Map-less inventory and location for an RFID-based robot

Victor Casamayor Pujol

TESI DOCTORAL UPF / ANY 2020

Director de la tesi
Rafael Pous
Deparatment Departament de Tecnologies de la Informació i les Comunicacions

To a world where robots work hard and humans can enjoy their lives.

Agraïments

En primer lloc voldria agrair al meu supervisor, el professor Rafael Pous, l’oportunitat de realitzar aquesta tesi, així com el seu suport, consell i confiança. Igualment voldria mencionar a tots els membres que han format part i formen part del grup de recerca UbiCA Lab, gràcies a ells sempre m’he sentit acompanyat i ajudat. En aquest sentit, voldria donar especialment les gràcies al Dr. Marc Morenza. He tingut la sort de compartir amb ell molts moments: difícils, intensos, alegres, emocionants però sempre molt edificants. Sense la seva empenta i suport aquesta tesi no hauria estat la mateixa.

També voldria agrair a tots els membres del Departament TIC i, en general, a tots els membres de la Universitat Pompeu Fabra per contribuir a un espai de treball amable i enriquidor. Voldria també mencionar especialment aquells companys i companyes amb els que he compartit assignatura i als membres del secretariat que sempre han facilitat la nostra feina. No puc oblidar-me de tots els membres de la biblioteca del campus del Poblenou que han donat un suport inestimable a la meva recerca.

Voldria agrair a Keonn i a tots els seus membres els recursos i les oportunitats que han posat al meu abast durant aquesta tesi. Especialment, voldria destacar el suport, l’enginy i les hores de dedicació del José Luis Sanz al desenvolupament dels robots.

Des d’un punt de vista més personal voldria destacar a les companyes i companys que he fet durant aquesta etapa, confio que ara passaran de companys a amics. Parlant d’amics, vull agrair especialment a la Laura tots aquests anys plegats a la universitat, la seva companyia i consell han fet ridículament fàcils i lleugers aquests anys.

Per últim voldria donar les gràcies als més propers. Especialment, al Gregorio i a la Maria, també coneguts com a pare i mare, és evident que sense ells tot això no hauria estat possible, però igualment voldria agrair el seu suport incondicional en tots els moments. A la Clara qui també m’ha donat suport i consells lingüístics molt valuosos. Espero que, tot i la distància, puguem seguir compartint moments i il·lusions. Finalment, voldria agrair a l’Anna haver estat al meu costat durant tota aquesta etapa, per suportar-me, per entendre’m i per estimar-me: Gràcies.

Abstract

This thesis presents a new paradigm for RFID-based inventory robots. This map-less operation increases the operative autonomy of the robots as they no longer require a mapping step. This new paradigm is based on the stigmergy concept.
Additionally, this new paradigm leads to a simplification of the robot design and allows the cooperation among multiple robots, increasing the robustness and scalability of the system while reducing its cost.
The stock-counting problem is defined and an algorithm based on stigmergy is proposed as a solution, which is initially tested in simulation, an later in real scenarios.
This thesis details the design process and development of two robots that can take advantage of this new paradigm and that are tested in a real environment, the library of the university.
Finally the thesis also presents a new RFID groups location algorithm aligned with the main characteristics of the new paradigm: simplification and efficiency.

Resum

Aquesta tesi presenta un nou paradigma per als robots d'inventari basats en RFID. Aquest no requereix un mapa de l'entorn, així s'augmenta l'autonomia operativa dels robots. El nou paradigma està basat en el concepte d'estigmergia.
A més, permet la simplificació del disseny dels robots, i de manera inherent, la coordinació entre ells. Així, la robustesa i l'adaptabilitat del sistema augmenta a la vegada que el cost es veu reduït.
La tesi descriu el problema de "stock-counting" i proposa un algorisme com a solució, inicialment es desenvolupa i prova en una simulació basada en grafs.
També es detalla el procés de disseny de dos robots per aprofitar els avantatges d'aquest nou paradigma. Els robots són provats a la biblioteca de la universitat, obtenint uns resultats molt satisfactoris.
Finalment, es presenta un algorisme de localització de grups d'etiquetes RFID que s'alinea amb les característiques del nou paradigma: simplicitat i eficiència.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 General context

Traditional retailers are increasingly adopting the RFID technology as the key technology in order to stay competitive against online retailers, such as Amazon. As an example, a survey from 2018 said that 24% of Amazon's revenue came from customers facing stock-outs at their local stores[1]. This decision is mainly taken due to the capacity of RFID technology for improving stock-counts leading to high-resolution stock management. Obtaining this level of detail on the stock management is a crucial step for traditional retailers. As an example, out-of-stocks and overstocks cost retailers 450$ billion per year (252$ billion in North America alone)[2].
Nevertheless, the situation due to the harsh competition by online retailers is getting worse. For instance, recent news from the United States of America have coined the term *retail apocalypse*. This is because during the first quarter of this 2019 more stores closed than during the whole

[1] https://www.ihlservices.com/news/analyst-corner/2018/06/24-of-amazons-revenue-comes-from-customers-who-experienced-out-of-stock-at-local-retailer/

[2] http://www.crmsearch.com/retail-robots.php

2018[3]. Obviously, this trend will cross the ocean and it is expected to be experienced also in Europe[4].
Nowadays, the Covid-19 crisis has deteriorate the situation of traditional retail. With the stores closed by law, only those with online sells can have a chance to stand against it. On the contrary, online retailers have continued servicing their customers and, in some cases, selling even more than before this crisis.
Fortunately for traditional retailers, RFID technology can improve more aspects than only stock-counts and its derivatives such as reduction of overstocking or elimination of stock-outs. Among several related features, it can enhance customer experience with faster and simpler points of sale, or it can help fulfilling online orders in stores in a single day. Also, it can improve security with loss prevention systems based on RFID, it can help detecting misplaced items or it can help to create statistics of the store such as a money-mapping to check the profitability of the different areas of the store.
In any case, the situation is critical and the main question retailers need to ask is: *How can we obtain enough profits from RFID technology in order to compete effectively against online retailers?*
The most common idea among retailers to fight back the online ones is using RFID technology to obtain high resolution information for stock management in order to become omni-channel retailers. This would integrate the best of the online world with the best of traditional retail. One could buy online but pick the product in the store or buy it in the store with personalized attention and then have it shipped to one's home, etc. The key concept behind it, is that one can buy a product or receive it using any channel available. In contrast to online retailers, traditional ones have the infrastructure of stores and warehouses to be able to become omni-channel.
The first step in the way of becoming an omni-channel retailer is reduc-

[3]https://www.businessinsider.com/retail-apocalypse-start-of-2019-more-store-closures-all-of-2018-2019-4?IR=T

[4]https://www.dw.com/en/as-us-uk-retail-apocalypse-deepens-eu-chains-grow-nervous/a-44271346

ing the gap between what retailers think they have in their stores and what actually is there. Common inventory accuracies for retailers that have not adopted RFID is below 80%. Retailers without RFID compute their stock-count by adding to their system the products that arrive to the store and subtracting those that are counted at the points of sale. But this results in a very inaccurate inventory due to shrinkage (errors and thefts). Another handicap for traditional retailers without RFID is that they misplace products in their own store. Customers usually change the positions of items or take them to the fitting rooms, and the item ends up left there or simply not in its original position. This issue, which is rather frequent, has an effect similar to an out-of-stock.
Leveraging RFID technology retailers will gain the knowledge about their items in three dimensions. They will know:

- what they have
- when they have it
- where they have it

Thus, the first dimension makes reference to what do they have in the store, and they can know it at Electronic Product Code (EPC) level, which uniquely identifies every item. Previously, the encoded information in a bar-code was at Stock Keeping Unit (SKU) level, which does not identify items uniquely, but families of items. As an example for an apparel store it would identify all the T-shirts of an specific model, color and size. Moreover, as previously stated, the accuracy on the stock-count also increases dramatically.
The second dimension is time. It describes the frequency in which the information about the store is available and updated. In previous systems without RFID the information might be always available, but not updated, a complete inventory of a store was done at most once per trimester. Now, with RFID, the update frequency of the information is key to reach a high-resolution stock management, some systems can provide real time inventory status.

Besides having a precise, complete and frequent inventory, another benefit from using RFID is knowing the location of all the items in their store, which can be understood as the third dimension provided by RFID. With this information retailers can add more value to their stores by creating money-mappings or they can optimize the picking of items, and also, by finding lost items they can improve the stock-counting results.
There are several RFID-based inventory systems which allow getting all these benefits from the RFID technology. Obviously, depending on the system the results obtained change.
There are four basic RFID systems available for retailers (Table 1.1 summarizes the systems):

- **Hand-held devices**. This is the most basic system, it consists of a hand-held device that contains a small antenna and a reader. It usually has a screen in order to configure some options and it gives visual or auditory cues when it is reading tags. They are the cheapest and easiest to adopt, however, they need human intervention to perform the stock counting, so they are not cheap to operate. In addition, in large spaces the system will be prone to human errors. Humans are not efficient performing tedious and repetitive tasks, so, when taking inventory in large spaces it is highly probable that some aisles will be unintentionally skipped. Depending on the store size and organization they allow at most one stock-count per day, but in medium or large stores, the full stock-count is obtained at the end of the week. In other words, the employees stock-count a few areas of the store each day, aiming at having counted the whole store by the end of the week. Last but not least, they do not provide location data.

- **Smart shelves**. This system consists on adding readers and antennas in all the shelves of the store. It provides precise, automatic and practically real-time stock counts. Also, it can provide location of the items with the only uncertainty of its position inside its own shelf, so it is very accurate. The downside of this system is the very high acquisition and installation cost.

- **Overhead RFID systems**. It consists on adding many RFID antennas and some readers hanging from the ceiling. So, taking inventory is automatic and almost real time. It has worse stock-counting accuracy than the previous system (smart shelves) but it is slightly cheaper, however, its price is still a big barrier for retailers. Regarding location, it can estimate the location of the items with an uncertainty of a few meters.

- **Inventory robot**. This solution mixes two different technologies: RFID and robotics. It allows almost automatic inventory with very high accuracy on stock-counting. However, it is not real time inventory, nevertheless, it can offer a daily inventory. In terms of location it has an accuracy of less than 1m. The system is not as expensive as the previous two, but it has not yet penetrated the market because it is not as affordable as a hand-held device, in terms of an initial investment.

Table 1.1 summarizes the RFID inventory systems. It can be observed that the system providing the best trade-off between cost and the three axis of the RFID knowledge is the inventory robot. It has the best inventory and location accuracy, together with smart shelves. Its acquisition cost is lower than the latter and the installation cost almost negligible, nevertheless, the operation cost is slightly higher, although, it is low compared to the hand-held devices. Also, it can have a daily update on the stock count, which is less frequent than smart shelves or overhead systems, but it is enough for a high-resolution stock management.

1.2 Motivation

Inventory robots are seen by most retailers as the best alternative to take inventories using RFID technology. However, they have a huge drawback, they are not as autonomous as one could expect. Inventory robots require a set up process that depends on the store, so it has to be done at each store and it requires an amount of time proportional to its size. Moreover,

	Hand-held devices	Smart shelves	Overhead systems	Inventory robot
Inventory Accuracy [%]	85	99	85	99
Location Accuracy [m]	None	0.5	3	0.5
Time resolution	Weekly	Real time	Real time	Daily
Acquisition cost [k€/m^2]	Low	High	High	Medium
Installation cost [k€/m^2]	Low	High	High	Low
Operation cost [k€/m^2]	Medium	Low	Low	Low

Table 1.1: RFID inventory systems comparison

this set up process has to be repeated if the store changes its layout.
This process is usually called mapping. It consists on moving the robot through the store while its sensors capture the characteristics of the store's layout, creating a map. Afterwards, the robot will be able to recognize its location in this map, and therefore, in the store. In addition, linked to this process, there is another one that adds a set of navigation goals to this map, commonly known as waypoints, which are locations on the map that the robot will have to reach in order to complete the inventory or location mission.
This mapping process allows the robot to navigate through the store, and therefore, to perform any mission. However, having a map updated to the latest store layout ensures that the robot will be able to cover the entire area of interest, without forgetting any part. In addition, having an updated map significantly reduces the risk of a robot getting lost in the store. A robot is lost when its estimation of its location in the store is different from its real location. These kind of situations can lead, in the best scenario to an aborted mission, but, in the worst scenario, into physical damages due to a robot crash if it tries to enter into forbidden areas such

as stairs, doors, elevators, etc.
The main concern for retailers, is that performing this process requires training employees of the store, who periodically will have to devote part of their time to this process instead of performing their usual store operations or simply helping customers.
To sum up, the mapping process is making the adoption of inventory robots by retailers more difficult, a solution that could help them stay competitive with respect to online retailers.

1.3 Contribution

The main contribution of this thesis focuses on the development of a map-less method allowing RFID-based inventory robots to take inventory of any space without the need of any prior knowledge or any mapping set up process. The method requires only that the items to be inventoried are tagged with RFID labels.
During this thesis, an RFID-based inventory robot that is able to take advantage of the map-less algorithm has also been developed. Due to the characteristics of the map-less method, the overall cost of the developed RFID-based inventory robot has been reduced by roughly an order of magnitude with respect to the current RFID-based inventory robots in the market.
Moreover, it has been shown that the map-less algorithm can be applicable to multi-robot decentralized systems, which greatly improves system characteristics such as scalability or robustness.
Simulations and real scenario experiments show the validity and feasibility of the map-less method using the robot developed during this thesis.
Finally, an RFID-based location algorithm for groups of RFID labels has been developed. This algorithm takes some hypothesis, such that the need of changing the RFID parameters with respect to an inventory mission, and it adds simplifications in line with the designed robot that allow improvement of the location results while making them more sensitive to the user needs.

1.4 Organisation

The thesis is divided into 6 chapters. The second chapter is the background which covers the main topics of the thesis. The third chapter describes the new map-less paradigm for taking inventory. The fourth chapter follows the design of a robot in order to test the map-less inventory paradigm. Chapter five validates the algorithm and the robot developed in a real environment. The sixth chapter presents a new method for locating groups of RFID tagged items. Finally, the seventh chapter is devoted to the overall conclusions and future work.

Chapter 2

BACKGROUND

2.1 RFID Technology

RFID stands for Radio Frequency IDentification. This technology is able to identify an electronic label by illuminating it with radio frequency waves. A key advantage of this technology is that one does not need direct visual contact between the antenna that emits and the electronic label (or tag) that stores the information. In addition, these electronic labels are usually passive, so they use the energy from the requesting wave in order to respond with its encoded information. This technology allows the unique identification of several tagged items simultaneously. These key assets of RFID technology enable the evolution from bar-codes to RFID tags.

An RFID system has three main elements: the reader that emits the requesting wave, receives the responding wave and decodes the information; the antennas that radiate the emitted signal and capture the response; the electronic labels that store the information to be retrieved and communicate with the reader via the antennas.

All electronic labels or tags used in the context of this thesis are passive, and all of them contain the Electronic Product Code (EPC) of the product. The total number of codes possible with the 96 bits that form this EPC code makes it possible to identify every physical item in the world

uniquely. Again, this is a huge benefit with respect to bar-codes, where the storage of data is limited and it usually only encodes the SKU (Stock Keeping Unit) that represents a group or family of unique items. For instance, in retail usually encoded as 13 digits and in a bar code, all T-shirts for a given model, color and size would share the same SKU code.
The Over The Air (OTA) RFID protocol standard used in this research is GEN2 defined by GS1[1] which uses UHF (Ultra High Frequency) at around 900 MHz. This standard defines three sessions (S0, S1 and S2) in which the reader and the electronic labels can communicate. This sessions define the protocol of communication between these two agents. In session S0 the tags will always keep responding to the reader request as long as they receive the RF wave. Session S1 sleeps the electronic labels after their response for a few seconds, up to 1 minute, depending on the label. Finally, session S2 sleeps the tag after its response while it is illuminated, so to get a second response from that tag one must stop illuminating it with RF waves and then illuminate it again to have it responding again. These sessions are designed to enhance different uses of RFID technology. For instance, S2 usually allows the best results in stock counting in very dense environments, this is achieved because those tags with less capacity to answer the call have an easier opportunity when all the surrounding tags are sleeping due to a minimization of interactions between RF waves. On the contrary, if the objective is to get as many readings as possible of the surrounding tags, then S0 fits this need, as all the tags that can answer will do so continuously.
All the RFID systems used during this thesis are Commercial Off-The-Shelf (COTS) systems that are commercialized by Keonn[2]. The RFID readers used are of the model AdvanReader-150.03[3] and the antennas are Advantenna-SP11 [4]. This research aims to enable a real application with RFID components, therefore, it is relevant that the hardware used is not modified.

[1]https://www.gs1.org/epc-rfid
[2]https://www.keonn.com/
[3]https://www.keonn.com/rfid-components/readers/advanreader-150.html
[4]https://www.keonn.com/rfid-components/antennas/advantenna-sp11.html

At the frequency in which the RFID standard works, the radio waves suffer many interactions with the environment. The waves bounce against many different materials, also they are absorbed by other materials, like water. Thus, creating a physical model is not feasible without taking into account the environment, which means that creating a physical model that works in any condition is not possible. Actually, the defined sessions (S0, S1 and S2) help different applications to succeed, despite this interaction with the environment. In any case, for location purposes it would be very helpful being able to model an RFID system, but as it is complex and very dependent on the environment. The model that could be obtained might not work properly in any condition.
Nevertheless, several studies have created models in which a distance to a tag is modelled with the Received Signal Strength Indication (RSSI) of the returning signal such as in [1]. Also in some occasions, but less frequently the phase of the returning signal is used also for computing the distance as in [2], in these cases it is usually more complex due to the size of the wave length, which is around 30cm. Due to this short wave length and the possible bounces, it is very unlikely to infer the distance to a tag by comparing the initial phase with the final one. Other techniques use both the RSSI and the phase of the signal trying to take advantage of all the available data, such as in [3]. Finally, there are some works in which there is also a model not only for the location but also for the occurrence of detection. This is relevant because not receiving the signal of an RFID tag does not mean that the tag is not there, and these kind of events are important if a probabilistic approach for the location estimation is taken.

2.2 RFID-based inventory and location

As it has been introduced in section 1.1 RFID has become the de facto standard to identify products in retail. It enables a faster and more accurate stock-count of the products in a store, but it also enhances the traceability of the products during their whole life-cycle. With RFID it

is possible to know the exact origin of any given product. This leads to a better understanding of all the processes in the business and helps optimizing them.

This work focuses on inventorying and locating all products in a store or a library, but it does not assess other benefits beyond improving the knowledge of what, when and where are the items inside this space, such as how the customer experience can be enhanced or the benefits that can be obtained in the supply chain management.

In this regard, the figure that will be used to assess the performance of stock-counting is the accuracy. Accuracy represents the amount of RFID tags that have been detected over the total amount of tags present in the store. However there are two characteristics that are worth mentioning. First, RFID tags are universally unique, therefore it is enough with detecting them once, in contrast with bar-codes, because in that case it is not only required to obtain the SKU but also to keep a count of each SKU in order to know the stock-count of the store. Second, there may be many RFID tags in the environment, not only those that are labeling products, therefore, only tags that belong to the store should be counted. This list of tags that belong to the store is usually referred to as the base-ground, base-line or ground-truth of tags.

This leads to a problem: how can one be sure of the accuracy obtained if there is an uncertainty over the amount of tags that are of the store. In some occasions, by decoding the tags that are not on the base-line it would be possible to know if they belong to the stock of the store or if it is just a random tag detected. However, this option will not always be available, in this sense a method to assess the whole base-line is detailed in [4] in which after getting the detections repeatedly from several devices and integrating them with the perpetual inventory record a better and more reliable base-ground can be obtained.

Regarding the location of RFID-tagged items, the figure used to assess the results obtained will also be an accuracy. But, in this case, it will be computed as the amount of items that are located correctly over the total amount of items assessed.

2.3 Mobile autonomous robots

In the last years, autonomous robots have been changed from a science fiction promise to an everyday technology, but still, there are several issues that need to be solved in order to get all the benefits from this technology.
Our image of robots is shifting from those humanoid robots that appear in science fiction to autonomous robots that have a precise task to perform and that are specially designed for it, such as all the vacuum cleaners that now live in many homes. One of the robots that will soon have a huge impact in everyday life will be autonomous vehicles, again far from those humanoids and without the need to be different than any current car. In this sense, the design of autonomous robots can take advantage of focusing on the task to be solved without having to care about the humanoid aspect.
When the task that an autonomous robot has to perform is collaborative, specifically with humans, the robot is called cobot, from collaborative robots. In this sense robots for retail will be cobots. There are many different tasks to perform in a store, from stock-counting to helping customers or moving goods from the warehouse to the shop window. Some of these tasks will be more suitable for a robot and others for a human, but some collaboration will be needed among both agents in order to maximize efficiency. Our robot is collaborative, but not with humans yet, however, we have the vision of the stores as spaces where different types of robots perform tasks with the collaboration of the store's staff in order to provide the best possible service to customers.

2.3.1 Mobile robots

This thesis focuses on autonomous mobile robots that operate in indoor environments. There are several categories of this type of robots depending on their purpose. In this case, the work is based on the so called service robots that will perform tasks for humans.
Regarding indoor environments, there are two main issues that impact the

technology. On one side, there is no GPS signal available, so there are no means to obtain a global localization. In this context, localization makes reference to the position and orientation of the robot, also known as robot pose. On the other side, there is usually no direct sunlight allowing the use of sensors that would otherwise be blinded by it.

Mobile autonomous service robots for indoor spaces require very specific robotic technologies in order to navigate these environments. In the first place, the technology that enabled these systems is called Simultaneous Localization and Mapping (SLAM), see [5] and [6] for a complete explanation on its history, developments and the math behind it. Briefly, by using SLAM, robots are able to create a map of the environment while simultaneously estimating their own localization in this map. There are several SLAM algorithms. However, a Bayesian approach was first presented in [7] which led to the solutions that are currently used more frequently such as the one presented in [8] and [9].

In order to create a map, any robot requires a sensor able to detect landmarks on the environment. The most common sensor to generate maps is a LIDAR that stands for Light Imaging, Detecting, and RAnging. This sensor emits light and by sensing the reflected light it measures the distance to an obstacle. The advantages of this sensor at mapping are that the range of the sensor is of several meters with high precision, typically lower than 1cm. In addition, due to its fine controlled rotation, they can achieve an angular precision much better than 1^o. A LIDAR can provide a reading of the distances to obstacles in 360^o with high accuracy, in less than a second. The drawback of this type of sensors is that there are surfaces that can not be detected. For instance, they can not detect glass, transparent surfaces nor very dark colors that absorb light. Depth or RGB-D cameras and Sound NAvigation Ranging (SONAR) sensors can also be used for mapping. The first are more suited for 3D mapping and require higher processing resources, while SONARs have a low angular accuracy to obtain good maps.

Maps created using SLAM are images in which each pixel represents a scaled space in reality. Maps can have resolutions such as 1cm per pixel or 5cm per pixel. Depending on the application, having a finer resolution

may be required. Obviously, fine resolutions consume many more resources. Each pixel of the map can have three values, which can vary depending on the implementation, but always represent one of these states: the space is free, so the robot can navigate through it; the space is occupied, so the robot can not go through it; the space is unknown, so the sensors of the robot have not yet reached that position on the map. Figure 2.1 shows an example of a map taken in a real store, in this case, the resolution is 1cm per pixel, the white color represent free space, gray color the unknown space, and black color the obstacles. Notice that the interior of obstacles is gray, because it is an area that the sensor can not reach.

Figure 2.1: Example of a robot map for navigation using SLAM. White pixels are the space where the robot can navigate, black pixels are obstacles and gray pixels represent spaces that are unknown to the robot.

Once the map is created, the robot can navigate through the environment.

Usually, a robot will navigate by following navigation goals, which are poses on the map that the robot needs to visit, commonly known as waypoints. To do so, a planner computes the path, pixel by pixel, from the robot position to the position of the navigation goal. The planner runs an algorithm that usually tries to minimize the cost of the computed path. So, if moving from any pixel to an adjacent one has the same cost, the shortest path will have the minimum cost. Usually these planners implement the A* or the Dijkstra's algorithm to do the computation. The planning process iterates from the new position of the robot to the next navigation goal. Figure 2.2 show an example of a map with navigation goals that the robot must reach to complete the area.

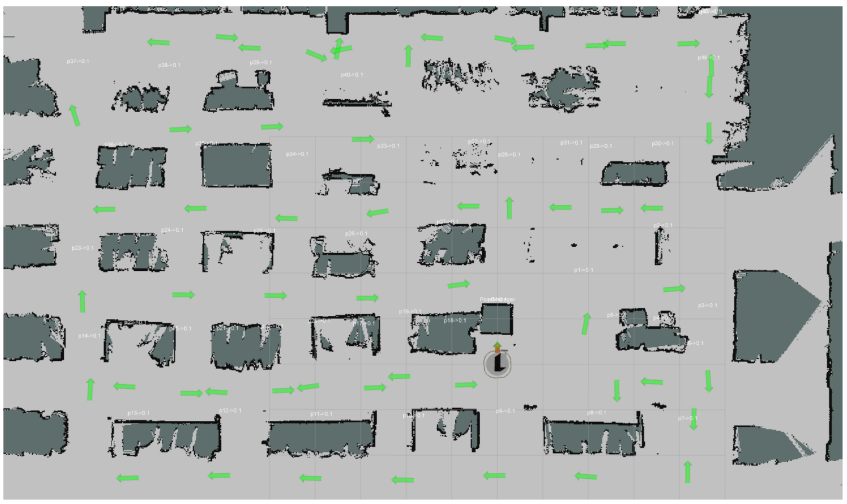


Figure 2.2: A visualization of a robot map with all the navigation goals, in green, that the robot needs to reach before completing its journey. Notice that the navigation goals can also have a target orientation, not only a target position, they are poses in the map.

However, this navigation approach will only work in a very static environment, in which the layout does not change with respect to the moment when the map was created, and where there are no people nor objects

around that can move. Therefore, on top of the map there are other type of maps called costmaps that can be used by the robot's navigation to add live information. For instance, these costmaps can collapse 3D obstacles sensed by a depth camera on the map, so the map includes obstacles that cannot be seen with a LIDAR, which is limited to a fixed height. An example of this are tables. From a LIDAR perspective tables are just 4 points because a LIDAR can only detect the legs, and the robot could try to pass through these points. But from a depth camera perspective, the table is a 3D obstacle and can not be traversed. Thus, all the sensors of the robot will contribute the detected obstacles to a costmap and therefore the planner will adapt the path to the newly sensed obstacles. As a consequence, the path created by the planner will not be the shortest, but the one with the lowest cost according to the costmaps. The obstacles on the costmap will add a cost value to each pixel, the closer to an obstacle the higher the cost up to a maximum cost that it is called lethal, and that the robot can not visit.

The planner that creates the path from the robot position to the next navigation goal using a stored map and a global costmap is called the global planner, but the one that controls the navigation in order to avoid unexpected obstacles or that are moving is the local planner. This planner computes very short trajectories from the position of the robot to some centimeters ahead, a few meters at most using the current observations by the sensor to compute a local costmap. This trajectory is recomputed very frequently and is the one that controls the actual movement of the robot. There are two main approaches for the local planner. On one side there is the Trajectory Rollout, see [10], in which all the possible trajectories from the current pose of the robot until the next pose are computed and evaluated in a forward simulation, and then, the best is chosen. On the other side there is the Dynamic Window Approach (DWA), see [11], in which all the possible speeds are computed and evaluated in a forward simulation, and again, the best one is chosen. In general, the second approach is more efficient, as the search space for velocities is lower than for trajectories, nevertheless, if the acceleration limits of the robot are low, the Trajectory Rollout can perform better than the DWA.

Finally, in order to briefly review all indoor navigation technologies it is important to mention localization. Actually this is a simpler problem than SLAM, since it just requires the robot to localize itself in a map, instead of localizing the robot in the map while generating the same map as SLAM does.
Localization algorithms first require a prior knowledge of the initial position of the robot. Then, they work in two steps, called prediction and correction. The first step uses data from internal sensors such as accelerometers, gyroscopes or wheel encoders, and based on the data received from them and a kinematic model of the robot, they estimate the following position of the robot in the map. Then, the second step uses a sensor that can detect the landmarks of the map, usually a LIDAR. In this step the pose of the robot is corrected by adjusting it according to the landmarks detected. These type of algorithms are solved in the context of an Extended Kalman Filter (EKF), in which the uncertainties of the sensors can be taken into account. However, to increase efficiency and to allow multiple simultaneous hypothesis on the robot pose an Adaptive Monte Carlo Localization (AMCL) algorithm is usually used.

2.3.2 ROS

The Robot Operating System [5] (ROS) is an open-source framework created by Open Robotics[6]. Firstly, it provides communication channels between all the processes running simultaneously on the robot, therefore one can have dedicated software packages, called nodes, for every sensor or process running in the robot, and they can communicate through messages, services or actions provided by the ROS framework. Therefore, synchronization of events is very easy to achieve. Secondly, it provides many libraries and packages, some of them created by the community, with state of the art methods and algorithms that are open-source and well documented. This way, it is not required to program all the navigation algorithms to operate a robot. Instead, just by properly parameteriz-

[5] https://www.ros.org/
[6] https://www.openrobotics.org/

ing and using the Navigation Stack[7] from ROS where all the previously mentioned algorithms and technologies are implemented it is possible to have almost any robot navigating. The ROS paradigm has always a node, called master, that registers all other nodes and allows the communication among them. Therefore, although it looks like a distributed system, if the master node fails all the other nodes will fail too. In this sense, a new version of ROS, ROS2, is now being developed and among other improvements it will make the system fully distributed eliminating the need of a master node. By the time that this thesis started ROS2 was not yet developed, and by the time of it ending ROS2 still does not have all the features of ROS available, so it has not been considered, nevertheless, at some point a migration of the packages here developed will be required.

2.4 Retail robots

Currently there are three types of robots intended to be used in retail. There are the RFID-based, the vision-based and the combination of both, RFID and vision. In general terms, the RFID-based robots will navigate through the store to read all the RFID labels aiming for a full inventory of the store products. The vision-based robots will move through all the aisles with shelves containing products looking for missing products (stock-outs) and checking if the prices are correct. Finally there are some robots that combine both technologies in order to provide a stock-count and simultaneously check for missing products or whether the prices are correct.

In general, RFID-based retail robots will decrease their efficiency if customers are in the store while they are operating. However, there are two good reasons to operate robots while customers are in the store. First, they can become an attraction to customers and, second, if they are continuously taking inventory a higher visibility on the store stock can be achieved. The case of vision-based retail robots is different, they require to identify out-of-stocks on the shelves and this is required while cus-

[7] https://wiki.ros.org/navigation

tomers are in the store, therefore, they require a navigation system better prepared to deal with people moving in the store.

2.4.1 Available robots for retail

The market for inventory robots in retail is large and there are several robots trying to get their share. The most promising robots are shown next, in strict alphabetical order of the company.

Marty from Badger Technology

This robot is a vision-based inventory robot. According to the information provided by the company it is able to detect holes on shelves, it checks the price integrity of the products, and planogram compliance. It can also detect hazards in the store, such as a glass jar fallen in an aisle, and report it.
Regarding the operation of the robot there is no explicit information on the procedures, however, it says it can autonomously operate safely alongside shoppers and employees of the grocery retail environment.
In December 2019 they have announced the largest roll-out of robots for the grocery industry, they are providing nearly 500 multi-purpose robots to GIANT/MARTIN's and Shop&Shop grocery stores.
In terms of the design of the robot, as it can be seen in Figure 2.3, it has two rotating LIDARs one on top for 360º scanning and another one in its base pointing in the direction of the movement, this second one is not 360º, but it might scan more than 180º. It also has depth cameras on top and bottom of the robot. These two types of sensors are basically for navigation purposes. In terms of data acquisition it has several high resolution cameras in one of the sides that are surrounded by several back-lighting LEDs to improve the camera's vision. This robot takes advantage of the ROS framework.

Figure 2.3: Marty's robot from Badger Technology.
Source: https://www.badger-technologies.com/media-kit.html

Bossa Nova 2020 from Bossa Nova Robotics

This robot is also a vision-based inventory robot. It is designed to detect holes in the shelves and out-of-stocks, it can check the price of the products and it has bar-codes readers to better identify them. It can also create panoramic images of the aisles to better visualize their state. And finally it can check the planogram compliance of products.
Again, the operation of the robot is not disclosed.
They have been testing their robots in Walmart for several years and they are now preparing a roll-out of robots in 1,000 stores. This will be the largest roll-out of inventory robots.
Regarding its design, as it can be seen in Figure 2.4, it has rotating LIDAR at the base with a view of more than 180º, for navigation purposes it also has several depth cameras in the front and rear at the bottom. The data capture is done with the depth cameras but also with all the cameras that are placed on the side of the robot. Some of the optics that they are using are designed for this specific purpose, it also has a custom dedicated

computer to process all the data gathered [8].

Figure 2.4: Bossa Nova 2020 robot from Bossa Nova Robotics.
Source: https://www.therobotreport.com/bossa-nova-2020-inventory-robot-includes-sharper-vision-for-brick-and-mortar-retailers/

Tagsurveyor from Fetch Robotics

This is an RFID-based inventory robot. It is focused more for warehouses than stores. It can count the tagged stock to improve inventory tracking. It can also be used to verify inbound and outbound activity at the docks.
In terms of operation it can gather data 24/7 with its autonomous charging capabilities. They offer on-demand deployment in hours without the need of any extra infrastructure. There is no information available about where they have tested their robots, but they have partnered with Surgere in order to offer this solution to a broader market, they have named this robot ROBi.
Regarding the design of this robot (see Figure 2.5) it has a rotating LIDAR

[8] from: https://www.therobotreport.com/bossa-nova-2020-inventory-robot-includes-sharper-vision-for-brick-and-mortar-retailers/

in its base reaching more than 180º for navigation purposes, no other sensors are clearly identified or announced for this purpose. Regarding the RFID system it has three antennas at different heights and inclinations in only one side covering 82º, reaching even the tall shelves. The system has a maximum range of 7.6m and they claim that in a single pass they can detect all the RFID tags.

