phi \end{bmatrix} = \frac{1}{x_{D_0}^2 + y_{D_0}^2} \cdot \begin{bmatrix} x_{D_0} \cdot x_D + y_{D_0} \cdot y_D \\ \\ y_{D_0} \cdot x_D - x_{D_0} \cdot y_D \end{bmatrix} \tag{5.16}$$

The angle ϕ can be extracted from the following equations Eqs. 5.17 and 5.18.

$$\phi = \pm \arccos \left[\frac{x_{D_0} \cdot x_D + y_{D_0} \cdot y_D}{x_{D_0}^2 + y_{D_0}^2} \right] \tag{5.17}$$

By using Eq. 5.17 two different angles are extracted, in order to verify the correct one which represents which rotation direction, ϕ is to be calculated using Eqs. 5.18 and find the matching angles.

$$\phi = \pm \arcsin \left[\frac{y_{D_0} \cdot x_D - x_{D_0} \cdot y_D}{x_{D_0}^2 + y_{D_0}^2} \right] \tag{5.18}$$

A block diagram showing the workflow of the DMLS, and the full handshake between the *Detector* and the *Pawn*, is shown in Fig. 5.5.

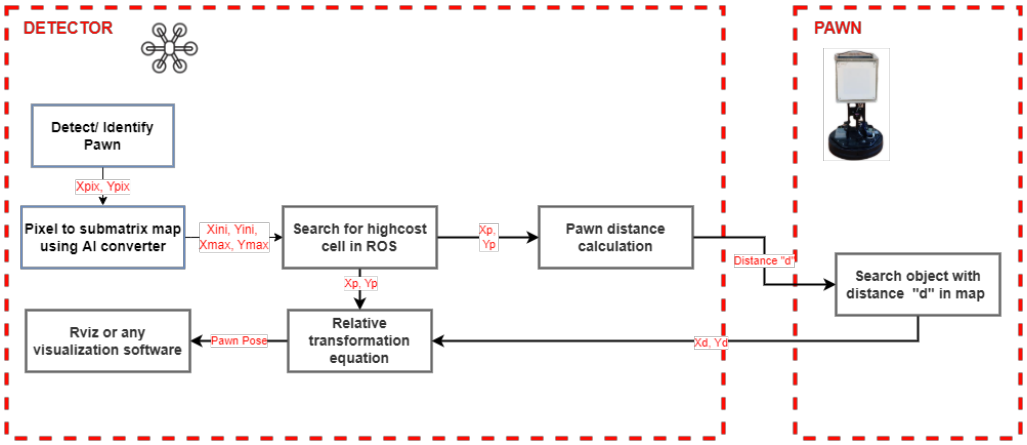


Figure 5.5: DMLS method Workflow and Block diagram.

5.6 Comparison with QR-code pose estimation method

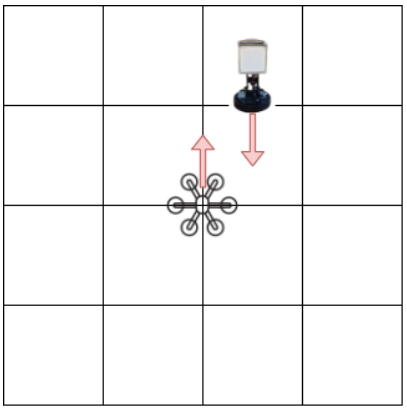
5.6.1 Scenario 1: Pose estimation at $d = 1.00m$

In order to validate the performance of our localization method, the results are compared with the ones from using the conventional QR-code pose estimation

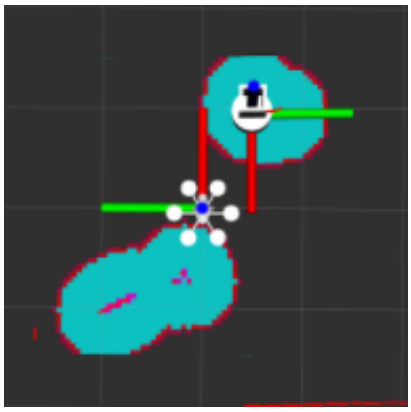
method for robotics [C26]. In Scenario 1, we analyze and compare the results of both the DMLS and the QR-code localization methods on both robots, the *Detector* and *Pawn*. The *Pawn* is placed at one-meter distance from the *Detector* in the x-axis, which represents the front facing direction of the *Detector* map-frame. The *Pawn* was placed in 4 different positions on the y-axis, which were: $\vec{r}_P = (1.00m, 0.65m)$, $\vec{r}_P = (1.00m, 0.25m)$, $\vec{r}_P = (1.00m, -0.25m)$, $\vec{r}_P = (1.00m, -0.65m)$. At each position, 4 different sub-experiments are conducted. In each, the *Pawn* is oriented to a different relative-orientation: $\phi = 0°, 90°, -90°, 180°$, ending up with a total of 16 sub-experiments. The tests were executed on a 3m x 3m grid with 0.5m x 0.5m subdivisions in the laboratory and in *Rviz*. The *Detector* was in a static position with a fixed orientation for all the tests.

Experiment 1A, Scenario 1, Using DMLS

In this experiment, the DMLS is used for localizing and relative orientation measuring of the *Pawn* in the *Detector* map-frame for all of the 16 sub-experiments of Scenario 1. In Fig. 5.6, an example of having the *Pawn* positioned at $\vec{r}_P = (1.00m, 0.65m)$ and $\phi = 180°$ is shown. Fig. 5.6a shows the sketch of the scenario done in the laboratory, and Fig. 5.6b shows an *Rviz* illustration of the local cost-map of the *Detector* used to obtain an accurate estimation of the position and orientation of the *Pawn* within the *Detector* map-frame. The red axis indicates the direction each robot is facing ("x-axis").



(a) Sketch of the scenario in the laboratory.



(b) *Rviz* illustration of the scenario from local cost-map observation by the *Detector*.

Figure 5.6: Scenario 1, Experiment 1A. *Pawn* positioned at $\vec{r}_P = (1.00m, 0.65m)$, $\phi = 180°$ from *Detector*.

Experiment 1B, Scenario 1, Using QR-pose estimation method

In Experiment 1B, the QR-code pose estimation method is used for localizing the *Pawn* in the *Detector* map-frame for all of the 16 sub-experiments of Scenario 1. In Fig. 5.7, an example where the *Pawn* is positioned at $\vec{r}_P = (1.00m, 0.25m)$ and $\phi = 180°$ is shown. Fig. 5.7a shows the sketch of the scenario done in the laboratory, and Fig. 5.7b shows an image of the detected QR code and corresponding pose estimation. The results are presented in Table 5.1, which shows that from the 16 sub-experiments, only in 8 the QR-code pose estimation method was able to get a successful detection of the *Pawn* by the *Detector*. The QR-code pose estimation method failed to identify and read the QR-code when it was not entirely visible from the camera's field of view (FOV). This shows that for having a pose estimation, it is vital for the QR-code to fully appear in the captured camera frame, whereas for the DMLS this is not a restriction, as long as the CNN is able to identify the *Pawn*, and a pose estimation using DMLS can be accomplished by only the partial appearance of the *Pawn* in the camera FOV. Among all the results using the QR-pose estimation method, only those that were successful were used to calculate the absolute mean error, and its variance, in Table 5.1.

(a) Sketch of the scenario in the laboratory.

(b) Screenshot of the *Detector* detecting a QR code on the *Pawn*.

Figure 5.7: Scenario 1, Experiment 1B. *Pawn* positioned at $\vec{r}_P = (1.00m, 0.25m)$, $\phi = 180°$ from *Detector*.

5.6.2 Scenario 2: Pose estimation at $d = 1.40m$

In Scenario 2, to further test the range of detection with this setup, the same methodology is repeated as in Scenario 1, but this time increasing the distance between both robots to $1.40m$, in the front axis of the *Detector* (x-axis), since

Pawn pos.	Lab ϕ	DMLS ϕ	QR-code ϕ
$\vec{r}_P = (1.00m, 0.65m)$	$180°$	$180°$	$-$
$\vec{r}_P = (1.00m, 0.65m)$	$-90°$	$-93°$	$-$
$\vec{r}_P = (1.00m, 0.65m)$	$90°$	$90°$	$-$
$\vec{r}_P = (1.00m, 0.65m)$	$0°$	$0°$	$-$
$\vec{r}_P = (1.00m, 0.25m)$	$180°$	$180°$	$179°$
$\vec{r}_P = (1.00m, 0.25m)$	$-90°$	$-81°$	$-90°$
$\vec{r}_P = (1.00m, 0.25m)$	$90°$	$96°$	$86°$
$\vec{r}_P = (1.00m, 0.25m)$	$0°$	$0°$	$-3°$
$\vec{r}_P = (1.00m, -0.25m)$	$180°$	$180°$	$178°$
$\vec{r}_P = (1.00m, -0.25m)$	$-90°$	$-81°$	$93°$
$\vec{r}_P = (1.00m, -0.25m)$	$90°$	$92°$	$93°$
$\vec{r}_P = (1.00m, -0.25m)$	$0°$	$0°$	$2°$
$\vec{r}_P = (1.00m, -0.65m)$	$180°$	$180°$	$-$
$\vec{r}_P = (1.00m, -0.65m)$	$-90°$	$-85°$	$-$
$\vec{r}_P = (1.00m, -0.65m)$	$90°$	$99°$	$-$
$\vec{r}_P = (1.00m, -0.65m)$	$0°$	$0°$	$-$
Mean		$\mu = 3.643$	$\mu = 2.229$
Variance		$\rho^2 = 13.274$	$\rho^2 = 1.536$

Table 5.1: Results of the experiments in Scenario 1 using the DMLS and QR-pose estimation methods.

it was observed that the QR-code method did not consistently work with greater distances. For this experiment, the *Pawn* is also placed in 4 different positions from the *Detector*, which were: $\vec{r}_P = (1.40m, 0.80m)$, $\vec{r}_P = (1.40m, 0.25m)$, $\vec{r}_P = (1.40m, -0.25m)$, $\vec{r}_P = (1.40m, -0.75m)$. At each position experiments were conducted while placing the *Pawn* at 4 different orientations, obtaining 16 sub-experiment results in total.

Experiment 2A, Scenario 2, Using DMLS

In Experiment 2A, Experiment 1A in Scenario 1 is repeated but in Scenario 2. We conclude from the results of all obtained poses and the absolute mean error shown in Table 5.2, that at a higher distance, the DMLS method achieved to localize the *Pawn* in all 16 sub-experiments while maintaining almost the same performance.

Experiment 2B, Scenario 2, Using QR-pose estimation method

In Experiment 2B, Experiment 1B of Scenario 1 is repeated but in Scenario 2. Since the QR-code cannot be read when the QR-code is only partially in the FOV of the camera, only 8 of the 16 sub-experiments were successfully executed. The pose estimations and the absolute mean error of the QR-pose estimation method are shown in Table 5.2.

Pawn pos.	Lab ϕ	DMLS ϕ	QR-code ϕ
$\vec{r}_P = (1.40m, 0.8m)$	$180°$	$180°$	$-$
$\vec{r}_P = (1.40m, 0.8m)$	$-90°$	$-81.41°$	$-$
$\vec{r}_P = (1.40m, 0.8m)$	$90°$	$92.79°$	$-$
$\vec{r}_P = (1.40m, 0.8m)$	$0°$	$0°$	$-$
$\vec{r}_P = (1.40m, 0.25m)$	$180°$	$180°$	$173.551°$
$\vec{r}_P = (1.40m, 0.25m)$	$-90°$	$-81.334°$	$-93.63°$
$\vec{r}_P = (1.40m, 0.25m)$	$90°$	$96.379°$	$81.674°$
$\vec{r}_P = (1.40m, 0.25m)$	$0°$	$0°$	$-1.757°$
$\vec{r}_P = (1.40m, -0.25m)$	$180°$	$180°$	$177.957°$
$\vec{r}_P = (1.40m, -0.25m)$	$-90°$	$-91.32°$	$-91.584°$
$\vec{r}_P = (1.40m, -0.25m)$	$90°$	$89.088°$	$84.119°$
$\vec{r}_P = (1.40m, -0.25m)$	$0°$	$0°$	$-7.248°$
$\vec{r}_P = (1.40m, -0.75m)$	$180°$	$173.38°$	$-$
$\vec{r}_P = (1.40m, -0.75m)$	$-90°$	$-83.25°$	$-$
$\vec{r}_P = (1.40m, -0.75m)$	$90°$	$98.213°$	$-$
$\vec{r}_P = (1.40m, -0.75m)$	$0°$	0	$-$
Mean		$\mu = 3.673$	$\mu = 2.515$
Variance		$\rho^2 = 13.493$	$\rho^2 = 6.326$

Table 5.2: Results of the experiments in Scenario 2 using the DMLS and QR-pose estimation methods.

5.6.3 Scenario 3: Pose estimation in case of different robots in range

In scenario 3, since our robots are expected to operate within the presence of other team member robots of different types, forms, and sizes also in more complex environments, the experiments done in Scenario 1 and 2 are repeated but this time involving an extra robot, known as *Adversary Robot*. The *Adversary Robot* has a size and footprint bigger than the *Pawn*, meaning that in some scenarios, it could

partially or fully distort the detection of the *Pawn* by blocking the LoS between the *Detector* and the *Pawn*.

Scenario 3.1, Robots on different sides of the Detector's FOV

In Scenario 3.1, the *Pawn* and *Adversary Robot* are placed at each side and in-front of the *Detector*'s FOV. This concludes that the *Detector* would visually sense one robot at each side of the image camera-frame. Fig. 5.8 illustrates the scenario. The positions of all robots in the *Detector*'s map for scenario 3.1 can be found in Table 5.3.

Scenario	$\vec{r}_D$	$\vec{r}_P$	$\vec{r}_A$
3.1	$(0.00, 0.00)$	$(1.00m, -0.25m)$	$(1.00m, 0.25m)$
3.2	$(0.00, 0.00)$	$(1.00m, -0.25m)$	$(1.50m, 0.00)$
3.3	$(0.00, 0.00)$	$(1.00m, -0.25m)$	$(0.50m, -0.25m)$

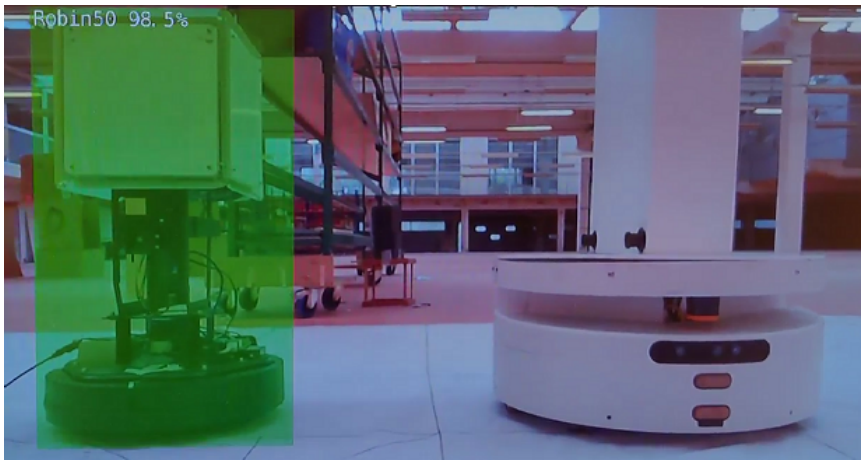
Table 5.3: Relative position of the *Detector*, *Pawn*, and *Adversary* robots in Scenarios 3.1, 3.2, and 3.3 with respect to the *Detector*.

Experiment 3.1A, Scenario 3.1, Using DMLS

In Experiment 3.1A in Scenario 3.1, the DMLS is used for localizing the *Pawn* in the *Detector*'s map-frame. It can be clearly seen from Fig. 5.8a and the results shown in Table 5.4 that the *Detector* has successfully identified, distinguished, and localized the *Pawn* from the *Adversary Robot*.

Experiment 3.1B, Scenario 3.1, Using QR-code pose estimation method

In Experiment 3.1B in Scenario 3.1, the QR-code pose estimation method is used to estimate the pose for the *Pawn*. It can be clearly seen from Fig. 5.8b and the results shown in Table 5.3 that the *Detector* has successfully identified, distinguished, and localized the *Pawn* from the *Adversary Robot*. The reason was that the QR-code was fully visible to the *Detector*, which was able to obtain a pose of the QR-code.



(a) Detecting the *Pawn* using the DMLS method.

(b) Detecting the *Pawn* using the QR-pose estimation method.

Figure 5.8: Scenario 3.1, Experiment 3.1A and 3.1B. The *Pawn* and *Adversary Robot* placed at each side of the *Detector*'s FOV.

Scenario 3.2, Robots positioned on the same side of the Detector's FOV

In Scenario 3.2, the *Pawn* and the *Adversary Robots*, are placed on the same side of the map. However, the *Pawn* was placed closer to the *Detector* than the *Adversary Robot*, as shown in Fig. 5.9. The positions of all robots for Scenario 3.2 can be found in Table 5.3.

Experiment 3.2A, Scenario 3.2, Using DMLS

In Experiment 3.2A, the DMLS method is used for localizing the *Pawn* in the *Detector*'s map-frame. It can be clearly noted from Fig. 5.9, and the results shown in Table 5.4, that the *Detector*, has successfully identified, distinguished, and accurately localized the *Pawn* from the *Adversary Robot*, even though more cells had a high-cost value in the RoS sub-matrix.

Experiment 3.2B, Scenario 3.2, Using QR-pose estimation method

In Experiment 3.2B, the QR-code pose estimation method is used for localizing the *Pawn* in the *Detector*'s map-frame. It can be clearly noted from Fig. 5.9b and the results shown in Table 5.4, that *Detector*, has successfully identified, distinguished, and accurately localized the *Pawn* from the *Adversary*.

Scenario	Real ϕ	DMLS ϕ	QR-code ϕ
3.1	180°	180°	179°
3.2	180°	180°	180°

Table 5.4: Provides the results of Experiments in scenario 3.1 and 3.2 using the DMLS and QR-code pose estimation method.

(a) Detecting the *Pawn* using the DMLS method.

(b) Detecting the *Pawn* using the QR-pose estimation method.

Figure 5.9: Scenario 3.2, Experiment 3.2A and 3.2B. The *Pawn* and *Adversary Robot* are placed at one side of the *Detector*'s FOV while the *Pawn* is closer to the *Detector*.

Scenario 3.3, Robots placed on the same side, with different relative distances to the Detector

In Scenario 3.3, the *Pawn* and the *Adversary Robot*, are placed on the same side of the *Detector*'s FOV, but this time the *Pawn* was behind the *Adversary Robot*, hindering its visualization and detection of the *Detector*, making it non-LoS.

The Experiments in Scenario 3.3, as shown in Fig. 5.10 failed, as none of the methods could detect the *Pawn*. In Fig. 5.10a, an illustration of how the DMLS fails to detect the *Pawn* is shown, while Fig. 5.10b shows the QR-code pose estimation method failing to detect the QR-code.

5.6.4 Scenario 4: continued pose detection in $360°$

The aim of the Experiment in scenario 4 is to test the detection robustness of both methods. For doing so the *Pawn* is continuously rotated CW with $30°$ increments. For each rotation, both methods are tested to verify if they were able to achieve a successful detection of the *Pawn*. In total, 16 different sub-experiments were tested on both methods. The *Pawn* was placed at $d = 1.00m$ in the x-axis from the *Detector*. The locations of the *Pawn* and the *Detector* can be found in Table 5.5.

(a) Fail detection of the *Pawn* using the DMLS method.

(b) Fail detection of the *Pawn* using the QR-code pose estimation method.

Figure 5.10: Scenario 3.3, Experiment 4A and 4B. The *Pawn* and *Adversary Robot* are placed at one side of the *Detector*'s FOV while the *Pawn* is hidden from the *Detector*.

$\vec{r}_D$	$\vec{r}_P$
$(1.00m, 0.00m)$	$(1.00m, 1.00m)$

Table 5.5: Relative position of each robot in Scenario 4 to the *Detector* map-frame.

Experiment 4A, Scenario 4, Using DMLS

In Experiment 4A, the DMLS for is used for localizing the *Pawn* in the *Detector*'s map-frame for all 16 sub-experiments of Scenario 4. It can be clearly noted from the results shown in Table 5.6 that the *Detector* has successfully detected and accurately localized the *Pawn* in all 16 orientations.

Experiment 4B, Scenario 4, Using QR-pose estimation method

In Experiment 4B, the QR-code pose estimation method was used for localizing the *Pawn* in the *Detector*'s map-frame for all 16 sub-experiments of Scenario 4. It can be clearly noted from the results shown in Table 5.6 that the *Detector* has successfully detected and accurately localized the *Pawn* in all 16 orientations.

Real ϕ	DMLS ϕ	QR code ϕ
$0°$	$0°$	$359°$
$30°$	$30°$	$28°$
$45°$	$46°$	$41°$
$60°$	$56°$	$62°$
$90°$	$84°$	$93°$
$120°$	$126°$	$127°$
$135°$	$140°$	$138°$
$150°$	$155°$	$147°$
$180°$	$180°$	$178°$
$210°$	$215°$	$206°$
$225°$	$229°$	$224°$
$240°$	$240°$	$236°$
$270°$	$270°$	$268°$
$300°$	$297°$	$311°$
$315°$	$317°$	$319°$
$330°$	$335°$	$328°$
Mean	$\mu = 2.898$	$\mu = 3.599$
Variance	$\rho^2 = 5.30$	$\rho^2 = 5.754$

Table 5.6: Provides the results of Experiments in Scenario 4 using the DMLS and QR-pose estimation method.

It is important to note that both methods were able to recover after a faulty detection and successfully detect and localize the *Pawn* when the parameters that affected the detection were present. Table 5.7, shows a fault tolerance comparison for both methods with different conditions:

Method	Net. com. between Robots	Robot relative ori.	LoS
DMLS	Not tolerant	Tolerant	Not tolerant
QR-code	Tolerant	Not tolerant	Not tolerant

Table 5.7: Shows fault tolerance comparison between both methods in different situations.

From Table 5.7, we can see that both models require different environment conditions to operate as needed. For example, the DMLS requires that the *Detector* and the *Pawn* are able to connect through a wireless network, whereas the QR-code pose estimation method does not, however, the latter is sensitive to the relative orientation of the QR code as it fails to detect the QR code at some angles. Finally, both methods require that both robots are in LoS to able to work properly.

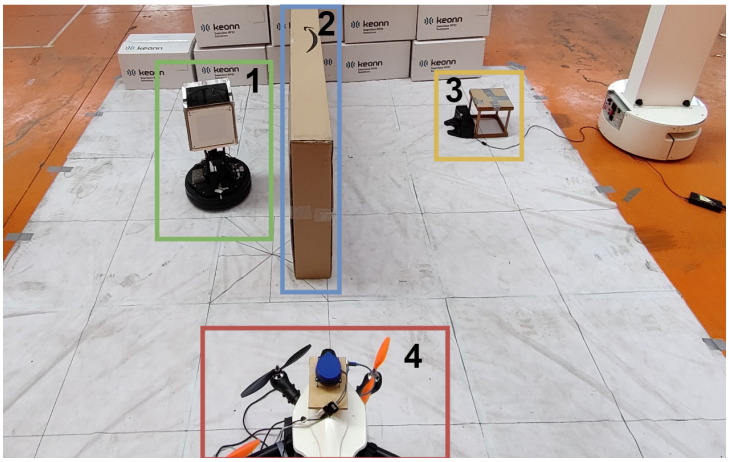
5.7 Applications

5.7.1 Laboratory Experiment

The docking station and the robot docking mechanism are designed to work together, providing a mechanical and electrical connection between the charging system and the robot [C27]. This communication in our case is composed of infra-red LoS transmitted signals from the docking-sensors on the *Pawn*, with the receiver on the *Docking Station*. Taking advantage of the full capacity of the CNN used by the DMLS, the CNN model was trained to detect multiple important entities in the environment, one of them being the *Docking Station*. We simulate a scenario of the *Pawn*, during an end of a mission in an un-explored environment, where it starts searching for the *Docking Station*, in order to self-charge and continue towards task completion. However, the *Pawn* fails to detect the *Docking Station* due to an obstacle blocking the LoS communication between the *Pawn* and the *Docking Station* as shown in Fig. 5.11a. The *Detector*, however, happens to detect both the *Pawn* and the *Docking Station* using the pre-trained CNN model on board. Utilizing the computed *Pawn* pose using DMLS, the *Detector* can share with the *Pawn* the exact pose of the *Docking Station* in the *Pawn*'s map-frame.

In Figs. 5.11a and 5.11b the entity tagged as "1" is the *Pawn*. The entity tagged as "2" is the obstacle blocking the LoS between the *Docking Station* and the *Pawn*. The entity tagged as "3" is the *Docking Station*. The entity tagged as "4" is the *Detector* which cannot be seen in Fig. 5.11a since it is the source point. It can also be seen how both *Pawn* and *Docking Station* are being identified and detected with a high level of confidence.

(a) POV of the *Detector* during the test.



(b) Scenario picture of the test

Figure 5.11: Laboratory illustration of the Application scenario having Multi-robots collaboration using DMLS.

5.7.2 Simulation Experiments

Feature Location Sharing Using Vision

In Simulating Scenario 1, Gazebo simulation platform is used for simulating Scenario 1 that is shown in Fig. 5.12a, a Gazebo world model is designed for the purpose of showing how a group of UGVs using the DMLS, can collaborate to share important features in the environment.

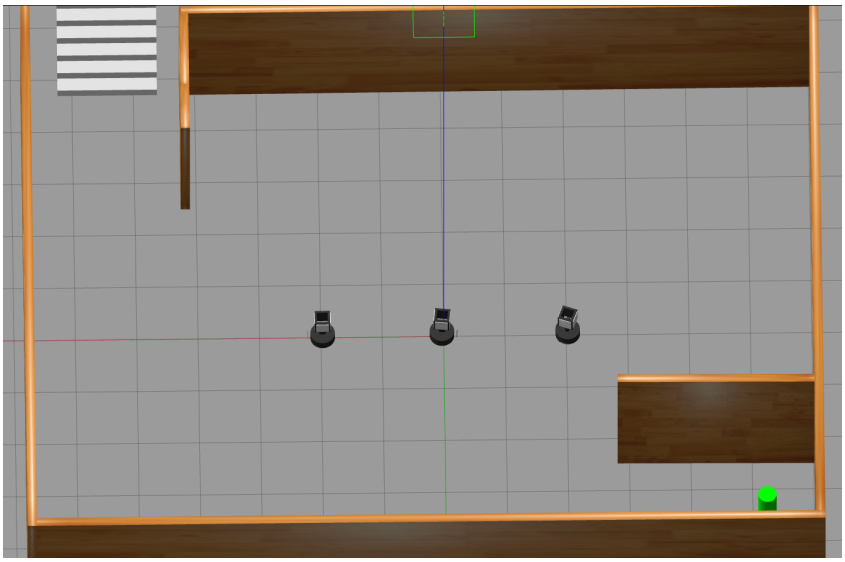
For this Scenario, 3 UGVs will be used, one *Detector* and 2 *Pawns* called *Pawn-1* and *Pawn-2*. The UGVs will initially spawn 1m away from each other. The *De-*

tector's initial position would be in the middle and the rest at the sides. First, the *Detector* will detect *Pawn-1*, due to the fact that it is in LoS with the *Detector*, lays within its local costmap detection region, and is detected by the CNN.

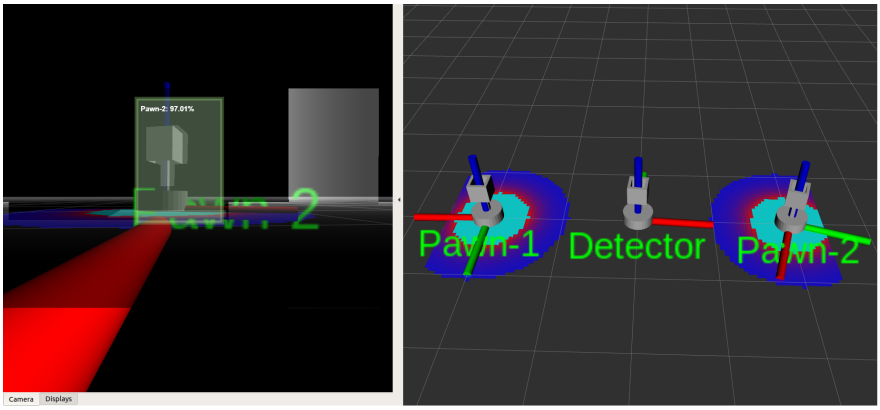
Then the *Detector* does a $360\,\mathrm{deg}$. rotation to check if it is able to detect more robots in the area. As a result, *Pawn-2* will also be detected and successfully localized using the DMLS. Fig. 5.12b, shows the detection of both *Pawns*. The left side of the image shows the camera output visually detecting the robots and the right shows what is able to be sensed in the *Detector's* frame.