Figure 2.5: Tagsurveyor from Fetch robotics.
Source: https://fetchrobotics.com/products-technology/datasurvey/tagsurveyor/

AdvanRobot from Keonn Technologies

This is an RFID-based inventory robot for retail stores. It can take daily scans of the store and provide stock-counts with an accuracy higher than 99.5%. It also provides the location of the detected items.
It has performed several demos and pilots during the last six years but the retailers are not disclosed. It holds a patent in the US as an automated inventory taking movable platform [9] and two more patents have been filled. Regarding its operation, first it needs to be manually navigated across

[9] https://patents.google.com/patent/US9939816B2

the space in order to create a map of the store. Once the map is created AdvanRobot autonomously navigates the environment reading all RFID tags, see [12] for a detailed explanation on this topic. It has more than 10h of operation between charges and it can autonomously charge using its recharging station. As far as we know, all the other robots operate in a similar way, however, it is not publicly disclosed.
As it can be seen in Figure 2.6 the robot has a depth camera on the top and bottom and a rotating LIDAR in the base with more than 180º for navigation purposes. Inside it has 4 RFID antennas per side and 2 RFID readers. This robot uses ROS.

Figure 2.6: AdvanRobot from Keonn Technologies.
Source: https://www.keonn.com/systems/view-all-2/inventory-robots.html

LoweBot from Lowe and Fellow Robots

In 2016 Lowe with Fellow Robots created LoweBot, an RFID-based inventory robot. The robot should be able to help the customers and staff from the store find products and take inventory in real-time.
There is not much more information on how it operates or where it has been tested. From the information on Fellow Robots, now called FellowAI[10], it can be seen that they have, at least, 2 platforms, one that it is not autonomous for 3D mapping spaces, see Figure 2.8. And the other platform, the LoweBot, Figure 2.7, which would be able to navigate the 3D maps and take real-time inventory.
From Figure 2.7 we can see that it has at least one LIDAR at the base for navigation purposes, but we can not see if it has depth cameras or other sensors. Similarly for the RFID subsystem, from Figure 2.7 the components or their distribution can not be identified. On the other side, from Figure 2.8 it can be observed that it has a 3D LIDAR on its top and 4 RFID antennas at each side.

Figure 2.7: LoweBot
Source: https://www.fellowai.com/

Figure 2.8: FellowAudit

[10]https://www.fellowai.com/

Tory from Metralabs

This RFID-based robot is basically dedicated to taking inventory with an accuracy above 99%. In addition, they claim that other functionalities can be added on demand, such as a shopping assistant. MetraLabs offers also a product called *Tory To Go* which is a similar robot but it is driven manually, it is thought for very tight shops
This robot has been operating since 2015 at different shops from German retailer Adler. They claim to be the first inventory robot in continuous live operation in retail.
Its operational procedures are not disclosed, however, they say that it has low operational costs and that it has 18h of autonomy and only 4h for charging.
From Figure 2.9 it can be observed that the robot has a LIDAR at the base and one depth camera on top, in the middle there is another sensor but from the information gathered it can not be determined which one it is. Regarding the RFID subsystem it has 3 antennas per side with the option of adding two more antennas on top, as seen in Figure 2.9.

Stockbot from PAL Robotics

This is an RFID-based robot for inventory and location, although the latest versions also have cameras on a side in order to check the planogram compliance and the prices.
To operate the robot it is required to first map the store and then through a web GUI the inventory areas and its scheduling can be defined, then it will perform inventories as configured.
This robot is now running at Decathlon Singapore Labs Store, also in the past it was also being tested in MediaMarkt.
From Figure 2.10 it can be observed that it has 2 depth cameras on top and a LIDAR at the bottom for navigation purposes. In addition it has 4 RFID antennas per side and three cameras on one side to provide the visual intelligence.

Figure 2.9: Tory from MetraLabs.
Source: https://www.metralabs.com/en/rfid-robot-tory/

Tally from Simbe Robotics

This is both a vision-based and RFID-based robot for retail. Therefore, it can take inventory with the RFID system and check prices and the planogram with its cameras.
In order to set Tally into operation it first requires to map the store, which is done by Simbe engineers and then it can operate on its own. If the store layout changes they claim that Tally can adapt, however, they do not specify how it can be done. It has a charging station and it can dock and charge by itself, it takes 90 minutes to charge and it does not specify the battery capacity, but it is said it can scan the entire shop in one shot.
Tally has been tested in Target stores and in Giant Eagle stores, also some tests have been performed in Decathlon.
As it can be seen from Figure 2.11 it has a LIDAR at the bottom and at

Figure 2.10: Stockbot from PAL Robotics.
Source: http://pal-robotics.com/robots/stockbot/

least 3 depth cameras in the lower part of the payload for navigation purposes, in their website they say that it has 40 sensors to safely navigate with people in the store and that it is the service robot that can navigate in the tightest spaces. In its side it has around 12 cameras without extra light sources for capturing data from the shelves and a modular RFID antenna design optimized to read at short range, up to 4m.

2.4.2 AdvanRobot

This inventory robot was originally created with significant contributions by previous members of our research group. For a detailed description of the robot see [12], [13] and [4]. The starting point of this thesis is during its development, therefore, it is taken as the state of the art in RFID-based inventory robots and one of the objectives of this thesis is to improve the detected shortcomings.
The key feature of AdvanRobot is its capacity of linking the navigation with the RFID detections. For this purpose, the navigation algorithm has two thresholds that can stop the robot or resume its path. The threshold

Figure 2.11: Tally from Simbe Robotics.
Source: Press kit from https://www.simberobotics.com/news/

to stop the robot will be triggered when the amount of new tag identifications per second goes above it. Then the robot will be stopped but twisting (rotating around its central axis) in order to maximize the reads of the surrounding tags. It will not continue its path until the threshold to resume is triggered. This will happen when the tag identifications per second go below this threshold. This simple but very effective feedback algorithm makes the robot achieve outstanding inventory accuracy levels, above 99.5% in the shortest possible time.

AdvanRobot operative procedure

The AdvanRobot is designed to inventory large spaces, it has more than 10h of battery autonomy, enough to take inventory of almost any store in a single night. In any case, due to scalability and operability issues, the first

thing that the user of the AdvanRobot needs to do is divide the space into areas. In terms of scalability, working with a smaller map consumes less resources, also creating larger maps with SLAM can lead to maps with more errors. Regarding operability, it can be desirable for the user to only take inventory of an area of the store, in order to have a fast stock-count of a certain type of product.
Then, the operative procedure of the AdvanRobot is divided in two parts, there is a recognition (mapping) phase and an inventory phase.
The recognition phase consists of manually navigating the robot in order to create a map of the area to inventory and to define the path that the robot should follow during the process. Therefore the first step is to place the robot in front of an initial Quick Response (QR) code identifying the area to recognize. Once the robot identifies the QR code, the interface of the robot allows the user to navigate it through the area. Once the entire area has been mapped, the robot has to be navigated in front of the QR code of the following area, at this point the map for the area will be saved and the process finishes. Then one needs to repeat this process for each area of the store.
Therefore, the user has to divide the space into different areas and each area has a starting QR code and a finishing QR code, which will become the starting QR code for the following area. The positions of the QR codes can only change if the recognition phase is performed again. Adding one QR code at the start and one at the end, which is the start of the following area, allows a continuous path through the whole store between areas and some overlapping in consecutive areas, which reduces the risk of the robot getting lost during transitions.
The inventory phase can be done on areas which are already recognized, it is not possible to perform an inventory of an area that has no map. To start an inventory phase, the robot has to be placed in front of the QR code of the first area to inventory and with an interface one can select all the following areas that require inventory. Then the robot starts the inventory at that point and finishes at the ending QR code of the last area selected to inventory. It can also navigate back to the starting point, to make the pick-up procedure of the robot easier.

AdvanRobot ongoing operative procedure development

The current version of the AdvanRobot has a docking station, so the first area of the store is always the one containing the docking station. However, at this moment this feature is not fully implemented in terms of operability, for instance, the robot will not start an inventory if it is not in the docking station, therefore, it doesn't have the option to start the inventory on a different area. However, these features will soon be integrated in the new version with the docking station. Additionally, there is an ongoing development that will allow an automatic set up for the inventory mission, which will not require the user to move the robot in front of the desired QR code and it will autonomously move back to the charging station.

Discussion

The operative procedure of AdvanRobot requires manually mapping the store before the robot can take inventories. Moreover, if the layout of the store changes significantly the robot might need a new map in order to be able to operate. Operating with old maps can lead to failed missions. Therefore, the mapping stage is required regularly as the store's layout changes.
Also, AdvanRobot requires an employee of the store to start the inventory and to take the robot back, once the inventory is done, to charge. During one of the largest tests performed with the AdvanRobot in US, the robot was left to be operated by the employees of the store. One of the outcomes of the experience was that several times the employees forgot to start the robot or to take it back for charging, leaving the robot without battery for the day after.
After several tests with AdvanRobot in different stores, we realized that from all the issues reported the one that worried retailers the most was the need to periodically create a map of the store.
Therefore, the first and main goal of this thesis will be developing a method for taking inventory of retail stores in which there is no need for a map of the store, providing full operative autonomy to the new inventory robot.

2.5 Exploration techniques

If the robot has to take inventory of a store it needs to navigate along all its aisles and zones. *So, how can we do that without a map?*

2.5.1 Classic exploration techniques

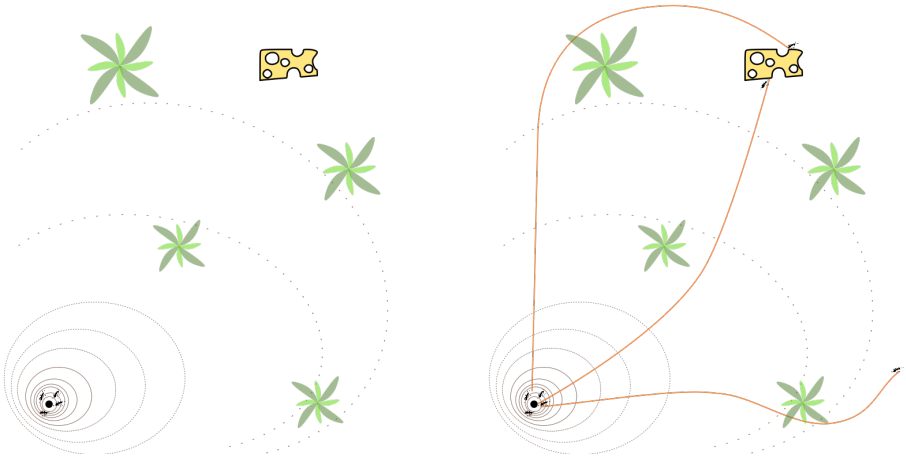
The simplest answer would be to move it randomly. After a very long period of time it might have counted all the stock of the store. But, the time to perform the mission can be very long, very unpredictable, and providing figures on inventory performance versus time would not be possible.
Another option could be using a similar algorithm than the one used by cleaning robots, which consists on moving on a fixed direction until it is not possible to continue and then turn, and start moving straight again. This method is good for not very big areas and for spaces that are wide. In a store with several aisles, the robot might take too long turning and not moving in a relevant direction.
A third option is to use exploration techniques to guide the robot. These techniques run a SLAM process which allows discovering the environment while, on top of it, there is the exploration layer that decides the navigation goals for the robot, therefore, it is able to create the entire map by itself.
The current exploration paradigm is called frontier based exploration, [14]. It moves the robot towards the exploration frontiers of the current map. In a conventional 2D robot map there are 3 types of areas: free of obstacles, occupied by obstacles and unknown. Therefore, the frontier based exploration sends the robot towards the zones that directly connect areas free of obstacles with unknown areas.
Therefore, this technique requires more computational power than the SLAM, as it solves SLAM while it looks for new frontiers in the current map to set the following navigation goals. In addition, exploration is an open problem basically due to the trade-off exploit-explore that needs to be solved. It consists on how to balance actions that take the robot

into new spaces (exploring), so the navigation goals are set in the frontiers, or towards known ones (exploiting). If many exploration actions are taken consecutively the map that is being generated can be wrong. This is because the SLAM technique that is running in the background requires visiting known spaces in order to correct the cumulative errors associated to dead reckoning. For instance, in [15] they tackle this problem with an approach based on entropy, but there are several approaches for solving this problem.
In our case, the exploration would be done in every mission since the robot can not know if the layout of the store has changed. In this sense there is work that addresses the problem of a continuous and lifelong exploration, such as [16] and [17].
However, without information about RFID tags in the control loop, the robot would be solving a problem of discovering an area and not stock-counting it. Therefore, in our new paradigm the RFID readings have to enhance the inventory process, this is already something that is achieved by the AdvanRobot by using two thresholds to link the RFID and the navigation algorithms. This has been proven to be crucial in order to achieve outstanding inventory results.
Taking advantage of RFID technology in order to enhance the exploration problem has also been used. For instance, there are techniques that use RFID tags in order to keep track of the areas visited [18] and [19]. However, these techniques have the RFID labels added and customized for this purpose but in our environment the RFID tags will be as product labels, so no specific distribution or special functionality can be given to them.

2.5.2 Stigmergy

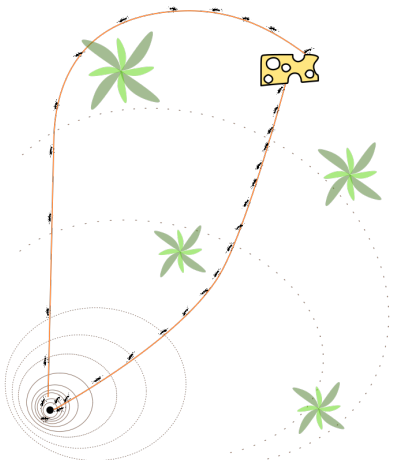
After analyzing several exploration options, we found that some insects have exploratory behaviors in which a single insect finds a resource to be exploited by the whole colony, but it is able to communicate the location of this resource to the whole colony without the need of a map nor by directly communicating the location of this resource.
The foraging behavior of ants was first discovered by [20]. It consists in

the following: first ants start randomly looking for food and while they move, they leave a trail of pheromones; then, when they find a source of food, they get some and they go back to the nest following their own trail of pheromones. This behavior reinforces their trail and more ants from the nest will follow it to the source of food. Moreover, when another ant finds the source of food through a less optimal path, as the optimal or better path is more reinforced due to a higher number of ants passing through, due to the fact that it takes a shorter time, the ant will use it to get back to the nest. This will lead all the ants to follow the shortest path from the food to their nest. See Figure 2.12 for a schematic example.
Thus, stigmergy consists of modifying the environment with the use of pheromones in order to store information in it that can be used by other agents. So, we can say that communication or coordination between agents is done indirectly through the environment.

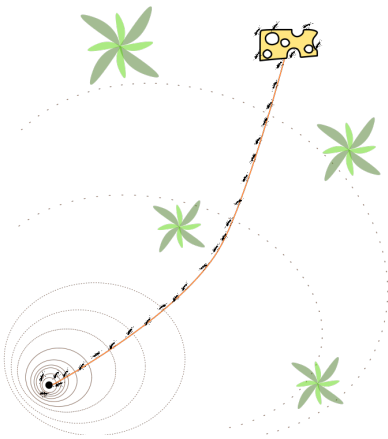


(a) Initially ants are in their nest. They do not know where the source of food is. Therefore, they start to randomly explore their nest surroundings.

(b) After a while, some ants have found the source of food while others have not. The ones that found it, return to the nest leaving a trail of pheromones.



(c) Other ants from the nest start following those pheromone trails that lead to food. The shortest path is completed faster, and therefore, is more reinforced by pheromones.



(d) Finally, the shortest path is the only one used by the ants, the larger amount of pheromones has discarded all the other options.

Figure 2.12: Schematic of ants foraging behavior.

2.5.3 Stigmergy & Robotics

We take the stigmergy definition as the ability to modify the environment in order to gain indirect communication between agents from [21]. Very simple agents are able to solve complex problems using stigmergic behaviors. As an example, with stigmergic algorithms robots do not need to obtain global localization [22]. This is a very relevant characteristic. The information required for the agent to solve the task is embedded in the environment. Therefore, the agent does not require to know its global position to decide its next action, but it is enough to sense the surrounding environment.
Moreover, robots running this type of algorithms are, in general, more robust due to two factors. On the one hand, the robot requires fewer sensors and less computational resources, which directly translates into a simpler system which is inherently less prone to failure. On the other hand, these algorithms are applicable to multi-robot decentralized systems, and therefore, the failure of a single robot does not compromise the final outcome of the task being performed [23]. In fact, until all the robots have failed the task is not considered a failure.
Several researchers have been working with stigmergic algorithms in robotic mobile platforms. For instance, in [22] robots perform exploration of different areas without the need of a global localization, taking advantage of a stigmergic algorithm. In this case, the robots deploy some markers to inform about the locations already visited. Another problem that uses stigmergic algorithms is cooperative cleaning. In [24], the problem is tackled by detecting the dirt, and then, using its presence or absence as information to know whether an area has already been cleaned or not.
There exist other studies, besides this work, which use RFID tags to support simulated pheromones for their stigmergic algorithm. For instance, [25] claims to be the first research to use the RFID technology to support pheromones. In their work, they use the read/write characteristic of an RFID tag to leave pheromones (write), so other robots can track them (read). However, their experimental results with real robots are not very promising. In [26] agents can deploy, read and write on RFID tags so they

claim that a rapid exploration of unknown areas can be achieved. Nevertheless, only simulation results are presented. In [27] they use an experimental floor full of RFID tags and a swarm of robots in order to solve complex problems such as the formation of gradient maps. They present successful results in both simulation and experimentation with real robots.

2.5.4 Area coverage

Intuitively the area coverage problem is related to the problem of taking inventory of a store. In fact, the area coverage problem consists on covering the entire space with a sensor or actuator, or both. Taking into account that the RFID antennas are a sensor the objective for the inventory problem would be covering the whole store with the range of the RFID antennas. However, the real objective is not covering the space but counting all the RFID labels present in the space. Even with the best RFID system it is possible to cover the whole store but to miss several tags requiring more than one pass through certain areas. Therefore, both problems can be related but they are not exactly the same.
The area coverage for unknown spaces consists on moving an agent through the entire space, so that it can visit the whole area once or keep on repeating the task of covering the space. This problem can be solved by stigmergic behaviors.
If the space to cover is represented as an unknown graph, for the area coverage problem the goal is achieved after going through all the nodes of the graph. In the case of the stock-taking task, the agent does not need to move through all the nodes of the graph, but identify all the labels of environment to fulfill the task. In the stock-taking problem, the agent might need to visit more than once a node, because the action of reading RFID labels is not deterministic. Therefore, being in a node does not imply reading all the labels.
Therefore, it is relevant to study if solutions for the area coverage in unknown graphs work on solving the stock-counting problem, this will be assessed in section 3.4.1.

The following three methods are interesting as they can be solved using stigmergic behaviors:

Brownian motion.

The simplest way to solve the area coverage task is to let the agent move randomly. Obviously, this is not efficient in time. In [28] they present time bounds for covering the entire space. Therefore, is not efficient time-wise, but with enough time it is finally achieved. In addition to that, in [29] they prove that using a type of Brownian motion called Lévy flights, which is a heavy tailed distribution that allows covering long distances in a single step, the results are better than with pure Brownian motion.

Node counting

This is a typical method for online graph exploration. It consists on increasing by one a value added to each node that the agent visits and then moving to the adjacent node with the lower value.

LRTA*

The learning real-time A* algorithm is similar to node counting. However, in this case, the value of the node that is being visited is increased by the maximum value of all the neighbors. Then the agent moves towards the node with the lower value.
In [30] and [31] they compare Node counting and LRTA* extensively and they formally present the similarity between area coverage and online graph exploration problems.

2.6 RFID tag location with robots

Providing the RFID system with the capability to locate items has been a topic of research for several years, and yet, there is no general solution adopted by the community. The complex physics of the RFID technology

coupled with the environment makes the problem of locating the RFID labels still open.
For instance in [32] they use the received RSSI from the reference tags to train a neuronal network to then locate other tags. The training with the reference tags is crucial, in terms on how to locate these tags with respect to the reader. They improve a previous method called LANDMARC, but the results of absolute location capabilities are not clear. Other studies focus on the RSSI to model the location, for instance, in [1] they locate moving tags or fixed tags through modelling the RSSI decay, but the method is used with direct line of sight between the tag and the reader. They claim a resolution as good as 15cm. However, results in crowded environments, in terms of many tags responding simultaneously or without direct line of sight are not presented.
Some studies use the phase of the returning wave in order to estimate the location of RFID tags. In [2] there is an RFID reader on top of a robot and mixed with its odometry they can correct the position of the robot by analyzing the phase received. It requires several tags on the ceiling and the trajectories of the robot are very controlled in the environment. Additionally, the precision on locating tags is not really assessed as they focus on the robot location.
There are other previous works that used RFID tags to improve the robot position as in [33]. They are one of the first to mix both technologies. They created an RFID detection model for the robot and by using a technique similar to the one used for solving the SLAM problem they are able to locate the RFID tags in the surroundings of the robot. They do not present quantitative values on the precision of the RFID tag location because they are centered on using their study to improve the robot localization, moreover, only 100 tags are present and located in the environment. The scalability of their method with several thousand tags is not ensured, usually SLAM techniques require large computational efforts.
Nevertheless our focus is in locating RFID tags with a reader that it is on top of a robot, but the location of the robot is obtained by other means. In [34] they create a fuzzy model in order to locate tags with a mobile robot. In fact, they create two models one based on the RSSI and the

other on the occurrence of detection. For both models the distance and the angle between the robot and the tag are used. Although they use a complex model, one of their main conclusions is that the locating precision improves when the robot approaches the tag.
In [35] they use a Bayesian filtering method, similar to the one used in [33], but, in addition, they change the emission power of a COTS RFID system achieving this way a 50cm location error. This approach is also similar to the one taken in [4] where a recursive Bayesian estimation is performed on the RSSI of the RFID tags detected in order to estimate their 3D location. The training and the tests are performed in the university library with thousands of tags, in fact, the data-set is open an accessible in [36]. The method, however, requires training for the model of the RSSI behavior with respect to the relative position of the robot antennas. Unfortunately, the characteristics of the RF waves make the training very dependent on the environment. The results are shown with respect to the same environment where the training is performed achieving mean distance errors of 0.6m.

Chapter 3

MAPLESS INVENTORY

In order to become omni-channel, traditional retailers can benefit from an inventory robot. However, they do not want to spend work-hours of their employees to operate an inventory robot. They require the solution to be as autonomous as possible. This chapter presents a new paradigm to take inventory that does not require creating a map, it is map-less. This solution provides the operational autonomy required by the users of an inventory robot. Additionally, the map-less paradigm improves other characteristics of the inventory robots such as cost, robustness and scalability.

3.1 Solution hypothesis

In order to determine if the new paradigm for inventory robots is feasible, the first step will be to define the problem of stock-taking. Then, a stigmergic solution will be proposed and simulation tests will be run in order to assess its feasibility at the end of the chapter.
In general, stigmergic behaviors allow very simple agents, such as ants, to perform complex tasks as a swarm. Given this consideration, the research hypothesis is: *simple robots in terms of hardware and software are able to perform complex tasks, such as stock-counting all the RFID tagged items in a confined space, by using a stigmergic behavior.*
Accordingly, the solution hypothesis is presented as follows:

- RFID tags in the environment will provide indirect communication among robots.
- RFID tag readings will drive the current state of the mission.
- RFID tag detections will attract the robots:
  - Attraction will take place towards areas with new detections.
  - Attraction will be lower to areas with already identified tags.
  - Attraction will not happen towards areas without tags.

In consequence, it seems possible to perform an exploration of the area of interest while taking inventory of it. In addition, as the RFID labels will guide the navigation of the robot, the process of taking inventory will be enhanced.

3.2 Stock-taking task

Briefly, the task of stock-taking consists of identifying all the labels that are inside a confined space. The task assumes that during its process labels will not move, be added nor removed from the space.
In this new paradigm, the RFID labels take the role of pheromones and they change the environment that the robot senses. Therefore, every time that a robot reads a label it will increase and check the counter associated to this label. The list of counters linked to RFID labels will be shared and updated among the robots, depending on the implementation it can be accessed from a centralized server or shared p2p.
This way, robots will move towards the direction where there are more new labels to discover, by following RFID tags that have been read only once. With this simple behavior the robot will be able to inventory an area.

3.2.1 Formal description

The stock-counting problem can be formally described through a graph-based approximation by defining an environment, a state of the problem and two actions that agents can take.
The environment of the task is represented by an undirected, unweighted and connected graph $G(V, E)$ where $V = \{v_i\}$ are the nodes and $E = \{e_{jk}\}$ the edges. Nodes represent subareas, small enough, so that they can be considered visited by an agent placed in it. So, if we put a circle of radius R, which is the maximum reading distance of the RFID system, on every node of the graph, the circles will cover the entire area that contains RFID labels. The latter, implies that the amount of nodes required to have a good approximation of the environment has to be large.
There is an edge $e_{ij} \in E$ between two nodes $v_i, v_j \in V$ if there is an accessible path (without obstacles) between v_i and v_j. The environment contains a set of labels L distributed in the space, representing stock items and defined as $L = \{l_1, l_2, l_3, \ldots, l_N\}$. Each node $v_i \in V$ contains an unknown number of labels $L_i = \{l_n\} \mid l_n \in L$, therefore, $L_i \subseteq L$. Each label only belongs to one node.
The state shows how the task develops at each time step m, where $m \geq 0$. It is represented by a set $S^m = \{s_k\}$, in which each element $s_k =< l_n, c_n >$ is a tuple containing a known label, l_n, and a value associated to the number of times this label has been identified, c_n, where $c_n > 0$.
The initial state is $S^0 = \emptyset$ at time $m = 0$, and the final state after M steps is $S^M = \{s_k\}$ where $|S^M| \leq N$. Notice that the operation $|A|$ over the set A represents its cardinality. Therefore, if all the labels from all nodes are read, then $|S^M| = N$. It is worth mentioning that the environment is unknown as is the total amount of labels in it. Therefore, it is not possible to undoubtedly know if the task is completed.
Initially, as $G(V, E)$ is unknown only the initial node v_{start} is given. The set of adjacent edges from a node $v_i \in V$ is denoted as $A(v_i)$ and, as G is connected, $A(v_i) \neq \emptyset$. Also, $succ(v_i, e_{ij})$ represents the adjacent node of v_j reached by traversing the edge $e_{ij} \in A(v_i)$. The visited node at time m can be denoted as the agent pose, $P^m_{agent} = v_i$, which can be understood

as a part of the problem's state. However, it does not have any correlation with the completion of the task.
The agent solving the task has two possible actions. These are similar to functions, but they change the state of the problem or the pose of the agent. So the agent can read labels in its current node or through an edge, this action will modify the current state S^m of the task. The second action that the agent can perform is traversing an edge to change its current pose.

- $R(v_i, e_{ij}, m)$ is the action of reading the surrounding labels at the time m, from the node v_i, in the direction of e_{ij} or in the same node v_i. We define the resulting set of labels as $L^m_{v_i e_{ij}} = \{l_n\} \mid |L^m_{v_i e_{ij}}| \leq |L_i|$. Notice that when the action is on the same node is expressed as $R(v_i, \emptyset, m)$, and returns $L^m_{v_i \emptyset}$. This action is probabilistic and atomic, meaning that the same action will produce different outcomes each time and that it is indivisible, primitive and cannot be interrupted.

- $T(v_i, e_{ij})$ is the action of traveling from node v_i through the edge e_{ij}. It returns $succ(v_i, e_{ij})$ and modifies $P^m_{agent} = v_i$ to $P^{m+1}_{agent} = v_j$. Note that this action requires an amount of time so the time step m is increased.

We define $L^m_{v_i}$ as the union set of all the labels read from node v_i. So, the labels read in the node are denoted as $L^m_{v_i \emptyset}$ and for each adjacent node the labels read are denoted as $L^m_{v_i e_{ij}}$. Therefore, $L^m_{v_i} = L^m_{v_i \emptyset} \bigcup L^m_{v_i e_{ij}} \forall e_{ij} \in A(v_i)$. And S^{m+1} as the state at time $m+1$ being the update of the state S^m with $L^m_{v_i}$ according to the definition below. At this point, four operators or functions need to be defined:

- *Labels*(S^m) which returns the set of labels L^m from the state S^m.

- *counters*(S^m, L^m) which extracts the list of counters c^m associated to the set of labels L^m from the state, S^m.

- *Update*(S^m, $L^m_{v_i}$). This operation, called updated method, returns the state S^{m+1} after updating the current state S^m with the readings

obtained from the node $v_i \in V$ at time m. To do so, the first elements of the tuple $s_k =< l_n, c_n >$, $\forall s_k \in S^{m+1}$, are created by the union of the two sets *Labels*$(S^m) \cup L^m_{v_i}$. Then, the counters c_n from the labels that were already in the state, such that $l_n \in L^m_{v_i}$ and $l_n \in$ *Labels*(S^m) are increased by a value which depends on a selected approach to solve the problem. And the counters from new labels obtained at the last reading action, c_n such that $l_n \in L^m_{v_i}$ but $l_n \notin$ *Labels*(S^m), are initialized with a value that also depends on a selected approach to solve the problem.

- *end_condition*(S^{m+1}, m). This function evaluates to *True* if the state of the task, S^{m+1}, reaches a predetermined condition, for instance, if $|S^{m+1}| \geq 99\%|L|$. It also evaluates to *True* if the amount of time steps, m, reaches its predetermined maximum. Otherwise, it evaluates to *False*.

3.2.2 Attraction u

We define the attraction as a measure obtained from the environment. This measurement is given by the RFID tags and provides the agent with the required information to choose the next movement. The notation of the previous section is kept. Formally, let $u^m_{v_i e_{ij}}$ be the attraction sensed by the agent at node v_i from the direction of the edge e_{ij} at time m, with $L^m_{v_i e_{ij}}$ being the identified RFID tags in that direction.
Hence, we define the attraction as a positive real number such that:

$$u^{m+1}_{v_i e_{ij}} = |L^m_{v_i e_{ij}} - \textit{Labels}(S^m)| + \frac{|L^m_{v_i e_{ij}}|}{\sum \textit{counters}(S^{m+1}, L^m_{v_i e_{ij}})} \tag{3.1}$$

Where the difference between two sets, $A - B$, is the set containing the elements of the first set A but not elements that are common between sets A and B.
As it can be seen in equation 3.1, the attraction is composed of two terms. The first term is always ≥ 0, and, it greedily attracts the agent towards areas with new labels. The second term is positive and always ≤ 1. Therefore, this second term guides the agent when the first term is zero or there

is a draw between directions, favoring those that have been visited less times.
The first term just adds one per each new tag identified in the specified direction. The second term evaluates how many times a direction has been visited. It is represented by the number of identified tags in that direction divided by the sum of their associated counters c_i. It behaves similarly as $1/x$ would do with the increase of reads of a single tag, so an area that has been visited several times, becomes being less attractive.
In other words, if there are new tags, the first term will dominate over the second one and it will drive the agent towards the areas not yet visited. The second term of the equation will be used to break ties in favor of the areas that have been visited less times, the behavior will be similar when there are no new tags detected.
If we return to the stigmergic analogy, agents will move leaving a trail of pheromones (RFID tags identified) that will make their path less attractive to other agents by decreasing both terms of the attraction of the nodes visited. So, all agents will feel attracted towards the areas not yet visited, where the amount of pheromones is lower.

3.2.3 Algorithm

In order to solve the stock counting problem a stigmergic inspired algorithm for multi-agent systems is developed. It is stigmergic because it uses the RFID tags in the environment to guide the actions of the agents and as a communication channel between them, which means that the state, S^m, is shared across the agents and can be read and updated by any of them.
The notation follows the one described in subsection 3.2.1. Before diving into the algorithm, two operators have to be defined. First, *one-of(X)* returns one random value of the set X. Second, *arg-max*$_{x \in X}$ $f(x)$ returns the set of values of x that maximizes $f(x)$. It is a set of values because several elements of X could maximize $f(x)$.
Algorithm 1 is used to solve the stock counting problem. Firstly, the time is initialized $m = 0$, the initial state is assigned so $S^0 = \emptyset$ and the agent

starts in a random initial node v_{start}.
At this point, the main loop starts until the *end_condition*(S^{m+1}, m) is met. So, the node v_i is given by the agent pose, P^m_{agent}, and its set of identified labels is initialized, $L^0_{v_i} = \emptyset$. Then, the agent reads the labels on the current node and adds them into $L^m_{v_i}$. Secondly, for each possible direction $e_{ij} \in A(v_i)$ the agent read tags $R(v_i, e_{ij}, m)$, adding each of the obtained sets into $L^m_{v_i}$.
Thirdly, S^{m+1} is created using the *Update* function. Note that c_i may depend on the choice of the update method, even if the straightforward decision is $c_i = 1$ for a label not in S^m and $c_i := c_i + 1$ for a label already in S^m.
Fourth, $u^{m+1}_{v_i e_{ij}}$ is calculated using Equation 3.1 for every $e_{ij} \in A(v_i)$.
Finally, the best next direction, e_{ik}, is the one with the maximum value $u^{m+1}_{v_i e_{ij}}$. If there is a draw in two or more different directions or there are no tags read, the next direction is chosen at random. Then, the robot travels and the agent pose, P^{m+1}_{agent}, is changed. Consequently, the step time increases $m := m + 1$. At this point if no final condition is met, the algorithm repeats the process for the new node.
As previously stated, it is not possible to know if the task is complete since the total number of tags N is unknown. However, there are at least four conditions that can be considered as finishing conditions for the algorithm.

- A determined amount of time has passed, so $m \geq$ fixed threshold.
- The agent has identified a predefined amount of tags, so $|S^{m+1}| \geq 99\%|L|$.
- The attraction sensed in all directions is below a certain threshold for several iterations.
- There are several algorithm steps without new tags read.

The two first finishing conditions reach a positive milestone in the task completion. On the contrary, the last two conditions force the algorithm to

stop due to not progressing on the task completion. In any case, these conditions require a previous knowledge of the environment and the multi-agent system, however, their values can be optimized with repetitions of the same task.