After successful detection and localization between the robots, all robots would be able to share their local costmaps with each other as shown in Fig. 5.13. This will increase the environment awareness of each robot using the help of the others. The robots will then pursue the tasks that they are designed for. In the case of this Scenario, the robots will move in random directions exploring the map.

In Figs. 5.14a and 5.14b, the *Detector* has discovered and successfully located important features in the environment. Not only features that are beneficial to the robots like the charging/docking station are shared when discovered and localized, however, using the DMLS robots can warn each other of dangerous zones. An example of localizing surfaces unsuitable for the robots to navigate through (i.e, stairs) is shown in Figs. 5.15a and 5.15b. The *Detector* uses just the first part of the DMLS algorithm to detect and localize these features, as in this case the orientation of the objects is not important.



(a) Gazebo illustration of the Scenario.



(b) Rviz Detection of Pawn-1 & 2.

Figure 5.12: Simulation Scenario 1: The Detector localizing and sharing the positions with all robots.

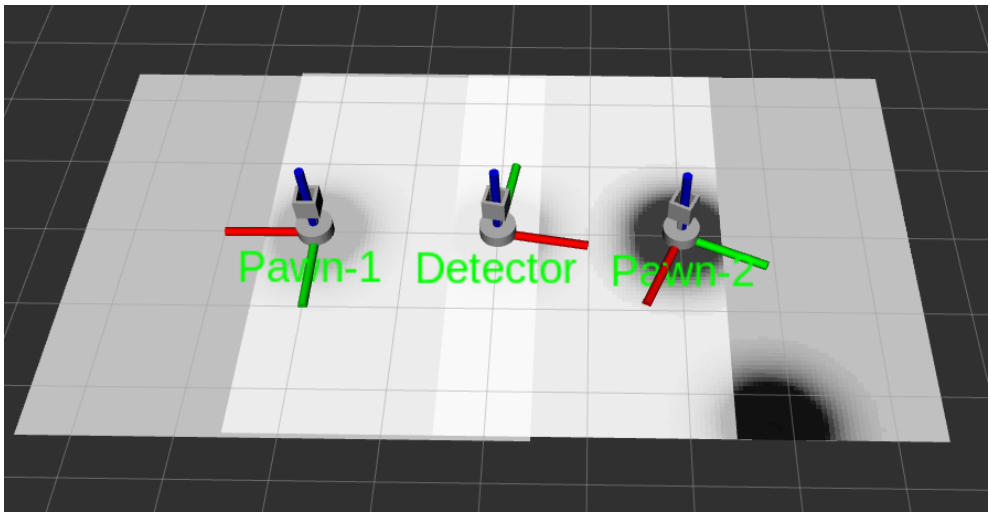
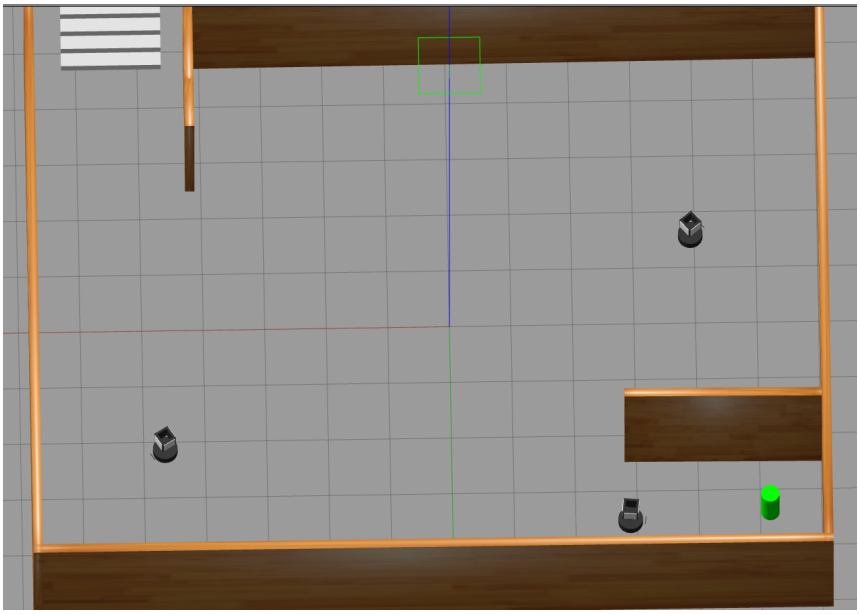
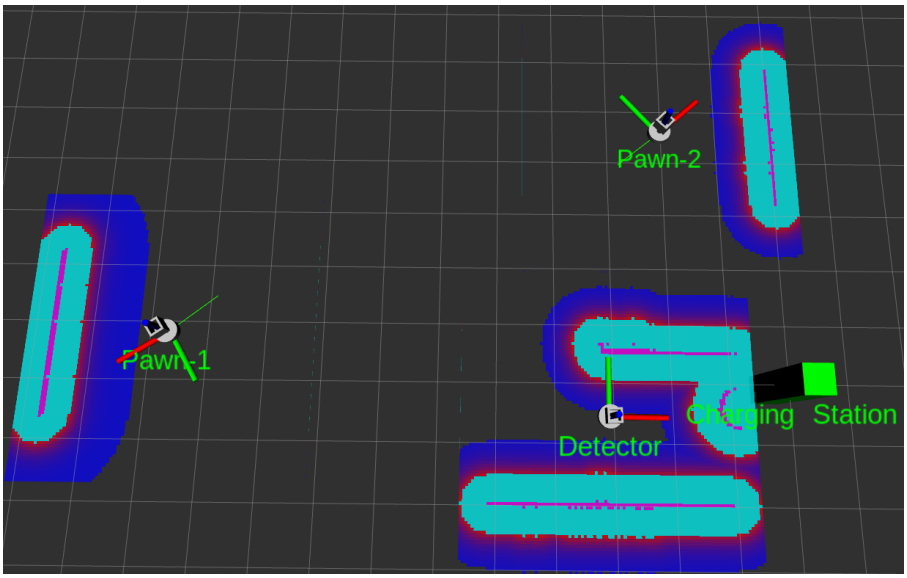


Figure 5.13: Simulation Scenario 1: An illustration from the Detector's frame showing local costmaps from all robots.

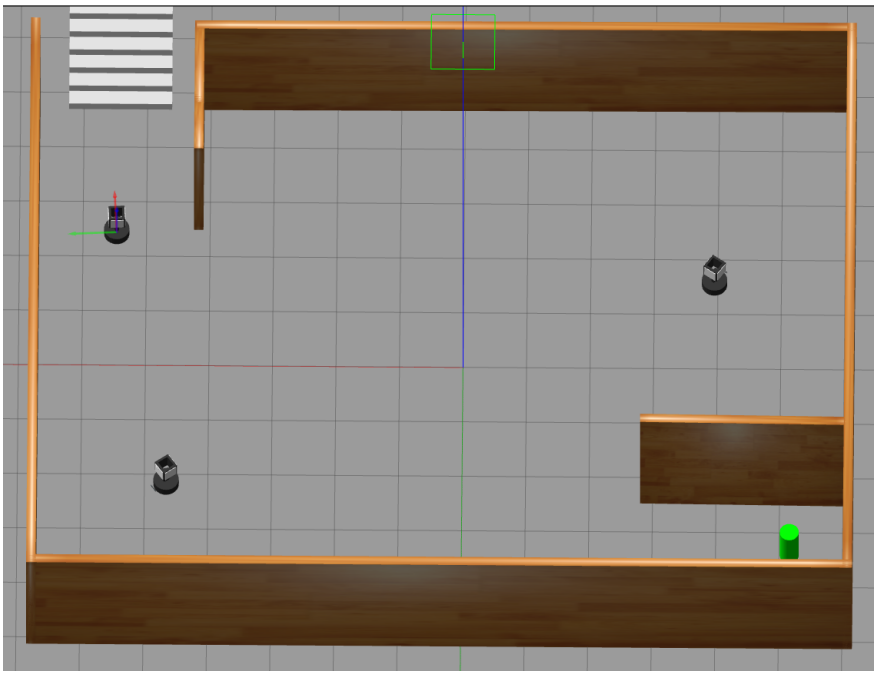


(a) Gazebo illustration of Localizing the Docking station.

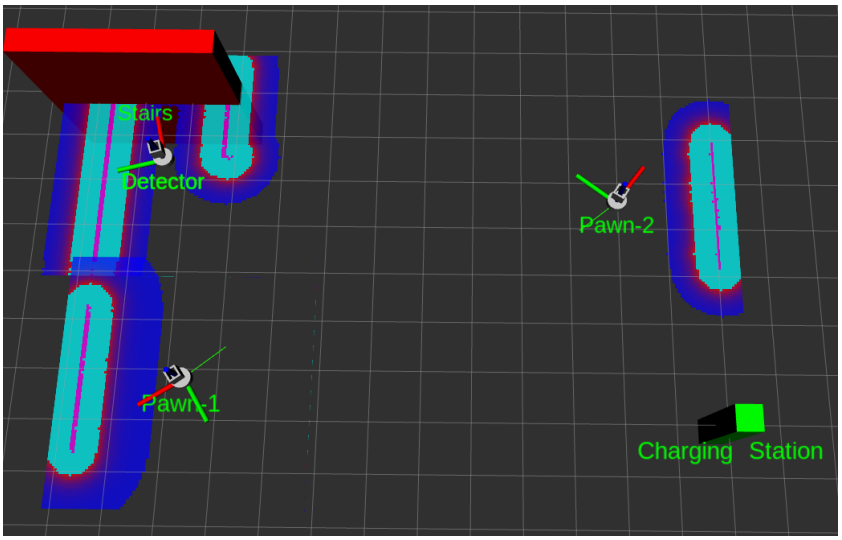


(b) Rviz illustration of Localizing the Docking station.

Figure 5.14: Simulation Scenario 1: Localizing and sharing the docking station position with all robots.



(a) Gazebo illustration of Localizing the Stairs.



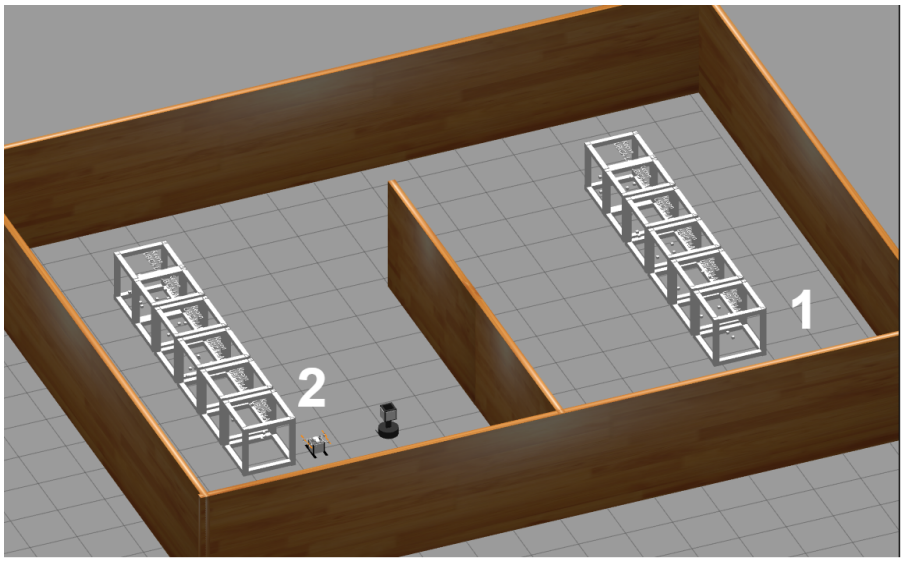
(b) Rviz illustration of Localizing the Stairs.

Figure 5.15: Simulation Scenario 1: Localizing and sharing the position of the stairs with all robots.

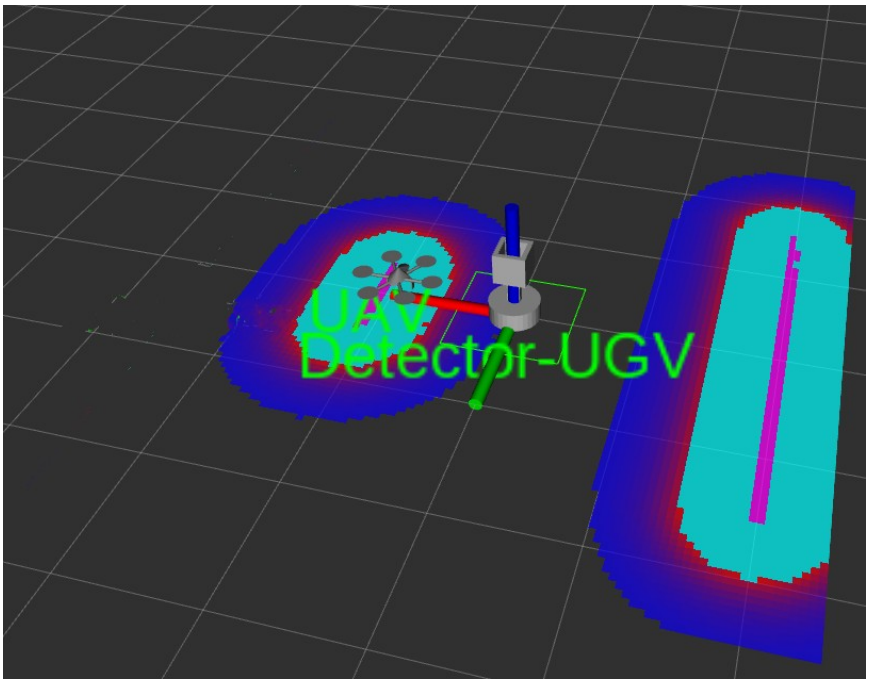
Collaborating in the existence of RFID tags

In Simulation Scenario 2, shown in Fig. 5.16a, a gazebo world model is designed to show that using the DMLS, heterogeneous inventory robots can collaborate in environments where RFID sensors exist for the purpose of autonomous map-less navigation and completing an inventory mission.
The goal is to increase the overall team performance of completing an inventory mission in environments where products are represented by RFID tags, such environments are warehouses or retail shops. The robots will exploit the heterogeneity feature in the team for exploring new regions that have RFID tags.
Inventory UGVs have proven to perform well during the recent years [C28]. They are robust machines that can carry heavy payloads which allows large power sources to be mounted onboard. This able them to operate for a considerable amount of time. However, these robots are slow, considered limited, and not adequate for exploration purposes.
On the other hand, UAVs are adequate for exploration purposes due to their agility, maneuverability and are undependable of the ground surface type. However, UAVs suffer from low flight time and have weight and size constraints for the payload they can lift. This Scenario focuses on the exploitation of the benefits of both types of robots for completing an inventory task.
Two heterogeneous robots are used. This time the *Detector* will be a UGV. The *Pawn* will be a UAV. Both robots will be equipped with an RFID reader [C29] and

directional antennas [C30]. The map will contain 2 shelves. Both are placed far from each other and have a wall almost separating them as shown in Fig. 5.16a.



(a) Gazebo illustration of Scenario Simulation Scenario 2



(b) Rviz illustration Detection of Pawn.

Figure 5.16: Simulation Scenario 2: The Detector localizing the Pawn.

Each shelf contains 6 boxes, each box contains a group of RFID tags. The non-detected RFID tags will be represented as grey cubes. The RFID tags that are detected by the *Detector* would be represented as blue cubes. Finally, the RFID tags that are detected by the *Pawn* would be represented as red cubes. RFID technology is simulated using a ROS-based Gazebo plugin [C31]. Since the *Detector* uses RFID stigmergic-based navigation [C32], therefore it would only navigate in the region where it can detect new RFID tags. This implies that since the shelf on the right side marked as (1) in Fig. 5.16a, would be considered outside of the detection range, the *Detector* would not be aware of the existence of the RFID tags on that side. It will only read the RFID tags on the left side marked as (2) in Fig. 5.16a and shown in Fig. 5.17.

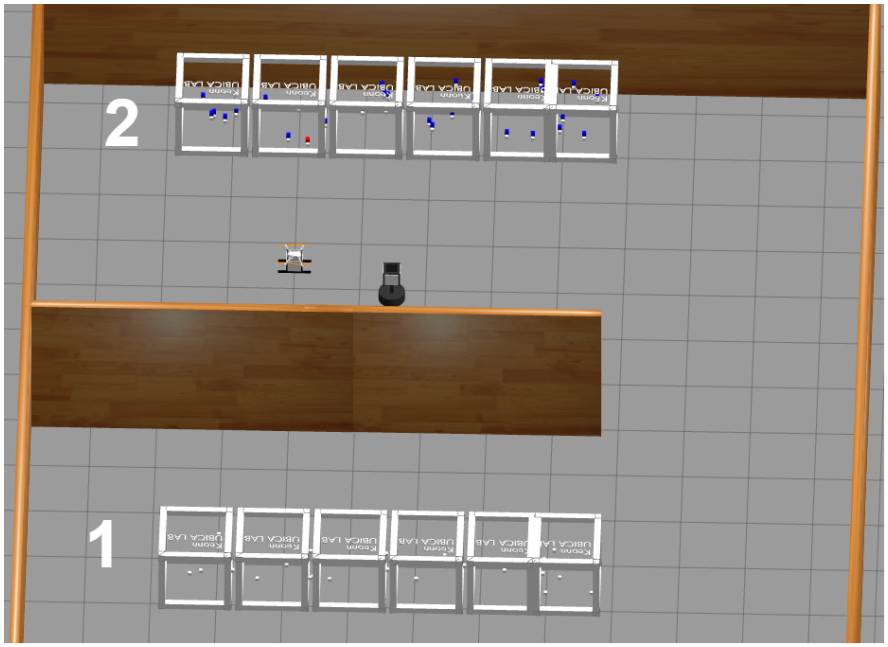


Figure 5.17: Simulation Scenario 2: Gazebo illustration of the RFID tag detection located on the left shelves (2).

Since the *Pawn* operates as a "scout", it continuously searches for new zones where populations of RFID tags exist and shares the location of where it was able to detect these new RFID tags. The *Pawn* uses an autonomous navigation scheme, which moves in a chosen direction as long as no obstacles are detected in its direction of movement. The *Pawn* will choose a different direction in case of an obstacle is detected in the direction of movement. In Fig. 5.18, more red cubes can be seen that represent the detected RFID tags while the *Pawn* is flying through the environment. As soon as new RFID tags are detected by the *Pawn*, as

shown in Figs. 5.18 and 5.19b, the location coordinates of where the new RFID tags are read will be relayed to the *Detector*. The *Detector* therefore will move to new regions where new tags are explored by the *Pawn*, each time it receives this data.

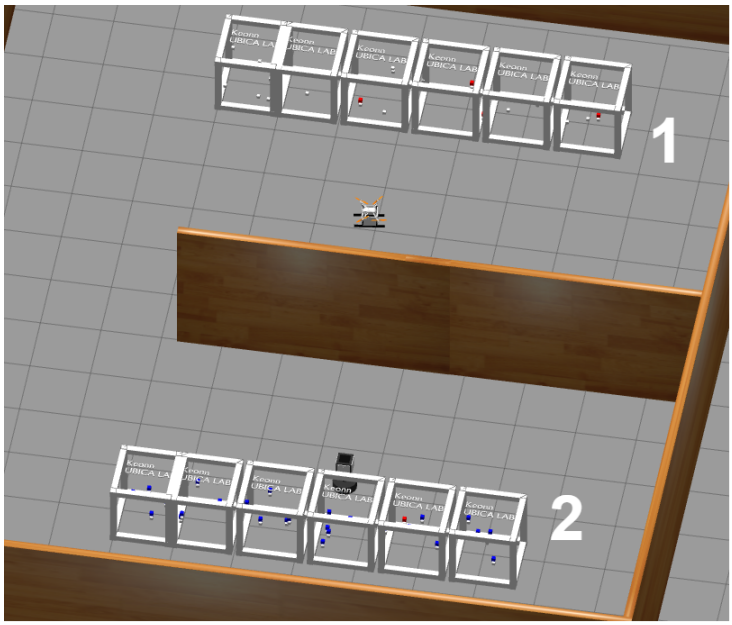
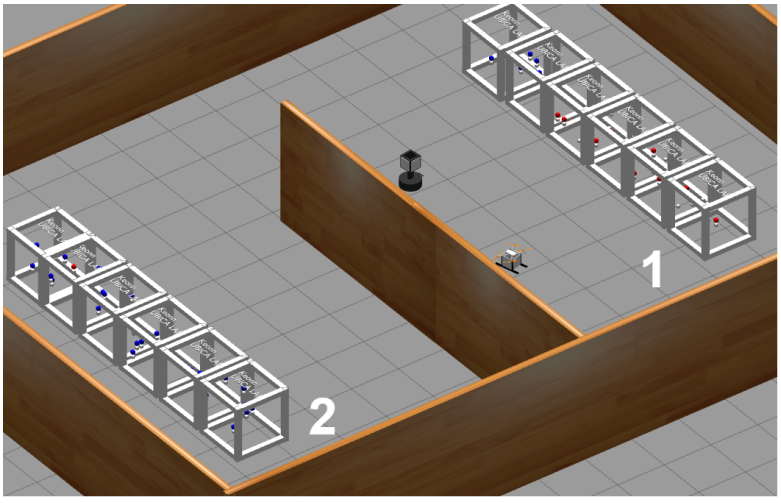
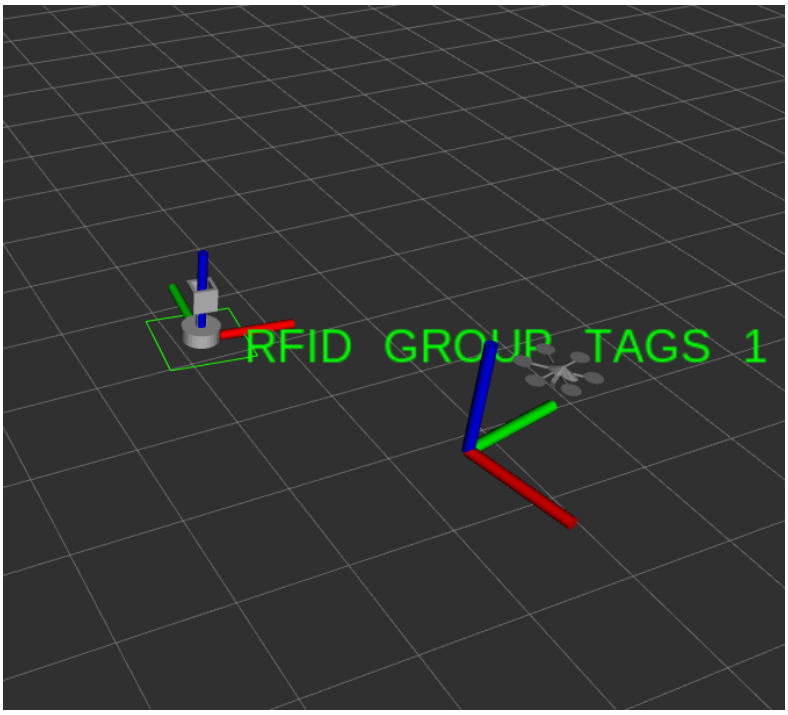


Figure 5.18: Simulation Scenario 2: Gazebo illustration of the RFID tag detection located on the right shelves (1).



(a) Gazebo illustration of the Detector and Pawn positioned at the newly discovered RFID tags zone.



(b) Rviz illustration of the Detector and Pawn positioned at the new discovered RFID tags zone.

Figure 5.19: Simulation Scenario 2: The Detector Moving towards the new explored zone of RFID tags, sent by the Pawn.

5.8 Conclusions

Driven by the growing interest in multi-robots trying to achieve collective intelligence from individual simplicity, a strategy based on ROS that is used to localize and measures the relative orientation for distributed multi-robots is presented, for the case where the team robots are not connected within their transform trees. This enables heterogeneous multi-robots to share valuable resources among themselves. The proposed technique does not require modification of the environment, such as placing QR codes, beacons, or complex sensors, nor it requires prior knowledge of the map for it to function and obtain good results, as do most currently used localization methods. We also prove empirically that it is computationally inexpensive. It has been tested on several low computation power single board computers (SBC) such as Jetson-Nano and on different robot platforms. The proposed method relies only on the existing basic hardware of an autonomous robot in multi-robot's scenarios. Any robot that can use the basic

navigation and path planning packages provided by ROS will be able to run this localization model. At least one robot in the multi-robots group is equipped with a visual sensor, and all robots in the group must be able to communicate with each other.

Though the DMLS method, as illustrated in experiments, proves to have accurate results and good performance similar to the QR-code pose estimation method in most cases, it still has some limitations. These limitations are addressed in Section 5.9.

5.9 Future Work

During intensive experimentation on our proposed localization method, some limitations and opportunities were discovered:

a) *Reducing the RoS in the map*, by improving the algorithm to compute a smaller RoS, which indicates the MLH obstacle to be the robot. This will significantly reduce the possibility of the *Detector* to wrongly locate the *Pawn*.

b) *Increasing the resolution of the local cost-map cells*, by increasing the resolution of the map cells, and changing it from approximately $2.5cm$ per cell to just a few millimeters per cell. This will lead to a higher resolution in extracting the exact position of the *Pawn*.

c) *Using AI to analyze the map*, designing a neural network (NN) whose input data consists of the local cost map of the robot by converting the map-frames to consecutive images. This data-set will be fed into the NN during the detection phase. The goal is to predict and distinguish the robot shape in a map, from different shapes and patterns of different obstacles.

d) *Using 3D mapping to increase method's efficiency*, eliminating intermediate steps such as the Detector landing in order to be detected by the Pawn. Increasing the dynamism of the overall workflow of the DMLS Framework.

Chapter-5 References

[C1] I.J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on Robotics and Automation*, 7(2):193–204, 1991.

[C2] F. Chenavier and J.L. Crowley. Position estimation for a mobile robot using vision and odometry. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2588–2593 vol.3, 1992.

[C3] A.L. Barker, D.E. Brown, and W.N. Martin. Bayesian estimation and the kalman filter. *Computers and Mathematics with Applications*, 30(10):55–77, 1995.

[C4] Feng Lu and Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 935–938, 1994.

[C5] Lynne Parker. Current state of the art in distributed autnomous mobile robotics. In *Distributed Autonomous Robotic Systems 4*, pages 3–14, 01 2000.

[C6] Victor Casamayor-Pujol, Marc Morenza-Cinos, Bernat Gastón, and Rafael Pous. Autonomous stock counting based on a stigmergic algorithm for multi-robot systems. *Computers in Industry*, 122:103259, 2020.

[C7] Vinicio Rosas-Cervantes, Quoc-Dong Hoang, Soon-Geul Lee, and Jae-Hwan Choi. Multi-robot 2.5d localization and mapping using a monte carlo algorithm on a multi-level surface. *Sensors*, 21:4588, 07 2021.

[C8] Teresa A. Vidal-Calleja, Cyrille Berger, Joan Solà, and Simon Lacroix. Large scale multiple robot visual mapping with heterogeneous landmarks in semi-structured terrain. *Robotics and Autonomous Systems*, 59(9):654–674, 2011.

[C9] Karol Hausman, Jörg Müller, Abishek Hariharan, Nora Ayanian, and Gaurav S Sukhatme. Cooperative multi-robot control for target tracking with onboard sensing. *The International Journal of Robotics Research*, 34(13):1660–1677, 2015.

[C10] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(3):376–382, 1991.

[C11] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios. Multi-robot collaboration for robust exploration. *Ann. Math. Artif. Intell.*, 31:7–40, 10 2001.

[C12] K. Kato, H. Ishiguro, and M. Barth. Identifying and localizing robots in a multi-robot system environment. In *Proceedings 1999 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human and Environment Friendly Robots with High Intelligence and Emotional Quotients (Cat. No.99CH36289)*, volume 2, pages 966–971 vol.2, 1999.