Algorithm 1 RFID-based stock counting algorithm

1: $m := 0$
2: $S^m := \emptyset$
3: $v_{start} :=$ *one-of*(V) ▷ v_{start} is selected randomly
4: $P^m_{agent} := v_{start}$
5: **while** not *end_condition*(S^{m+1}, m) **do**
6: $v_i := P^m_{agent}$
7: $L^m_{v_i} := \emptyset$
8: $L^m_{v_i} := L^m_{v_i} \cup R(v_i, \emptyset, m)$ ▷ The agent reads in current node
9: **for all** $e_{ij} \in A(v_i)$ **do**
10: $L^m_{v_i} := L^m_{v_i} \cup R(v_i, e_{ij}, m)$ ▷ Reading through every edge
11: **end for**
12: $S^{m+1} :=$*Update*$(S^m, L^m_{v_i})$ ▷ Update of the state
13: **for all** $e_{ij} \in A(v_i)$ **do**
14: calculate $u^m_{v_i e_{ij}}$ ▷ Use equation 3.1
15: **end for**
16: $e_{ik} :=$*one-of*$($*arg-max*$_{e_{ij} \in A(v_i)} (u^m_{v_i e_{ij}}))$ ▷ Edge of max attraction
17: **if** $e_{ik} := \emptyset$ **then**
18: $e_{ik} :=$*one-of*$(e_{ij} \in A(v_i))$ ▷ The edge is chosen randomly
19: **end if**
20: $P^{m+1}_{agent} := T(v_i, e_{ik})$ ▷ The agents travels from node v_i to v_k
21: $m := m + 1$ ▷ Traveling requires time, m is increased
22: **end while**

Algorithm example

An example of the process is shown here in order to clarify the concepts defined. However, in the interest of reproducibility, the following charac-

teristics have been simplified. The probability of detecting an RFID label will be 1 and the range for detecting labels will be also 1, so the agent will be able to detect labels from its node and from the neighbors. This is motivated to create an example that can be easily followed and with final results that do not change due to the stochastic behavior of the RFID detections.
Consider the graph of Figure 3.1 representing an environment in which the distribution of RFID labels is as seen in Table 3.1. The task will be finished if any of the following two end conditions is met:

- The mission reaches its 10th time step ($m = 10$)
- The agent identifies more than 99% of the labels ($|S^{m+1}| \geq 99\%|L|$)

The agent starts at node $v_{start} = v_8$, as seen in Figure 3.2, it has been randomly selected among all possible nodes. From that position it first reads the RFID labels of its current position and then for every adjacent node. This way the state of the problem can be updated and the attraction computed. In this first step of the algorithm the largest attraction is towards node v_7.
Therefore, in this example, the agent travels to v_7, as seen in Figure 3.3. In this scenario there is only one available direction, so after reading the labels, it senses an attraction towards node v_8 of 0.5. In any case, the agent only option is moving back to node v_8.
Now the agent is again in node v_8, see Figure 3.4. But, it senses the environment differently than before. Therefore, after reading all available labels, nodes v_3 and v_9 pull it with the same attraction, 0.5. At this point, the agent has to randomly select one of the two options. In this example, the agent moves towards v_9.
From this point the same process will be repeated. The agent will read the surrounding labels and then it will travel towards the node with the maximum attraction, see from Figure 3.5 to Figure 3.8.
The task ends at node v_2, see Figure 3.8, when the agent has detected all labels present in the environment, triggering the second end condition presented, $|S^{m+1}| \geq 99\%|L|$.

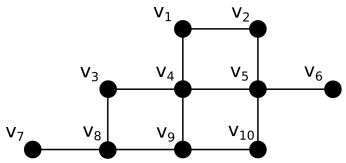


Figure 3.1: Graph representing the environment to stock-count.

Node	**RFID labels**
v_1	$\{l_1\}$
v_2	$\{l_2, l_3, l_4\}$
v_3	$\{l_5\}$
v_4	$\{l_6, l_7\}$
v_5	$\{l_8, l_9\}$
v_6	$\{l_{10}\}$
v_7	$\{l_{11}, l_{12}\}$
v_8	$\{l_{13}, l_{14}\}$
v_9	$\{l_{15}\}$
v_{10}	$\{l_{16}, l_{17}, l_{18}\}$

Table 3.1: RFID labels distribution

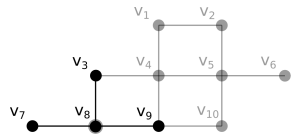


Figure 3.2: Starting node set randomly to v_8. The agent at node v_8 reaches nodes v_3, v_7 and v_9. The maximum attraction push the agent to v_7.

1. $m = 0$
2. $v_8 = P^0_{agent} = \text{one-of}(V)$
3. $R(v_8, \emptyset, 0) = \{l_{13}, l_{14}\}$
4. $R(v_8, e_{87}, 0) = \{l_{11}, l_{12}\}$
5. $R(v_8, e_{83}, 0) = \{l_5\}$
6. $R(v_8, e_{89}, 0) = \{l_{15}\}$
7. $S^1 = \{(l_5, 1), (l_{11}, 1), (l_{12}, 1), (l_{13}, 1), (l_{14}, 1), (l_{15}, 1)\}$
8. $u^0_{ve_7} = 3$
9. $u^0_{ve_3} = 2$
10. $u^0_{ve_9} = 2$
11. $P^1_{agent} := T(v_8, e_{87})$

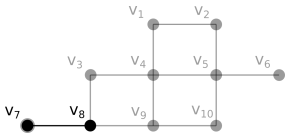


Figure 3.3: Agent at node v_7. It only reaches node v_8 and it only has one accessible path. So, for any attraction sensed it will travel back to v_8.

1. $m = 1$
2. $v_7 = P^1_{agent}$
3. $R(v_7, \emptyset, 1) = \{l_{11}, l_{12}\}$
4. $R(v_7, e_{78}, 1) = \{l_{13}, l_{14}\}$
5. $S^2 = \{(l_5, 1), (l_{11}, 2), (l_{12}, 2), (l_{13}, 2), (l_{14}, 2), (l_{15}, 1)\}$
6. $u^1_{ve_8} = 2/4$
7. $P^2_{agent} := T(v_7, e_{78})$

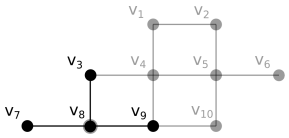


Figure 3.4: The agent is again at node v_8. But, its perception of the environment has changed with respect to the initial step. The highest attraction is sensed towards v_3 and v_9. So, the next direction is picked at random, being e_{89}.

1. $m = 2$
2. $v_8 = P^2_{agent}$
3. $R(v_8, \emptyset, 2) = \{l_{13}, l_{14}\}$
4. $R(v_8, e_{87}, 2) = \{l_{11}, l_{12}\}$
5. $R(v_8, e_{83}, 2) = \{l_5\}$
6. $R(v_8, e_{89}, 2) = \{l_{15}\}$
7. $S^3 = \{(l_5, 2), (l_{11}, 3), (l_{12}, 3), (l_{13}, 3), (l_{14}, 3), (l_{15}, 2)\}$
8. $u^2_{ve_7} = 2/6$
9. $u^2_{ve_3} = 1/2$
10. $u^2_{ve_9} = 1/2$
11. $P^3_{agent} := T(v_8, e_{89})$ Random choice between e_{89} and e_{83}

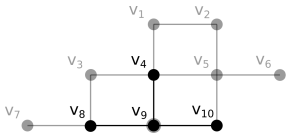


Figure 3.5: The agent is at node v_9. It reaches nodes v_8, v_4 and v_{10}. The highest attraction is towards node v_{10} with three new labels.

1. $m = 3$
2. $v_9 = P^3_{agent}$
3. $R(v_9, \emptyset, 3) = \{l_{15}\}$
4. $R(v_9, e_{9\,8}, 3) = \{l_{13}, l_{14}\}$
5. $R(v_9, e_{9\,4}, 3) = \{l_6, l_7\}$
6. $R(v_9, e_{9\,10}, 3) = \{l_{16}, l_{17}, l_{18}\}$
7. $S^4 = \{(l_5, 2), (l_6, 1), (l_7, 1), (l_{11}, 3), (l_{12}, 3), (l_{13}, 4), (l_{14}, 4), (l_{15}, 3), (l_{16}, 1), (l_{17}, 1), (l_{18}, 1)\}$
8. $u^3_{ve_8} = 2/8$
9. $u^3_{ve_4} = 3$
10. $u^3_{ve_{10}} = 4$
11. $P^4_{agent} := T(v_9, e_{9\,10})$

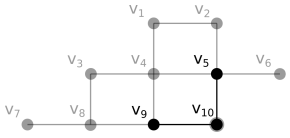


Figure 3.6: The agent is at node v_{10}, besides labels in v_{10}, it reads from nodes v_9 and v_5. The highest attraction is sensed towards node v_5 with 2 new labels.

1. $m = 4$
2. $v_{10} = P^4_{agent}$
3. $R(v_{10}, \emptyset, 4) = \{l_{16}, l_{17}, l_{18}\}$
4. $R(v_{10}, e_{10\,9}, 4) = \{l_{15}\}$
5. $R(v_{10}, e_{10\,5}, 4) = \{l_8, l_9\}$
6. $S^5 = \{(l_5, 2), (l_6, 1), (l_7, 1), (l_8, 1), (l_9, 1), (l_{11}, 3), (l_{12}, 3), (l_{13}, 4), (l_{14}, 4), (l_{15}, 4), (l_{16}, 2), (l_{17}, 2), (l_{18}, 2)\}$
7. $u^4_{ve_9} = 1/4$
8. $u^4_{ve_5} = 3$
9. $P^5_{agent} := T(v_{10}, e_{10\,5})$

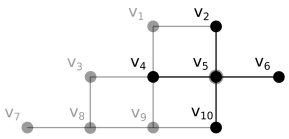


Figure 3.7: The agent is at node v_5, from there it can also reach nodes v_2, v_4, v_6 and v_{10}. Two of these nodes have new tags, but node v_2 has more than v_6. Therefore, the agent will travel to v_2.

1. $m = 5$
2. $v_5 = P^1_{agent}$
3. $R(v_5, \emptyset, 5) = \{l_8, l_9\}$
4. $R(v_5, e_{5\,4}, 5) = \{l_6, l_7\}$
5. $R(v_5, e_{5\,10}, 5) = \{l_{16}, l_{17}, l_{18}\}$
6. $R(v_5, e_{5\,6}, 5) = \{l_{10}\}$
7. $R(v_5, e_{5\,2}, 5) = \{l_2, l_3, l_4\}$
8. $S^6 = \{(l_2, 1), (l_3, 1), (l_4, 1), (l_5, 2), (l_6, 2), (l_7, 2), (l_8, 2), (l_9, 2), (l_{10}, 1), (l_{11}, 3), (l_{12}, 3), (l_{13}, 4), (l_{14}, 4), (l_{15}, 4), (l_{16}, 3), (l_{17}, 3), (l_{18}, 3)\}$
9. $u^5_{ve_4} = 2/4$
10. $u^5_{ve_6} = 2$
11. $u^5_{ve_{10}} = 3/9$
12. $u^5_{ve_2} = 4$
13. $P^6_{agent} := T(v_5, e_{52})$

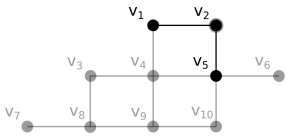


Figure 3.8: This is the final step of the task. From node v_2 it can read the missing label l_1 that is in node v_1. Therefore, the *end_condition*(S^{m+1}, L) is met, $|S^7| \geq 99\%|L|$, then the task is over.

1. $m = 6$
2. $v_2 = P^6_{agent}$
3. $R(v_2, \emptyset, 6) = \{l_2, l_3, l_4\}$
4. $R(v_2, e_{2\,1}, 6) = \{l_1\}$
5. $R(v_2, e_{2\,5}, 6) = \{l_8, l_9\}$
6. $S^7 = \{(l_1, 1), (l_2, 2), (l_3, 2), (l_4, 2), (l_5, 2), (l_6, 2), (l_7, 2), (l_8, 3), (l_9, 3), (l_{10}, 1), (l_{11}, 3), (l_{12}, 3), (l_{13}, 4), (l_{14}, 4), (l_{15}, 4), (l_{16}, 3), (l_{17}, 3), (l_{18}, 3)\}$
7. $u^6_{ve_1} = 2$
8. $u^6_{ve_5} = 2/6$
9. $P^7_{agent} := T(v_2, e_{21})$

3.3 Simulation methodology

This section presents simulation tests performed with *Matlab*[1]. These tests will determine the exact behavior of the map-less algorithm and assess its feasibility.

3.3.1 Simulation considerations

Before presenting the simulation tests, it is required to clarify some implementation issues that arise. Some of these issues will require to take assumptions and make hypothesis to get the simulation running.

Environment

The environment is modelled as a connected and undirected graph. The connectivity of each node is at most 4 and traversing any edge has the same cost. It is assumed that each node covers a circle with a radius of 1m. Therefore, the distance to the center of a neighbor node is at 2m.
It is assumed that items in a store do not occupy the whole floor, there are areas of the store which are accessible, from a navigation point of view, but that does not contain tagged items. In order to take this into account, the graph representing the environment has nodes with RFID labels and nodes without them. However, from a node with labels there is always a path to any other node with labels that does not require traversing a node without labels. So, it can be understood as an inner connected subgraph with RFID labels.
At this point, a helper function has to be defined. It will be necessary to evaluate the results of the tests. This function is called *empty(v_i)*, and given a node of the graph it returns *True* if the node has no labels, otherwise, *False*.
Also, stores do not have an evenly distribution of labels, there will be areas with a higher density than others. To accommodate this idea, the nodes with labels will have a randomly selected number, ranging from 1

[1] https://www.mathworks.com/products/matlab.html

to 50 labels.
Figure 3.9 represents one of the environments used in simulation. The nodes in red contain tags and form a connected subgraph. It can be observed that there are some blue nodes within the red ones, however, there are no isolated nodes with tags.

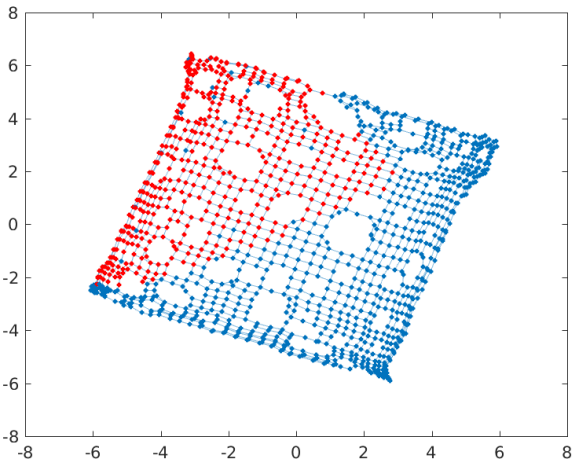


Figure 3.9: Example of graph used for simulation. Only the red nodes contain RFID tags. Note that both the entire graph and the red subgraph are connected.

RFID detection model

Due to the complexity of physically modelling the RFID interaction with the environment we choose a probabilistic approach for the action of reading tags, called R. In this sense, we introduce a probability of detecting an RFID tag, p_{detect}, which depends on two other probabilities, the reaching probability, $p_{reach}(d)$, and the reading probability, $p_{read}(d)$.
The reaching probability is applied to the adjacent nodes. It models the capacity for the RFID waves to reach a certain area. RFID waves are very sensitive to the environment, for instance, a metallic shelf can block and

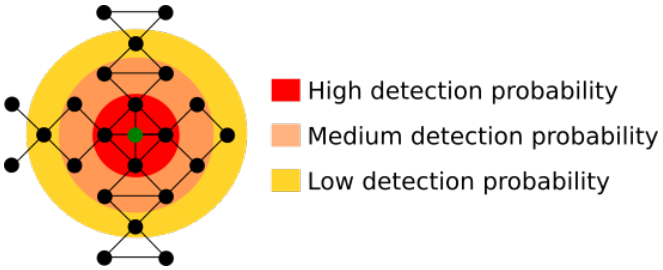


Figure 3.10: Scheme of the RFID range in a simulation environment, the shape of the graph is totally randomized. The central node, in green, is the position of the robot.

make them bounce, while a shelf full of books could absorb them preventing them to go forward or bouncing. Therefore, this probability models the response of the RFID waves to the environment. At this point it is required to make an assumption based on the experience obtained working several years with the RFID technology:
The RFID waves can reach at most a node that is at distance 3 of the agent's node, where distance is the number of edges between two nodes, see Figure 3.10. This would give a reading range of 7 meters for the RFID system, which is a commonly used value. The probability of reaching, $p_{reach}(d)$, a node decreases with the distance, so to reach a neighbor node, at distance 1, the probability is higher than to reach a neighbor at distance 2. The labels in the current node are always reached, so the reaching probability is 1.
The reading probability, $p_{read}(d)$, is applied to each RFID label reached by an RF wave. It models the capacity of the label to respond to the reader's wave. There are several causes that make a label not respond. For instance, if tags are very close to each other, or if a tag polarity is orthogonal to the antenna's or if a tag is touching a metallic shelf.
The main difference between the reaching and the reading probability is that the reaching probability is conditional. In this sense, when the agent reads towards an edge, once the RF wave does not reach a certain node, the following nodes in range will not be reached. On the contrary, the

reading probability is applied to an entire node, so all the labels in it can be detected.
Therefore, the RFID detection probability is obtained as in Table 3.2, notice that $P_{detect} = P_{reach} * P_{read}$.

RFID range	**0**	**1**	**2**	**3**	$\geq$ **4**
$\mathbf{P_{reach}}$	1	0.8	0.4	0.2	0.0
$\mathbf{P_{read}}$	0.98	0.8	0.5	0.2	0.0
$\mathbf{P_{detect}}$	0.98	0.64	0.2	0.04	0.0

Table 3.2: RFID model of detection probabilities as a function of the RFID range.

It is important to highlight that the detection probability is not applied directly to the labels, but first the reaching probability is applied to an entire node, and then, the reading probability to each of the labels. Also, if a neighbor node is not reached, the following nodes will not be reached.
The values of Table 3.2 are chosen carefully to provide high detection rates in the proximity of the system, while decreasing the detection rate quadratically with the distance. This probabilities not only include the behavior of the RFID system but also the environment. Therefore, their values are given based on an average evidence of the RFID system behavior.

Agent

The stigmergic nature of the map-less algorithm makes it suitable for multi-agent systems, therefore, in simulation several agents will be performing the task simultaneously. Their position will be identified by the agent's pose, $Pose^m_{agent}$, and they all will be able to check and modify the state of the system, S^m, with their RFID detections.
Interactions between agents will not be simulated, not in terms of RFID detections nor in navigation. So, two agents sharing the same pose will not see affected their RFID detections or navigation.
We understand a robot as a physical implementation of an agent that is

able to interact with a physical environment. We take the definition of agent from [37].

3.3.2 Figures of merit

The evaluation of the algorithm and comparison of methods is based on the time to complete the task, each unit of time is a complete iteration of the developed algorithm. As opposed to the real world, the actual stock is known, therefore, the task is considered completed when the agent has identified at least a 99% of the RFID labels, this value is chosen because is close to an upper bound of the expected accuracy on stock-counting given the technology [12]. This value is the stock-counting accuracy and it is computed as seen in Equation 3.2, take into account that the notation used is the same used in section 3.2.1.

$$accuracy(m) = \frac{|\textit{Labels}(S^m)|}{|L|} \quad \in [0, 1] \tag{3.2}$$

The figure to evaluate the efficiency is the average visiting time, $\hat{V}$, which is defined as the amount of time steps to reach the target accuracy, in our case 99%, divided by the amount of nodes with RFID labels in the graph, see Equation 3.3.

$$\hat{V} = \frac{m \mid accuracy(m) \geq 99\%}{|\{v_i \mid empty(v_i) = False\}|} \tag{3.3}$$

The average visiting time, $\hat{V}$, models the time required by the system of agents to achieve an inventory accuracy of at least 99%. The number of nodes with RFID labels will change at each simulation test, therefore, to have a fair comparison between tests, it is required to normalize the figure of merit by the amount of nodes with RFID labels. That is the reason why we are not comparing the number of steps to reach a certain accuracy but, this divided by the amount of nodes with RFID tags. To clarify the figure, an average visiting time of 1 means that the agents require as time steps to perform the inventory mission as nodes with RFID labels in the graph. Therefore, the lower the value the more efficient the system is.

3.4 Tests

Three sets of tests are performed to assess the feasibility of the developed stigmergic inspired algorithm.
The first set of test aims to analyze three area coverage algorithms at stock-counting. These methods will be adapted to stock counting, and also, some characteristics of the problem will be simplified to take into account that these methods were not created to solve this problem.
The second set of test will check the behavior of the attraction proposed, see Equation 3.1, with two different update methods.
The third set of test is to assess the performance of the algorithm with the increase of the amount of agents participating in the task.
In brief, the first 2 set of test will help define the final characteristics of the algorithm. The third set of test will assess the feasibility and scalability of the map-less method with respect to the number of agents.
At all the tests, there are 4 parameters that are randomized in every repetition, these are:

- Graph size and shape
- Number of tags per node
- Total amount of tags
- Initial position

Notice that the initial position of the agent is always at a node with RFID labels. It is not considered the possibility of an initial position where the agent can not read any RFID label. This situation would only force the agent to move randomly until an RFID label is read, and then, start using the method proposed.

3.4.1 Area coverage algorithms

The differences between the area coverage problem and the stock-counting are significant. The main difference is with respect to the goal of the problem, at area coverage the agents will try to go through every node of the

environment, whereas the stock-counting problem only aims at identifying the RFID labels of the environment. In addition to that, the RFID technology allows identifications from a certain distance and, therefore, visiting all the nodes is not required to take a full inventory of the environment. However, the RFID detections are not deterministic, depending on several factors a tag inside the range of detection may not be detected. Therefore, it might require the agent to visit a node several times to complete the count. Nevertheless, both problems require algorithms that are able to move through an unknown environment.

To obtain a better understanding on the behavior of these methods two characteristics of the RFID system are simplified. The maximum distance to detect tags is set to 1 instead of 3 and the probability of detecting tags, p_{detect}, is also set to 1. So, the action of reading tags for this test is deterministic. These two modifications reduce the gap between the area coverage problem and the stock-counting. In the first place, the requirement of not having to visit every node is reduced by shortening the range. In the second place, a deterministic detection model gives a higher level of relevance to the visits of the node and less to the RFID detections.

At this set of test three algorithms that solve the area coverage problem will be compared, they are the node counting, the LRTA* and the random motion, see section 2.5.4 for more detailed information on them.

The node counting algorithm is modified, so that the attraction only uses the counters associated to each label, as seen in Equation 3.4. Also, the update of the state, $Update(S^m, L^m_{v_i})$, initializes the counters, c_k with the value of 1, and each time a label is seen again, it increases its counter, c_k, by 1.

$$u^{m+1}_{v_i e_{ij}} = \sum \textit{counters}(S^{m+1}, L^m_{v_i e_{ij}}) \tag{3.4}$$

In addition, to behave as similar as possible to the real node counting algorithm. The agent will not be attracted towards the largest attraction, but towards the lowest. So, the operator that takes the argument that minimizes a function must be defined, $\textit{arg-min}_{x \in X} f(x)$. Therefore, the agent will take the edge given by this operator. However, the minimum value will always be 0, which is the value for an edge without labels, therefore,

the edges without RFID detections will not be an option, unless it is the only option. All the other characteristics will be similar as in the Algorithm 1.
The LRTA* algorithm will work similarly than the node counting algorithm, so it will use the attraction defined in Equation 3.4, and it will also use the previously defined operator to choose the next direction for the agent. Therefore, its difference will only affect the update function, $Update(S^m, L^m_{v_i})$. In this case, each time a label, l_k, is identified, the value of its counter, c_k, is set to the highest value of a counter plus one of a detected label in the same direction, as seen in Equation 3.5.

$$c_k = max(counters(S^{m+1}, L^m_{v_i e_{ij}})) + 1 \tag{3.5}$$

Lastly, the random motion will always choose the next direction for the agent randomly, without taking into account the detected RFID labels.
Table 3.3 shows more details on the simulation parameters. Notice that the number of nodes in the environment was chosen randomly with a minimum of 10 nodes and a maximum of 1400. Then, the number of nodes with labels is also randomly selected, but it is at least a 50% of the total. Next, for each node with labels the number of labels is randomly selected between 1 and 50.

	Values
Repetitions	240
Graph size	rand(10,1400)
Total labels	from 40 to 19000
$\mathbf{P_{detect}}$	1
RFID Range	1

Table 3.3: Characteristics for the simulation of the area coverage tests.

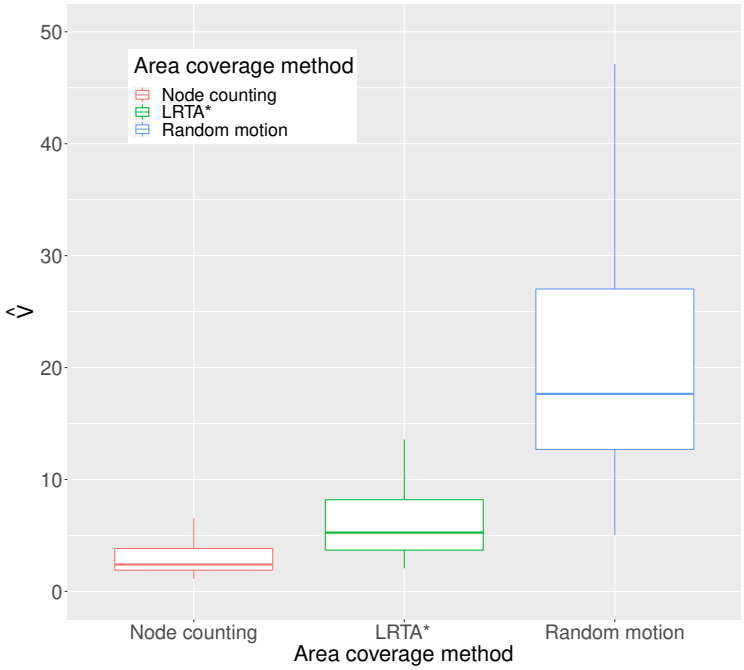


Figure 3.11: Box-plot comparison of area coverage methods at stock-count. The horizontal axis presents the three methods and the vertical axis the average visiting time.

Results

After running the set of tests, the results obtained can be observed in Figure 3.11. It shows a box-plot of the resulting average visiting time, $\hat{V}$, for each algorithm. Notice that in the horizontal axis there are the area coverage methods, in the vertical axis we can observe the average visiting time, $\hat{V}$.
It is important to highlight that in the results presented the methods always reach the 99% on the accuracy, as it is defined in Equation 3.3.
From Figure 3.11 it can be seen that the random motion is the worse method at stock-counting. Its median $\hat{V}$ is above 15, which means that for each node with RFID labels the algorithm has to perform 15 steps. It

is also the one with the larger deviation on the results obtained and it had few outliers that took higher average visiting times, but for visualization reasons they have been removed from the graph.
The differences between the node counting method and the LRTA* are relevant, but both methods clearly outperform the random walk. The node counting algorithm has performed better at stock counting than the LRTA*. The median of the LRTA* is slightly above 5, and, for the node counting is around 2.5. If we look at the average values of $\hat{V}$ the node counting is at 3.2 while the LRTA* is at 6.7, in addition to that, the dispersion on the results is higher for the LRTA*, showing that it is also a less reliable method than the node counting.
Such a significant difference between LRTA* and node counting was not expected, actually from [30] it is said that the LRTA* is more efficient than the node counting at area coverage, so, a similar behavior was expected here. Figure 3.12 shows a comparative between the two methods in which the horizontal axis shows the number of nodes with labels and the vertical axis the number of algorithm steps to reach the 99% of accuracy. In Figure 3.12 all the performed tests can be observed, for graphs with a small number of nodes with labels both methods perform similarly. However, at the larger graphs, which are more significant, the node counting performs clearly better than the LRTA*.
Our hypothesis is that with the LRTA* algorithm there might be nodes on the graph with the labels not yet identified that are surrounded by nodes that have been visited many times, so the agent has trouble reaching these type of nodes. Mainly because the update value for the LRTA* increases drastically the value of the counters, faster reducing the attraction sensed. In any case, the exact reasons of why this is happening are not in the scope of this work and it is left as an open topic to study further.
Nevertheless, the results clearly show that updating the state by increasing the counters, c_k, by one improves the efficiency on the task completion.

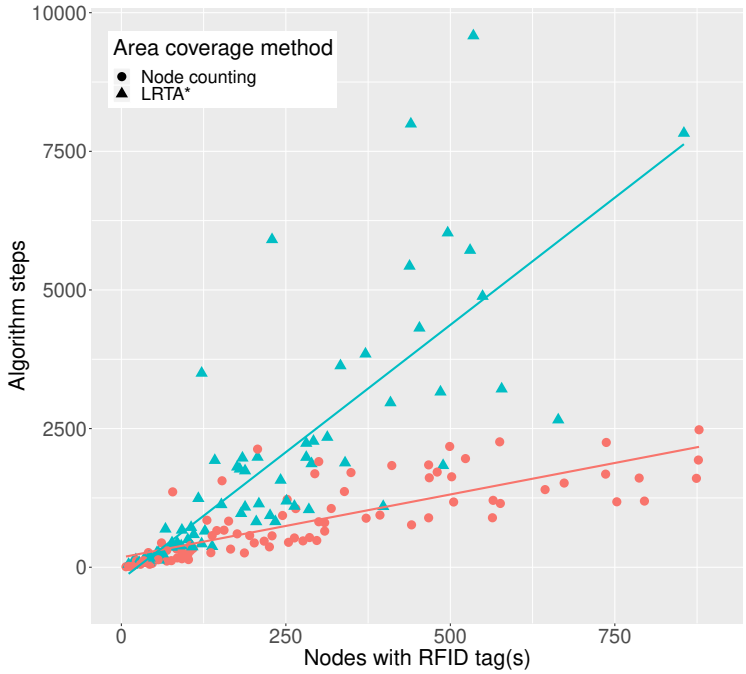
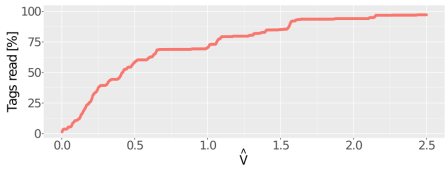


Figure 3.12: Comparison between the node counting and the LRTA* algorithms. The x axis shows the number of nodes with RFID tags of the graph and the y axis the amount of algorithm steps to reach an accuracy value of 99%. The lines are linear regressions just to show the trends.

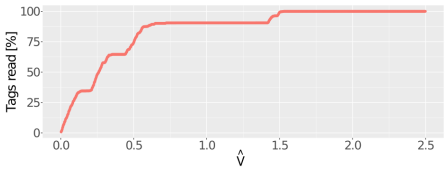
3.4.2 Update method

The previous test shows the best update method for an attraction, which is only based on visiting the less recurring nodes. However, it is required to analyze how these update methods perform with our proposed attraction, see Equation 3.1. Therefore, this second set of test aims at deciding which is the best update method for the proposed attraction.
Two update methods are compared: the one used by the node counting algorithm and the one used at the LRTA* algorithm. Briefly, the first method initializes any c_i at 1. After, each time the label is read the c_i is increased by 1. The second approach sets the c_i as the maximum value of

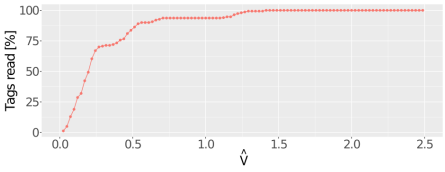
the read labels in that same direction plus 1.
Taking into account that for any of the two methods initially the agent will be driven by the amount of new RFID labels discovered, the change of the update method should only affect the end of task, during the asymptotic part.
Figure 3.13 shows the percentage of new tags discovered by a single robot as a function of the average visiting time. This way it is possible to compare the performance in different scenarios. In this intermediate result, it can be observed how the behavior is always similar, there is an initial phase in which the robot discovers many new tags, but then the discovery becomes nearly asymptotic. Our assumption is that the update method will guide this asymptotic phase of discovering RFID labels.
This set of tests will be also used to observe the behavior of the probabilistic approach on the RFID detections. So, the tests will be performed changing the range of the RFID system, see Table 3.2. Test will have an RFID range of 1, 2 or 3, named short, mid and long respectively. We do not expect relevant differences on the performance, however, a slight improvement is expected as the range increases. See Table 3.4 for more details on the simulation parameters.



(a) Graph with 9750 RFID labels.



(b) Graph with 2460 RFID labels.



(c) Graph with 300 RFID labels.

Figure 3.13: Example of the accumulated amount of RFID labels discovered per algorithm iteration.

	Values
Repetitions	1380
Graph size	rand(10,1300)
Total labels	rand(60,25000)
$\mathbf{P_{detect}}$	As in Table 3.2
RFID Range	All

Table 3.4: Characteristics of the simulation of the update method tests.

Results

Figure 3.14 shows the results obtained as a box-plot with the average visiting time, $\hat{V}$, for each update method and for each range.
Observing the results, firstly it is worth mentioning that for any range or update method, the map-less algorithm together with the attraction presented in Equation 3.1 performs better than a deterministic node counting method at stock-counting, as seen in Figure 3.11. Specifically, the $\hat{V}$ median obtained with the node counting algorithm is at 2.5 and with the proposed attraction is around 1.5.
Regarding the update method, there is no clear evidence over which of them works better with the attraction presented. For short and long range the node counting update method performs slightly better, but this is not happening for the mid range. In this sense, the decision on which update method to use will not depend on the results obtained here, but on the results obtained at the previous set of tests, see subsection 3.4.1. Therefore, the selected update method is the one that is initially determined and the one that works as the node counting algorithm, in this sense the counters will always be increased by 1.
Last but not least, the results do not show a real difference between the three ranges. The best results are obtained at long range, it is also the range with less variation. But, the difference with mid or short range are low. Observing the results, it is unexpected that the mid range results are worse than the short range. In general terms, as the probability of detection decreases with the range, see Table 3.2, it is expected not to greatly

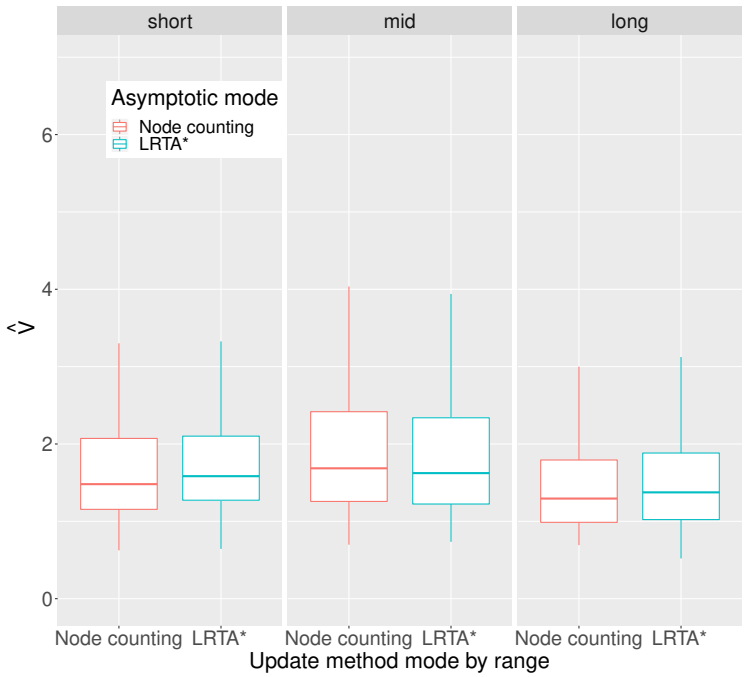


Figure 3.14: Box-plot comparison of the stock-counting algorithm for 2 update methods and 3 ranges.

improve the performance when increasing the range. But, it is not expected that the mid range performs worse than the short range. At least, it should perform equally. In any case, the differences are small and they could be a consequence of the randomly selected characteristics for every test.