[C13] Cullen Jennings, Don Murray, and J.J. Little. Cooperative robot localization with vision-based mapping. *Proceedings - IEEE International Conference on Robotics and Automation*, 4:2659 – 2665 vol.4, 02 1999.

[C14] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8:325–344, 06 2000.

[C15] Guilherme AS Pereira, R Vijay Kumar, and Mario FM Campos. Localization and tracking in robot networks. *Departmental Papers (MEAM)*, page 39, 2003.

[C16] Ashley Stroupe, Martin Martin, and Tucker Balch. Distributed sensor fusion for object position estimation by multi-robot systems. *Proceedings - IEEE International Conference on Robotics and Automation*, 2:1092–1098, 01 2001.

[C17] I. Gavrilut, V. Tiponut, A. Gacsadi, and C. Grava. Cnn processing techniques for multi-robot coordination. In *2007 International Symposium on Signals, Circuits and Systems*, volume 1, pages 1–4, 2007.

[C18] Abdussalam A. Alajami, Guillem Moreno, and Rafael Pous. Design of a uav for autonomous rfid-based dynamic inventories using stigmergy for mapless indoor environments. *Drones*, 6(8), 2022.

[C19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[C20] Anis Koubâa et al. *Robot Operating System (ROS).*, volume 1. Springer, 2017.

[C21] Eitan Marder-Eppstein. MoveBase move base is a navigation package for robots., 2018. Accessed: 2022.

[C22] Filip Novotny. Costmap is a cell cost based map creator package for robots. `http://wiki.ros.org/costmap\_2d`, 2021. Accessed: 2021.

[C23] Andy Rowlands. Fundamental optical formulae. In *Physics of Digital Photography*, 2053-2563, pages 1–1 to 1–62. IOP Publishing, 2017.

[C24] TF transform package for tracking multiple coordinate frames over time. `http://wiki.ros.org/tf`. Accessed: 2022.

[C25] Dave Hershberger. Rviz a 3d visualization tool for ros. `http://wiki.ros.org/rviz`, 2022. Accessed: 2022.

[C26] Filip Novotny. QR_pose_estimator visp auto tracker is a qr code pose estimator package for robots. `http://wiki.ros.org/costmap\_2d?distro=noetic`. Accessed: 2021.

[C27] Guangming Song, Hui Wang, Jun Zhang, and Tianhua Meng. Automatic docking system for recharging home surveillance robots. *IEEE Transactions on Consumer Electronics - IEEE Trans Consum Electron*, 57:428–435, 05 2011.

[C28] Marc Morenza-Cinos, Victor Casamayor-Pujol, Jordi Soler-Busquets, Jos Luis Sanz, Roberto Guzm, and Rafael Pous. *Development of an RFID Inventory Robot (AdvanRobot)*, pages 387–417. Springer International Publishing, Cham, 2017.

[C29] Keonn. Keonn's AdvanReader 160 keonn's advanreader 160. `https://keonn.com/components-product/advanreader-160/`, 2022. Accessed: 2022.

[C30] _Keonn. Keonn's AdvantennaSP11 keonn's advantennasp11. `https://keonn.com/components-product/advantenna-sp11/`. Accessed: 2022.

[C31] Abdussalam Alajami. RFID Plugin abdussalam alajami, guillem moreno, rafael pous, ros wiki plugin. http://wiki.ros.org/RFIDsensor_Gazebo_plugin, 2022. Accessed: 2022.

[C32] Francis Heylighen. Stigmergy as a universal coordination mechanism : components , varieties and applications. In *Human Stigmergy: Theoretical Developments and New Applications.*, 2014.

5.10 Overall Conclusion and Future Work

We concluded that collective intelligence can be achieved from individual simplicity in heterogeneous (Ground and aerial) team robots. Using the DMLS method increases the collaboration capacity between heterogeneous robots in a team, which increases the overall performance of pursuing a task. However, the benefits still come with a penalty of increasing the complexity of the team members, as more algorithms, hardware, and software packages would be installed on the robots to achieve this.

Furthermore, although the team robots would extend their environmental awareness and sense of their environment using the DMLS method, they would still face the limitation that is associated with their structure type, for example, a UGV would still face complications navigating in un-prepared surfaces or terrains such as smooth surfaces, roads, rails, .etc. A UAV would not be able to endure the flight time required for completing an inventory mission of a huge warehouse, as its structural characteristics limit it from carrying such heavy power sources that would enable it to sustain aerial inventory needed to cover all the warehouses.

Therefore, the final approach in the pursuit of the study within this thesis was to design a hybrid robot that shares the characteristics of both types of robots, the "UAV and the "UGV", for the goal of increasing the performance of an individual task-based robot. The study that discusses the solution design is presented in the following Chapter 6.

Chapter 6

UNMANNED HYBRID AERIAL-GROUND VEHICLE DESIGNS

6.1 UHAGV v4

6.2 Abstract

The problem of increasing the operation time and extending the mobility of the limited energy capacity small-sized Unmanned Aerial Vehicles (UAVs) is studied in this paper. This paper presents a hybrid robot design and a simplified energy-efficient locomotion strategy solution that enables ground mobility locomotion for UAVs. The proposed strategy does not require installing motors for ground mobility, but, it exploits the UAV for generating taxiing locomotion. The proposed hybrid robot design confines important characteristics of "Unmanned Ground Vehicles (UGVs)" and "UAVs" for the purpose of exploiting the benefits and mitigating some drawbacks that are associated with each robot type. The design consists of a quadcopter, 2 large freely rotating passive side wheels, 3 small front wheels for ground stabilization, and an autonomous custom braking system onboard.

IEEEkeywords: Hybrid robot, heterogeneous robot, mobile robots, energy-efficient UAV, taxiing

6.3 Introduction

In recent years, small-sized Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs), have generated a lot of interest in civilian applications. Such applications include retail [D1], logistics [D2], research [D3], and others [D4], [D5]. The continuous development of these machines and recent innovations have helped them enter a wide range of hitherto unexplored domains. Advancement in computer systems, miniaturization of electronics, artificial intelligence, and composite materials is propelling the development of UGVs and UAVs.

The different body structures and forms of both types of robots, UGVs and UAVs, give them different behavioral characteristics, features, and different advantages between them. These features and advantages can be manifested in performing different tasks that are adequate for the robot type. For example, UAVs are superior in terms of mobility due to their ability to utilize the 3D space to get from one place to another. However, if the terrain is UGV friendly, a UGV would cross the same distance with consuming less energy as both carry the same payload. A UGV can be designed to carry a heavy payload (i.e a 10kg payload) for a longer distance and a longer operation time than a UAV carrying the same payload. Hybrid or Self-heterogeneous UGV-UAV robots (UHAGVs), are designed to compromise the trade-off between UAVs and UGVs, making them attractive for researchers to further investigate their potential [D6]. Table 6.1, illustrates the major beneficial differences between UAVs and UGVs. The goal behind the design

Robot Type	**Power consumption**	**Payload capacity**	**Terrain Restrictions**	**Operation Time**
Aerial based	HIGH	LOW	Air Only	LOW
Ground based	LOW	HIGH	Ground Only	HIGH
Self Heterogeneous	MODERATE	MODERATE	Air & Ground	HIGH

Table 6.1: The advantages and drawbacks of different types of robots

of the UHAGV model that is presented in this paper is to create a uni-system machine that confines the benefits of the two types of robots, aerial and ground, while mitigating the disadvantages that are associated with each robot type, therefore, increasing the performance of the resulting vehicle.

6.4 Previous Work

New explorations and design innovations of hybrid ground/aerial robots have been investigated recently by researchers. An interesting approach in [D7] uses passive wheels attached to the frame of a UAV for bridge inspections. The wheels enable it to climb and run on the bridge surface. Research in [D8] presents a design of a UAV that also uses wheels and uses the approach of skateboard steering trucks, which use lateral tilt to affect steering to exploit efficient rolling locomotion to travel long distances on a smooth surface. A similar design in [D9], presents a new all-around Air–Land–Sea UAV that uses floating wheels for mobility. The wheels of this design are used for automatic battery charging as well. Researchers in [D10] introduce a new design of a flying robot with Adaptive Morphology for Multi-Modal Locomotion. They present a prototype that can use its wings to walk on the ground and fly forward. However, these hybrid robot designs either lack the autonomy and the ability to perceive their environment or construct their own local map path plan/navigate autonomously through their environment. Other hybrid designs change their form for extending the mobility [D11], [D12].
In [D13] and [D14], a group of researchers propose a hybrid robot design and add autonomy to it by equipping the robot with various sensors such as a 3D Lidar, a depth camera, a wheel encoder, and a powerful processing unit. Although the authors claim that ground mobility in their platform achieves a reduction in power consumption by 5 times than when flying, the power-hungry processing unit and sensors make their model very power-consuming during operation in any locomotion type.

In this paper, we propose a self-heterogeneous robot that is designed to use a much-simplified mobility approach than other approaches that are mentioned in Section 6.4. The proposed simple design requires much less power-consuming sensors and a much less powerful processing unit with the minor cost of adding lighter-weight hardware, which, enables aerial and ground autonomy in navigation, while reducing costs and power consumption.

6.5 UHAGV Model

6.5.1 Hardware Structure

The UHAGV hardware structure can be explained into 3 main blocks as shown in Fig. 6.1:

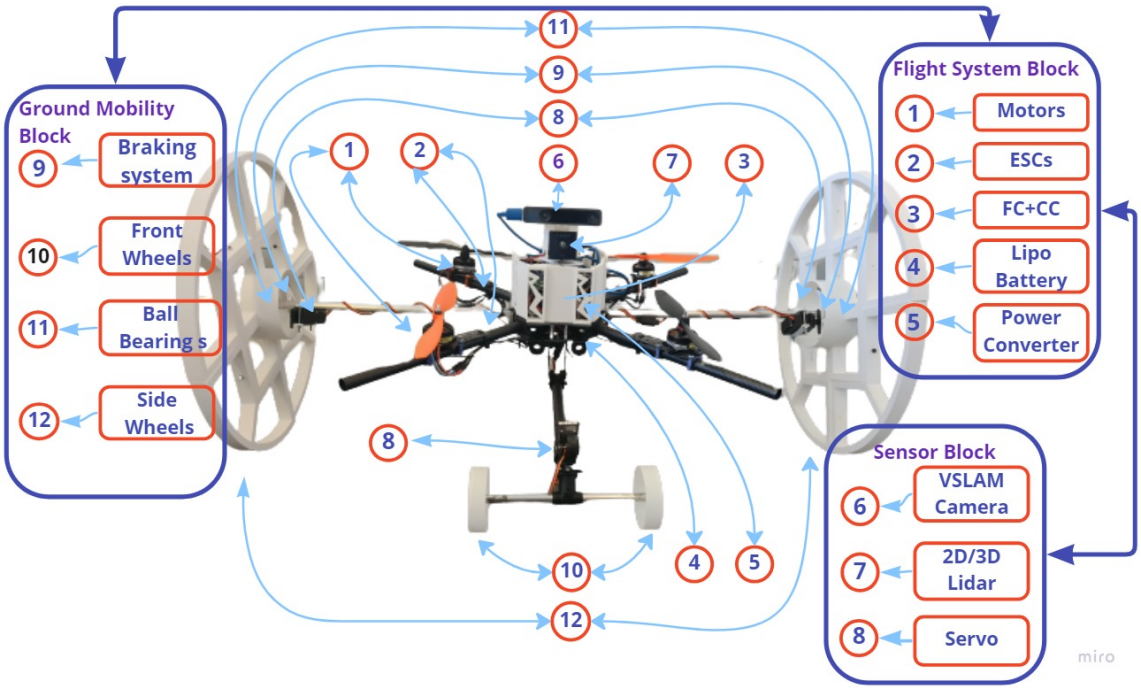


Figure 6.1: Illustration of the Hardware configuration of the UHAGV.

1. *Flight System Block:* this block is considered the main drive block of the robot. It contains the force-generating hardware that is necessary for mobility, ground or aerial. This block contains a companion computer (CC), for processing all the data from the sensor block and is responsible for controlling the navigation decisions of the robot, a Flight controller (FC), for controlling the flight dynamics-related parameters through the direct control of the Electronic Speed Controllers (ESCs) and motors, and other necessary components for enabling a stable flight.

2. *Ground Mobility Block:* this block contains the hardware that enables the ground mobility locomotion for this robot. It contains two lateral large wheels, which move freely on the main aluminum rod using frictionless bearings. The aluminum main rod is fixed to the body of the UHAGV. Two small front wheels are fixed to the base of the UHAGV for two reasons: the first reason is to fix a relative pitch angle of the UAV to the ground, the second reason is to prevent the UAV from over-flipping around the aluminum shaft when braking. Finally, this block also contains a braking system, which consists of carbon fiber rods with high friction pads, these rods are connected to high-torque digital servos. The whole braking system is also fixed directly on the aluminum rod or so-called main axis.

3. *Sensors Block:* the sensor block contains two types of sensors: A proximity sensor and a localization or a VSLAM sensor. The proximity sensor is composed of a low-resolution, low-computational 2D/3D point-cloud camera. The localization sensor is a low-power consumption VSLAM-enabled tracking camera that has built-in IMUs for increasing localization accuracy.

Fig. 6.1 and Table 6.2, illustrate hardware specifications and configuration of the UHAGV.

Hardware Component	Type	Number
Flight controller (FC)	Pixhawk 2.4.8	1
Electronic Speed Controllers (ESC)	60Amps SimonK	4
Motors	Brushless T-Motors 900kv	4
Companion Computer(CC)	RaspberryPi 4	1
Proximity Sensor	2D/3D Lidar	1
Energy Source	4cell 6,000 mAh LIPO battery	1
Position Hold Sensor	VSLAM Camera	1
Side Wheels	Light-weight large wheels	2
Front Wheels	small wheels	3
Servo	Linear electric servo	2

Table 6.2: The Hardware description of the proposed UHAGV robot design.