3.4.3 Multi-agent system

This last set of tests is developed in order to assess the results obtained for the stock-counting task if the number of agents is increased. As a stigmergic inspired algorithm, it is specially useful for a large number of agents, the communication through the environment is expected to coordinate the agents performing the mission. They will be sharing an updated list with

their RFID readings, so they should be able to avoid areas that others have already visited.
The simulations have been done increasing the number of agents, from 2 to 10 first, and then for 20 and 50 agents. Although the graph sizes are set randomly, if the number of agents increases the minimum and maximum possible graph sizes also increase. The initial pose of each agent is determined randomly but at a node with RFID labels, and more than one agent can start in the same node.
See Table 3.5 for more details on the simulation parameters.

	Values
Repetitions	1700
Graph size	rand(10,2900)
Total labels	rand(150,65000)
$\mathbf{P_{detect}}$	As in Table 3.2
RFID Range	3

Table 3.5: Characteristics of the simulation of the multi-agent system tests.

Results

From Figure 3.15 it can be observed how the average visiting time, $\hat{V}$, decreases with the increase of the number of agents performing the task. In addition, as it is a logarithmic scale, also the variance on the results is decreased with the amount of robots.
Therefore, the performance of the system increases, but to assess the scalability of the solution, it is required to penalize the fact that the solution requires more agents. To do so, in Figure 3.16 it can be observed, in logarithmic scale, the average visiting time multiplied by the number of agents that are composing the system. In this case, the shape of the results is almost flat, showing that no matter the size of the team or the space that the system will be able to scale without any additional cost, but also without an additional gain in time. It can be observed that the value obtained

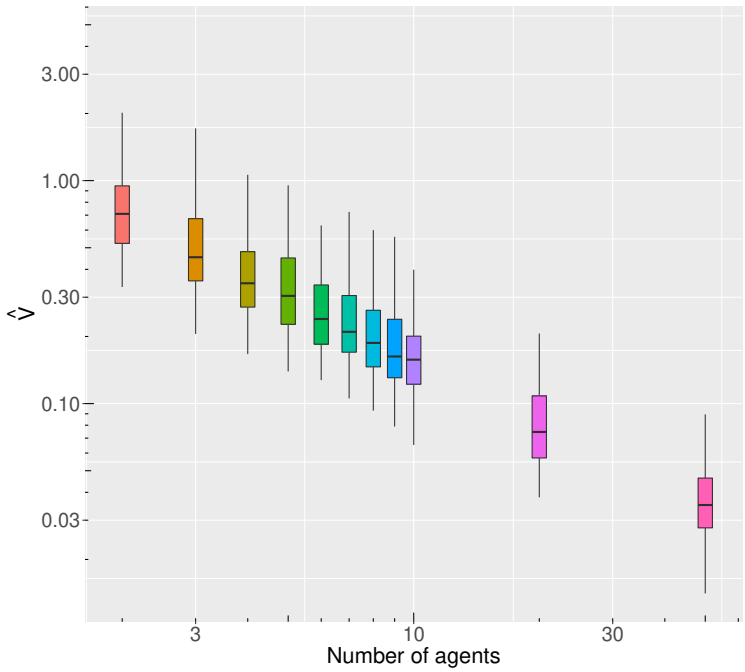


Figure 3.15: Box-plot comparison of the stock-counting algorithm for an increase number of agents.

is almost always 1.5, showing that the performance of the algorithm is maintained with the increase of agents.
Regarding the parallelization of the task, Table 3.6 shows the average of the number of nodes visited per agent. Obviously, with the increase of the number of agents the quantity of nodes visited per agent decreases, also the standard deviation of this average decreases. Nevertheless, it never reaches 1%, meaning that in all cases the behavior is very similar. It is also worth mentioning that the total percentage of nodes visited increases also with the number of agents. In this sense, the increase of number of agents does not keep the efficiency at the same level. If the task is finished with 2 agents visiting less than the 75% of the nodes, it could be thought that the same could happen with 10 robots. Nevertheless, as the size of

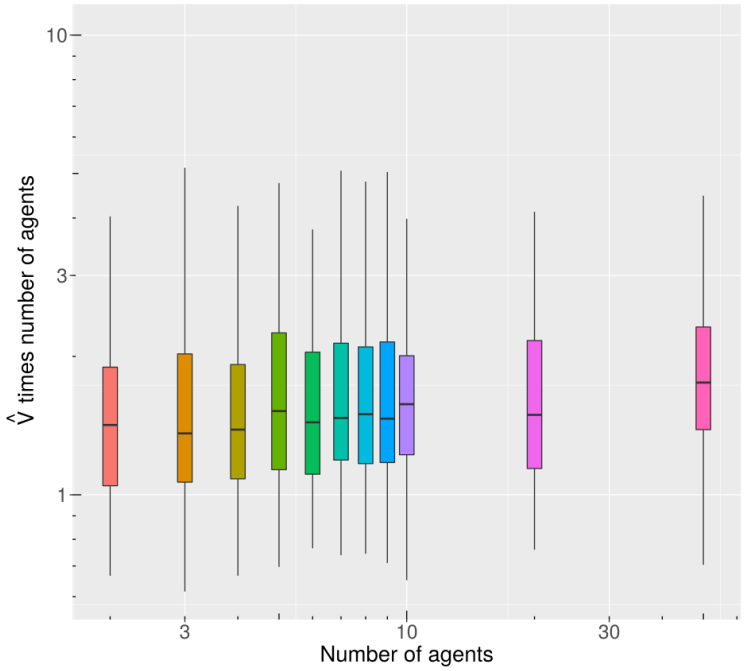


Figure 3.16: Box-plot comparing the average visiting time, $\hat{V}$, multiplied by the number of robots of the system.

the graphs and the initial position of the agents are randomized, if graphs where the 2 agents work are not perfectly scaled with the ones with 10 agents it is understandable that some overlapping occurs. This fact could explain the reason why in Figure 3.16 the result with 50 agents is slightly higher than the others. Nevertheless, it is important to understand that in simulation the interactions between RFID systems and robots are not considered. Therefore, the performance of a real RFID system can change in presence of multiple robots reading simultaneously. Also, the navigation could get worse if, for instance, several robots are simultaneously trying to enter a narrow aisle.

	Average of nodes visited	Std. deviation of nodes visited	Total of nodes visited
2 agents	36.1 %	0.78 %	72.3 %
3 agents	25.6 %	0.32 %	76.8 %
4 agents	21.2 %	0.43 %	84.7 %
5 agents	18.3 %	0.21 %	91.3 %
6 agents	14.7 %	0.35 %	87.9 %
7 agents	12.7 %	0.27 %	89.2 %
8 agents	11.3 %	0.39 %	90.8 %
9 agents	10.8 %	0.15 %	96.8 %
10 agents	10.7 %	0.19 %	107.4 %

Table 3.6: Percentages of nodes visited by each robot as a function of the number of robots of the system.

3.5 Conclusions

By approaching the stock-counting problem with a graph, and then, simulating it, it has been proved that this new stigmergic-based paradigm can be used by RFID-based inventory robots in order to stock count any environment with RFID-tagged items.
This new paradigm will work without a map, providing a higher level of autonomy to robots. In this sense, the algorithm has been compared to other algorithms that solve a similar problem, such as the area coverage, and has performed better at stock-counting by using the attraction defined in Equation 3.1.
Also, as it is a stigmergic inspired algorithm it is ready to be used by several agents simultaneously, achieving a high level parallelization, hence scalability. For instance, taking the percentage of nodes with tags visited per each robot and calculating an average for each repetition, it can be observed that the difference between agents is around 1%, meaning that all of them visit a similar amount of nodes, and therefore, the task is extremely well distributed among them.
At this point, the next logical step is to test the presented solution in a real

environment with a real robot. The next chapter describes the process for designing an RFID-based inventory robot capable of taking advantage of the developed map-less algorithm.

Chapter 4

ROBOT DESIGN

The main outcome of the work described in this chapter is the design of a robot capable of taking inventory following the map-less algorithm developed in Chapter 3. Currently there is not an available robot designed to perform stock-count with the map-less algorithm. In addition, there are some assumptions in the algorithm that may not be easily satisfied. Therefore, the design itself is considered a scientific outcome. Thus, this chapter pretends to create new knowledge through the design following the techniques and perspectives of the design science [38].

4.1 Design science

The design science is a set of rules and techniques that allow performing research through the design.
There are no rigid constraints to perform this type of research, however, some guidelines can be found in [39]. These guidelines cover from the design, which, for instance, requires producing a viable artifact, to the evaluation, where a minimum level of rigorousness is required.
In this chapter the design has to solve two nested problems. The outer most is to have a robot that can take advantage of the stigmergy-based algorithm in order to take inventory of an unknown environment. The

inner problem is to create an RFID system able to determine the relative orientation between the RFID label and the platform. Both problems will be faced simultaneously by means of a design that deals with them.
This chapter presents two iterations on the design of the robot as it can be seen in Figure 4.1.

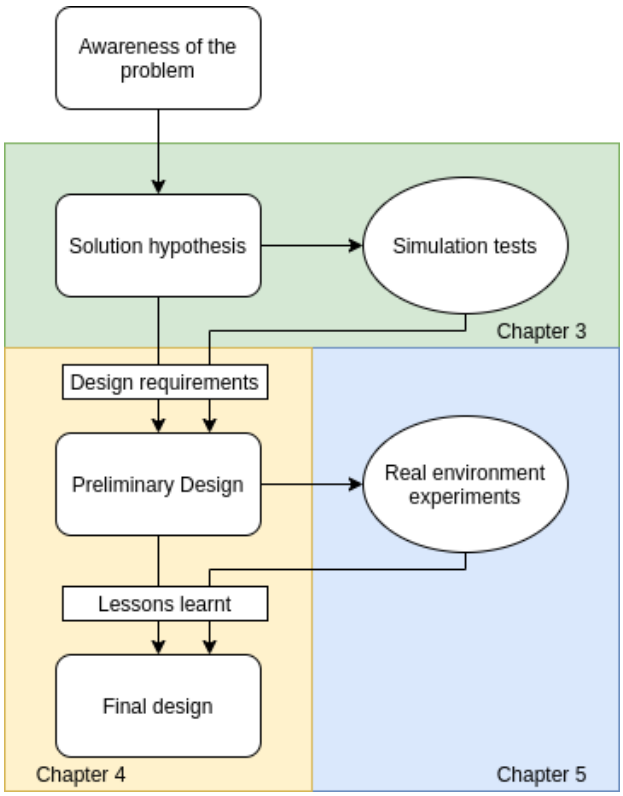


Figure 4.1: Design process in the context of the map-less inventory paradigm development.

This chapter will present the requirements specification to fulfill the needs of the design of a robot. According to them, a preliminary design will be developed an a set of tests will be done in a real environment, these tests are detailed in Chapter 5. The final part of this chapter details the lessons learnt from the preliminary design and the set of tests, resulting

in the development of a final design of the robot for the map-less inventory paradigm. This final design focuses on increasing modularity, cost efficiency, compactness and increasing the functionalities of the development.

4.2 Design requirements

The main requirement of the design is proving the validity and feasibility of the algorithm by taking it to a real context. Nevertheless, the following requirements set a starting point to design a prototype robot that implements the map-less algorithm. At the end of this section a table of the requirements is presented as a summary, see Table 4.1.
From an overall point of view, the main requirements for the design are the following:

- The designed robot has to be cost-effective. This way, it will be disrupting with respect to the other inventory robots, which are expensive.
- All components used for the design have to be commercial off-the-shelf (COTS). The designed robot has to develop a feasible application and using experimental components would compromise its development as a real application.
- It has to be ROS compatible to take advantage of the open-source resources available from the community and to leverage the experience obtained from working with the current AdvanRobot.

These initial and general requirements will affect all the systems of the robot. Now, the following requirements will be specific of each system. Regarding the RFID system:

- The RFID payload has to detect the relative orientation between the robotic platform and the identified RFID tag. This will allow the algorithm to decide in which direction to plan the following movements. This requirement is of the utmost importance.

- The system requires inventory accuracies as good as the current state of the art for RFID-based inventory robots.

In terms of the robotic base, the map-less algorithm requires the following:

- A mobile robotic base to move through the different corridors of the store.
- Sensors for detecting and avoiding obstacles in the surroundings.
- An holonomic drive with a circular footstep to ease its navigation in narrow and intricate aisles.
- It has to be capable of carrying an RFID payload of at least 2kg.
- It requires having a battery life of at least 2h.

The holonomic drive with a circular footstep makes the robot able to turn around in its same position avoiding any possible collision. In the first estimation in terms of RFID system needs, it has been calculated that the weight of the required payload shall not exceed the 2kg. Also, it has been considered that within 2 hours the system has to show enough progress in an inventory mission to evaluate its performance.
Finally, the system requires a brain, or computer, with the following requirements:

- Enough computational power to:
  - Navigate detecting and avoiding unexpected or sudden obstacles
  - Execute the map-less algorithm
- Means of wireless communication to gather data and communicate with other systems.

Table 4.1 summarizes the requirements.

Global HW & SW	Cost-effective COTS components ROS compatible
RFID payload	RFID tag direction detection Inventory accuracy as high as any RFID-based inventory robot
Robotic mobile base	Holonomic drive Obstacle detection Circular footstep Payload weight $>$ 2kg Battery life $>$ 2h
Brain	Navigation with obstacle avoidance Executing the map-less algorithm Wireless communication

Table 4.1: Summary of requirements specification

4.3 Preliminary design

The preliminary design shows how the previous stated requirements are met and results in a preliminary version of the robot, which will be tested in a real environment, see Chapter 5.

4.3.1 RFID payload

The requirements specify that the RFID system has to be able to determine the direction in which the tags are detected. This requirement is not achievable given a reader and a single and fixed antenna. The characteristics of the RF waves do not allow knowing the relative direction between the RFID tag and the antenna at the moment of the detection. Therefore, given the RSSI and/or any characteristics of the received wave, creating

a model from which the system can extract the relative direction of the wave is not feasible. This means that the solution for this requirement has to be in the design.

Thus, if the RFID system is not composed by a single and fixed antenna there might be chances to detect the relative direction between the tag and the antenna. For instance, if the single antenna is moving and its movement is finely controlled, it is possible to determine a relative orientation between the RFID tag and the antenna. For instance, if the detection of the tag is done while the antenna is on the right side of the robot, then it could be assumed the RFID tag is in the right side of the robot. As explained before, in the same way a model can not be created, in this case occlusions and multi-path could mislead some readings, however, once several detections are performed, most of them should lead towards the correct side.

However, having an antenna turning around the robot adds a complexity layer to the system that will drastically increase its cost and decrease its reliability.

Instead of having a single antenna moving, an equivalent can be a system with several fixed antennas, each of them with a different orientation. Therefore, the RFID tag direction will be given by the orientation of the antenna that performs the detection. RFID tags are unique, but due to multi-path effects is possible that the same tag is detected by more than one antenna. There are techniques to avoid these conflicts such as only taking into account the detection with the highest RSSI or the one with the highest read-rate, which is a parameter given by the reader that inform on the number of detections performed by the antenna. Another option is simply use the first or the last detections.

In any case, the more antennas with different orientations the better precision on the relative orientation, however, there are two clear limitations regarding the number of antennas. On one side, the RFID reader has only 4 channels for antennas, adding more antennas requires multiplexing the signal and this leads to a decreased performance. Also, adding a second reader to avoid the losses of the multiplexer increases substantially the cost of the overall system. On the other side, the robot is conceptually

small and compact, therefore, there is not much space to add a large number of antennas.
Thus, the proposal for the design is four antennas placed creating a square: it will have one antenna towards the front; one towards the back; one on the left and the last one on the right side.
Our main design hypothesis is that an RFID tag read from the frontal antenna will be, in most cases, in front of the robot. Therefore, all tags read from the frontal antenna are treated as if they were ahead of the robot. The same behavior is expected for each antenna. Using only four antennas that are perpendicular between them reduces the chance to suffer interactions or to have an RFID tag that can be read from more than one antenna at the same time. Although the RF waves will bounce causing detections on antennas that do not have the RFID tag in front of them, it is expected that this will not be the general case. Nevertheless, in the case of receiving detections of the same tag by several antennas, only the first detection will be considered. This keeps the system as simple as possible for a situation that it is not expected to happen often.
The second requirement for the payload should be satisfied by adding to the configuration of 4 antennas an RFID reader with enough capacity. In this sense, the one selected is the same one that it is set in the Advan-Robot, it can read between 350-400 labels per second and it has a Single Board Computer (SBC) on board that can communicate with the system through an Ethernet wire.
Regarding the RFID payload, the last design consideration is the height at which the antennas will be located. At this point of the development the largest set of tests is planned to be performed at the university library. Therefore, the center of the antennas will be placed at 1m from the ground, this way, the antennas are almost in the middle of the bookshelves. Figure 4.2 shows the initial design of the RFID system for the robot. This is the one used for the experimental tests in Chapter 5.

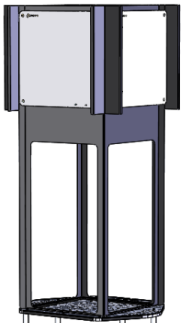


Figure 4.2: Initial RFID payload configuration, 2 of the four antennas are visible. The RFID reader is inside the cube that holds the four antennas. Also, the structure that will elevate the antennas at 1m height can be observed.

4.3.2 Robotic platform

Robotic mobile base

After performing a survey among different robotic mobile bases, the selected base is the Turtlebot-2[1], see Figure 4.3.
The Turtlebot-2 satisfies several requirements, it is a circular base with holonomic drive, which provides the capacity of rotating around its central vertical axis. According to the specifications given by the manufacturer, it can carry a payload of at most 5kg and it has two different battery configurations, at this point the smallest one is picked (2200 mAh Li-Ion), but having the option of doubling the battery capacity secures that the battery life requirement will be satisfied.
Moreover, the Turtlebot-2 has some useful sensors in its default configuration. It has encoders at the wheels and a gyroscope to get odometry measurements. It has a bumper, which is a contact sensor that can detect collisions at 3 points of its circular footprint. It also has infra-red sensors that can recognize reflectors and light detectors that can notice changes on the floor, such as when getting close to descending stairs.

[1] https://clearpathrobotics.com/turtlebot-2-open-source-robot/

Finally, all its hardware configuration options are ROS compatible.

Figure 4.3: Turtlebot 2

Obstacle detection

A key asset of this new version of inventory robot is its capacity to operate without the need of a map, therefore, a LIDAR is no longer a requirement. Instead, the only sensor that is required is one to detect obstacles. Taking into account that the robotic mobile base has already obstacle detection sensors such as the bumpers, only one among a LIDAR, a SONAR or an RGB-D camera is considered for the task.
In any case, the LIDAR option is not considered for this preliminary design as the available sensors in the market are too expensive.
The SONAR option is discarded as the main obstacle detection sensor due to its low angular precision which can affect the ability of the robot to navigate in narrow aisles. There exist belts of sonars that by comparing obstacle detections between sonar neighbors can improve the angular precision, however, adding so many sensors decreases the overall system

Figure 4.4: Orbbec Astra. Source:
https://orbbec3d.com/product-astra-pro/

reliability and does not significantly reduce the cost.
Regarding an RGB-D camera, it has the advantage with respect to the other sensors that can sense obstacles in 3 dimensions and that has a resolution of around one centimeter. However, it requires higher computational power to have a good update rate compared to other sensors.
After analyzing pros and cons of each type of sensor, the selected one is the RGB-D camera, precisely the Orbbec Astra[2] due to the experience obtained of working with it on other robots and that it can be purchased together with the Turtlebot-2 as a complement, see Figure 4.4.

Brain

The first candidate for the brain is a Single Board Computer (SBC), this decision is based on several aspects. They are very economic, the algorithm computational requirements are not high, so, it should be suitable for a simple SBC. Also, some of them are ROS compatible and most of them have a wireless communication module, as specified in the requirements. However, after deciding that the main sensor for obstacle avoidance is an RGB-D camera, the sensor is tested in two possible SBCs but the results obtained are not satisfying. Therefore, the SBC as a brain is discarded.
Then, the only option available for the brain is a NUC[3]. The main draw-

[2]https://orbbec3d.com/product-astra-pro/
[3]https://www.intel.es/content/www/es/es/products/boards-kits/nuc.html

Figure 4.5: Nuc i3 and power bank. Sources:
http://images.bit-tech.net/content_images/2015/02/intel-nuc-kit-nuc5i3ryk-review/nuc5i3ryk-1-1280x1024.jpg
https://www.litionite.com/product/tanker-mini/

back of using a NUC is that it has no internal battery and it consumes too much power to drain it from the Turtlebot, therefore, it requires an external battery. So, after a short survey on available power banks, it is decided to use the Litonite Tanker Mini[4]. Figure 4.5 shows both components.

4.3.3 Software

In terms of software design, the requirements specification of the hardware and its preliminary design bounded the possible options. From section 4.2 there are some requirements that directly affect the software design. For instance, it has to be ROS compatible and it requires navigation capabilities with obstacle avoidance. Actually, using ROS complements the second requirement, because there are open-source packages that with some tunning can be used for navigation with obstacle avoidance. Also, the drivers for the sensors and the Turtlebot platform are also ROS compatible. Therefore the packages designed to implement the algorithm have to also be part of the ROS ecosystem.
In addition, as the RFID payload is using an off-the-shelf reader, called

[4]https://www.litionite.com/product/tanker/

Advanreader-150.03[5], it has already its own software which can be interacted with a REpresentational State Transfer (REST) interface.
Therefore, from one side a software package is required that can interact with the reader, and then, transmit the information obtained by the reader to the main algorithm. On the other side the main algorithm requires to interact with ROS nodes, and therefore it has to be inside the ROS ecosystem.
Given all that the final software running in the robot is composed of four main pieces, see Figure 4.6.

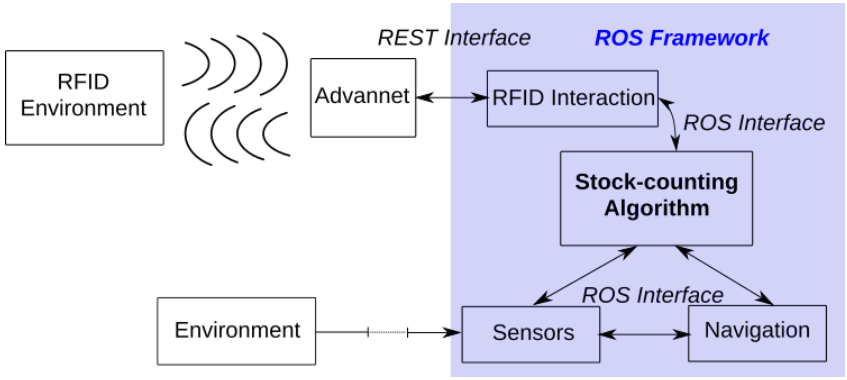


Figure 4.6: Software architecture scheme

The main piece of software is the implementation of the map-less algorithm for real robots, more information will be given later in this section. In any case, this implementation is much more complex than in the simulated graph due to the need to control a real environment. For the reasons previously explained, this piece of software is developed under the ROS framework and in Figure 4.6 called Stock-counting Algorithm.
The other package developed, is the interface between the stock-counting algorithm and *AdvanNet*, called RFID Interaction. It uses the REST interface to communicate with the RFID reader, it decodes the received information and it reports the relevant information to the main algorithm using messages inside the ROS framework.

[5]https://www.keonn.com/rfid-components/readers/advanreader-150.html

The third piece of software running in the robots is Keonn's *AdvanNet*[6], it runs on-board the RFID reader and it reads the information from the detected tags. Other software packages can interact with *AdvanNet* through a REST interface which allows commanding the reader as well as retrieving information obtained from the tags.
The last piece is the navigation software, which is used almost out of the box from the *Navigation Stack*[7] of ROS, and it is commanded and controlled by the stock counting algorithm using ROS interfaces.
Although one of the main benefits of using a stigmergic algorithm is its decentralization, due to implementation restrictions the system is centralized, as the ROS framework is, and all the communication depends on a single node. Nevertheless, the stigmergy-based algorithm for stock counting implementation is transparent to this centralization.

Stock counting algorithm

Algorithms 2 and 3 show the pseudo-code of the map-less algorithm's implementation for real robots, in Figure 4.6 called Stock-counting Algorithm. It is composed of four main procedures that run simultaneously.
In this section the notation developed in Section 3.2.1 is kept, however, some modifications and new definitions are required to adapt to the real environment.
First, we define tag data, T_{data}, as the data associated to an RFID label detection. $T_{data} = \{l_k, c_k^*, a_k, t_k, r_k, p_k\}$ is a tuple of six elements: the label id, l_k; the label counter, c_k^*; the antenna port, a_k; the timestamp of the detection, t_k; the Received Signal Strength Indication (RSSI), r_k and the label estimated position, p_k.
Note that the counter, c_k^*, is a local counter of the state, S^m, which will be used to update the counter, c_k. To differentiate between both, the local counter has an asterisk. The antenna port, a_k, is the port of the reader through which the label is detected. The RSSI is the power in the returned signal. Usually, high values of RSSI mean that the label is close

[6]https://www.keonn.com/software/advannet-software-drivers.html
[7]http://wiki.ros.org/navigation

to the antenna and that the returning wave comes from a direct path. On the contrary, when the RSSI is low the tag is usually far from the antenna or does not come from a direct path. In any case, high values of RSSI are more determining, as there are too many reasons why values can be low.
Take into account that the data received from the reader does not have the counter or the estimated pose, however, they are computed and added in the algorithm.
Along with this, $W = \{T_{data_i}\}$ is defined as the list of all the detection data tuples obtained during an algorithm step, this list is restarted at the start of each step. It behaves as a time window, so it is used to take into account only the detections performed during a single algorithm step. If the same tag is detected more than once, only the first detection is kept in the list, W. Given that the value r_k is known, another option would be to keep the detection with the highest RSSI. In this implementation the first is kept because the reader operates with S2, therefore, a tag will only answer once while it is being interrogated, therefore, the amount of answers from the same tag is limited, as it requires to be interrogated twice in the same algorithm step. Additionally, the fastest answer the antenna would get is through the most direct path, which should have the highest RSSI.
Second, the $Update$ function has to be redefined. Now the state of the task, $S^m = \{s_k\}$, where $s_k =< l_k, c_k >$, is updated with an element, $s_k^* =< l_k, c_k^* >$ extracted from T_{data}. So the function $Update(S^m, s_k^*)$ returns the state, S^m, with $s_k =< l_k, c_k^* >$, where the previous counter, c_k has been substituted by the counter, c_k^*, from T_{data}. However, if $c_k^* \leq c_k$ the state does not change. Notice that the state S^{m+1} is not obtained directly from the update function, now the updates are done for each label individually.
Third, the function $locate(a_k, t_k, r_k)$ has to be defined. This function estimates the position of the tag read as a function of the antenna, a_k, that performed the detection, the timestamp of the detection, t_k, and the received signal strength, r_k. This function has two steps. It first estimates the pose of the tag with respect to the antenna that performed the location. It places the tag in the direction of the antenna at a distance that it is a function of r_k. The implementation uses a simplified function to

obtain the distance in meters, see Equation 4.1. Notice that -40dbm is near the maximum value that can be obtained from any label and -80dbm is approximately the minimum. For values in between these two limits a linear interpolation is used.

$$\begin{aligned} d &= d_{min} && \text{if} \quad r_k \geq r_{max} \\ d &= d_{max} && \text{if} \quad r_k \leq r_{min} \\ d &= d_{min} + \frac{(r_k - r_{max})(d_{max} - d_{min})}{r_{min} - r_{max}} && \text{if} \quad r_{min} < r_k < r_{max} \end{aligned} \tag{4.1}$$

Where $d_{min} = 0.2m$, $d_{max} = 2m$, $r_{min} = -80dbm$ and $r_{max} = -40dbm$. The second step transforms the tag location from the antennas coordinate frame to the odometry coordinate frame, which is an external frame with respect to the robot. Any location requires a reference frame to be defined, the antennas coordinate frame is linked to the robot, therefore, if the robot moves, this movement has to be applied to the tag location to keep this location fixed in the environment. Another option to solve this is using a reference frame, such as the odometry coordinate frame which is external to the robot, so when it moves, the location does not need to change, it is always fixed. To do so, the timestamp of the detection, t_s, allows retrieving the transform between both frames from ROS.
Fourth, the available directions for the next move, $A = \{e_i\}$, are defined by the antennas of the robot. Candidate directions are: front, right, left and back. Therefore, the function $Check(costmap)$ returns $A = \{e_i\}$ with those that do not find an obstacle through a straight line of 1.3m, which is the maximum distance that the robot will travel in an algorithm iteration. Notice that the function $Check$ does not require to know the position of the robot, for definition it will be in the center of the costmap.
Fifth, $G^m_{e_i}$ has to be defined as a tuple with as many lists as available directions, for example, if only two directions are available, $G^m_{e_i}$ is a tuple with two lists. These lists contain the labels of the tags detected in the correspondent direction. Therefore the function $Group(l_k, p_k, A)$ has to be defined. This function assigns l_k into a list of $G^m_{e_i}$ and each list is associated to an e_i. To do so, it is required to transform again the location of each detected tag, however, now it is done from the odometry reference

frame to the antennas reference frame.
One could think that this step is unnecessary as the antenna that detected the tag is already known. But, detections are done while the robot is moving and, as it has been explained, the location of the tag is defined with respect to an external coordinate frame. Therefore, when the robot stops and the possible directions need to be evaluated, it is required to relate the orientations of the antennas with respect to the location of the tags detected. To do so, the locations are transformed back to the antennas coordinate frame. At this step, no overlap is considered between the regions defined by the orientations of the antennas.
Sixth, the attraction, $u^m_{e_i}$, is computed as shown in Equation 3.1 but instead of using $L^m_{v_i e_{ij}}$ it uses the lists in $G^m_{e_i}$.
Finally, function, $Move(e_i)$, has to be defined. $Move(e_i)$ sends a navigation goal, which is a new pose with respect to the current pose of the robot, in the direction of e_i and a distance that it is randomly selected between 0.7m and 1.3m from a uniform distribution. The *Navigation Stack* of ROS receives this goal and moves the robot accordingly, which results in an update of the agent pose, $P^{m+1}_{agent} = Move(e_i)$. In some occasions the robot will perform an extended move, where the distance to travel will be increased by a factor of 1.5, in this cases the function used will be called $ExtendedMove(e_i)$.
Now it is possible to follow Algorithms 2 and 3. The implementation assumes that each robot of the system has its own copy of the state, S^m, so they communicate peer-to-peer to share the detections obtained and update the state.
It starts by executing four procedures: *RFID interface*, *Update environment*, *Navigation control* and *Stigmergic stock-counting*. All procedures are active until the $end\_condition$ is met.
The *RFID interface* procedure receives the information from the *RFID interaction* package. This package is connected to the reader through a REST interface. Each time that the reader detects an RFID label, it transfers the associated data, T_{data}, by means of the REST interface to the *RFID interaction* package and this package converts it into a ROS message that it is sent to the *RFID interface* procedure. Then, the *RFID*

interface checks that the label, l_k, is in the ground-truth, if it is not, is discarded. Although, all labels in the environment are being detected, because it is known that the ground-truth 100% accurate, it might be not convenient to led the robots follow labels which origin is not known. Or, for instance, if the robot has to stock-count only a determined area of the environment, the ground-truth filtering allows keeping the robot in the right area.
Then, it checks if it is a new label, according to that, it creates or updates the associated counter, c_k^*, and it adds the counter to T_{data}. Finally it sends T_{data} to the *Stigmergic stock-counting* procedure and $s_k^* =< l_k, c_k^* >$ to the other robots.
The *Update environment* procedure is completely asynchronous from the other procedures and it just updates the state, S^m, for the robot that receives data from other robots of the system.
The *Navigation control* procedure waits until the next direction for the robot, e_i, has been received. If e_i is not empty it will execute the function $Move(e_i)$ in order to send a navigation goal to the *Navigation Stack* of ROS. However, if the direction is *back* and the last direction was also *back*, then it increases the movement distance by a factor of 1.5 in order to avoid navigation oscillations. The robot's moving direction is towards the front, it does not move laterally or backwards, therefore, choosing back twice in a row indicates an oscillation. In case that $e_i = \emptyset$, then it moves the robot towards any available direction, this choice is random. This procedure also interacts with the *Navigation Stack* of ROS to cancel any current goal if required and with the *Stigmergic stock-counting* procedure to notify if the robot is pursuing a goal or is stopped, and, to transfer the current costmap. For simplicity this lines are omitted in the Algorithm 2
The procedure *Stigmergic stock-counting* is composed by three internal procedures that execute sequentially: *Get candidate directions*, *Compute attraction* and *Best direction*.
It starts with the procedure *Get candidate directions*. It is just receiving T_{data} from the *RFID interface* procedure until the robot stops moving. So, if T_{data} is received and the robot is moving, it updates its current state, $Update(S^m, s_k^*)$, then it computes the estimated tag position

$p_k := locate(a_k, t_k, r_k)$ and, finally, adds the value T_{data} to the window list W. Once the robot stops moving, it receives the current costmap of the robot and it extracts the available directions, $A := Check(costmap)$. If data is sent while the execution is taking place and the robot is not moving, this data waits in a queue until this procedure is again available.
Once A is obtained, the next procedure, *Compute attraction*, starts. Each detection, T_{data}, in the windows list, W, is grouped according to its relative orientation with the robot, $G^m_{e_i} \leftarrow Group(l_k, p_k, A)$. These relative orientations are limited to the ones in A. Then, it computes the attraction for each $e_i \in A$. If $A = \emptyset$, then it sets the attraction as $\varnothing$.
Once the attraction is computed, the last procedure, *Best direction*, starts. This procedure simply decides the direction of maximum attraction, $e_i :=$ *one-of(arg-max*$_{e_i \in A}(u^m_e)$*)*, so the direction that maximizes the value of attraction, in case of having more than one direction with a maximum value of attraction, the best direction is selected randomly between them. In case that there is not attraction, it sets e_i as $\varnothing$. Finally, it sends e_i to the *Navigation control* procedure.
So the robot starts moving and, again, the procedure *Get candidate directions* erases the window list, W, and starts receiving and processing data from the procedure *RFID interaction*.

Stock counting algorithm example

In order to clarify the procedure of the map-less stock-counting algorithm, the following figures, from Figure 4.7a to Figure 4.7e presents a schematic of the process.
Consider the red dots as EPCs already read in a previous iteration and the green dots as EPCs read for the first time in this iteration. Obviously, there might be more EPCs but if the robot has not read them, it does know they exist. So, Figure 4.7a presents the robot in the middle of an environment with tags that has been detected for the first time and others that have been already detected before.
In Figure 4.7b the robot has obtained the costmap of its surroundings and it has found an obstacle, the gray rectangle, in its back. Therefore, in Fig-

Algorithm 2 Stigmergic stock-counting pseudo-code

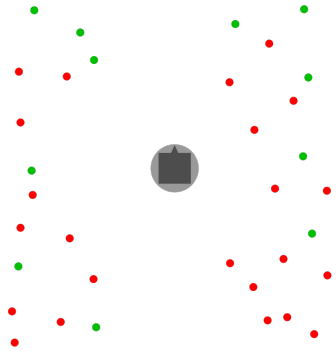
Start procedure: RFID Interface, Update environment, Navigation Control, Stigmergic stock-counting
procedure RFID INTERFACE(T_{data})
    while not *end_condtion* **do**
        if T_{data} received from *RFID interaction* **then**
            if l_k in ground-truth **then** ▷ Filtering step
                if $l_k \in S^m$ **then**
                    $c_k^* := c_k + 1$
                else
                    $c_k^* = 1$
                end if
                $T_{data} \leftarrow c_k^*$ ▷ Adds c_k^* into T_{data}
                Send T_{data} to *Stock-counting procedure*
                Send $s_k^* =< l_k, c_k^* >$ to others *Upload environment*
            end if
        end if
    end while
end procedure
procedure UPDATE ENVIRONMENT(s_k^*)
    while not *end_condtion* **do**
        if s_k^* received from others **then**
            Update(S^m, s_k^*) ▷ Update own state from others
        end if
    end while
end procedure

```
25: procedure NAVIGATION CONTROL(e_i)
26:     while not end_condtion do
27:         if e_i received then
28:             if e_i not ∅ then
29:                 if e^{m-1} = e_i = back then
30:                     P^{m+1}_{robot} := ExtendedMove(e_i)        ▷ Larger move
31:                 else
32:                     P^{m+1}_{robot} := Move(e_i)
33:                 end if
34:             else
35:                 P^{m+1}_{robot} := Move(One-of(A))              ▷ Random move
36:             end if
37:             e^m = e_i                                           ▷ Store last direction
38:         end if
39:     end while
40: end procedure
41: procedure STIGMERGIC STOCK-COUNTING
42:     while not end_condtion do
43:         Get candidate directions(T_data)
44:         Compute attraction(A)
45:         Best direction(u^m_e)
46:     end while
47: end procedure
```

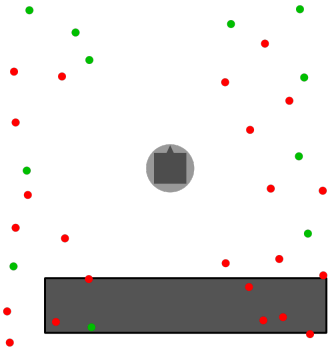
Algorithm 3 Stigmergic stock-counting internal procedures

procedure GET CANDIDATE DIRECTIONS(T_{data})
    Restart W ▷ Restart of the time window list
    while Robot is moving **do**
        if T_{data} is received **then**
            $Update(S^m, s_k^*)$ ▷ Update own state
            $T_{data} \leftarrow p_k := locate(a_k, t_k, r_k)$ ▷ Estimate tag location
            $W \leftarrow T_{data}$
        end if
    end while
    $A := Check(costmap)$ ▷ Determine candidate directions
end procedure
procedure COMPUTE ATTRACTION(A)
    for all $T_{data} \in W$ **do**
        $G_{e_i}^m \leftarrow Group(l_k, p_k, A)$ ▷ Group tag labels wrt. directions
    end for
    if $A \neq \emptyset$ **then**
        for all $e_i \in A$ **do**
            $u_{e_i}^m := f(S^m, G_{e_i}^m)$ ▷ Compute the attraction
        end for
    else
        $u_{e_i}^m := \emptyset$
    end if
end procedure
procedure BEST DIRECTION(u_e^m)
    if $u_e^m \neq \emptyset$ **then**
        $e_i :=$ *one-of(arg-max*$_{e_i \in A}(u_e^m)$ ▷ Choose direction of max. attraction
    else
        $e_i := \emptyset$
    end if
    Send e_i to Navigation Control
end procedure

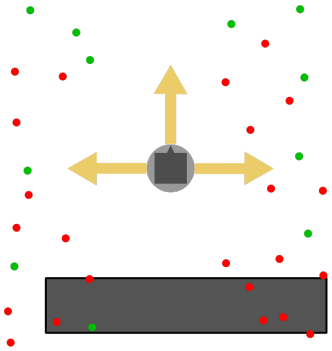
ure 4.7c the robot checks its costmap and realizes that it has three out of the four possible directions: front, right and left.
Then, in Figure 4.7d, it calculates the attraction felt through each of its three possible directions. The division of the surrounding environment is done without overlap, therefore, each RFID tag counts towards one direction. Finally, in Figure 4.7e the robot decides that the next best direction, given by the maximum attraction, is towards the front. Clearly is the direction that presents more new detections.



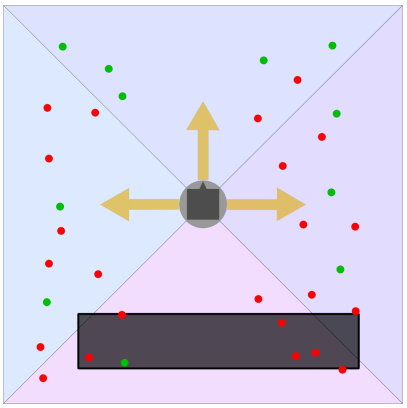
(a) The robot detects the RFID labels in its environment. The red dots represent the tags that had already been detected. Green dots represent those tags that have been detected for the first time.



(b) At this step, the robot is aware of the position of the obstacles in its surroundings. The gray rectangle represents an obstacle, such a shelf.

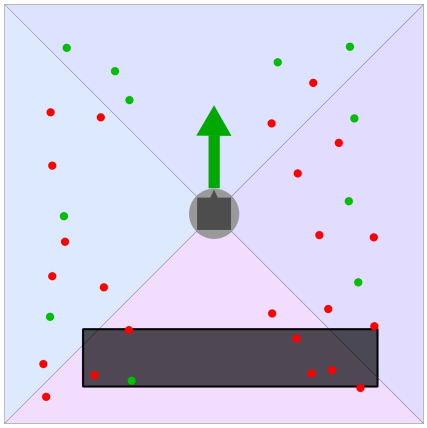


(c) The robot has only 4 relative orientations to move: front, back, left and right. It matches these directions with the obstacles of the surroundings and decides that only 3 out of the four directions are available.



(d) At this step, the robot assign the detected labels at each of the three possible directions. Obviously, the tags that are towards the not available direction are not considered.

Figure 4.7: Schematic representation of the robot performing a step of the algorithm.



(e) Finally the direction with the largest attraction is selected to set the next navigation goal, which is a position around a meter further in the selected direction.

Figure 4.7: Schematic representation of the robot performing a step of the algorithm.

4.3.4 Designed robot

Once the design has been specified in terms of software and hardware, the prototype obtained can be seen in Figure 4.8. Two units of this prototype have been built during the thesis.

Figure 4.8: Prototype of the inventory robot

Figure 4.8 shows the cubic shape on top of it that contains the RFID antennas as well as the reader. The metallic structure below the cube, places the center of the antennas 1m from the ground ensuring that the antennas will easily read the tags placed at a similar height. Then, the structure

is linked to the own structure of the Turtlebot-2 with its characteristic hexagonal shape and the 4 metallic pillars, this structure holds the NUC, the RGB-D camera and the battery for the NUC, although this last component can not bee seen from this perspective. Finally at the bottom there is the Turtlebot-2 base.
Table 4.2 summarizes the cost of the main components of the selected design. As explained at the beginning of the chapter, the intention was to create a cost-effective prototype. In order to obtain all the components, some of them are leased and the others have been acquired with the money of a prize won at the university and thanks to funds from the university program Maria de Maeztu.

System	**Element**	**Cost (€)**
Payload	1 x Reader	1300
	4 x Antenna	400
Platform	Turtlebot-2	650
	External battery	150
	RGB-D camera	150
	Intel NUC i3	350

Table 4.2: Prototype components cost details.

This prototype is the one used for the experiments in a real environment detailed in Chapter 5. The following sections of this current chapter explain the next design iteration for this robot taking into account the results and the learnings from the experiments.

4.4 Lessons learnt

Chapter 5 will prove that both the algorithm and the robot can take inventory of an unknown environment thanks to the new map-less paradigm for stock-counting. This section gathers the lessons learnt during the initial design and at the experimental tests that will be shown in chapter 5. Table 4.3 summarizes the issues found.

Stability	The prototype has stability issues and it wobbles.
Obstacle detection	The prototype has difficulties to detect obstacles at its sides.
Autonomous charging	The external battery of the design prevents the robot to autonomously charge.
Handle	The prototype does not have any handle for transportation.

Table 4.3: Summary of issues encountered

During the tests it has been detected that the initial design of the robot is not very stable, mainly due to the height of the RFID payload. The robot wobbled significantly at every start and stop of its movements, and also, if its speed was not properly adjusted it could have fallen after a sudden brake. Therefore, the final design will require to reduce its height.
The use of only a RGB-D camera for obstacle detection has complicated some aspects of the navigation given the algorithm developed. The camera has a 60^{o} field of view and its range was reduced to two of meters for computational and precision issues. This two factors prevented the robot to properly detect obstacles at its sides. Therefore, when the robot selected a direction towards one of its sides, it did not know that there was an obstacle. This resulted in the robot performing long and weird trajectories trying to avoid an obstacle that unexpectedly appeared.

Additionally, while the initial design was created and the tests were performed new low cost LIDARs appeared in the market. Using this type of sensor will solve the problem of detecting obstacles at the sides as they have a field of view of 360º. Moreover, this type of sensors opens the door to new features such as creating maps to add a finishing condition to the system by analyzing the coverage performed by each robot, or to provide RFID tag location capabilities to the system.
The initial design also lacked of an autonomous way of charging the battery, this was forced due to the external battery, which could not be charged together with the Turtlebot. Therefore, a charging station was not really useful as the external battery will required a user to plug it for charging. Therefore, the final design will remove the external battery in order to provide the robot with autonomous charging.
A minor issue was transportation, the initial design did not have any handle to lift the robot for transportation.
Finally, the Turtlebot-2 has been proven as a very suitable robotic base. It is in the market since October 2012, which means that it has a huge community with lots of experience that can help whenever an issue is faced. In addition, it is developed as open hardware[8], which if at any point its production stops, there will be manuals to continue.

4.5 Final design

This section covers all the changes in order to end up not only with a robot that can take advantage of the map-less algorithm developed, but, a robot that has all the capabilities than any other RFID-based inventory robot with an increased autonomy, robustness, scalability and with a cost an order of magnitude lower than any other inventory robot. Additionally the design will enhance the modularity and compactness of the robot's concept.

[8] http://freedomdefined.org/OSHW

4.5.1 RFID payload

The RFID payload will be re-designed for 2 reasons:
The first is to change the height of the RFID system to provide better stability to the robot.
The second reason comes due to the analysis of the radiation pattern of the antennas, as it can be seen in Figure 4.9. Both diagrams are not com-

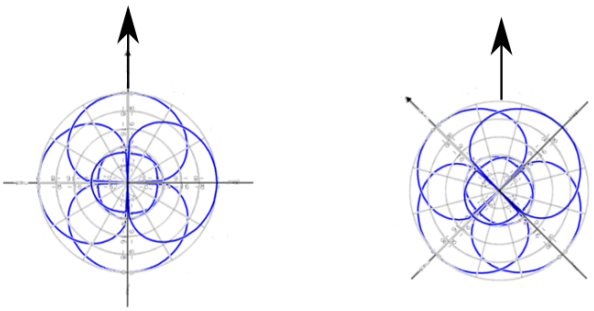
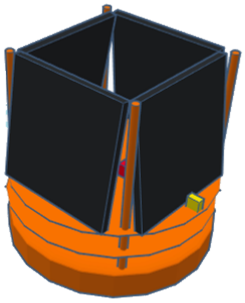


Figure 4.9: Radiation diagram of four antennas represented as a point in the center of the diagram. On the right side the configuration of the current robot, on the left side the diagram of a robot having the four antennas rotated 45^{o}.

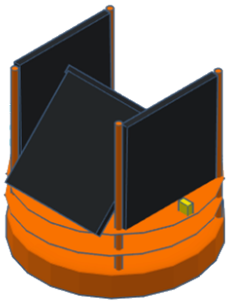
pletely realistic, as the antennas are represented as a central point on each of them. Nevertheless, it can be observed in Figure 4.9 that the current configuration has the central part of the radiation lobes of the frontal and back antennas aligned with the movement of the robot, and therefore, most of the RFID labels will not be aligned, as they are usually in the shelves on each side. On the contrary, the configuration proposed on the right side, has the 4 lobes with a contribution towards the sides, which could potentially improve the RFID detection capabilities.
Three RFID system configurations have been proposed in order to reduce the height of the RFID system while trying to keep a similar accuracy of RFID detections. Figure 4.10 shows the tested configurations. Notice that all the options have the center of the antennas at around 30cm from the

ground.



(a) Same antenna configuration as the initial design but placed at around 30cm from the ground, named Square.

(b) Antenna configuration rotated 45^{o} with respect the initial design, placed at around 30cm from the ground, named Diamond.



(c) Antenna configuration with the frontal and rear antennas as in the current configuration but the lateral antennas tilted up to better capture the highest RFID labels. Also the whole system at around 30cm from the ground, named Tilted.

Figure 4.10: Proposed antennas configurations for the new design of the robot.

In Figure 4.10a, it is shown the same configuration as the current robot

but with the antennas placed at around 30cm from the ground. From now on known as Square configuration.
Figure 4.10b presents a configuration where the antennas have rotated 45^{o}, and therefore, the maximum of the radiation pattern is towards the sides of the robot, as seen in Figure 4.9. From now on known as Diamond configuration.
The last proposed configuration, see Figure 4.10c, has the frontal and rear antennas as in the initial configuration, but the antennas on each side are tilted up in order to improve the detection capacity in height despite reduction of the robot's height. From now on known as Tilted configuration.
Two tests have been planned in order to assess the best configuration for the antennas. The first test is to determine the RFID reading accuracy of the proposed antenna configurations with respect to the height of RFID labels. The second test is to assess the overall performance of the RFID system in an environment with many RFID tags at different heights and orientations. Both tests are compared against the previous RFID system version, this will be used as a control test to assess the newly proposed configurations.

Height test

This test aims at detecting the maximum height and the change of performance in reading RFID tags as their height increases.
Given the radiation pattern of the antenna, we could have an intuition on the maximum height at which the readings could happen, however, the tests will be done in an environment with multi-path effects, as in the real operation, therefore, the readings can be different than those that are deducted from the radiation pattern.
Besides the environment, there are 2 parameters that can influence the results:

- The configuration of the antennas
- The height of the antennas

Therefore, the tests will be performed with an RFID system put directly on the robotic base. Also, the previous version of the robot will be used to evaluate the influence of height, it will also be used to assess if there is an improvement or a diminish on the reading capabilities of the system. In any case, the reduction of the height is not optional.

Test characteristics
The systems that will be tested are the following:

1. Previous design, see Figure 4.8
2. Square configuration, see Figure 4.10a
3. Diamond configuration, see Figure 4.10b
4. Tilted configuration, see Figure 4.10c

The target RFID labels are inside in a cardboard box with 100 labels as it is shown in Figure 4.11 it has 20 rows of 5 RFID labels each, the rows are 400mm wide and the total height of the box is 1185mm.
The box will be placed at three different heights: ground height, at 740mm from the ground and at 1320mm (740mm + 580mm), so the highest row of tags will be above 2.5m. In addition, for each height three passes at different distances from the box to the center of the robot are performed. Distances are: 400mm, 700mm and 1000mm. Figure 4.11 presents a scheme of the tests.

Results
Table 4.4 summarizes the results obtained. As expected, with the increase of the height the amount of missing labels increases for all the configurations. It is also worth mentioning that the amount of missing labels for the previous configuration is minimal, although a test was removed because the results were too different from the overall behavior, and considered an outlier.
Regarding the three proposals, the Square and the Diamond configurations performed better than the Tilted configuration, that had more problems to detect tags at the third height, so tags from 1.3m to 2.5m.

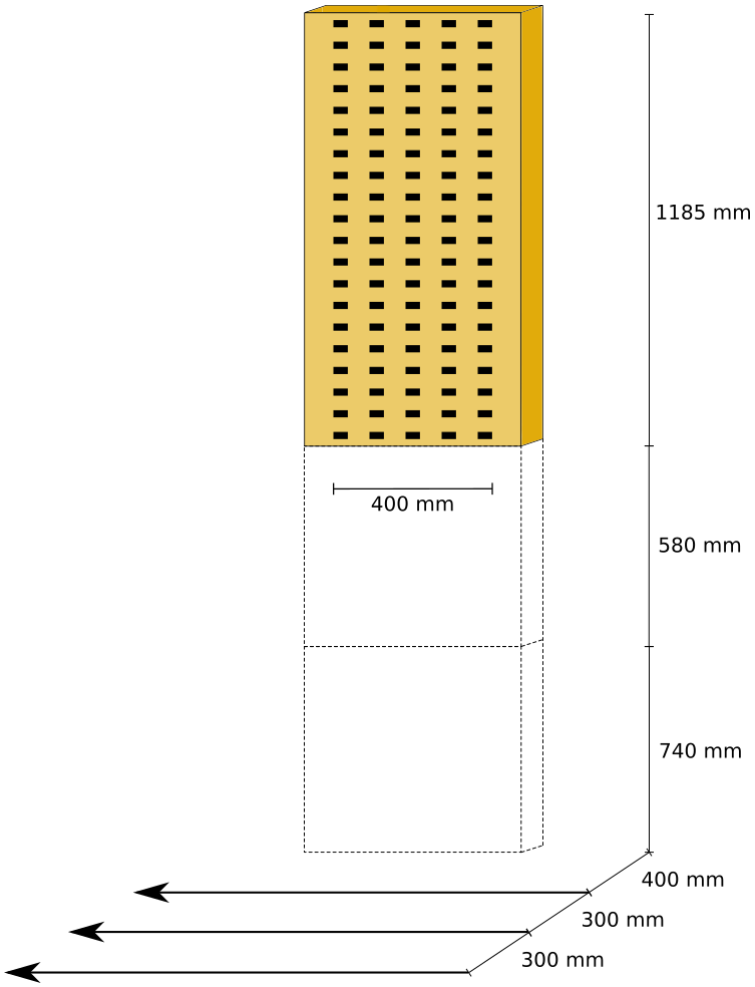


Figure 4.11: Scheme of the test for assessing the performance of the RFID system with respect to the height of RFID tags.

Environment test

This experiment aims to compare an overall reading performance. The proposed configurations will move through an environment with more than a thousand of RFID tags at different heights and orientations.

Configuration / *Tests*		**Previous**	**Tilted**	**Square**	**Diamond**	**Total**
	D1	0	0	0	0	**0**
Height 1	*D2*	1	1	2	0	**4**
	D3	NA	2	0	1	**3**
	D1	0	2	3	1	**6**
Height 2	*D2*	0	0	1	1	**2**
	D3	0	4	0	3	**7**
	D1	0	4	2	1	**7**
Height 3	*D2*	0	3	3	2	**8**
	D3	0	6	1	3	**10**
Total		**1**	**22**	**12**	**12**	

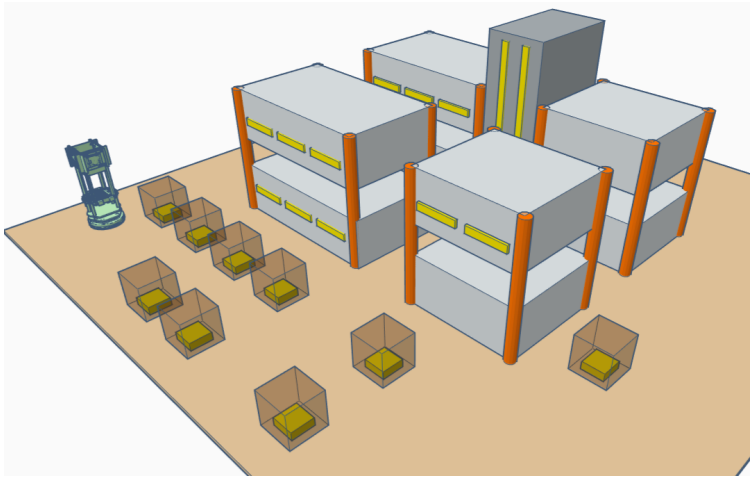
Table 4.4: Summary of results. Each value of the matrix is the number of unread out of 100 total tags for the configuration in a determined test. D1 makes reference to the test performed at 400mm from the box, D2 at 700mm and D3 at 1000mm.

Test characteristics

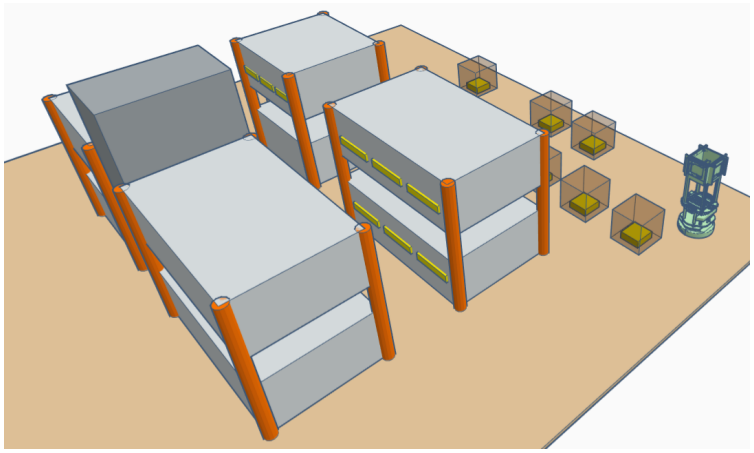
Figure 4.12 shows two perspectives of the space that the system has to cover. In Figures 4.12a and 4.12b can be observed that the RFID tags can be found inside cardboard boxes, hanging from shelves at two different heights and in two large strips following a column. For the sake of clarity, this test is performed at Keonn's warehouse, but to better show the environment a 3D representation has been created.

Each configuration will be moved twice through the environment as shown in Figure 4.13. It is important to take into account that the movement is controlled by us at a fixed speed, so it does not adapt to the detections obtained by the system.

Our ground-truth contains more than 1,300 RFID tags. However, it is filtered and only those tags that have been read at least once by any configuration, will be counted. As the purpose of the test is to compare the different antenna configurations, this should not affect the final conclusions. Finally the filtered ground-truth contains 1,265 RFID tags.



(a)



(b)

Figure 4.12: Environment for the test. The yellow blocks represent RFID tags, the total amount of tags is 1,265.

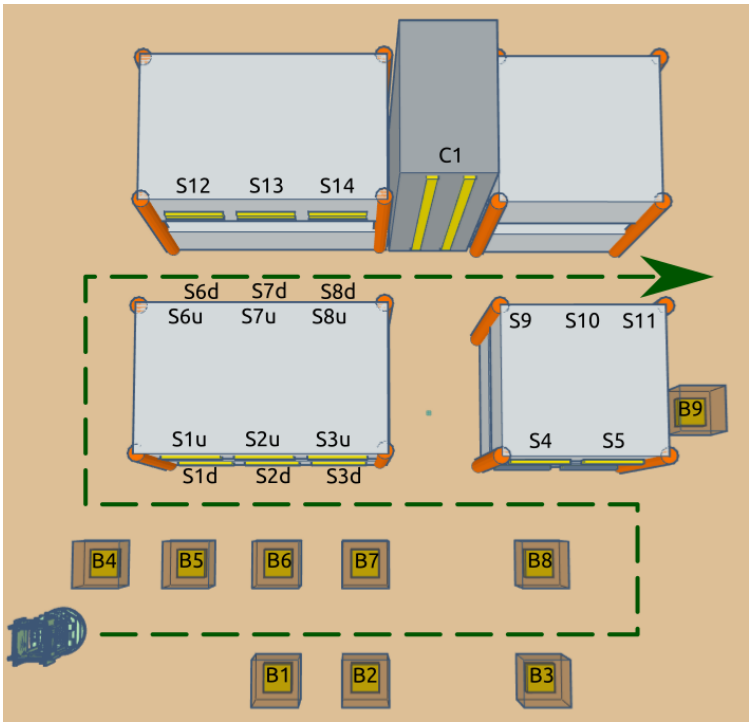


Figure 4.13: Path to cover for each of the configurations to test.

Results

Table 4.5 summarizes the results obtained. Each value represented is the average of both passes performed by each configuration.

	Previous [%]	**Tilted [%]**	**Square [%]**	**Diamond [%]**
Boxes	81.35	82.38	71.57	75.52
Shelves	99.36	87.99	88.84	83.64
Column	100	93.33	96.67	86.67
Overall	92.61	88.31	82.41	82.41

Table 4.5: Summary of results

The best configuration is the Previous design, however, if we observe the results regarding the RFID tags inside boxes the Tilted configuration performs better. Nevertheless, when reading the tags at the shelves the Square configuration performed better than the Tilted. From an overall perspective the Diamond and the Square configuration have the same results, however, the Square performs clearly better at reading the tags on shelves.

It is worth mentioning that the tags on the shelves are more representative than the ones inside the boxes. Inside the boxes RFID tags are packed and close to the floor, these conditions are not very common in the store. On the contrary, the tags hanging from the shelves are more similar than those in a store in terms of density and distribution.

Chosen configuration

The clearest conclusion obtained from the tests performed is that the previous configuration is the one that performs the best.

Regarding the height test, the new configurations that worked the best were the Square and the Diamond. In contrast with the environment test where the Tilted configuration outperformed the other two. Nevertheless, at the environment test we can see that the Square configuration performs better at reading the tags in the shelves, which is more relevant.

In conclusion, the chosen configuration will be the Square. There are two main reasons, first the results obtained by this configuration are the more regular as it performed well in both tests. Second, the only difference with the previous configuration is its height and it will be proved, in Chapter 5, that it works fine in terms of RFID readings.
Nevertheless, as it has been observed that the height plays an important role in terms of RFID detections, it has been decided to increase the height as much as possible but without incurring in stability issues. For this reason the final position of the RFID payload will have its top at 60cm, ensuring a high stability and being able to move under tables, that are usually at 70cm.

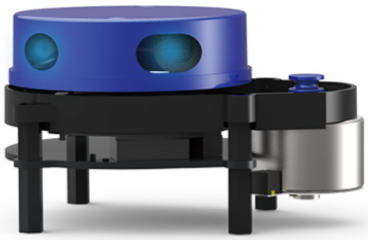
4.5.2 Robotic platform

Obstacle detection

The LIDAR technology is relatively new and, therefore, expensive. However, the increased interest on autonomous vehicles is driving many companies to make an effort and develop better and cheaper LIDARs, so that, they can have its component in an autonomous vehicle.
These efforts lead us to find two 360º LIDARs available at a price around 100USD by the end of the validation tests explained in Chapter 5. These LIDARs are the RPLidar A1[9] and YDLidar X4[10], see Figure 4.14. Both work in a similar way, they have a light emitter that sends a modulated laser signal, and, a light detector that receives the signal after its reflection against a surface. By having the time difference between the emission and the reception, it is able to compute the distance. Then due to the movement of its sensors it achieves a 360º field of view.
After a brief assessment of both sensors, we decided to stay with the YDLidar due to its more efficient power management. Briefly, this sensor is able to control the power received and, therefore, to stay off if it is not being started through its driver, while the RPLidar starts its rotation as

[9] https://www.slamtec.com/en/Lidar/A1
[10] http://www.ydlidar.com/products/view/5.html

(a) RPLidar A1. Source: https://www.slamtec.com/en/Lidar/A1



(b) YDLidar X4. Source: http://www.ydlidar.com/products/view/5.html

Figure 4.14: New 360º LIDARs available in the commercial market.

soon as one plugs it.
Adding a LIDAR to the system affects it in three ways. First, the internal data flow and its processing is lighter than with an RGB-D camera, therefore, using an SBC as a robot brain becomes an option.
Second, the navigation strategy changes as the robot goes from a sensor that has a narrow field of view but in 3D, to a flat field of view, but at 360º and with a faster update rate.
Finally, having a LIDAR sensor will add SLAM capabilities to the robot. Although it has been proved that the map is not required for inventorying an entire space, it can add relevant features. Briefly, it allows the system to locate items in the space. It facilitates the returning of the robot to its charging station, which leads to a more efficient balance between the power to inventory and the power to come back to the charging station. Finally, a finishing condition can be added by analyzing the map, such as there are no new places to move. More details will be given around these features in section 4.5.3.
The main downside of using a LIDAR sensor instead of an RGB-D camera is that the detection of obstacles remains in a single plane. This endangers the navigation as some obstacles can not be detected. To solve this issue, a SONAR sensor can be added in the sense of the moving di-

rection. This way, the robot is able to detect obstacles out of the LIDAR’s plane, in addition, it can detect obstacles that the LIDAR can not, such as mirrors, glasses or light absorbing surfaces.
The only added difficulty is that the SONAR has a very low angular precision, usually several tens of degrees. So, if the sonar detects an obstacle in an extreme of its field of view, it will mark the whole field as occupied, which can prevent the robot from moving through narrow aisles. To deal with this, the maximum range of the SONAR has been limited in such a way that the maximum size of an obstacle detected by the sonar will have the robot’s diameter.

Brain and robotic mobile base

Removing the RGB-D camera from the system allows using less powerful SBCs like a *Raspberry Pi*[11] (RPi). This has two benefits for the system. First the price of the brain is reduced almost an order of magnitude. Second, the battery consumption for an RPi is lower than for a NUC. This allows removing the external battery for the brain.
The Turtlebot 2 has several power outputs, by means of DC/DC converters it is possible to provide the required power to each component. Therefore, with the Turtlebot’s large battery (4400mAh) it is possible to feed all the robot’s components during almost three hours while the robot is on operation. Therefore, the initial requirement of having at least 2 hours of battery is maintained.

4.5.3 Software

This subsection briefly describes improvements that are currently under research. So they have been started during this thesis but they have not been fully tested or implemented. Nevertheless, this final iteration of the robot design is ready to implement them.

[11] https://www.raspberrypi.org/

Docking station

Once the mission is over or the battery is almost finishing the robot requires the ability to return to its charging station. In this sense, the Turtlebot 2 has a charging station, see Figure 4.15, that can be purchased and it has also a software package[12] that can drive the Turtlebot into the charging station if it is in sight and at less than five meters.

Figure 4.15: Turtlebot docking station. Source:
https://www.roscomponents.com/358-thickbox_default/docking-station-tb2.jpg

Therefore, our concern is reaching the area from which the Turtlebot can autonomously dock. To do so, two alternatives are under study.
The first is placing a battery assisted passive tag (BAP tag) on the docking station. This type of tags allows detection up to 30 meters. This way, if the robot is far from the docking station, it could detect the tag and by following a similar navigation method than the one used for taking inventory it could be able to come back to the docking station, until the infra-red sensor of the Turtlebot detects it.
The second alternative is running a SLAM process on the background while the robot is taking inventory. This would allow locating the robot as well as the docking station, so at the end of the mission a navigation goal from one pose to the other could be given. In this case, the SLAM process would not influence the inventory mission, but while it is being

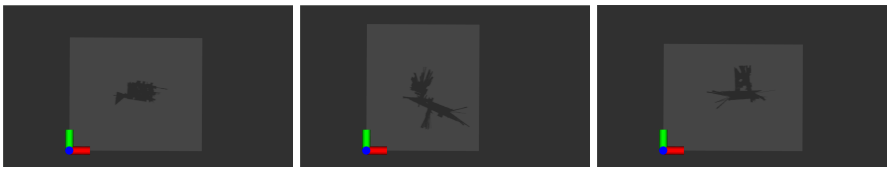
[12] https://wiki.ros.org/kobuki_auto_docking

performed a map would be generated. Additionally, the Turtlebot requires reaching a position around 5 meters from the docking station with line of sight. This is not a hard requirement and the resolution of a map to achieve this does not need to be high, therefore, the computational requirements of the SLAM process should not be high.