6.5.2 UHAGV Notations

Fig. 6.2 shows the notations of the proposed UHAGV model, and is explained in the following:

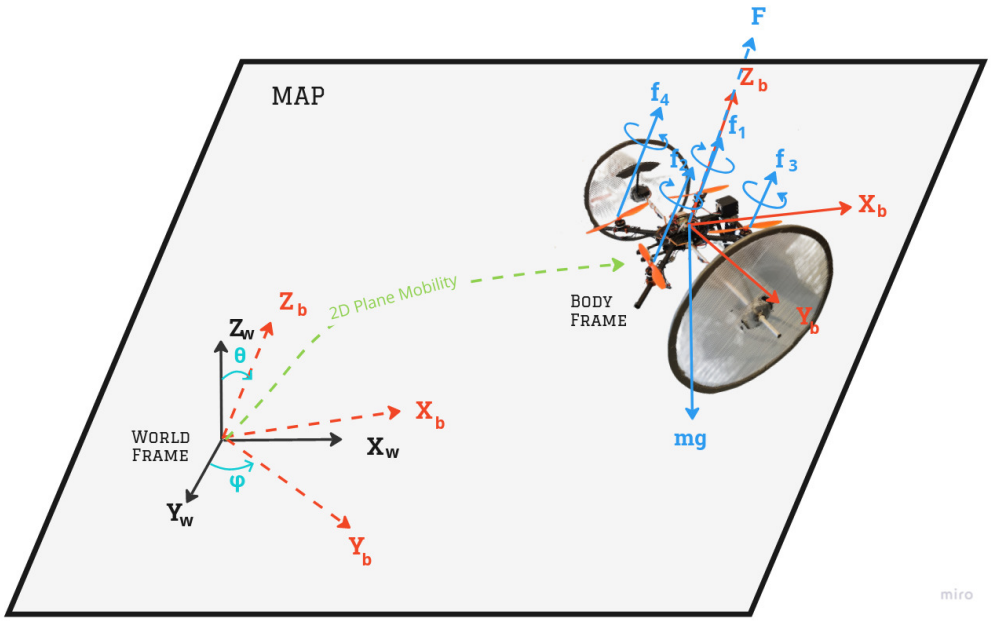


Figure 6.2: A visual illustration of the axis, forces, and angles of the UHAGVs frame to the world's frame.

$\vec{W} = \vec{X_w}, \vec{Y_w}, \vec{Z_w}$	The world inertial fixed frame
$\vec{B} = \vec{X_b}, \vec{Y_b}, \vec{Z_b}$	The UHAGV body fixed frame
$f_i \in \mathbb{R}$	The force generated from the $i'th$ motor along the axis
$F = \sum f_i \in \mathbb{R}$	The sum of the thrust applied from all motors on the UHAGV
$m \in \mathbb{R}$	The mass of the airframe and payload
$g = 9.8m/s^2$	The gravitational acceleration
$N \in \mathbb{R}_{>0}$	The diameter of the wheel

6.6 Navigation

The UHAGV is able to perceive its environment using sensors onboard. Through the 2D/3D pointcloud and self-localization data, a 2D/3D local costmap representing the environment will be constructed. 2D Global and Local path planners are used to generate the path toward the desired target while maintaining a safe distance from obstacles. The generated costmaps and trajectory path planners will

be used for both (ground and aerial) locomotion types on the UHAGV. Fig. 6.3, shows a block diagram of the navigation skeleton that governs the locomotion behavior of the UHAGV.

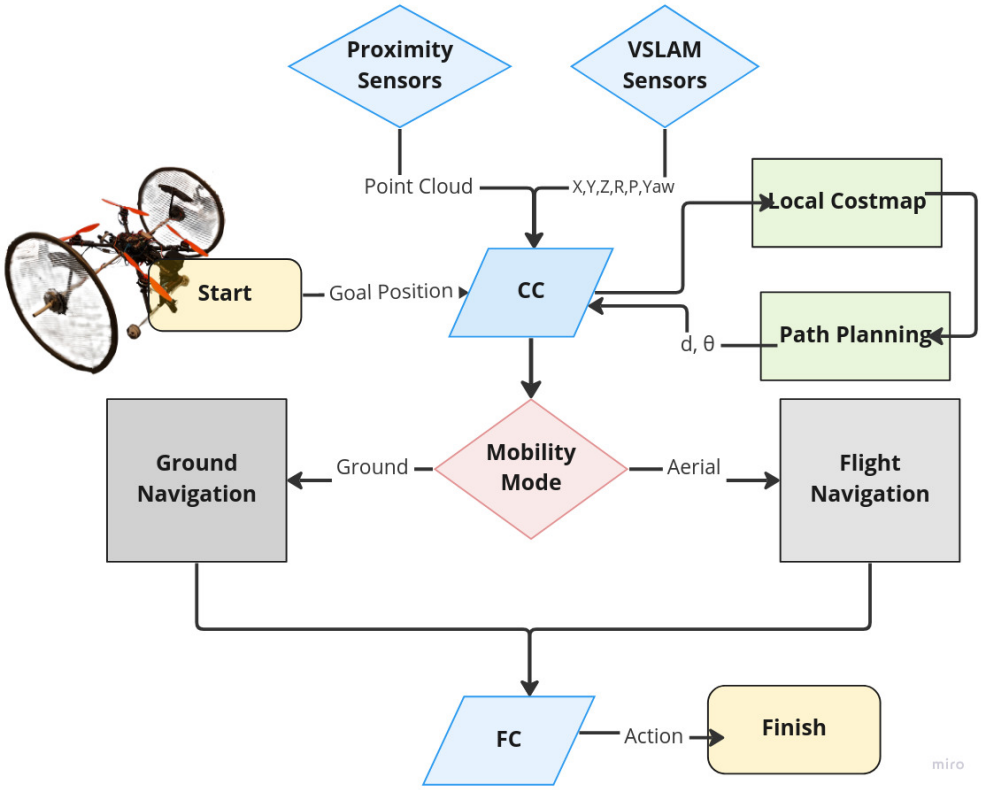


Figure 6.3: A block diagram of the navigation skeleton that governs the locomotion behavior of the UHAGV.

6.6.1 Aerial Navigation:

The classic way of small-sized UAV-guided navigation is to focus on the absolute position of the vehicle without taking care of the velocity vector. Besides the stationary flight capability, the UAV is able to move towards any trajectory in space employing longitudinal and lateral motions where the heading angle is maintained at the origin. However, this strategy is not adequate for the proposed UHAGV design, as most sensors onboard point towards the front direction of the body frame. Non-holonomic navigation is used which enables the UAV to satisfy desired dynamic behavior while tracking the geometric path. Thus, the UAV dynamic model is arranged in such a way that the lateral motion (roll rotation) can be canceled and longitude (pitch motion) and yaw angles would be used. The

UHAGV while flying, is able to reach any position fined in space by its absolute position and the yaw angle, using the forward and heading motions.

6.6.2 Ground Navigation:

The UHAGVs ground navigation is based on 2 main pillars: the motion controller and the navigation controller. The UHAGVs initial state would imply having the brakes actively locking the wheels of any movements, therefore, resisting any ground movements using the brakes regardless of any motor activity.

1. *Motion controller:* to generate forward motion, motors on the UHAGV would need to spin with a constant pre-set thrust value. The UAV body surface plane would be inclined relative to the ground with a pre-fixed pitch angle. The results of this setup enable the generation of a force in the forward direction. For rotating motions (yaw direction), the braking system will lock on one side wheel, while applying the same thrust value as in forward motion. With only one motion-free wheel and the forward direction force, the UHAGV is able to turn in CW or CCW senses. Using this motion strategy, the information of the local path planner is translated using a custom-designed navigation controller to enable the UHAGV to follow the constructed geometric path toward the goal.

2. *Navigation controller:* the UHAGV navigation controller is designed to translate the information given by the planner into ready-to-use navigation information for the UHAGV. The navigation information is a set of position coordinates that guide the UHAGV to reach the target position as shown in Fig. 6.4. The navigation controller uses a strategy that follows the generated local path from the planner toward the final goal destination.

   The global and local planner constructs a path ($\mathbf{Path}$) towards the final goal destination, which is represented as a set of consecutive positions $\vec{p_i} = (x_i, y_i)$ that lead to the final target position. The $\mathbf{Path}$ and $\vec{p_i}$, are the navigation path messages[1] and geometry pose messages[2] that contains world-frame coordinates.The strategy consists of 3 main parts. Due to that the consecutive path positions $\vec{p_i}$ are very close to each other, an optimized path is designed that consists of the path positions $\vec{p_i}$ with a larger distance between them. The goal of this step is to adapt the global/local path to the UHAGV movements. The optimised new path consists of consecutive steps $\vec{S_i} = (x_{s_i}, y_{s_i})$.

[1] http://docs.ros.org/en/noetic/api/nav_msgs/html/msg/Path.html
[2] http://docs.ros.org/en/noetic/api/geometry_msgs/html/msg/PoseStamped.html

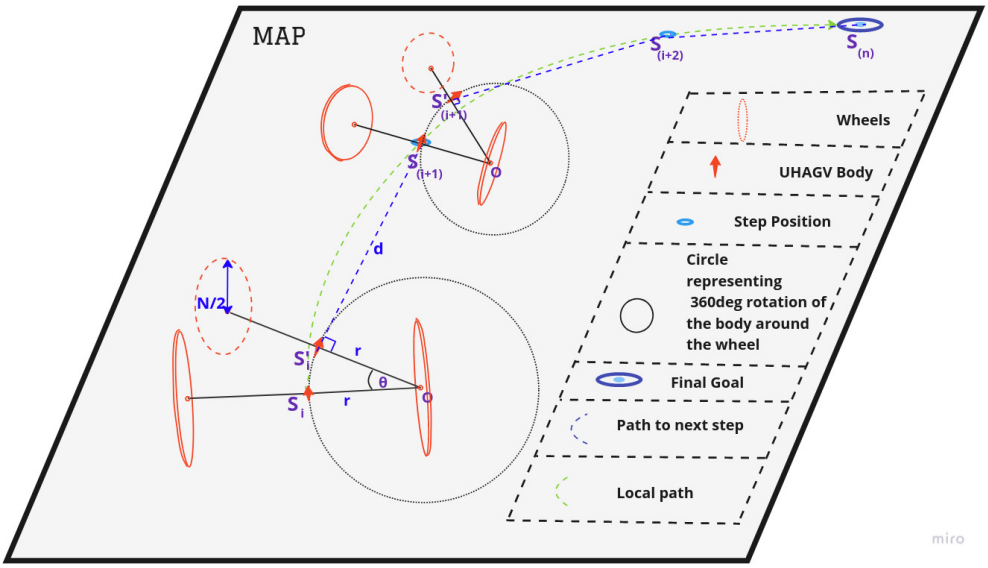


Figure 6.4: A visual illustration of the navigation strategy of the UHAGV controller.

Assuming that the initial position of the UHAGV is at $\vec{S_i} = (x_{S_i}, y_{S_i})$, the second part consists of calculating the position $\vec{S_i'} = (x'_{S_i}, y'_{S_i})$ which would be the result of rotating the body of the UHAGV along one of the chosen lateral wheels untill the the forward axis of the UHAGV points exactly towards $\vec{S_{(i+1)}} = (x_{S_{(i+1)}}, y_{S_{(i+1)}})$. In other words, until the UHAGV faces the next position step.

In order to calculate the point position S_i', we first compute the slope m_1 and m_2 of $\vec{S_{(i+1)}S_i'}$ and $\vec{S_i'O}$ simultaneously as shown in eq. 6.1.

$$m_1 = \frac{y_{S_{(i+1)}} - y_{S_i'}}{x_{S_{(i+1)}} - x_{S_i'}}, \tag{6.1}$$

$$m_2 = \frac{y_{S_i'} - y_O}{x_{S_i'} - x_O}$$

Since $\vec{S_{(i+1)}S_i'}$ is $\perp$ to $\vec{S_i'O}$ and $\vec{O} = (0, 0)$ we get the relation in eq. 6.2 and eq. 6.3.

$$m2 = \frac{-1}{m1} \tag{6.2}$$

$$x^2_{S_i'} + y^2_{S_i'} = x_{S_i'} \cdot x_{S_{(i+1)}} + y_{S_i'} \cdot y_{S_{(i+1)}} \tag{6.3}$$

Knowing the radius $r = 0.5$, which represents half the size of the main axis rod as shown in Fig. 6.4, and applying eq. 6.4 that satisfies the equations of a circle

on eq. 6.3, we get the eq 6.5.

$$x_{S_i'}^2 + y_{S_i'}^2 = r^2 \tag{6.4}$$

$$y_{S_i'} = \frac{r^2 - x_{S_i} \cdot x_{S_{(i+1)}}}{y_{S_{(i+1)}}} \tag{6.5}$$

Applying the value of $y_{S_i'}$ from eq. 6.5 on eq. 6.4, we get eq. 6.6.

$$(y_{S_{(i+1)}}^2 + x_{S_{(i+1)}}^2)x_{S_i'}^2 - (0.5x_{S_{(i+1)}})x_{S_i'} + (0.0625 - 0.25y_{S_{(i+1)}}) = 0 \tag{6.6}$$

We assume that the constants a, b, and c have the values as shown in eq. 6.7.

$$a = y_{S_{(i+1)}}^2 + x_{S_{(i+1)}}^2, \tag{6.7}$$

$$b = -0.5x_{S_{(i+1)}},$$

$$c = 0.0625 - 0.25y_{S_{(i+1)}}$$

Using the quadratic formula in eq. 6.8, $x_{S_i'}$ is calculated.

$$x_{S_i'} = \frac{-b \pm \sqrt{b^2 - 4a \cdot c}}{2a} \tag{6.8}$$

From eq. 6.8 and eq. 6.4 $y_{S_i'}$ is calculated.

The point $S_i{}' = (x'_{S_i}, y'_{S_i})$ will have 2 values, the point closer to the point S_i is the chosen position point where the UHAGV will move towards.

The movement decision of the UHAGV would be decided by the eq. 6.9.

$$Decision = \begin{cases} TurnCW, & \text{if } y_{S_i'} > y_{S_i} \rightarrow \text{the right brake activated} \\ TurnCCW, & \text{if } y_{S_i'} < y_{S_i} \rightarrow \text{the left brake activated} \\ Moveforward, & \text{if } y_{S_i'} = y_{S_i} \rightarrow \text{no brakes activated} \end{cases} \tag{6.9}$$

The third part is the case where the UHAGV decides to move forward, the UHAGV will move a distance of d from point (x'_{S_i}, y'_{S_i}) to point (x_{s_i}, y_{s_i}), which can be calculated by the eq. 6.10.

$$d = ||\vec{S_{(i+1)}} - \vec{S'}_i|| \tag{6.10}$$

6.6.3 Hybrid Navigation:

The UHAGV is able to switch between navigation strategies depending on the situation. The UHAGV, unlike a typical UGV, would be able to simply avoid obstacles in ground navigation and fly over them when needed to reach its goal as shown in Fig. 6.5.