Collaborative mapping

The map-less stock-counting algorithm is stigmergic, therefore, it is specially well suited for multi-robot systems or, even, swarm systems. The LIDAR included in the new design allows each robot to create a map of its environment. However, the SLAM process creates maps that are local, they are centered in the initial position of the robot and with its orientation and they suffer the local perturbations of the precise robot. So, if the robot slips or if there are misleading LIDAR readings the changes in the map are local. Additionally, the robots are performing a task that it is well distributed among them, as the results show, so, the local maps can have very little overlapping, meaning that they are not complete maps of the environment.
Therefore, creating maps and retrieving the maximum benefit from them is only useful if maps are shared, common and complete. Solving the problem of collaborative mapping is easier if the initial poses of the robots are known with respect a given reference. In our case, there is no prior knowledge of the environment, therefore, an initial given reference is not available. In this cases, where the initial positions are not known the problem becomes more complex [40].
One of the simplest solutions to this problem is using image processing algorithms that can rotate and translate images to stitch them. Therefore, by taking the individual maps, a common one can be created by stitching the individual maps, and, in addition, a transform can be established between each individual map and the common one. More interestingly, if the overlapping of the maps is low this method still works.
During this thesis a contribution has been made to the ROS package mul-

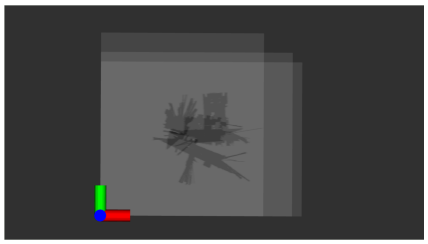
tirobot_map_merge[13] providing this transform relation between the maps and not only the image of the general map, as initially did the package.
An example has been developed to illustrate the process. Three maps of the same space are created by a robot following each time a different path on that space. Therefore, the maps are different until the end of the process. The three processes are recorded using internal ROS methods. Later, offline, the recorded processes are played and the map merging program executed to test it. This way it is not required to create maps every time an adjustment or a test has to be performed.
Figure 4.16 shows the starting maps of the robots, so what they capture from the surroundings at the start of the process, it is clear that they start from different locations, and therefore, there is no initial overlap. Then Figure 4.17a shows the three maps overlapped without any logic, at this stage the stitching process has not yet start.
In Figure 4.17b the stitching process has started. It is relevant noticing that there is not much overlapping between the maps, but the result is promising. The three maps can be observed by following the internal lines and the different gray shades. Finally, from Figure 4.18a to Figure 4.18d it can be observed how the stitching continues while the robots cover the whole space.
In the last map, Figure 4.18d, there are almost no internal lines and the gray shade is the darkest meaning that the three robots have covered the whole space, and the three maps completely overlap.



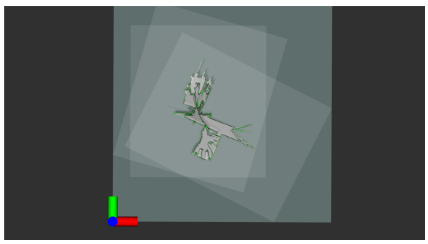
(a) Map 1 (b) Map 2 (c) Map 3

Figure 4.16: Initial maps

[13] http://wiki.ros.org/multirobot_map_merge

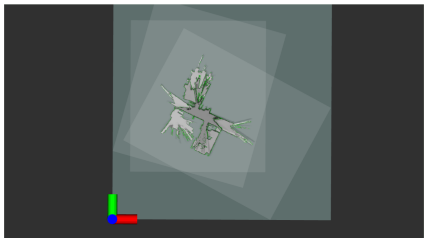


(a) The three maps without a logic transform

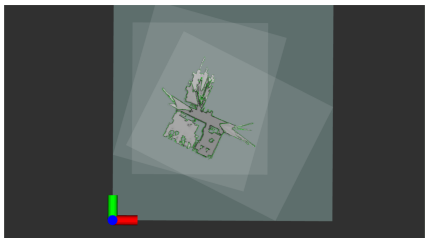


(b) The first transform for the three maps.

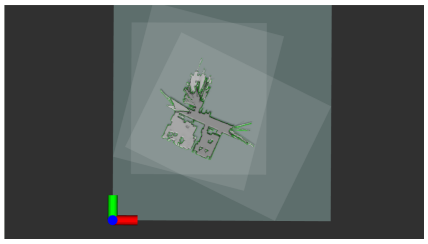
Figure 4.17: Initial maps



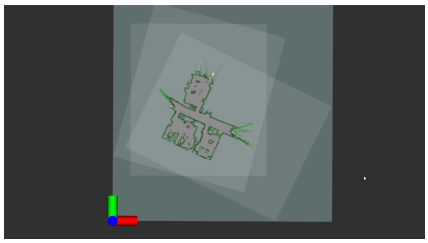
(a) Map merging intermediate state - 1



(b) Map merging intermediate state - 2



(c) Map merging intermediate state - 3



(d) Map merging final state

Figure 4.18: Process maps

If a SLAM process is running in the background while the map-less inventory is taking place. A collaborative mapping between the agents can be created and it could enhance the entire system performance. For instance,

it could send the most appropriate robots to areas that are not fully inventoried. It could allow using the finishing condition of having explored all the space. Or it could give item location capabilities to the entire system. Related to this, an interesting issue not tackled in this thesis is to check if these benefits can still be available in a decentralized system, or that, adding this type of features leads the system to a centralized one. In this sense, a further study is required.

Continuous inventory

In general, the RFID-based inventory robots are thought to perform an inventory or location mission, and then, go back to the docking station until the next mission starts.
However, if a fleet of RFID-based inventory robots is available, it could be possible to have a continuous inventory by having always several robots taking inventory of the space, while the rest are charging.
In this situation, the algorithm as is presented in this thesis would not work. Once all tags have been detected the attraction values would decrease fast and the robots would start moving randomly as the obtained values of attraction would be too small to be informative.
However, it could be possible to have a continuous inventory if a timeout or a weight that decreases with time is added to each EPC detection. This way, once an EPC has not been detected for a certain amount of time, it would be removed from the shared list or its weight would decrease to become irrelevant. Therefore, once it is again detected it counts as a new EPC allowing the system to continuously take inventory of the space.

Indoor location

One of the main reasons for having RFID robots is their capacity to locate the tagged items. In this sense, not having a map of the environment introduces a handicap to the system, as there is no fixed reference in which to place the located items. To overcome this issue two alternatives have been studied.
The first is to run a SLAM process in the background, so that, when the

robot locates an item with respect to itself, then it can place that item in the map, as it knows its reference in it. This possibility has already been discussed in the previous subsections.
The second alternative is to use Visual Light Communication (VLC) as an indoor Real Time Location System (RTLS). This technology modulates the LED lights that are illuminating a space, so, by means of a camera, such as the one in a smartphone, that captures this modulated light it is possible to determine the exact position of this camera with respect to the lights. This solution would require the integration of a camera on top of each robot, nevertheless, it has been observed that it is very precise and can be easily integrated with the system.
During this thesis, a testing package has been developed in order to transform the coordinates provided by the VLC system into a ROS message, so that it can be used by any robot.

4.5.4 Designed robot

The lessons learned from the initial design iteration, from the experimental tests of Chapter 5 and the support from Keonn staff's in the mechanical design led to the following design. Figure 4.19 shows the CAD design of the robot. From there, it can be observed that the height of the antennas has been considerably decreased and that the RFID reader, which in the previous design was inside the square created by the antennas, now is between the antennas and the Turtlebot base, further lowering the robot's center of gravity. In this design most of the robot components are in the same plane as all of them can feed directly from the Turtlebot's battery. It is relevant to observe the 360º degree view that the LIDAR has, and the location of the SONAR, which is in the frontal side and at a higher height than the LIDAR, in order to detect obstacles out of its plane. The bumper on the Turtlebot's base will be in charge of the obstacles with the lowest height.
Besides the Raspberry Pi 4 placed in one side, two DC/DC converters can be observed at each side of the Turtlebot's base, they are required to properly feed all the components from the Turtlebot's battery.

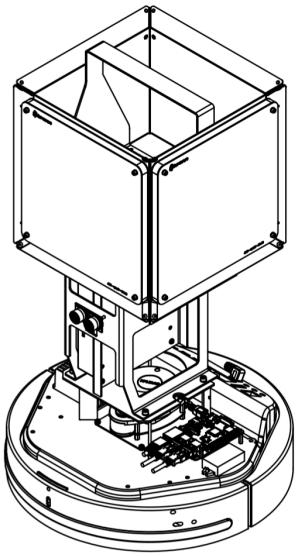
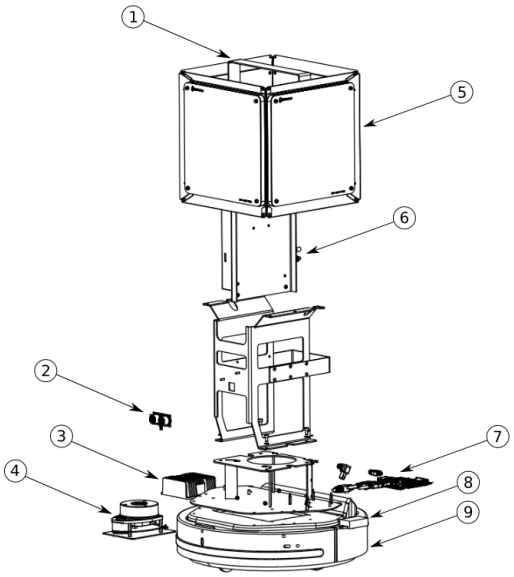


Figure 4.19: CAD of the final design of the robot.

In Figure 4.20 the most relevant parts of the robot are indicated, see Table 4.6 for reference.



1	Handle
2	SONAR - Parallax Ping)))
3	DC/DC converter 12V to 24V
4	YDLIDAR X4
5	Advantenna SP-11
6	AdvanReader-160
7	Raspberry Pi 4 b
8	DC/DC converter 12V to 5V
9	Turtlebot-2

Table 4.6: List of the main components of the final design

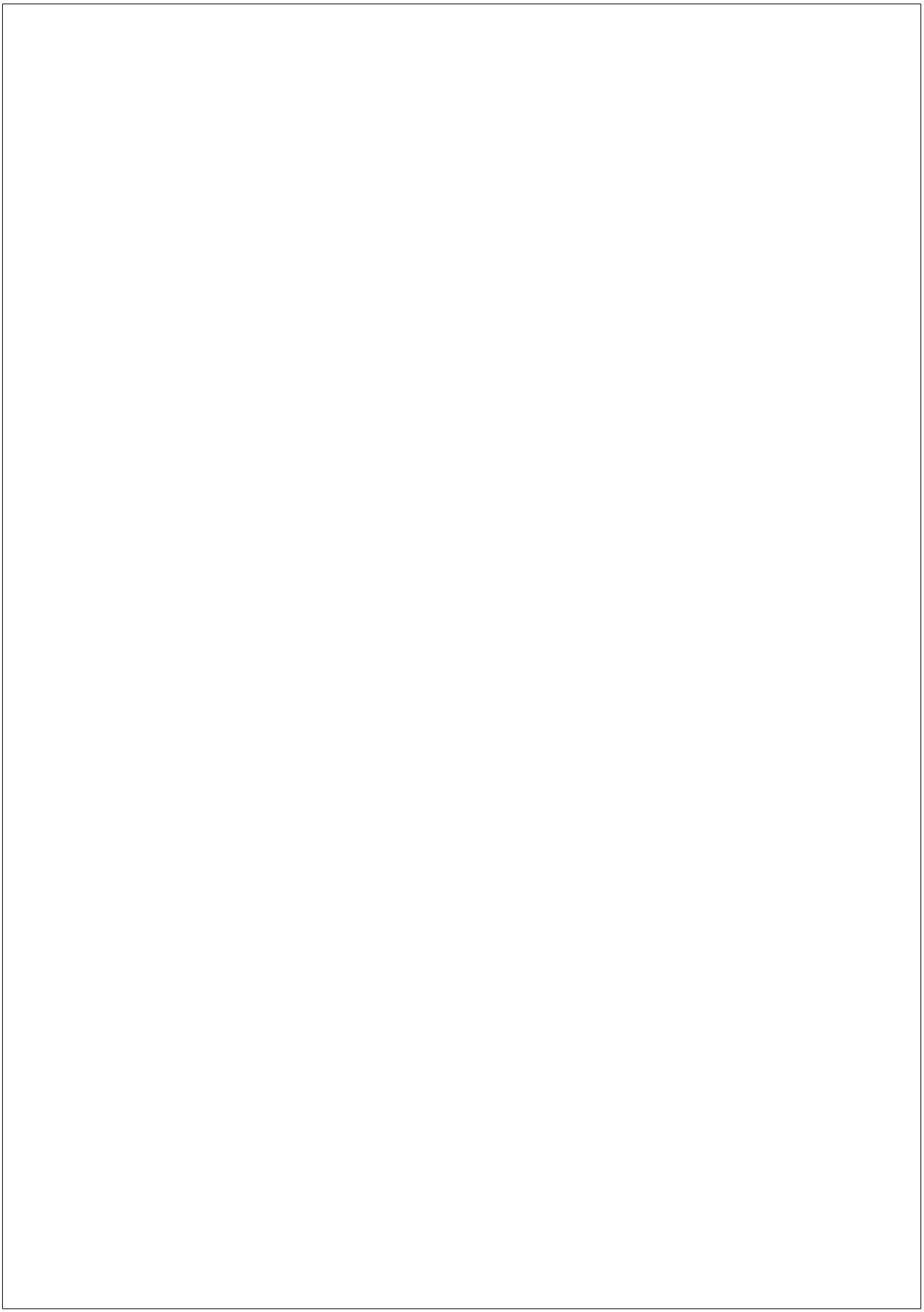
Figure 4.20: Final design parts.

In terms of modularity this design can easily separate the RFID payload from the rest of the components, this, apart from the clear advantages for maintenance, will also allow the use of different RFID payloads that could enhance the operation for specific cases.
This design has also pushed forward the cost-effectiveness of the robot, as it can be observed, the total cost is approximately 2650 €, 350 € less than the previous design and over 10 times cheaper than any RFID-based inventory robot on the market.
Figure 4.21 shows the robot built following the final design. It is now ready to be tested in a real environment to assess the new performance on RFID reading and to test the new functionalities extended with the new hardware and software available. However, due to the exceptional nature of the current situation with the Covid-19, the tests have been postponed and are not part of this thesis.

System	**Element**	**Cost (€)**
Payload	1 x Reader	1300
	4 x Antennas	400
Platform	Turtlebot-2 (large battery)	720
	YDLidar X4	100
	Parallax Ping)))	30
	Raspberry Pi 4	50
	Converters & wiring	50

Table 4.7: Designed robot components cost detail.

Figure 4.21: Final design of the robot.



Chapter 5

ALGORITHM & DESIGN VALIDATION

This chapter introduces the experimental validation of the map-less algorithm, presented in Chapter 3, and the validation of the prototype, presented in the first part of Chapter 4, in a real environment. In order to do so, the robot is taken to the library of the Campus Poblenou[1] (UPF).

5.1 Testing environment

The library is selected as the experimental environment for three reasons. The main one is that the library staff can provide us with their list of available books on the shelves. Therefore, an updated base-line of the expected books will be available. Second, they are flexible on the possible schedules to perform the tests. Finally, it is near the office where the robots are stored, which avoids transportation issues.
Additionally, the library is an interesting environment:

- The density of labels is higher than in an average store. This will stress and test the reading capabilities of the system.

[1] https://www.upf.edu/web/biblioteca-informatica/poblenou

- The aisles are wide and regular, therefore navigation issues should not arise. On the contrary, for RFID-based inventory robots that are based on map navigation, very regular environment can affect the robot navigation, as the robot might find itself lost due to the repetition of the same patterns in every aisle.

Nevertheless, the library presents the following challenges:

1. Only two aisles had books with tags, a total of 2,200 books are distributed among their shelves.
2. Reading tags inside books is difficult:
    - The books in the shelf are very packed and the density of RFID tags is high.
    - The bookshelves are metallic which create many interactions with the RF waves.
    - The paper contains water which absorb part of the RF power.
3. The environment is not totally controlled.

For each challenge an action will take place in order to perform the test with the best possible conditions, Table 5.1 shows a summary of the challenges and actions to mitigate them.

Challenge: Only 2 aisles with tagged books

This first issue is the most critical, a larger space is required in order to do the tests, a couple of aisles are not enough to test the algorithm. Therefore, the plan is to add more tags.
The first floor of the library contains a total of 35,000 books, which makes it not feasible to tag all of them. We cannot obtain enough RFID tags, they are cheap but the amount of tags is too large. Also, the procedure of tagging books requires a lot of time. It consists of the following steps:

1. Take all the books from a shelf without losing their order.

Challenge	Mitigation
Only 2 aisles have tagged books	Add tags to every bookshelf
Reading RFID labels inside books	Methodology evaluation and design ad hoc
Environment not totally controlled	Perform tests at the moments with less attendance

Table 5.1: Summary of challenges and mitigation actions for the tests in the library.

2. Read the first book bar code and codify it into an EPC
3. Stick the tag with the EPC codified inside the book, as seen in Figure 5.1
4. Repeat steps 2 and 3 for each book of the shelf.
5. Return the books to their shelf in their exact same order and take the next one.

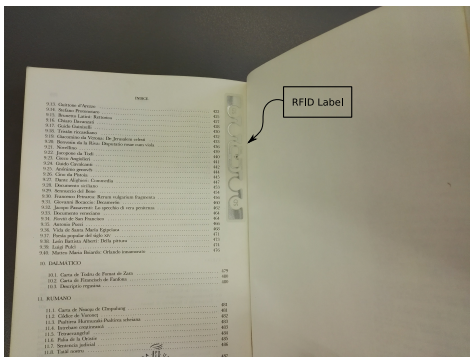


Figure 5.1: Example of a tag added to a library book

Therefore, we decided to tag the whole floor but only those books that are at 1m from the ground, which means only the third shelf of each book-

shelf. We managed to tag more than 4,100 books having tags on all books of the third shelf of the first floor of the library, which was only possible thanks to the collaboration of the library staff.
After adding these tags, we had an environment of 6,300 tags, 4,100 were distributed through all the aisles but at the same height and 2,200 tags were gathered in only 3 aisles but placed at all heights.

Challenge: Reading RFID tags inside books

Regarding the second issue, two actions are planned. The first consists on conducting a methodology analysis in order to better understand the RFID reading capacity of the developed robots in a difficult environment such as is the library. It will help understanding which are the books that can not be detected and it will let us adapt the methodology accordingly, see section 5.2.
The second action is in the design. The RFID payload is ad hoc placed at 1m height in order to be in the middle of a shelf, coinciding with the third shelf that has the majority of the RFID labels.

Challenge: Not fully controlled environment

The third issue can affect the tests in two ways. On one side, the library users can take books to their tables without notifying or they can misplace the books after checking them. This can affect the final inventory accuracy of the test. In fact, this is also considered in the following sections when the inventory accuracy computation has to be done. On the other side, people can block an aisle for many reasons, this can affect the navigation of the robot forcing it to take a direction that was not the selected by the algorithm. To deal with it, most of our tests are performed as early in the morning as possible, when the library is almost empty. Although, by the end of any test there were already many students around moving and taking books. Nevertheless, no tests had to be cancelled due to this,

at most, a few were stopped and resumed later.

Final considerations

Figure 5.2 shows the layout of the first floor of the library. Only the central part, of around 280m^2, is available for the robot to move freely. At each side of this central part tables are located for studying and on top or bottom of the floor there are closed rooms or offices. The aisles are 4.5m long and 1m wide, each aisle has 5 bookshelves on each side. In Figure 5.3, it can be seen the robot next to a bookshelf. The first shelf is situated at few centimeters from the ground, and then they are vertically separated 32cm.

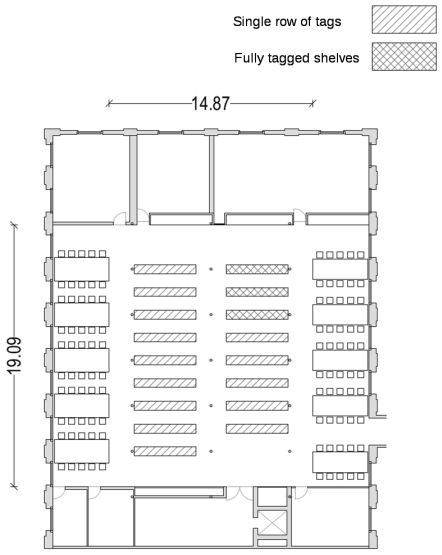


Figure 5.2: Layout of the first floor of the library. Values in meters.

Also, in Figure 5.2, it can be observed which are the bookshelves that contain tagged books in only the third row, and which ones have books

at all heights. In addition, the central area of the library has been divided in two sub-areas in order to constrict the space for some tests. Thus, the left column of aisles is block 1 and the one of the right is block 2, see Figure 5.5.

Figure 5.3: A prototype next to a library shelf.

5.2 Methodology

This section will define the methodology used to evaluate the performance, in terms of RFID detections, of the map-less algorithm for stock-counting.

5.2.1 Missing books test

Before defining a ground-truth of the books in the library to compute the accuracy, it is required to analyze the environment and its relation with the robot in terms of RFID readings. The test consists of performing inventory on the third shelf of some aisles, and then, investigate the reasons for not reaching a 100% of accuracy.
At this test, the accuracy is computed as follows: the RFID labels discovered by the robot are divided with the ones that are supposed to be there given the baseline provided by the librarians.
This test has been performed in two different scenarios, Figure 5.4 shows both of them. The first scenario, test A, is composed of three aisles and 5 bookshelves. The second scenario, test B, is composed of 4 aisles and 7 bookshelves. For both tests all labels that are not in the third shelf are filtered out of the test.

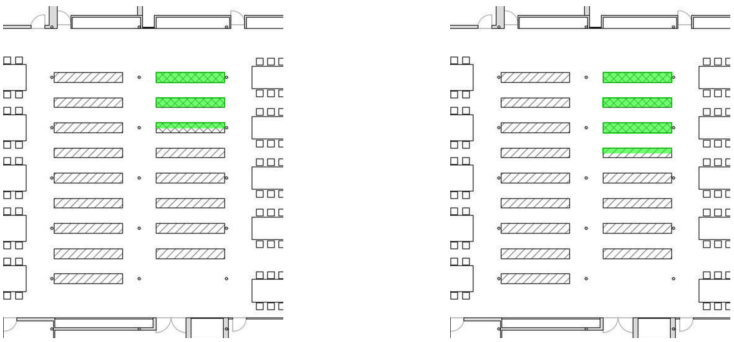


Figure 5.4: Aisles for preliminary test of missing books.

The main characteristics of these tests are shown in Table 5.2, in both cases the amount of available books is similar, but in the second test there have been 17 different repetitions, obtaining in some of them significantly better results, in terms of accuracy, than in the first test. However in any case never exceeding an accuracy of 97.92%.
The objective of this test is finding out why a 100% accuracy is never achieved. To do so, for each of the two tests, the books that are never detected in any repetitions are searched. Table 5.3 shows that 29 and 16

	Test A	Test B
Library aisles	3	4
Available books	665	770
Robots	2	2
Repetitions	4	17

Table 5.2: Characteristics of the preliminary test to determine the causes of the missing books

books are never detected for test A and B respectively, these books are then searched in their respective shelves.
The never detected books can be separated in three categories:
The first is composed by those books that are at the beginning or at the end of a shelf. This particular position means the book is in contact with a metallic plate. We know from other experiments that when these labels are very close to this metallic plate they are very difficult to read. However, the results obtained show that in some occasions it is possible to read them.
The second group are those books that are found on the shelf, but not in any particular position. In this category we have some books with labels that do not respond to the reader, because they are damaged. And some other labels that do work properly but for some reason the reader was not able to detect them.
The third category, and the largest for both tests with more than half of the books, is composed by those books that are not found in their shelves. They can be misplaced or lost, in any case, books that are unavailable, which the library ignores.

5.2.2 Accuracy specification

In general terms the inventory accuracy is the fraction of RFID labels discovered by the system divided by the total of RFID labels in the environment. Therefore, if a label is discovered by the robot but is not present in the count of the total labels of the environment, this label does not count

Books	**Test A**	**Test B**
Mean time [min]	11.7	14.5
Average accuracy [%]	92.8	96.2
Accuracy std. deviation [%]	2.9	1.9
Never detected	29 [4.4%]	16 [2.1%]
Touching metal	5 [0.7%]	2 [0.3%]
Not found	15 [2.3%]	10 [1.3%]
Found	9 [1.4%]	4 [0.5%]

Table 5.3: Results obtained in the preliminary test devoted to understand the causes of the missing books.

for the accuracy. Only labels that are expected to be found are used to compute the accuracy on inventory. This means that the robot can detect books that are in the library, but they are not expected to be there, in these cases they can not be counted.
The number of RFID labels in the environment is obtained with a list provided by the librarians, called $B_{\text{available}}$. However, due to the characteristics of the environment, there is a difference between the $B_{\text{available}}$ and the books that really are in the library. As it has been shown in the previous subsection, there is a significant amount of books that will not be detected due to 3 reasons: they are not actually there, they are blocked by metal or their RFID label does not work as expected.
There is no feasible way to determine which are going to be these books before doing the tests in order to remove them from $B_{\text{available}}$. From the preliminary tests we understand the reasons for some labels not to be detected, but we can not anticipate which are going to be missing.
Therefore, we will define four types of accuracy, based on the baseline used in order to provide the accuracy results:

- Raw accuracy
  It will be the accuracy computed directly according to the list provided by the librarians. The intuition behind this accuracy is that it

will under-estimate the performance of the robot.

$$\text{Acc}_{\text{raw}} = \frac{\text{Identified RFID tags}}{B_{\text{available}}} \tag{5.1}$$

- No metal accuracy
  This accuracy is computed by removing all the books that are in contact with metal from $B_{\text{available}}$, called $B_{\text{no metal}}$. It is possible to know which are these books because they are the first and last of every bookshelf, therefore, they can be removed from the list. This accuracy will be higher than the raw accuracy, however, as the percentage of books touching the metal not detected by the robot is low, the change on the performance should not be very relevant.

$$\text{Acc}_{\text{no metal}} = \frac{\text{Identified RFID tags}}{B_{\text{no metal}}} \tag{5.2}$$

- Found accuracy
  This accuracy is computed by removing from $B_{\text{available}}$ the books that are in the shelves but cannot be detected by the RFID reader, obtaining B_{found}. This accuracy would be higher than the raw accuracy, as books that for sure will not be detected are being removed from the ground-truth. However, it is not feasible to look for all these books in the library, it would require checking the books individually.

$$\text{Acc}_{\text{found}} = \frac{\text{Identified RFID tags}}{B_{\text{found}}} \tag{5.3}$$

- Trusted accuracy
  This accuracy is computed at the end of all repetitions of an experiment. In this case, the list of books that have been never detected is removed from $B_{\text{no metal}}$ obtaining B_{trusted}. It is worth mentioning that this accuracy requires several repetitions of an experiment. It makes no sense removing the books never detected if the repetitions are few as the bias on the ground-truth could be too high. This accuracy is an estimation of a not-found accuracy, however,

if the amount of repetitions is low the accuracy obtained will overestimate the performance of the algorithm.

$$\text{Acc}_{\text{trusted}} = \frac{\text{Identified RFID tags}}{B_{\text{trusted}}} \tag{5.4}$$

The results shown in section 5.3 will use the trusted baseline, B_{trusted}, in the case that the accuracy provided is not $\text{Acc}_{\text{trusted}}$ it will be notified.

5.2.3 End condition

The other figure that will be analyzed is the time the system takes to finish the inventory mission. As it has been previously explained, there is no way to assess whether the mission is completed. Therefore, two conditions have been established to finish the mission. The first is that the system has gone through all the aisles of the area to inventory, the second is that no new tags have been read in the last five minutes.
Both conditions are required because they are complementary. It may happen that the system revisits some aisles and therefore, there are no new tags read for the last five minutes, but there are several aisles the system has not yet visited. Additionally, the system could visit all the areas, but not thoroughly enough so that it is still detecting new labels.
Notice that the condition is based on the system, so the aisles require the visit of at least one robot, but all robots in the system have to be at least 5 minutes without new labels found. Meeting these two conditions can imply an overestimation of the time required by the system to finish the inventory, nevertheless, it is the approach that provides higher relevance to the accuracy obtained.
It is important to clarify that these conditions have been accomplished in the most accurate way possible. Nevertheless, the tests were performed in a non-controlled environment, and in some occasions it has been not possible to keep track of the robot's position while checking the amount of new detections or while making sure that no interference was produced with the regular users of the library.

5.2.4 Cooperation

Some tests are performed with a system formed by two robots. The cooperation between robots will be evaluated by analyzing the amount of new detections performed by each robot at every algorithm step. Ideally, it is expected that both detection rates follow a similar shape, in such a case a perfect coordination will have been achieved. On the contrary, if rates diverge with time it will indicate that one of the robots has performed most of the new identifications resulting in an unfair cooperation.
The plots that show this aspect have a third line that are the results obtained by the whole system, so the sum of each robot.

5.2.5 Starting point

At each repetition of each test, the initial pose of the robots will be changed to avoid similar patterns due to always detecting the same labels at the initial steps. Also, if the system is formed by more than one robot, the distance between them will be of at least one aisle. This way, interference issues in navigation and reading will be avoided at the very beginning of the test.

5.3 Tests

Three tests have been done at the library to assess the feasibility of the map-less algorithm. The first test takes only one robot and consists of stock-counting block 1 of the library floor. The second test takes two robots working together and consist of stock counting block 1. The third test uses two robots and consists of stock-counting the entire library floor. All these tests have only used the labels from the books that are in the third row.
Additionally, a fourth test has been done in order to assess the performance of the algorithm in an environment with an important amount of RFID labels that are not part of the ground-truth but that can not be filtered, therefore they can mislead the robots decisions. This test is done

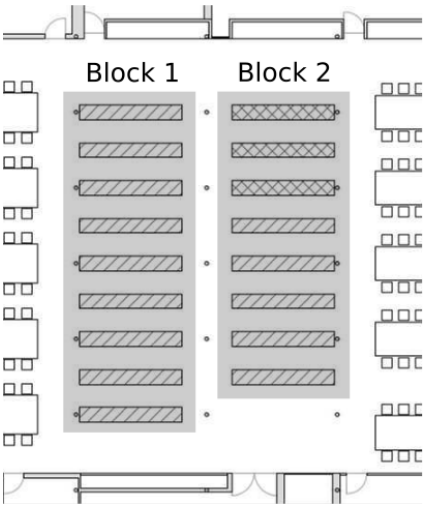


Figure 5.5: Library layout with the 2 separated blocks.

with 2 robots in block 2, where the labels on the third row are valid, but all other labels in different rows are not filtered.
In Figure 5.5 both blocks are drawn over the library layout for clarification.

5.3.1 One robot at block 1

For this first test the baseline given by the librarians consisted in 2,744 books. There are 14 repetitions, after them there were 15 books never detected. The books in contact with metal for this test are 75. Table 5.4 summarizes the characteristics of the test.
Figure 5.6 shows the results obtained on the first test. The results presented are based on the trusted baseline, which removes the books in contact with metal and the books never detected from the ground-truth provided by the librarians. In Figure 5.6 every dot is a repetition of the test. We can observe there is not a clear rule on how much time is required to inventory half the library. For instance, there are 5 repetitions above the 99%, but in terms of time they go from less than 65 minutes to more than

Test characteristics	
Repetitions	14
Raw baseline size	2,744
No metal baseline size	2,669
Trusted baseline size	2,654

Table 5.4: Characteristics of the test of a single robot taking inventory on block 1.

2 hours. Also, more time does not always imply better accuracy, one repetition lasted for almost two hours and it had the lowest accuracy of all.

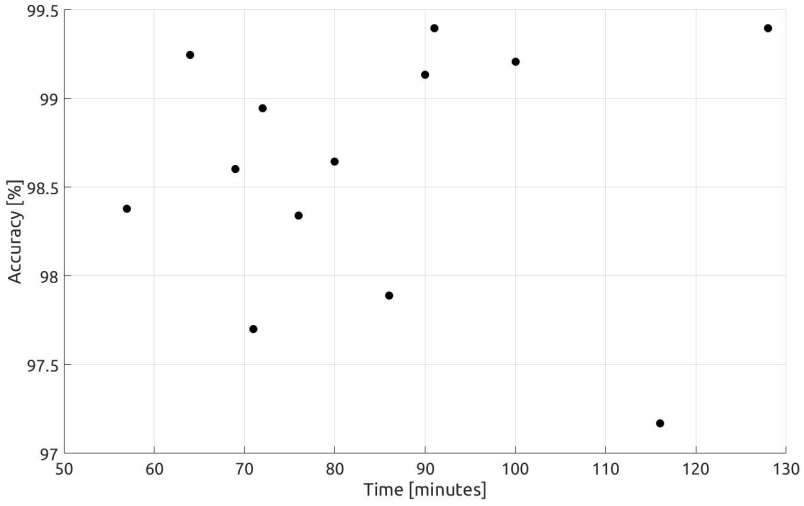


Figure 5.6: Results of the test with a single robot in the block 1 of the library. The accuracy showed in the figure is the trusted accuracy.

5.3.2 Two robots at block 1

The second test consisted of 2 robots taking inventory of block 1. The test is repeated 11 times, and there are also 11 books that are never detected. In total, the are 2,745 books with labels in this zone of the library, and 75 can be counted as touching the metal. Table 5.5 summarizes the test characteristics.

Test characteristics	
Repetitions	11
Raw baseline size	2,745
No metal baseline size	2,670
Trusted baseline size	2,659

Table 5.5: Characteristics of the test of two robots taking inventory on block 1.

Figure 5.7 shows the results obtained. It can be observed that all the repetitions lasted between 30 and 60 minutes, but most of them around 45 minutes. The variability on accuracy is similar than in the previous test, staying in a narrow 3% interval. As previously, the accuracy observed in Figure 5.7 is the trusted accuracy, so never detected books and books in contact with metal are removed from the baseline. Observing the results seems possible to affirm that 2 robots can finish the inventory mission with an accuracy of at least 97% in around 45 minutes.
The individual behavior of each robot of the system can be observed in Figure 5.8. In the first case of the two presented both robots behave in a very similar way, discovering almost the same amount of new tags per algorithm step. On the contrary, the second case shows that the 2 robots initially discover the same amount of labels, but then, robot 2 stays many algorithm steps without finding new labels until the last part of the repetition, when robot 1 stops finding new labels but robot 2 is able to complete the inventory mission.

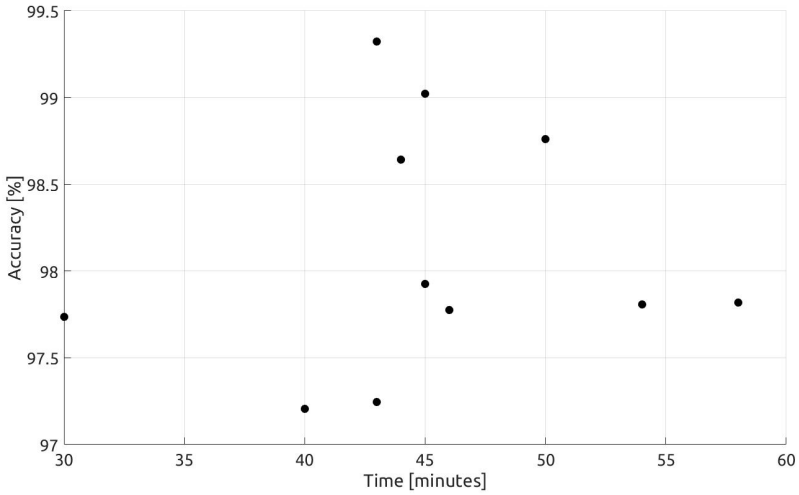


Figure 5.7: Results of the test with two robots in the block 1 of the library. The accuracy showed in the figure is the trusted accuracy.

5.3.3 Two robots at the entire library floor

The third test uses 2 robots and targets the entire floor of the library for stock-counting. In this case, the books out of row 3 that are present in block 2 are filtered out, so they will not influence the behavior of the robots.

There are 10 repetitions of this test. The baseline of labeled books is of 4,130 and 127 of them are considered to be touching the metal. After all the repetitions there are 30 books that have never been read. See Table 5.6 for a summary of the characteristics of the test.

The results obtained can be found at Figure 5.9, it can be observed that 9 out of 10 repetitions have a trusted accuracy between 98.5% and 99.5%. Regarding the time to complete the task, all repetitions take between 60 and 110 minutes. It is worth mentioning that if we consider only the repetitions that last between 75 and 110 minutes, and excluding the outlier, the accuracy is still more stable between 99% and 99.3%.

Figure 5.10 shows the individual behavior of the two robots for one of

Test characteristics	
Repetitions	10
Raw baseline size	4,130
No metal baseline size	4,003
Trusted baseline size	3,973

Table 5.6: Characteristics of the test of two robots taking inventory on entire floor of the library.

the repetitions. It can be observed that in this case both robots work in parallel detecting half of the new labels, they also show a very similar rate of discovery.

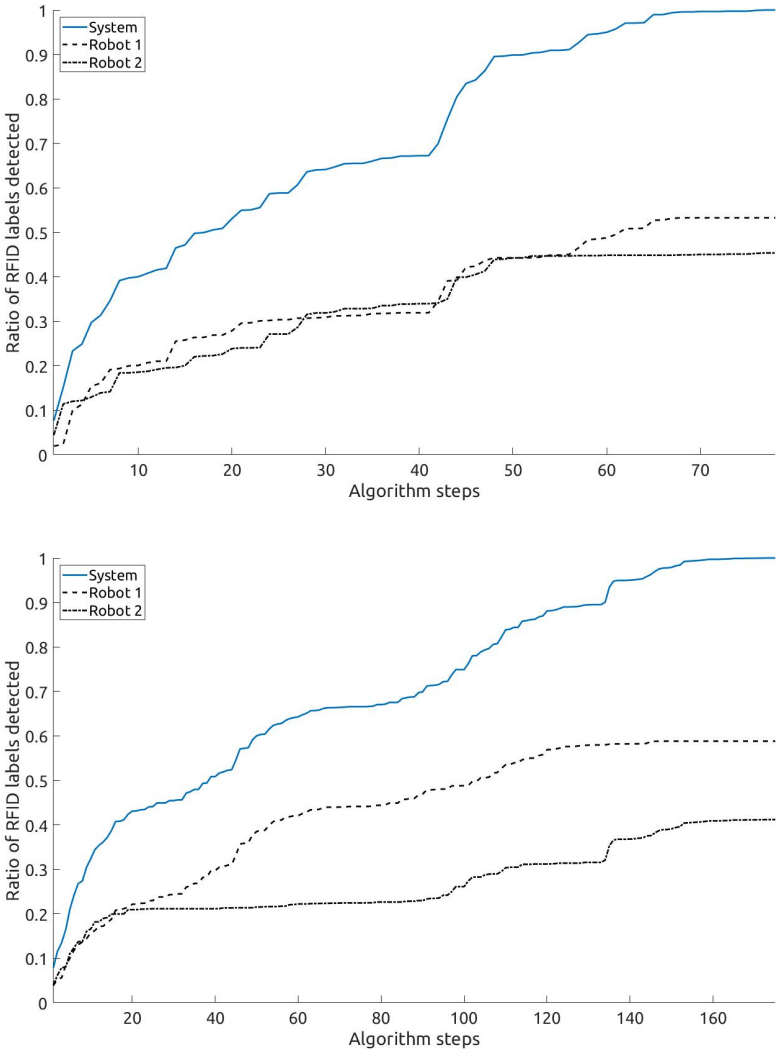


Figure 5.8: Two examples of the individual behavior of each robot during tests in the block 1 of the library. The ratio of RFID labels detected is obtained over the total of labels detected, so it does not measure an accuracy. The highest line is the sum of both lines.

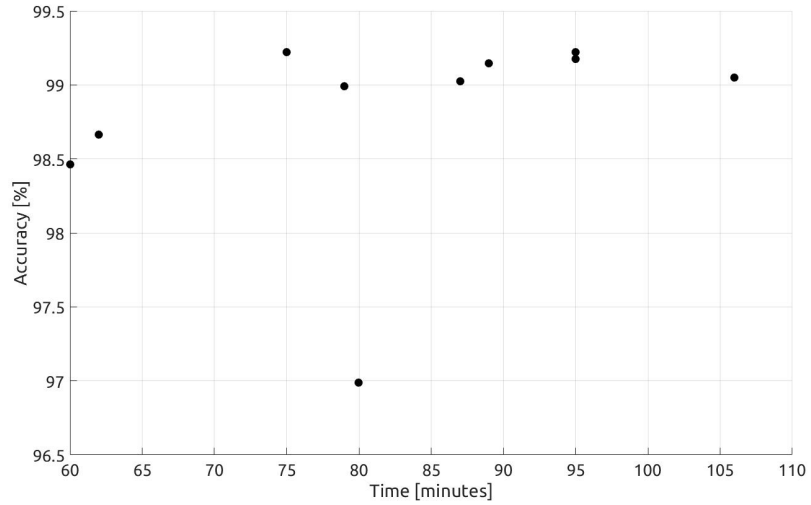


Figure 5.9: Results of the test with two robots on the whole floor of the library. The accuracy showed in the figure is the trusted accuracy.

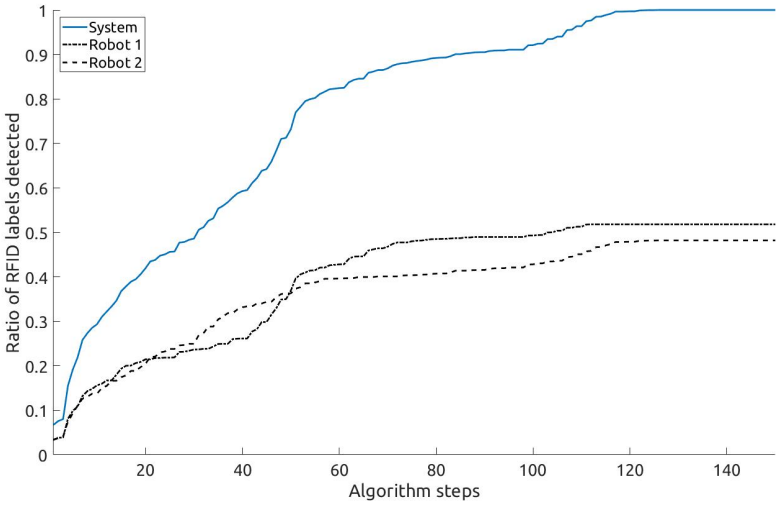


Figure 5.10: Example of the behavior of both robots during one of the repetition of the test on the whole floor of the library. The ratio of RFID labels detected is obtained over the total of labels detected, so it does not measure an accuracy. The System line is the sum of both robots.

Summary

In order to have a broader idea of the performance of the system, Figure 5.11 shows the three tests in a same graph. For each test, it plots the average value of time, and the average value of accuracy for all the repetitions of the tests. It also plots their respective standard deviations as error bars on top of the average values, even if it is not clear that the repetitions behave as normal distributions, the standard deviation provides a good intuition on the values obtained.
Figure 5.11 also shows the accuracies obtained taking into account three baselines defined in section 5.2. It can be seen that the difference between them is of around 0.5%, so the “no metal accuracy” is, in general, 0.5% higher than the “raw accuracy” and the “trusted accuracy” is 1% higher than the “raw accuracy”.
The results show two different groups, on one side there is the test with two robots and half the library (“block 1” at the legend of Figure 5.11). For this test the time spent is between 40 and 50 minutes and the accuracies are between 96% and 99%. On the other side, there are the tests with one robot at half library and with two robots at the whole library (“block 1” 1 robot and “Entire library” at the legend of Figure 5.11). In this case the time is doubled with respect to the previous group but the accuracies are slightly better from 97% and 99.5%.

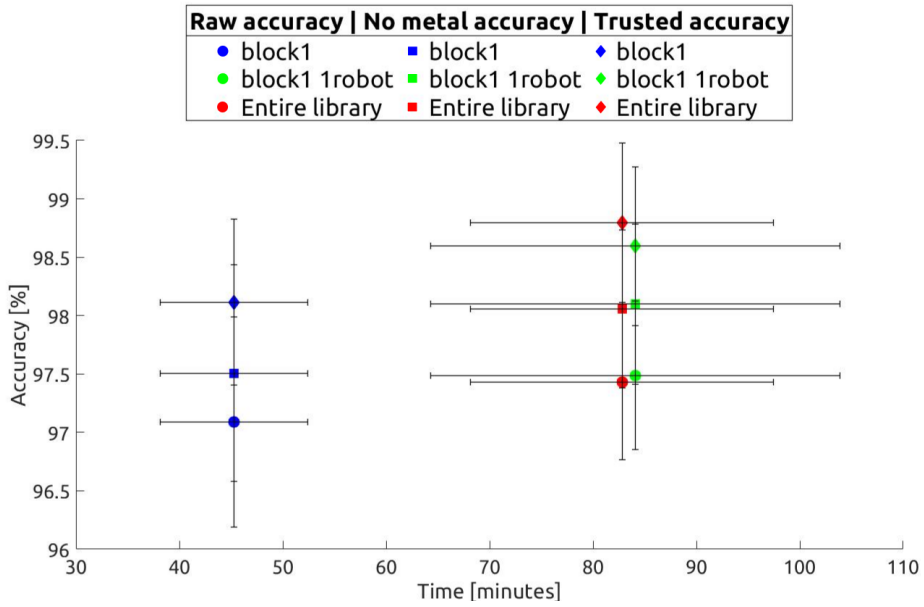


Figure 5.11: Average accuracy of the tests with respect to the time taken. The time required for each test is also averaged. Vertical lines show the standard deviation on the accuracy and horizontal lines the standard deviation on time.

5.3.4 Two robots at block 2 with misleading tags

This last test is with 2 robots and takes inventory of block 2. This area contains bookshelves that have books at all possible heights. During the tests there are no labels filtered inside the same block 2, therefore, labels from all heights can guide the robot. However, the accuracy is given only with respect to row 3. This test aims to assess the behavior of the algorithm in the presence of RFID tags that are not relevant to the inventory mission but cannot be filtered.
There are only 2 repetitions of this test, so it does not make much sense computing a trusted accuracy. With only two repetitions is not consistent enough to create a list of never detected books, in addition, they present very different results in time, but not in accuracy, see Figure 5.12. Therefore, the results are presented using the “no metal baseline”
This test has a baseline with 3,604 books, but only 1,396 are in row 3. Also there are 50 books in contact with metal. See Table 5.7 for a summary of the characteristics

Test characteristics	
Repetitions	2
Raw baseline size	3,604
Row 3 - Raw baseline size	1,396
Row 3 - No metal baseline size	1,346

Table 5.7: Characteristics of the test of two robots taking inventory on the block 2 with misleading tags.

Figure 5.12 shows the results obtained, it can be seen that the two repetitions have two different behaviors. In fact, the one that requires half the time has better accuracy values than the other. Nevertheless, take into account the differences in accuracy are small, less than 0.2%. The large time difference is given because in one of the tests the books from rows other than row 3 affected the decisions of the algorithm making the robots wander for too long on a few aisles.
In order to understand the individual behavior of the 2 robots of the sys-

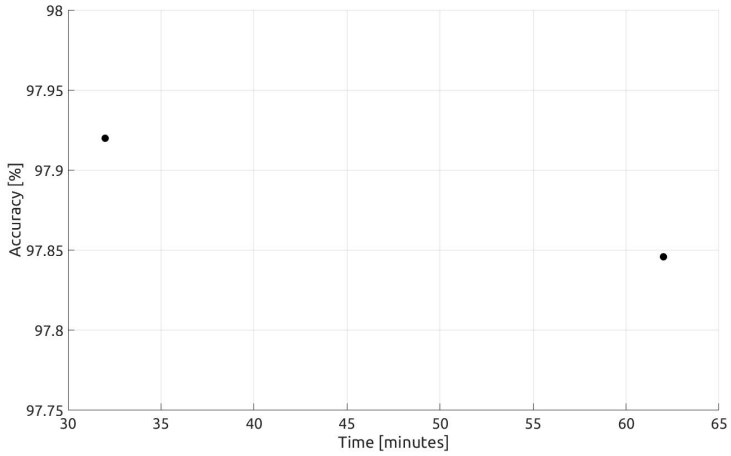


Figure 5.12: Results of the test with two robots in the block 2 of the library. The accuracy showed in the figure is the no metal accuracy.

tem it is relevant to look at Figure 5.13. First, notice that the ratio of RFID labels detected is computed over all the labels detected with both robots, so it includes books from all rows. Also, the figure only includes the labels detected for the first time, so if a robot is detecting the same labels many times, it just counts as one and only the first time. It can be seen that there is one robot, robot 1, that detects more than 60% of all the labels, which ends up being more than the double of labels than robot 2. This figure also explains the large time required to finish the inventory mission, as one of the robots clearly performs worse than the other lowering the performance of the whole system. This test shows that taking inventory of areas where tags can not be filtered can double the time required for inventorying the zone.

5.4 Conclusions

In general terms the results obtained are promising, the system is achieving inventory accuracy values as high as the ones expected for any cur-

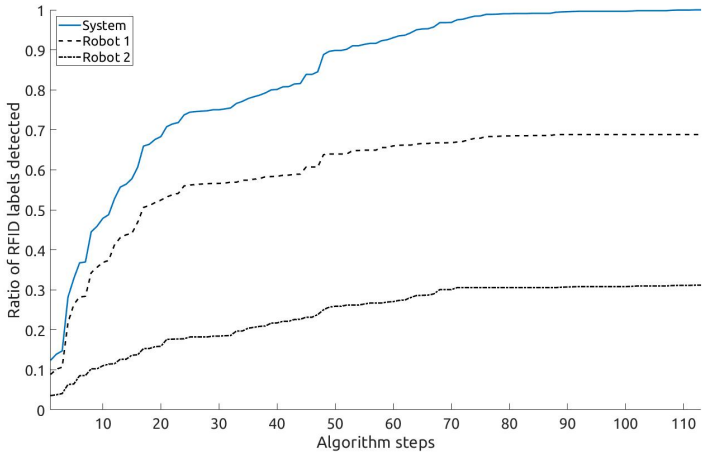


Figure 5.13: Example of the behavior of both robots during one of the tests in block 2 of the library. The ratio of RFID labels detected is obtained over the total of labels detected, so it does not measure an accuracy. The highest area is the sum of both areas.

rent RFID robot solution, between 97% and 99%. In addition, the time required for reaching these accuracy values is low enough to consider the proposed solution suitable for real life applications.

The library has been proven to be a challenging testing environment in terms of RFID readings, however, the results obtained prove that the prototype that has been developed for testing the algorithm has good initial conditions in order to be developed as a product.
The exact computation of inventory accuracies on large and not controlled environments is always a challenge. In this case, three different accuracies have been computed in order to provide the results. In this sense, it is possible to have a better understanding of how the system will behave in these types of environments.
Also, regarding the library environment, it is worth mentioning that the finishing conditions set for these tests were not always strictly followed.

Having track simultaneously of the number of new tags detected per robot and their amount of aisles visited while, for instance, trying to ask students to keep a proper distance to the robot was not always possible. Therefore, although the results are promising it is important to keep in mind that overall times and accuracies could be slightly different. In fact, the selected finishing condition have ensured high accuracy values, but it has increased the variability on the total amount of time required for the mission.

In terms of performance it is interesting to compare the results obtained at block 1, where the system has worked with 1 and 2 robots. In the case of using only one robot, it can be observed, see Figure 5.11, that the accuracy obtained is slightly higher, around 0.5% depending on the baseline used. However, in terms of time, having a system with 2 robots reduces in half the time required, it is also worth noticing that the variance on time is also reduced.

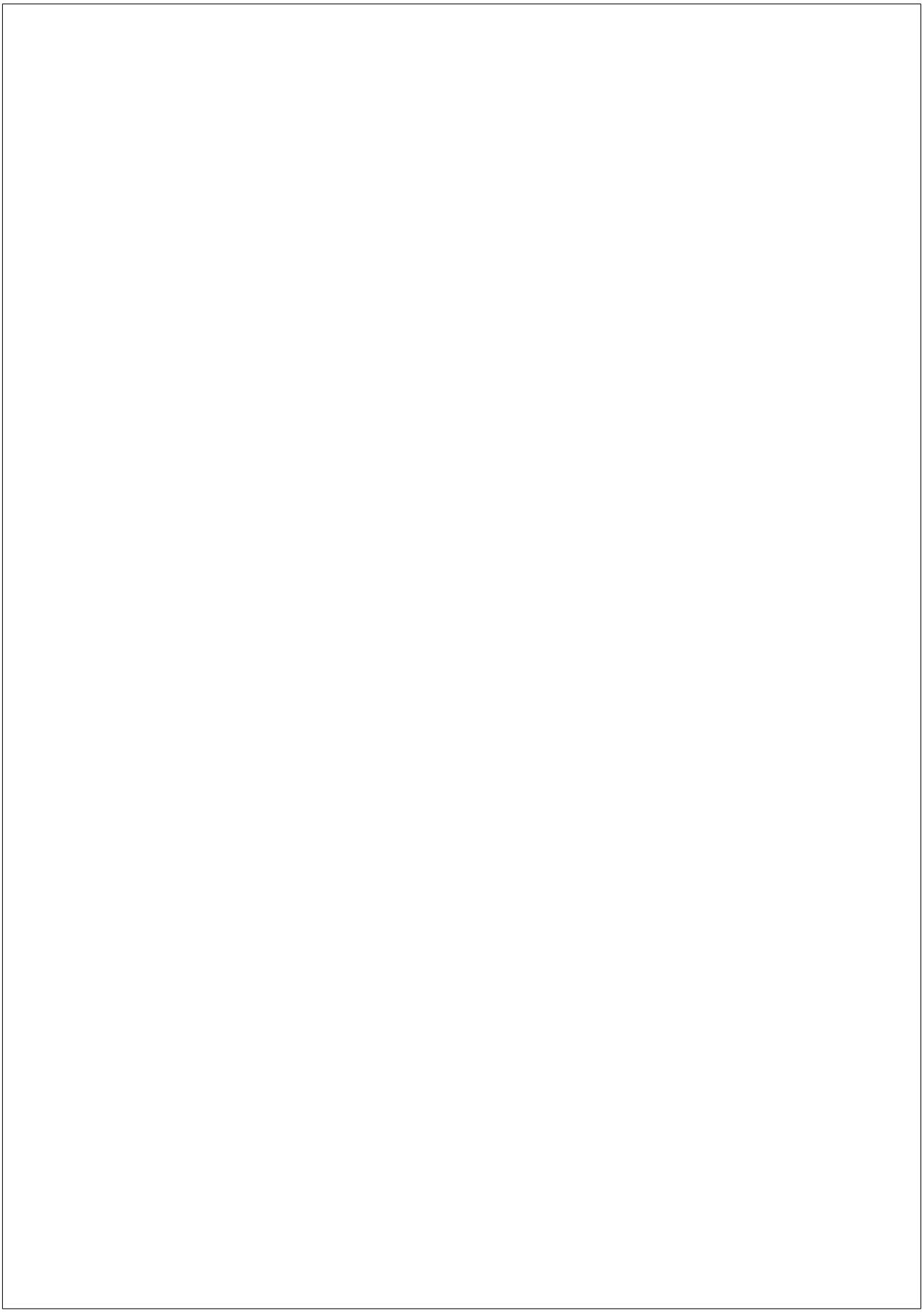
Another interesting comparison can be done if we contrast the system with two robots, but taking inventory of half library or the whole floor. In this case, it can be observed that the time is doubled if the space is doubled, but this also brings a small improvement in terms of inventory accuracy, 0.5% at the raw accuracy but almost 1% for the trusted accuracy.

The proposed algorithm makes the robots cooperate or, at least, parallelize the task they are performing. It has been seen during the tests that the level of parallelization is high when the robots perform as expected. So in these cases both robots discover around half of the tags. Also, it has been observed that in some cases if one of the robots got lost or could not find new RFID labels the other robot can finish the task, possibly spending more time but not risking the final value of accuracy. This is very remarkable because it greatly improves the robustness of the proposed solution in contrast to the solutions with a single robot.

The tests in the library have shown the suitability of the presented solu-

tion for inventory taking. In addition, it has been shown the improvement in the current state of the art of inventory robots, not only by being a fully autonomous solution, but also, by autonomously parallelizing the task, and by showing its robustness against failures. The system will finish the task while there is at least one operative robot.

The results have also provided some insight on the scalability of the solution, showing that it seems to be linear with the surface to inventory. Which means that if the space is doubled, by only doubling the number of robots, the time required to finish the task will be roughly the same. In addition, the system is prepared to handle as many agents as available. It is known that some interactions between the robots in terms of navigation and RFID readings happen, so if there are too many robots in a small place the performance of the system can get worse. However, this also means that can be an optimum of performance given by a determined number of robots in a fixed space. This makes the solution adaptable to any environment. Last but not least, the cost of the robots, as previously explained, has been reduced drastically with respect to the other inventory robots on the market. This fact allows the scalability feature of the solution to become a real asset to the system.



Chapter 6

RFID GROUPS LOCATION

Besides developing a map-less algorithm to enhance autonomy, robustness and scalability while reducing the cost of inventory robots, there are other ways to make inventory robots more useful to retailers. Location capabilities is a key enhancement as it is one of the features most wanted by retailers.

Knowing the location of tagged items can improve several aspects of a retailer's business. For instance, it improves the availability of products by finding those that are misplaced, it can help planning and picking orders, or it can provide insight about the places of the store where products are better sold.

This chapter presents a solution to locate groups of RFID labels in a real retail store by means of an inventory robot, which represents an improvement with respect to the current state of the art for two reasons: its efficiency and its convenience for the final user. The method developed also matches with the characteristics of the robot designed, which means that is kept as simple as the robot concept. Additionally, the method has low computational requirements to be compatible with the designed robot.

Although the method has been developed for the designed robot, specifically for the final design, which can have a map to reference items in it, due to organizational issues, the tests performed in this chapter are done

with an *AdvanRobot*[1] (see section 2.4.2 for more details on this robot.)

6.1 Problem description

Typically, this location problem is defined as the problem of locating an RFID tag in the environment through specifying its coordinates with respect to a map or a fixed coordinate system.
However, the problem tackled in this chapter is defined in a new way.
On one side, the problem consists on locating a group of RFID tags, instead of a single one. Usually this group of RFID tags will be a family of products, such as products that share the same SKU. In fact, when retailers need to know the location of a top-selling product, they do not care about the specific item sold, but the family of products that it belongs to. In this regard, it is more useful to group and locate the whole family of products.
On the other side, the reference used to locate the group of RFID tags will not be a set of coordinates but a zone of the environment such as a section in a store, or a landmark, such as a fixture. This change is also motivated by retailers, as they do not need the specific coordinates of a SKU in a map with an origin arbitrarily set by the initial pose of a robot. Instead, they need to know if the product is next to the main door, close to the fitting rooms or near the selling points. Additionally, they also want to know in which fixture the SKU is placed to obtain more precise knowledge about its layout performance, or to fulfill orders more efficiently.

6.1.1 Solution constraints and hypothesis

Before presenting the solution hypothesis for the location problem that it is being tackled, three constraints have to be taken into account in order to be as generic as possible:

[1] https://www.keonn.com/systems/view-all-2/inventory-robots.html

- There is no intervention on the store, no rearrangement of zones or fixtures is performed and no reference tags or other external devices are installed.

- The RFID technology used is COTS (Commercial Off-The-Shelf). In particular the RFID elements used are two *AdvanReader-150.04*[2] and 8 *Advantenna-SP11*[3], both found in the design developed in Chapter 4 and also in the AdvanRobot.

- Third, the amount of data to perform the computations can be large and the designed robot is not computationally powerful, therefore, the location method cannot be demanding on computational means.

To develop the simplified group location method two hypothesis are made:

- The best combination of RFID parameters, such as the RF power or the RF session, are not the same to take inventory than to locate items. The intuition behind this hypothesis is as follows: for the first case the objective is reading all the RFID tags at least once; for the second case is preferred to read the same tag several times and to avoid multi-path readings. Therefore, the RFID parameters (the RF power and the RF session) require different settings in order to optimize the location of RFID groups.

- A simple model for the RFID sensor may, on average, yield results as good, or better, as a very complex model. This hypothesis is taken because of the strong interaction of the RF waves with the environment. So, if the very complex model can properly work in a determined environment this will mean that the model has learnt not only about the RFID detection characteristics but also about the interaction with the environment, making this model not suitable for a different environment. On the contrary, a very simple model may not excel in any environment but can provide reasonably good results in all of them.

[2]https://www.keonn.com/rfid-components/readers/advanreader-150.html
[3]https://www.keonn.com/rfid-components/antennas/advantenna-sp11.html

6.1.2 Formal description

Notice that in this and the following sections, capital letters denote sets, capital letters with subindex denote subsets and lowercase letters are for elements of these sets.
The goal of the location problem is to classify each of the $|S|$ groups of RFID labels, G_p, into an area, a_n, from the set of areas of the environment, $A = \{a_n\}$, or into a fixture, f_m, from the set of fixtures in the layout, $F = \{f_m\}$, or both. The coordinates with respect to the robot's map defining each area are approximated by a polygon and denoted $CA_n = \{(x1, y1), (x2, y2), \dots\}$, and similarly, for each fixture are $CF_m = \{(x1, y1), (x2, y2), \dots\}$.
Each group of labels, G_p, is composed by several RFID detections, d_j. These will be initially stored in a database denoted $D = \{d_j\}$. The detections are the available information given by the RFID system to compute the position of each group of tags. Notice that d_j is a single tag detection, therefore, every time a tag is detected the detection, d_j, is stored in D.
A detection, d_j is mainly composed by the data retrieved from the robot at the moment of the detection. Specifically, $d_j = \{l_j, g_j, t_j, a_j, p_j, \hat{c}_j\}$ is composed by:

- l_j: The code identifying the RFID tag.
- g_j: The code identifying the group of products.
- t_j: The timestamp of the detection.
- a_j: The antenna port that produced the detection.
- p_j: The pose of the robot: its coordinates and orientation with respect to a map.
- $\hat{c}_j$: The estimation of the true RFID label coordinates, c_j.

The code identifying the group of products, g_j, is not directly obtained from the detection, but through a relation with l_j. This relation usually requires decoding the l_j to extract g_j or, in simpler cases, querying a table

that relates both identifiers. Therefore, once g_j is obtained from l_j, it is added to d_j.
The estimation of the RFID label coordinates, $\hat{c}_j$, is also computed after the detection and added to the data, d_j.
Notice that two different detections, d_j and d_k, of the same RFID label will only have $l_j = l_k$ and $g_j = g_k$ in common, but may have $\hat{c}_j \neq \hat{c}_k$, $t_j \neq t_k$, $a_j \neq a_k$ and $p_j \neq p_k$.
A group of labels, G_p, is composed by all the detections of the same group, so they share the same g_j. Therefore, $G_p = \{d_j \quad | \quad \forall_j \quad g_j = g_p\}$.
If the group G_p is composed by detections of a single RFID label, l_j, it can only have one position in the environment. But if a group is composed of several RFID labels, it can be located in several places in the environment. For clarification, one could have white T-shirts in a shop, each T-shirt has a different, l_j, but they all are part of the same group, G_p, as they share the same group code, g_p. It could be possible that half of the white T-shirts are at the entrance of the store, while the other half are next to the points of sale. In this case the group, G_p, would be in two different locations. Our experience shows that between 1 to 4 locations for a single group, G_p, in a shop is common.
Therefore, each of these locations, c_p^k, where the super-index k represents the cluster of detections that define this location, has to be related to an area or fixture of the store. In the case of the area it is simpler, as areas do not overlap and occupy the entire space. Therefore, the area that contains the coordinates c_p^k is the one that has to be selected.
On the contrary, fixtures do not occupy the entire space, therefore, the coordinates, c_p^k, of the cluster have to be the closest to the fixture to be assigned.
Thus, each result of the problem has to be composed by three elements: the group of tags, G_p; its assigned area, a_n and its assigned fixture, f_m.

6.1.3 Proposed solution

Before detailing the proposed algorithm, Algorithm 4, some definitions are required:
When a detection, d_j, is obtained, the function *estimate_location* computes the estimated coordinates of the RFID label detected. The model is as simple as possible, it estimates the location of the detection at a fixed distance with respect to the antenna that got the detection.
To do so, it uses the antenna port, a_j, and its location on the robot to determine a unit vector in the direction of the antenna with respect to the robot. Then it multiplies this vector by a constant factor, named *reading_distance*. This is a similar step that the one presented with the Equation 4.1, but in this case, to stay as simple as possible, instead of using the RSSI to determine the distance, a constant value is used. Finally, the coordinates are transformed from the robot frame to the map frame by means of the pose of the robot in the map, p_j, at the timestamp of the detection, t_j. It is worth mentioning that this computation may not be done at the moment of the detection, therefore, the pose of the robot needs to be queried by using the timestamp of the detection, t_j, which implies that time synchronization is required between the RFID system and the robot.
To define each cluster from the estimated coordinates of the detections a method called Density-based spatial clustering of applications with noise, *DBSCAN* [41], is used. This is a density based method that does not require assigning an initial number of clusters, and, given that the amount of locations that a group can have is unknown, this is a relevant feature. In addition, the *DBSCAN* is able to detect outliers, and therefore, an EPC detection that is far from any other reading can be discarded as an outlier. Two parameters drive the performance of DBSCAN in the implementation used, see [42]: *eps* which is the maximum distance to consider two samples of the same cluster and *min_samples* which is the minimum number of samples in a group to be considered a cluster. Therefore, if the number of RFID readings of a group is lower than *min_samples* or they are too scattered, the group will not be located
The *DBSCAN* implementation used returns all the coordinates given in

G_p but with a label k according to the cluster that they belong to, the list of the coordinates is named C_p. To further simplify the implementation, each cluster will be finally defined by the average value of the coordinates of all its samples.
Finally, two functions *assign_area* and *assign_fixture* are used to determine to which area or fixture each cluster belongs. In order to determine the relation between the area or the fixture and the cluster coordinates the *Python* library called *Shapely* [4] is used.
The *assign_area* simply iterates over all the existent areas until it finds the area that contains the coordinates of the cluster, then it just returns the id of the area, a_n. To do so, a function called *is_inside* is used, this function uses the *Shapely* library and returns true if the coordinates are inside the polygon.
The *assign_fixture* is similar, but if the coordinates are not inside a fixture, the distance between them is computed and stored in a list of distances, D_{G_p}. If at some point a fixture is found with the coordinates of the cluster inside, by means of *is_inside*, then it is returned. But, if all fixtures have been checked, then the selected fixture is the one with the shortest distance between the coordinates of the cluster and coordinates of the fixture.

Algorithm explanation

Algorithm 4 uses one procedure called "Compute group location" and three functions: *estimate_location*, *assign_area* and *assign_fixture*.
The procedure, "Compute group location", is applied at each group of labels, G_p. Initially the estimated coordinates for each detection are unknown, and are computed using the function *estimate_location* as previously explained. Once all the coordinates of a group are estimated, they are processed by *DBSCAN*, which provides the coordinates of each detection labeled with its corresponding cluster.
Therefore, the coordinates of each detection of the same cluster, C_p^k, are averaged to obtain the cluster coordinates, $\bar{c}_p^k$.

[4] https://pypi.org/project/Shapely/

Algorithm 4 RFID labels group location algorithm

procedure COMPUTE GROUP LOCATION(D)
  for G_p **in** D **do**
    for d_j **in** G_p **do**
      $\hat{c}_j$:=*estimate_location*(d_j)
      $d_j \leftarrow \hat{c}_j$ ▷ $\hat{c}_j$ is initially blank in d_j
    end for
    C_p, clusters := *DBSCAN*(G_p, eps, min_samples)
    for k **in** clusters **do**
      $\bar{c}_p^k$ = *average*(C_p^k)
      $G_p \leftarrow$ *assign_area*$(\bar{c}_p^k, A)$
      $G_p \leftarrow$ *assign_fixture*$(\bar{c}_p^k, F)$
    end for
  end for
end procedure

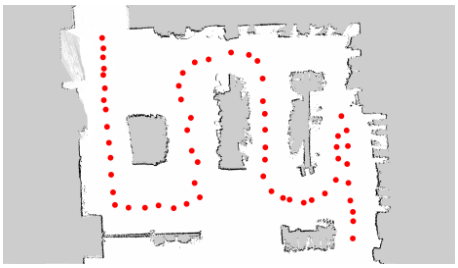
Then, it is possible to assign an area, a_n, and a fixture, f_m, for every cluster, k, of the group G_p, by means of the functions *assign_area* and *assign_fixture*.

Algorithm example

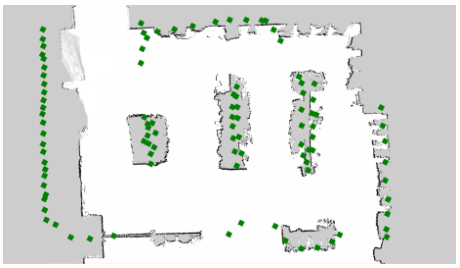
Figure 6.1 represents the algorithm steps to determine the position of a group. Initially, the red dots represent the robot poses where it has detected an RFID tag, see Figure 6.1a. Then, the location of each tag detection is estimated by transforming the pose of the robot with respect to the antenna that got the reading, see Figure 6.1b. Next, only the detections that belong to the same group are kept, see Figure 6.1c. Notice than steps two and three can change their order, depending on the way of storing the data, so the data can be separated by groups first, and then, estimate the EPC locations. The DBSCAN algorithm is then used to eliminate all the outliers and, in this case it also creates two clusters, see Figure 6.1d. Finally for each group the final location of the cluster is computed by averaging the positions of the members of the cluster, see Figure 6.1e.