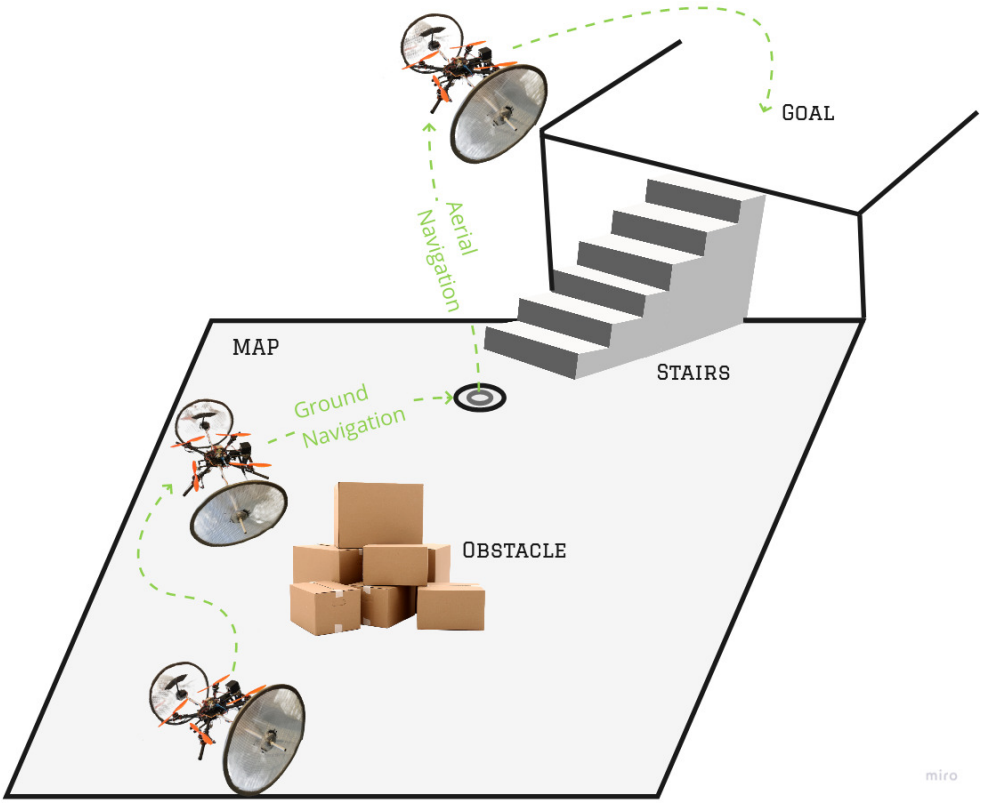


Figure 6.5: An Illustration of the UHAGV switching between ground and aerial navigation.

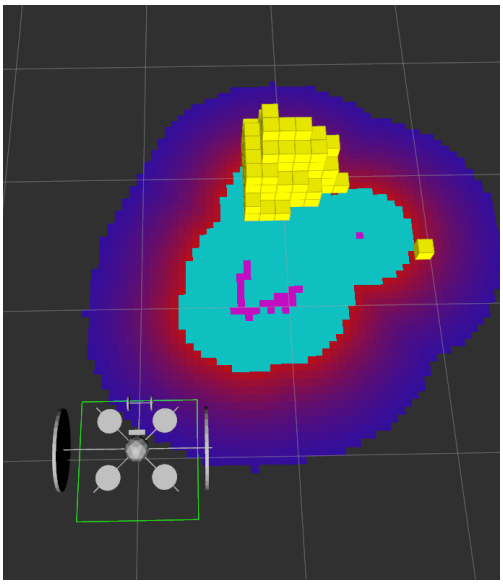
6.7 Experiments

The experiment scenario, as shown in Fig. 6.6a, was conducted in the laboratory. The purpose of this experiment is to test the ground mobility, navigation system, and maneuverability of the UHAGV in the presence of obstacles. The UHAGV would first construct a local costmap that relatively locates the obstacles of its surrounding environment as shown in Fig. 6.6b and then calculate the global and local path toward the desired goal. The algorithm in the navigation controller layer

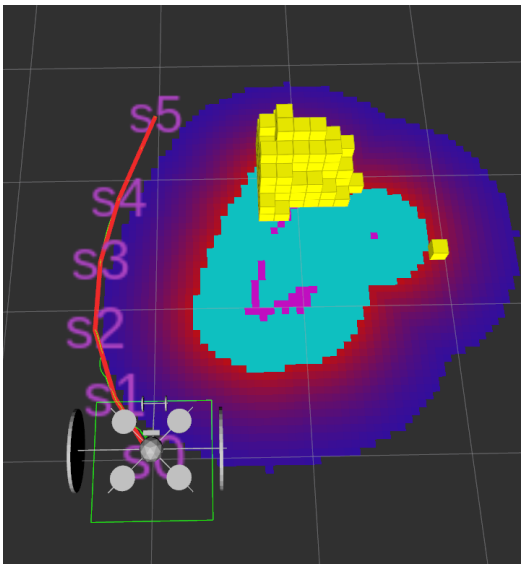
would then reconstruct a modified path as shown in Fig. 6.6c from the global path that then will be passed directly as a set of motion instructions to the FC.

Finally, Fig. 6.6d, shows the recorded successful path throughout the mission that the UHAGV was able to achieve.

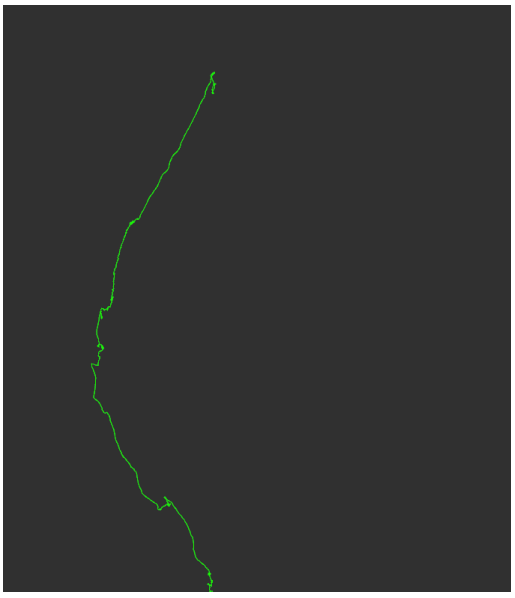
(a) Laboratory illustration of the scenario



(b) Rviz illustration of the constructed 3D local costmap.



(c) Rviz illustration of the constructed paths and steps.



(d) Rviz illustration of the trajectory that the UHAGV took.

Figure 6.6: Images that articulate the Experiment Scenario.

6.8 Conclusions

In this paper, a cost-effective self-heterogeneous robot is designed, developed, and tested. We show in this paper a different mobility and navigation technique that extends the mobility and operation time of a robot. The proposed design uses a combination of a taxiing strategy and friction-based brakes for ground mobility. The paper also shows that by using custom-designed brakes and a fixed relative pitch angle, we achieve to reduce a substantial amount of energy rather than consuming the energy for stabilizing the UAV body on the main axis as in previously designed models in the state of the art. The proposed technique also does not need to use a counter force generated by the motors to stop the robot, however, it uses a simple friction-based braking system that is applied on the wheels to stop the whole robot. As a result of reducing a lot of energy while moving on the ground surface rather than flying, the operation time is therefore increased, which can be very beneficial for increasing the quality of performance of task-oriented robots.

6.9 Future Work

1. *Autonomous adjustment of the pitch angle θ.* At this point, the UHAGV moves and navigates as a result of the F_x applied on its body, which is generated from having a constant thrust and a fixed pitch angle θ. If the UHAGV can autonomously set the fixed pitch angle, therefore, increasing or decreasing θ, it would be able to adjust the velocity without consuming more energy, while remaining the thrust values unchanged.

2. *Computation efficient 3D planner.* The UHAGV navigates following a set of 2D position goals that represent a path defined by a planner. A 3D planner can be used to help decide when to switch the type of navigation from ground to Aerial and vice versa, which would increase the autonomy of this type of hybrid robot, especially for very short obstacles to the ground.

Chapter-6 References

[D1] Nesrine Cherif, Wael Jaafar, Halim Yanikomeroglu, and Abbas Yongacoglu. 3d aerial highway: The key enabler of the retail industry transformation. *IEEE Communications Magazine*, 59(9):65–71, 2021.

[D2] Abdussalam A. Alajami, Guillem Moreno, and Rafael Pous. Design of a uav for autonomous rfid-based dynamic inventories using stigmergy for mapless indoor environments. *Drones*, 6(8), 2022.

[D3] Abdussalam AA Alajmi, Alexandru Vulpe, and Octavian Fratu. Uavs for wi-fi receiver mapping and packet sniffing with antenna radiation pattern diversity. *Wireless Personal Communications*, 92(1):297–313, 2017.

[D4] Alena Otto, Niels Agatz, James Campbell, Bruce Golden, and Erwin Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72(4):411–458, 2018.

[D5] Higinio González-Jorge, Joaquin Martínez-Sánchez, Martín Bueno, Arias, and Pedor. Unmanned aerial systems for civil applications: A review. *Drones*, 1(1), 2017.

[D6] Nur Shahida Roslin, Adzly Anuar, Muhammad Fairuz Abdul Jalal, and Khairul Salleh Mohamed Sahari. A review: Hybrid locomotion of in-pipe inspection robot. *Procedia Engineering*, 41:1456–1462, 2012. International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012).

[D7] Moyuru Yamada, Manabu Nakao, Yoshiro Hada, and Naoyuki Sawasaki. Development and field test of novel two-wheeled uav for bridge inspections. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1014–1021, 2017.

[D8] Jared R. Page and Paul E. I. Pounds. The quadroller: Modeling of a uav/ugv hybrid quadrotor. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4834–4841, 2014.

[D9] Nana Takahashi, Shuhei Yamashita, Yurina Sato, Yuta Kutsuna, and Manabu Yamada. All-round two-wheeled quadrotor helicopters with protect-frames for air–land–sea vehicle (controller design and automatic charging equipment). *Advanced Robotics*, 29(1):69–87, 2015.

[D10] Ludovic Daler, Julien Lecoeur, Patrizia Bernadette Hählen, and Dario Floreano. A flying robot with adaptive morphology for multi-modal locomotion. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1361–1366, 2013.

[D11] Mengjie Zhang, Bo Chai, Lijuan Cheng, Zhaowu Sun, Guang Yao, and Lei Zhou. Multi-movement spherical robot design and implementation. In *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1464–1468, 2018.

[D12] Scott Morton and Nikolaos Papanikolopoulos. A small hybrid ground-air vehicle concept. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5149–5154, 2017.

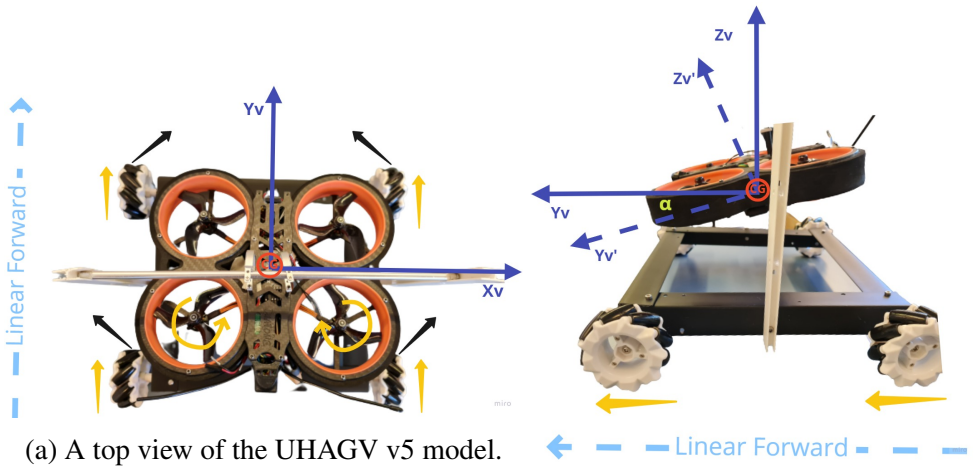
[D13] David D. Fan, Rohan Thakker, Tara Bartlett, Meriem Ben Miled, Leon Kim, Evangelos Theodorou, and Ali-akbar Agha-mohammadi. Autonomous hybrid ground/aerial mobility in unknown environments. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3070–3077, 2019.

[D14] Arash Kalantari, Thomas Touma, Leon Kim, Rianna Jitosho, Kyle Strickland, Brett T. Lopez, and Ali-Akbar Agha-Mohammadi. Drivocopter: A concept hybrid aerial/ground vehicle for long-endurance mobility. In *2020 IEEE Aerospace Conference*, pages 1–10, 2020.

6.10 UHAGV v5

During this thesis, various design layouts were tested for the goal of combining the heterogeneous characteristics and features of aerial and ground robots in one robot design.

One of these design models that is worth mentioning in this thesis is called UHAGV v5 model. The structure idea and operating mechanism of this hybrid robot design is that a UAV is able to use a ground platform with Omni-directional wheels that allow carrying different payloads, while at the same time, the UAV has the ability to detach and fly off when it does not need to use the ground platform or its payload. The ground platform is composed of 4 honolomic-type wheels, which are mounted on a square lightweight frame. The frame of the ground platform is attached to a base with an aluminum rod called the docking axis, which is specially designed for the UAV to be attached or docked on, as shown in Fig. 6.7. The UHAGV v5 model, as for the UHAGV v4 model, uses a taxiing locomotion technique for mobility while attached to the ground platform.



(a) A top view of the UHAGV v5 model.

(b) A side view of the UHAGV v5 model.

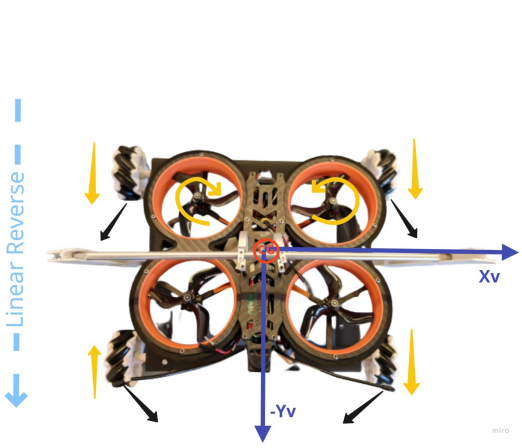
Figure 6.7: An illustration of the axes and direction of movement of all wheels and motors of the UHAGV when moving in the linear forward direction.

Mobility

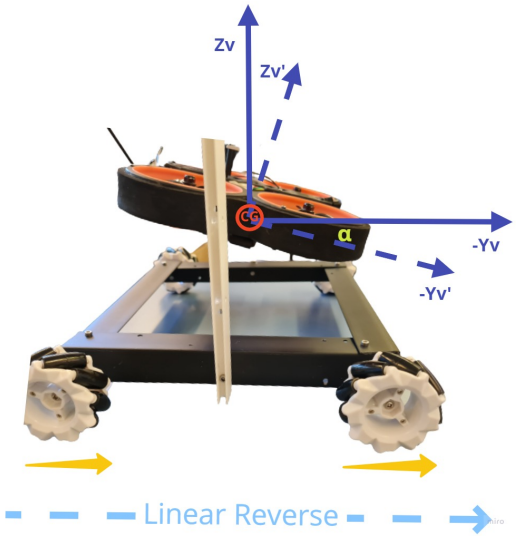
The ground mobility feature on this design operates as the following: *Forward movement:* in order to generate a forward movement for the UHAGV v5, the rear motors would need to spin faster than the front motors on the UAV, thus, inclining

the body of the UAV with a pitch angle α along the docking axis. Since the thrust of a UAV is the amount of upward force the UAV can produce when at full throttle, an adequate thrust constant value K_T and pitch angle α are set, which, therefore, generates a force towards the front axis (+ Y-axis) that enable the movement of the UHAGV in the linear forward direction. Figs. 6.7a and Fig. 6.7b, illustrate the schematics of the forward movement of the UHAGV.

Reverse movement: for generating a backward or reverse movement for the UHAGV v5, unlike the forward direction movement, the front motors would need to spin faster than the rear motors on the UAV, thus, inclining the body of the UAV with a pitch angle α along the main shaft axis. With an adequate thrust constant value K_T and a pitch angle α, a force towards the reverse direction of the body frame ($-$ Y-axis) will be generated to enable the movement of the UHAGV in the linear reverse direction. Figs. 6.8a and Fig. 6.8b, illustrate the schematics of the forward movement of the UHAGV.



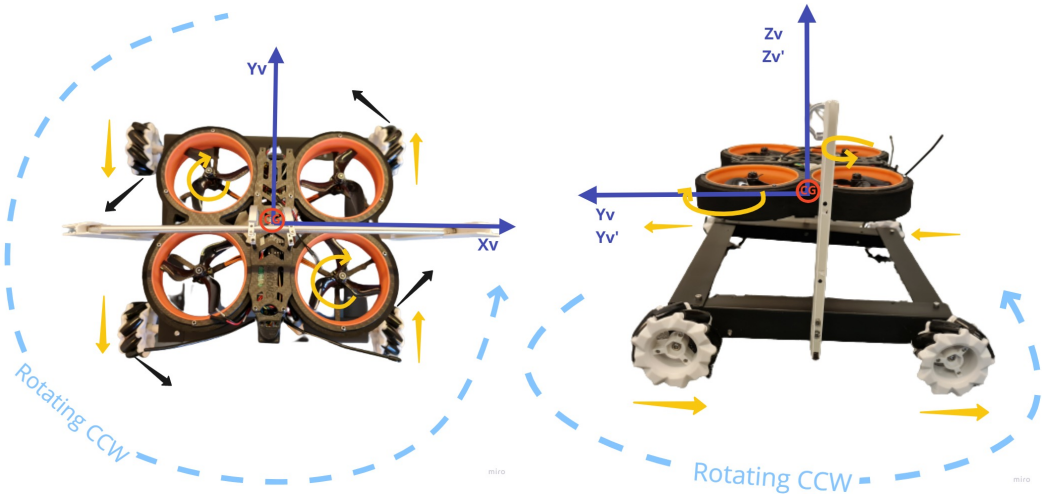
(a) A top view of the UHAGV v5 model.



(b) A side view of the UHAGV v5 model.

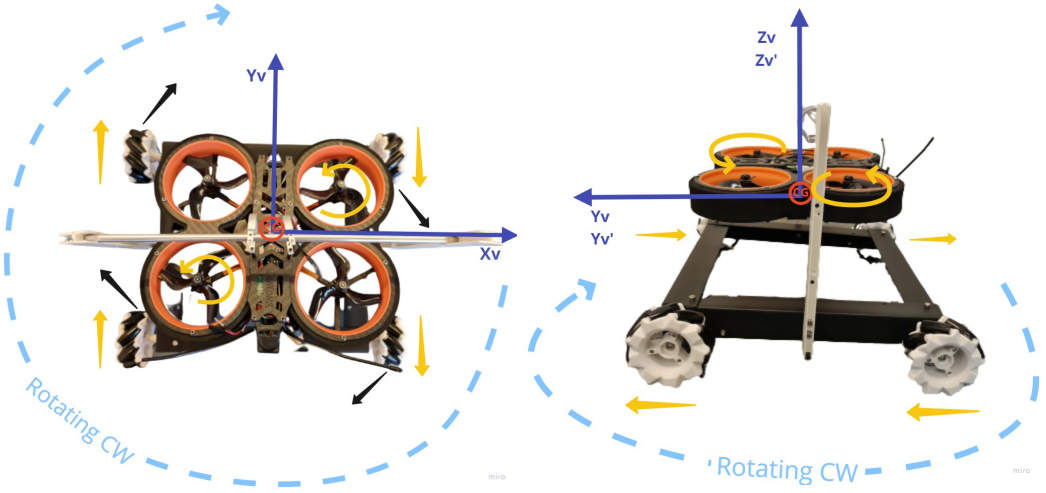
Figure 6.8: An illustration of the axes and direction of movement of all wheels and motors of the UHAGV when moving in the linear reverse direction.

Rotating movements: for applying rotation movements on the UHAGV v5, both in CW and CCW directions, the diagonal motors would need to spin faster than the other motors on the UHAGV. Applying an adequate thrust constant value K_T and a yaw angle β to generate a rotation force (Yaw) around the center point of the UAV and ground platform (Z-axis). Figs. 6.9a, 6.9b, 6.10a, and 6.10b, illustrate the schematics of the rotation movement of the UHAGV in CW and CCW.



(a) A top view of the UHAGV v5 model. (b) A side view of the UHAGV v5 model.

Figure 6.9: An illustration of the axes and direction of movement of all wheels and motors of the UHAGV when rotating in a counter-clockwise direction.



(a) A top view of the UHAGV v5 model. (b) A side view of the UHAGV v5 model.

Figure 6.10: An illustration of the axes and direction of movement of all wheels and motors of the UHAGV when rotating in a clockwise direction.

Navigation

The navigation and autonomous movement of this design will be investigated and designed in future work from the writing of this thesis.

Chapter 7

CONCLUSIONS

This thesis reveals a collective set of conclusions that shares one important goal, which is increasing the performance and automation of task-driven robots towards industry 4.0. This set of conclusions can be briefly described as the following:

1. *UAVs:* exploiting UAVs for performing autonomous inventory tasks in map-less environments was presented in this thesis. These powerful machines prove to be very useful for autonomous inventory missions, especially in large warehouses. However, the technology still faces many constraints and drawbacks compared to ground inventory robots. The major constraints were: the complexity of these machines (in terms of indoor navigation, stability, energy-computation power trade-offs), the high costs associated with experimenting with these machines for research purposes, and the major constrain of having a limited flight time, which makes the overall continuous charging time very time-consuming and research-inefficient. The proposed solution to mitigate the drawbacks of UAVs on research time and cost efficiency, was to design a simulation tool for simulating RFID technology, that enables the simulation of UAVs performing RFID-based inventory missions.

2. *Simulation:* the thesis proposes a simulation tool that enables the simulation of RFID technology that is designed to cooperate with most robot types and forms. Using the proposed simulation tool, more experiments were conducted on UAVs performing autonomous inventory missions without the risk of losing time on continuous charging, while reducing to null the costs of repairing the machines/environment from possible crashes during experimenting in the real world.

3. *Heterogeneous Collaboration:* this thesis also discusses how a team of heterogeneous robots, "aerial", and "ground", can benefit greatly by collaborat-

ing in known environments. The thesis exposes a multi-robot localization method that increases the overall team performance in unexplored environments. This thesis also articulates the benefits of the proposed method on an inventory mission performed by a heterogeneous team of robots in various scenarios. Although enabling the heterogeneous collaboration of a team of robots permits collective intelligence from individual simplicity, however, it will require pre-installing and running many algorithms on the robots which increases the complexity and reduces autonomy. The result of this conclusion is the design of a self-heterogeneous robot that shares the same heterogeneous features of 2 robots, "UAV" and "UGV".

4. *Self-Heterogeneous or Hybrid vehicles:* The thesis also discusses the design of a hybrid/self-heterogeneous robot for the purpose of increasing the performance of an individual robot. The hybrid vehicle design, which combines both features and characteristics of ground and aerial vehicles, was able to show initial experimental results that show advantages in reduced energy consumption, increasing the maximum operation time, and increasing the distance traveled. The proposed UHAGV design, theoretically, has a superior exploration performance than the other conventional ground or aerial robots, which is considered one of the main pillars on which this thesis is based.

Thesis References

[T1] Jack Collins, Shelvin Chand, Anthony Vanderkop, and David Howard. A review of physics simulators for robotic applications. *IEEE Access*, 9:51416–51431, 2021.

[T2] Maziar Arjomandi, Shane Agostino, Matthew Mammone, Matthieu Nelson, and Tong Zhou. Classification of unmanned aerial vehicles. *Report for Mechanical Engineering class, University of Adelaide, Adelaide, Australia*, pages 1–48, 2006.

[T3] Xingbang Yang and Xuan Pei. 15 - hybrid system for powering unmanned aerial vehicles: Demonstration and study cases. In Massimiliano Lo Faro, Orazio Barbera, and Giosué Giacoppo, editors, *Hybrid Technologies for Power Generation*, Hybrid Energy Systems, pages 439–473. Academic Press, 2022.

[T4] Regina Fazio Maruca. Retailing: confronting the challenges that face bricks-and-mortar stores. *Harvard Business Review*, 77(4):159–159, 1999.

[T5] Mikko Kärkkäinen. Increasing efficiency in the supply chain for short life goods using rfid tagging. *International Journal of Retail & Distribution Management*, 31:529–536, 10 2003.

[T6] Marc Morenza-Cinos, Victor Casamayor-Pujol, Jordi Soler-Busquets, Jos Luis Sanz, Roberto Guzm, and Rafael Pous. *Development of an RFID Inventory Robot (AdvanRobot)*, pages 387–417. Springer International Publishing, Cham, 2017.

[T7] Pous, R. and De Porrata-Doria, R. (2018) automated inventory taking moveable platform: advanrobot. `https://patents.google.com/patent/US9939816B2`. accessed November 14, 2019.

[T8] Keonn's AdvanReader 160 keonn's advanreader 160. `https://keonn.com/components-product/advanreader-160/`. Accessed: 2022.

[T9] Keonn's AdvantennaSP11 keonn's advantennasp11. https://keonn.com/components-product/advantenna-sp11/. Accessed: 2022.

[T10] Ali Abdul Khaliq. *From Ants to Service Robots: an Exploration in Stigmergy-Based Navigation Algorithms*. PhD thesis, Örebro University, 2018.

[T11] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. A comprehensive survey of visual slam algorithms. *Robotics*, 11(1):24, 2022.

[T12] Eitan Marder-Eppstein. MoveBase move base is a navigation package for robots., 2018. Accessed: 2022.

Chapter 8

LIST OF PUBLICATIONS

8.1 International journal articles

1. Alajami, Abdussalam A., Guillem Moreno, and Rafael Pous. 2022. "Design of a UAV for Autonomous RFID-Based Dynamic Inventories Using Stigmergy for Mapless Indoor Environments" Drones 6, no. 8: 208. doi: 10.3390/drones6080208 **[First-Auther]**

2. A. A. Alajami, G. Moreno and R. Pous, "A ROS Gazebo Plugin Design to Simulate RFID Systems," in IEEE Access, vol. 10, pp. 93921-93932, 2022, doi: 10.1109/ACCESS.2022.3204122. **[First-Author]**

3. Alajami, Abdussalam A., Nil Palau, Sergio Lopez-Soriano, and Rafael Pous, "A ROS-based Distributed Multi-robot localization and orientation Strategy for Heterogeneous Robots," in Intelligent Service Robotics 2022. **[First-Author, Accepted]**

4. Alajami, Abdussalam A. and Rafael Pous, "Design of an Energy-efficient Self-Heterogeneous Aerial-Ground Vehicle", in Drones journal, 2022. **[First-Auther, Under preparation]**

5. V. Casamayor-Pujol, B. Gastón, S. López-Soriano, A. A. Alajami, and R. Pous, "A Simple Solution to Locate Groups of Items in Large Retail Stores Using an RFID Robot," in IEEE Transactions on Industrial Informatics, vol. 18, no. 2, pp. 767-775, Feb. 2022, doi: 10.1109/TII.2021.3080670. **[Co-Author]**

8.2 Conference proceeding

1. A. A. Alajami, R. Pous and G. Moreno, "Simulation of RFID Systems in ROS-Gazebo," 2022 IEEE 12th International Conference on RFID Technology and Applications (RFID-TA), 2022, pp. 113-116, doi: 10.1109/RFID-TA54958.2022.9924062.**[First-Author]**

2. Alajami, Abdussalam A. and Rafael Pous, "Design of an Energy-efficient Self-Heterogeneous Aerial-Ground Vehicle", in ICARA 2023: 2023 9'th International Conference on Automation, Robotics, and Applications, 2023. **[First-Author, Accepted]**