Algorithm 5 RFID labels group location functions

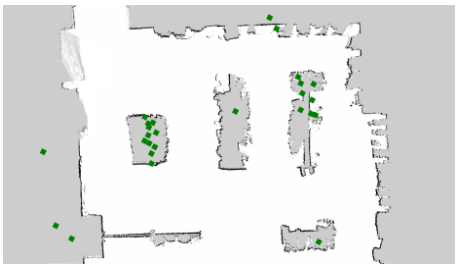
15: **function** *estimate_location*(d_j)
16: $\vec{a_p}$ = *antenna_vector*(robot_description, a_j)
17: $\hat{c}_j$ =*transform*($\vec{a_p} *$ reading_distance, p_j, t_j)
18: **return** $\hat{c}_j$
19: **end function**
20: **function** *assign_area*($\bar{c}_j$, A)
21: **for** a_n **in** A **do**
22: **if** *is_inside*($\bar{c}_j, CA_n$) **then**
23: **return** a_n
24: **end if**
25: **end for**
26: **end function**
27: **function** *assign_fixture*($\bar{c}_j$, F)
28: **for** f_m **in** F **do**
29: **if** *is_inside*($\bar{c}_j, CF_m$) **then**
30: **return** f_m
31: **else**
32: $D_{G_p} \leftarrow$ *distance*($\bar{c}_j, CF_m$)
33: **end if**
34: **end for**
35: **return** $f_m \mid$ *arg-min*$_{f_m \in F}(D_{G_g})$
36: **end function**



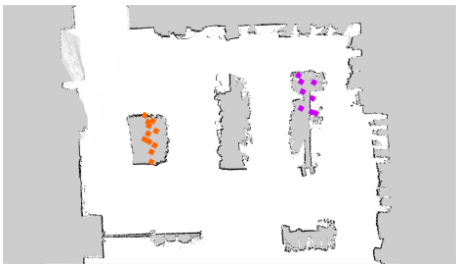
(a) Red dots are the location of the robot at every EPC detection.



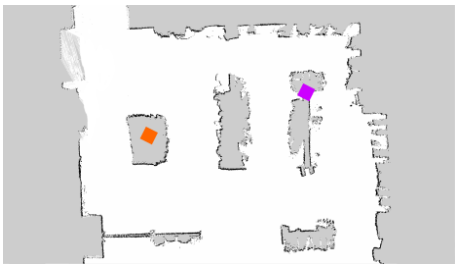
(b) Green dots represent the estimated location of each EPC detection.



(c) Only the EPCs belonging to the same group are processed.



(d) The DBSCAN algorithm clean outliers and separate 2 clusters.



(e) The position of each cluster is estimated as the average of all the detections location.

Figure 6.1: Description of the steps to estimate the location of a SKU.

6.2 Methodology

This section describes the methodology used in order to perform the tests explained in the following two sections. There are three aspects that require special attention: experimental procedures, parameter adjustment and evaluation.

6.2.1 Experimental procedures

In general, the location of items in a store is not known, or at least, there is no useful record on their location that can be used as a ground-truth. Therefore, it is required to manually record the coordinates of each group of products, its group code, g_k, their EPCs, the fixture they belong to and the area of the store they are placed. Also, the coordinates of each fixture and area of the store need to be recorded. All this data constitutes the ground-truth used to analyze the results of the algorithm developed.
Moreover, this recording has to be close in time with the data capturing of the robot. Ideally, the store should be closed and the operations of the staff stopped, if not, it is likely that the location of the recorded items changes with respect to the data captured by the robot.
Recording the location of each RFID label would provide more insight for the analysis, however, the amount of EPCs and the small difference in their location if they are part of the same group results into a prohibitive effort for the improvement that it can provide. Therefore in both experiments only the location of the group has been recorded, but not each individual RFID label.
Initially it is required to have a map of the store where the robot can locate itself. Therefore, previous to the ground-truth recording a map of the environment has to be created with the robot. Once the map is finished the location of the recorded groups, as well as the fixture or area coordinates can be referenced to it. It is important to remark that the map is for location purposes, but it might not be required for navigation.
When the map and the ground-truth are obtained, the next step is to move the robot around the environment to capture as many data points as possi-

ble from the RFID labels, therefore a complete dataset is obtained at each run of the robot.
Due to organizational issues, these experiments are performed with the AdvanRobot instead of the newly designed robot of section 4.5.4. Therefore, the robot is moved around the environment not following the RFID tags, but following navigation waypoints previously recorded to cover the whole area of interest. It is out of the scope of this thesis but kept as future work the analysis of the performance on the location given a robot that follows tags instead of the pre-recorded waypoints and that have antennas oriented towards four different directions, instead of two. In any case, the algorithm is expected to work for any robot and, as long as the robot detects enough RFID labels, the results should be similar.
Finally, when the dataset is completed the last step is analyzing the results obtained by the location algorithm.

6.2.2 Parameter adjustment

There are three sets of parameters to adjust for the developed algorithm:

- RFID system characteristics
- RFID detection model
- Clustering parameters

The first set is the RFID system characteristics, this includes the RFID session and the RFID transmitted power. The second set is the RFID detection model. As it has already been explained, it has been simplified as much as possible, therefore this set only includes the *reading_distance* as a parameter. It is reasonable to include this parameter in the first set, or to claim that it is directly given by the RFID power. However, due to the coupling with the environment this parameter is kept separated. The third set is related to the clustering method. The *DBSCAN* does not require the initial number of clusters but it requires the *eps* and the *min_samples* parameters.
One of the solution hypothesis states that the RF system parameters for an

inventory mission should be different than the ones for a location mission. In this regard, the main outcome of the experiments in the laboratory environment, see section 6.3, is adjusting the RFID system parameters to optimize a location mission as opposed to an inventory mission.
The *reading_distance* parameter is adjusted at each environment to capture some characteristics of the layout. The parameter for the laboratory environment, see section 6.3, will be fixed given the layout of the environment. For the real store environment, see section 6.4, the parameter will be adjusted given a captured dataset, the precise process is explained in section 6.4.
Finally, the *DBSCAN* parameters are only adjusted for real store experiments using one of the two datasets recorded.

6.2.3 Evaluation

The analysis of the results is based on two location accuracies, one with respect to the areas and the other with respect to the fixtures. So, if one creates a list $GA = \{G_p \mid a_{n_{\text{computed}}} = a_{n_{\text{ground-truth}}}\}$ where its elements have the same area computed than recorded in the ground truth, then the accuracy at area level is obtained with Equation 6.1.
The same procedure is done at fixture level, the list $GF = \{G_{gj} \mid f_{m_{\text{computed}}} = f_{m_{\text{ground-truth}}}\}$ is created and then the accuracy at fixture level is computed following Equation 6.2.

$$\text{accuracy}_{\text{area}} = \frac{|GA|}{|G|} \tag{6.1}$$

$$\text{accuracy}_{\text{fixture}} = \frac{|GF|}{|G|} \tag{6.2}$$

Where G is the set of recorded groups in the ground-truth that have enough RFID detections to be processed by the clustering process. Therefore, if the RFID tags from a group have not enough readings to compute a location from them, the group is not identified. Similarly, it is done at fixture level.

Again, the main challenge is that the ground-truth is not usually complete, therefore, these accuracies can only be computed over the recorded data. This adds another difficulty, if the ground-truth presents a single location for a group, but the clustering process decides that given the RFID detections there has to be two clusters and one of them coincides, then, the result taken is that the group is well located.
Additionally, to support the results the mean distance error, $\bar{d}$, is also computed. This is the average of the distances between the recorded location of a group and the computed location of the same group, see Equation 6.3.

$$\bar{d} = \frac{\sum_G \sqrt{(c^k - \bar{c}^k)^2}}{|G|} \tag{6.3}$$

The main objective of the development here presented is to locate the groups of products with respect to areas or fixtures, not to compare them at a coordinates level. However, this figure can provide precise insight about the method if the recorded coordinates of the groups are available.

6.3 Test in a laboratory environment

This experiment is performed to obtain the best RFID parameters to locate items in the environment with the algorithm defined.

6.3.1 Environment & Characteristics

The environment is similar to the one presented in Figure 4.12, but the RFID labels are on top of the boxes and there are four lines of boxes with a total of 49 boxes. Each line of boxes creates an aisle of around 1 meter wide, each box contains between 10 and 50 RFID tags, and the center of the box is considered the location of the group. Figure 6.2 shows the map recorded by the robot of the environment, the four lines of boxes can be seen in the middle of the map.
The following characteristics are precise of this experiment:

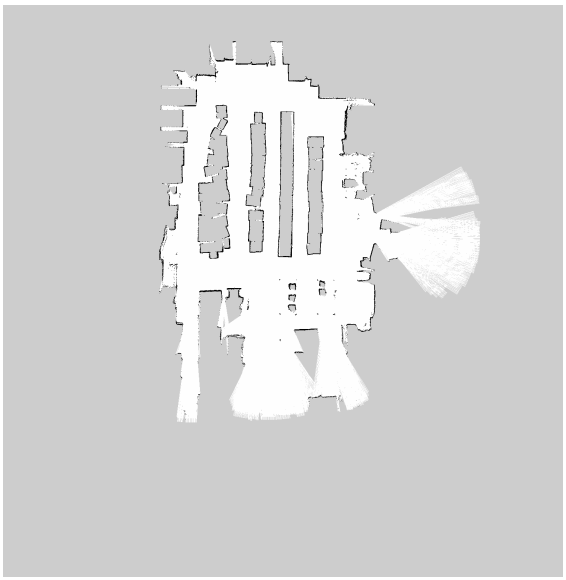


Figure 6.2: Map of the laboratory environment

- Each box is considered a fixture and due to the reduced dimensions of the environment there are not areas defined.
- The ground-truth is recorded just once at the beginning of the tests.
- The captured ground-truth is complete.
- The robot is commanded to perform 20 location missions to gather the data varying the RFID system characteristics: the power and the session.
- The parameter *reading_distance* for the RFID system is set to 0.5 meters with respect to the antenna, given that the aisles are 1 meter wide.
- The clustering process is skipped as each group of tags has only one possible location.

6.3.2 Results

The results obtained after the experiment are shown in Table 6.1. The Location accuracy column is the percentage of groups that are properly assigned to its box, $\text{accuracy}_{\text{fixture}}$, the distance mean error, $\bar{d}$, is computed as the average of the distances between the group computed location, $\bar{c}^k$, and its recorded location, c^k. The inventory accuracy is the number of labels discovered over the total of labels of the environment. The first two columns are the values given for the RFID parameters, notice that the initial values are those that excel at inventorying.

Test	**RFID power**	**RFID session**	**Location accuracy [%]**	**Distance mean error [m]**	**Inventory accuracy [%]**
1	30	S2	2.04	2.54	99.62
2	30	S2	2.04	2.48	99.62
3	25	S2	28.57	0.92	97.56
4	25	S2	16.32	0.90	97.56
5	25	S2	28.57	0.90	97.43
6	25	S2	18.36	0.91	97.43
7	25	S0	71.43	0.44	97.18
8	25	S0	71.43	0.46	97.18
9	20	S2	63.25	0.51	81.97
10	20	S2	55.10	0.51	79.86
11	20	S2	63.26	0.44	86.90
12	20	S2	55.10	0.47	86.90
13	**20**	**S0**	**81.63**	**0.36**	**84.96**
14	20	S0	79.59	0.37	84.96
15	15	S2	71.43	0.43	39.72
16	15	S2	67.35	0.44	39.89
17	15	S2	67.35	0.47	40.73
18	15	S2	71.43	0.45	41.70
19	15	S0	75.51	0.42	40.10
20	15	S0	77.55	0.43	40.10

Table 6.1: Results for the first set of tests

From the results it is clear that the best parameters for inventorying are

not those for location. The maximum location accuracy is obtained with an RFID power of 20dbm and the RFID session S0, see section 2.1 for a detailed explanation of the RFID sessions. It is worth mentioning that the decrease of the power helps the location accuracy, as most detections are done from the surroundings. However, if the power is too low, the number of labels detected decreases drastically affecting not only the inventory accuracy, but also to the location accuracy. Regarding the session, S0 allows maximizing the amount of detections, this clearly improves the location accuracy. As it can be observed in Table 6.1 for the same value of RFID power the location accuracy clearly improves with S0.
The distance mean errors behaves similarly than the location accuracy, it has the minimum with the same parameters than the location accuracy has its maximum. It is remarkable that this error is of only 0.36 meters.
Figure 6.3 shows a particular example of the results obtained. It can be observed on top of the recorded map of the robot all the RFID labels detections, which are the red dots, the estimated position in blue and the recorded position, which is the large green dot.
To sum up, the selected parameters for any location mission are the ones bolded in Table 6.1: RFID power of 20 dbm and RFID session S0.

6.4 Test in a real store

This experiment consisted on recording a sample of the ground-truth of a real store twice and for each of them perform a location mission. The two datasets are taken with only a week difference but they are totally independent. Additionally, the ground-truth is taken simultaneously to the robot operation, ensuring that the difference between the recorded and the captured data is minimal. However, the store has some minimal operation at the moment of the recording which results in some movements of the products.
The first dataset captured is mainly used to adjust the RFID detection model and the clustering parameters, the second to check the feasibility and accuracy of the method.

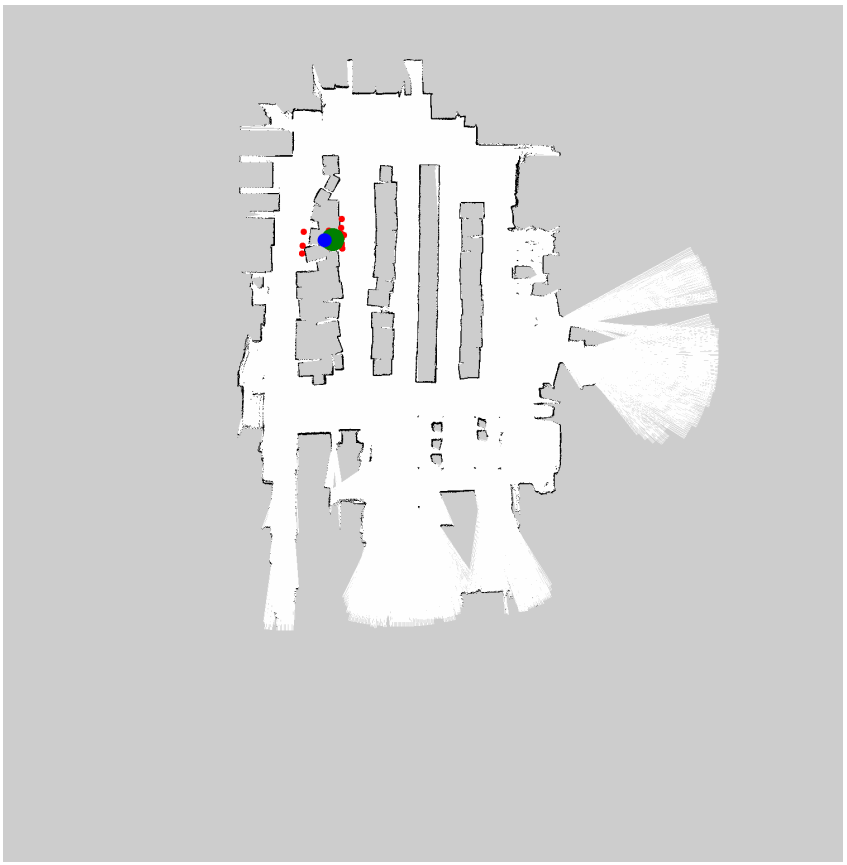


Figure 6.3: Example of the results obtained in the laboratory environment test. The red dots are each of the readings of the EPCs belonging to the same group. The blue dot is the estimated position of the group and the green dot is its recorded position. The results correspond to the 46th group of the 13th test on the first experiment.

6.4.1 Environment & Characteristics

The test store has a ground floor with 1,000 m^2 and around 10,000 different EPCs. The store associates have divided the ground floor in 11 different areas as a function of the type of products and the characteristics

of the area. Also they have identified 88 fixtures in which the products were placed.
In the store floor the products, identified with EPCs, are grouped in a range from 2 to 25 EPCs each group. Most of the EPCs belonging to the same group should be together. However, as previously stated, a group can be placed in several places of the store at the same time.
Figure 6.4 shows the 11 areas on top of the map of the store captured by the robot during the first day of experiments. The diversity of areas is clear from Table 6.2, the largest area has almost 200 m^2 and the smaller less than 20 m^2. Also it can be observed that the largest perimeter is not linked with the largest area, which means that the areas in general have irregular shapes. This diversity of shapes and sizes makes inefficient the use of geometrical characteristics of the areas in order to enhance the performance of locating items, such as by assigning a suitable *reading_distance*.
Figure 6.5 shows all the fixtures included in this experiment. The fixtures do not include all the space of the store, therefore, a group is assigned to a fixture if it inside or if it is closest.

Areas	**Perimeter** [m]	**Surface** [m^2]
a1	26.4	45.3
a2	20.5	17.2
a3	45.3	105.3
a4	35.9	74.5
a5	53.9	195.5
a6	55.3	158.7
a7	62.5	109.9
a8	46.5	115.6
a9	35.3	45.5
a10	42.6	94.1
a11	33.2	47.9

Table 6.2: Areas description

For both datasets the map recorded by the robot as well as the coordinates of all areas and fixtures are the same. Notice that the coordinates of the

Figure 6.4: Store layout divided by areas.

areas and fixtures are recorded manually.
The two datasets obtained are named, in Table 6.3, A and B respectively. Recording dataset A lasted for 1 hour and 40 minutes, which was the time that the robot needed to perform the location mission, dataset B lasted 40 minutes more. Both missions were very similar, but by the end the second one, the store opened and due to the increase of people in the store the robot required more time to finish. This provided more time to record the ground-truth, while the movement of products was not yet too high, which means that the dataset is considered valid.
Dataset B is more complete than dataset A as it has 115 more groups, and the robot captured around 250 more unique EPCs. In any case, both datasets are considered representative sample of the store. It is also worth

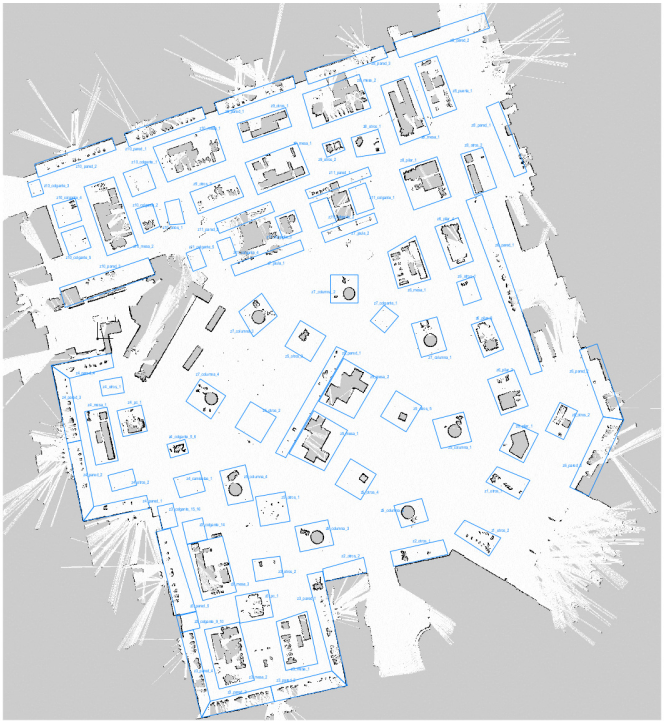


Figure 6.5: Store layout divided by fixtures.

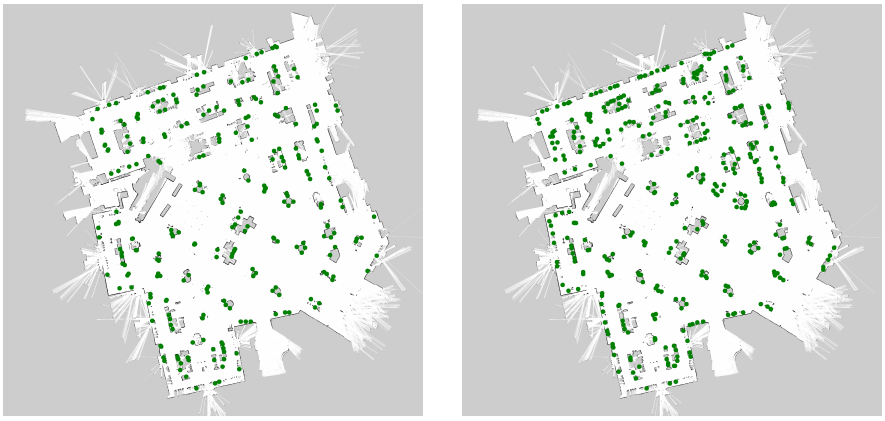
noticing that there is no ground-truth on the total amount of EPCs present at the store, therefore, it will not be possible to assess an inventory accuracy.
Dataset A will be mostly used to adjust the parameters, while dataset B will assess the results without any modification in the parameters with respect dataset A. Figure 6.6 shows the locations of the recorded groups on top of the map captured by the robot.

6.4.2 Parameter adjustment

Three parameters need adjustment before computing the final results, these parameters are: the maximum distance between RFID detections to be

Dataset	**Duration**	**Groups recorded**	**Unique EPCs read**
A	1h 40min	264	8546
B	2h 20min	379	8807

Table 6.3: Summary of the two tests performed



(a) Location of recorded groups from dataset A

(b) Location of recorded groups from dataset B

Figure 6.6: Recorded group locations

considered part of the same cluster, named *eps*; the number of RFID detections to consider a group, named *min_samples*; and the distance where the RFID detections are placed with respect to the robot when they are detected, named *reading_distance*. The *eps* and the *min_samples* are part of the *DBSCAN* clustering method, while the *reading_distance* is part of the RFID detection model.
As already has been explained, only the dataset A will be used to adjust these parameters, therefore, dataset B is kept untouched for the analysis of the results.

DBSCAN parameters adjustment

This adjustment is required because the parameters to drive the clustering of samples is linked with the store layout and the sparsity of the products on the store. This adjustment could be understood as the price to pay to avoid selecting a number of clusters beforehand. To perform this adjustment, the parameter *reading_distance* has been fixed to 1m.
In order to decide the best pair of values for the parameters, three figures of merit are analyzed, these are: the area location accuracy, the average distance error and the groups lost.
The *area location accuracy* is the percentage of groups that are correctly located in the expected area. In this case the increase of the clustering parameters decreases the accuracy because it can only be computed with the data retrieved, which means that the more clusters the higher probability of at least matching one of them properly. Take into account that increasing the clustering parameters reduces the number of clusters generated.
The *average distance error* is the distance between the recorded position of the group of EPCs and the computed location of the same group. This figure has a minimum because if there are too many estimated groups the average over all the samples increases this figure. On the contrary, if there are not enough groups the location of the group will be deviated from the true position and the figure will also increase.
Finally, the *groups lost* is the number of groups that have not enough samples to be a cluster and therefore, are not taken into account. This value is relative to the maximum number of groups considered.
Figure 6.7 shows the evolution of the previously stated figures of merit with the variation of the optimization parameters. The shared x axis is composed by two numbers, the first is the *eps* and the second the *min_samples*.
There are only two possible combination of parameters, 3,2 and 3,3, that have the average distance error is in its minimum while the number of groups of EPCs lost is zero. With both combinations the area location accuracy is the same, however, a $min\_samples = 3$ seems a better approximation as having only 2 readings of a group is low and it could easily

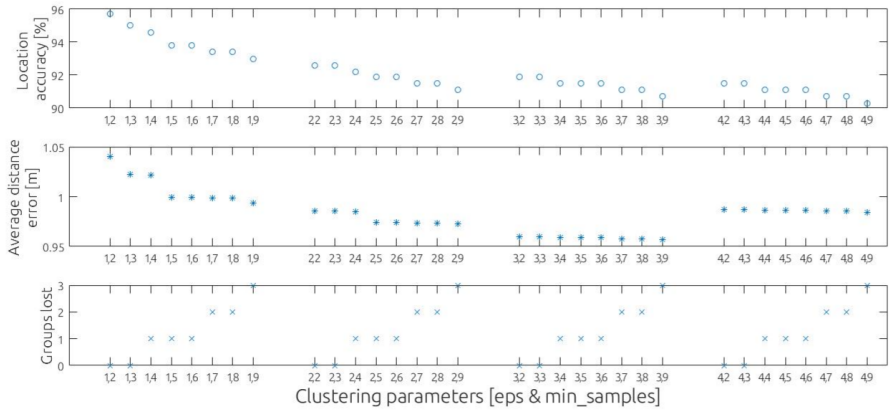


Figure 6.7: Analysis on the performance as a function of the parameters *eps* and *min_samples*. Notice that the parameters are on the x axis showing the *eps* next to the *min_samples* value.

create more groups out of a single EPC, which should be less common. Therefore, the parameters used are $eps = 3m$ and $min\_samples = 3$.

Reading distance adjustment

The second parameter adjustment process, again with only data from dataset A, is in regards of the RFID reading distance. It depends on the RF power and the environment, however, due to the physics of the RFID technology a fixed value is used. In [4] a study on the behavior of the RSSI shows that it can not be assumed that it is Gaussian, and actually, there is no single probability distribution that can describe the performance of the RSSI. Therefore, for this case, an approximation as the one used in Equation 4.1 is not used, only a fixed value to keep the model simple and the computational resources required as low as possible.
Therefore, once the clustering parameters have been fixed, the adjusting of the *reading_distance* is performed by analyzing the area accuracy results with respect to 5 different reading distances. Figure 6.8 shows the accuracy values obtained with *reading_distance* of: 1.2, 1.3, 1.4, 1.5 and

1.6 meters. Notice that the area location accuracy is computed over 1, not as a percentage.

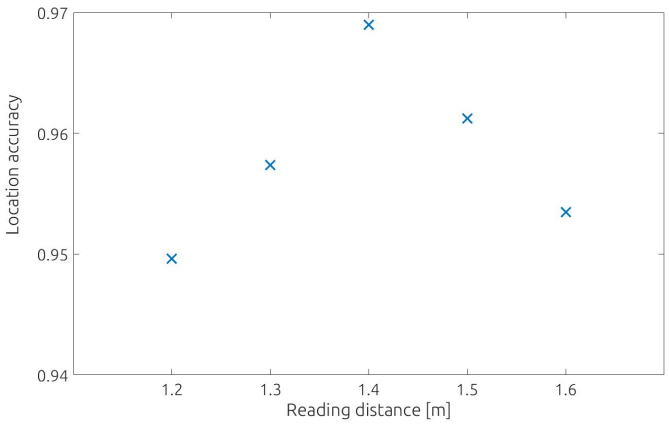


Figure 6.8: Location accuracy obtained from the dataset A as a function of the RFID reading distance. The location accuracy is computed over 1, not as a percentage.

In Figure 6.8 it can be observed that there is a maximum of accuracy, therefore, the best performance is at a fixed reading distance of 1.4 m.

6.4.3 Results

The results obtained for the datasets A and B are presented. For both datasets the accuracy with respect to the areas and with respect to the fixtures has been computed. As it has been already been discussed results with respect to dataset B have more credit, as the dataset A has been used to adjust the algorithm parameters. Table 6.4 summarizes the results for both datasets.
Where total groups are the amount of groups recorded at each dataset and identified groups is the amount of groups with enough data to be located inside an area or fixture. The area accuracy is the percentage of the identified groups that match the estimated area with its real area. Similarly,

Dataset	**Total groups**	**Identified groups**	**Area accuracy [%]**	**Fixture accuracy [%]**	**Mean distance error [m]**
A	264	257	96.9	94.6	0.67
B	379	366	95.9	92.1	0.73

Table 6.4: Summary of results on group location in a real environment.

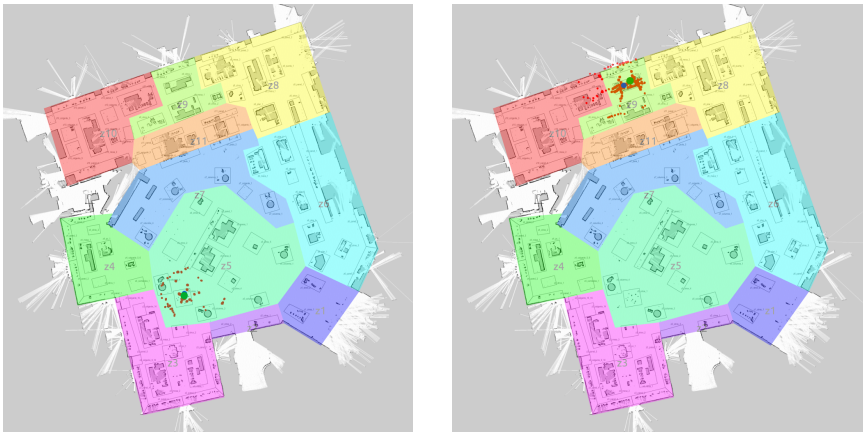
the fixture accuracy is the percentage of identified groups that match their captured fixture with the estimated one. Finally, the mean distance error is the average through all the identified groups of the distance between the recorded coordinates of the group and their estimated coordinates. In the case that the group is found in more than one place, the distance used is the shortest.

It can be observed in Table 6.4 that less than 3.5% of the groups recorded in test B have not been identified due to not having enough samples or having them too scattered. Decreasing the RFID power to obtain better location accuracy could lead to low accuracy on RFID detections and, therefore, low accuracy on location. However, it has been observed that the loss is minimal given the results obtained in location accuracy. Additionally, some of the differences between the total groups and the identified groups are also due to mistakes on the data recording, such as a group recorded with coordinates that are out of the map.

The results on location accuracy are computed based on the identified groups, not the total of groups as the main objective of the study is to assess the performance on locating groups of RFID labels, not detecting them is a consequence that it is not directly related to the algorithm but the parameters used.

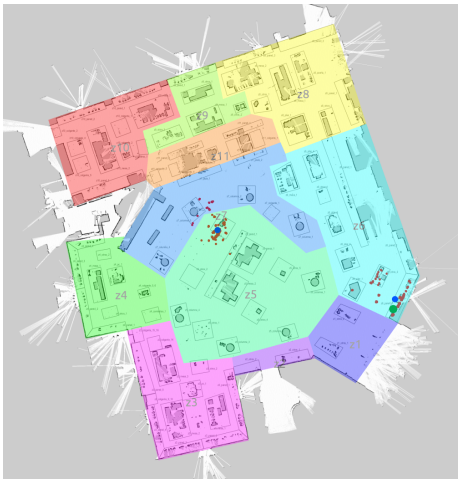
At area level the accuracy has reached a 95.9%, the results at fixture level are lower, a 92.1%. However, in both cases the accuracy is above 90%, which is considered remarkable given the customer expectations. In terms of the mean distance error, it is around 70 cm, given that the *reading_distance* is fixed the results are also remarkable taking into account that it is only 20 cm higher than in [35], but it is a test in a real environment and with a simpler and more efficient algorithm. Obviously, dataset

A presents better results, but adjusting the parameters with its data can have this type of influence. In any case, the results obtained with dataset B are considered valid and outstanding for the reasons explained.
The accuracy with respect to the areas is higher than with respect to the fixtures. This happens because the number of areas is much lower than the number of fixtures, and therefore, the complexity of the second is higher. However, it is worth mentioning that there are some cases in which the area is not properly located but the fixture it is. Usually, this is found when the group is situated at the edges of several areas but with only one fixture close to the group, Figure 6.9f is an example.
The next set of figures, from Figure 6.9a to Figure 6.9f, show some examples of the results in which on top of the layout captured by the robot the areas and the fixtures are drawn, as well as, the estimation of the position of every RFID detection, red dots, the estimation of the position of the group, blue dots, and the recorded position of the group, green dot. They are just illustrative to show the results obtained.
The results showed several examples, such as the ones in Figures 6.9e and 6.9f, where the group is placed in the wrong area, but it is next to its real one. Due to this, some experiments in a similar context have been conducted in which the areas have a small overlap. For example two contiguous areas have a shared zone of 1 meter. Assuming this overlap has increased considerably the accuracy, reaching in some cases a 100%.

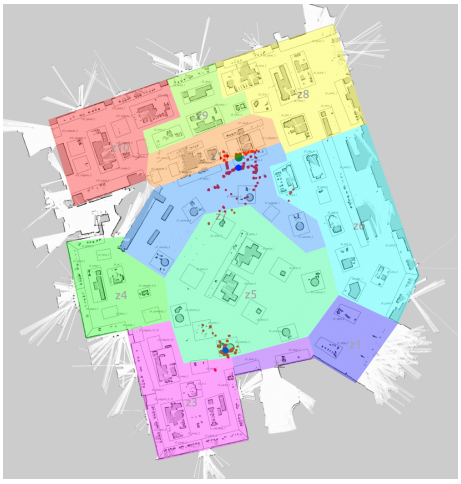


(a) Properly located group (b) Properly located group

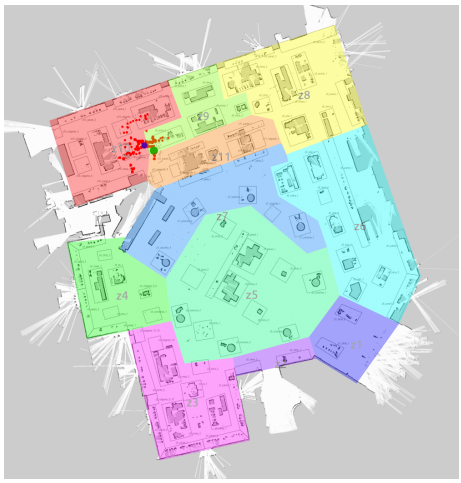
Figure 6.9: Sample of results of datasets A and B (left and right column respectively). The red dots are the EPC readings, the blue dots are the estimated position of the group and the green dot is the recorded location of the group.



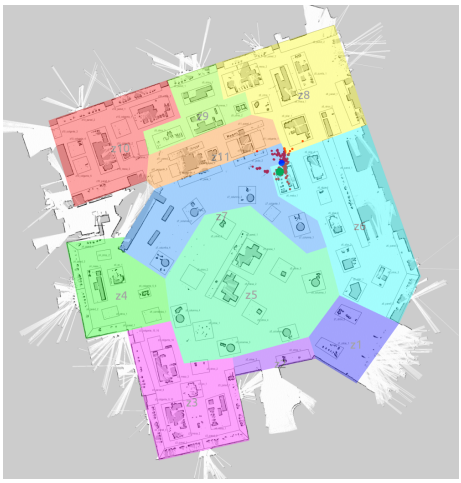
(c) Two locations for a single group



(d) Two locations for a single group



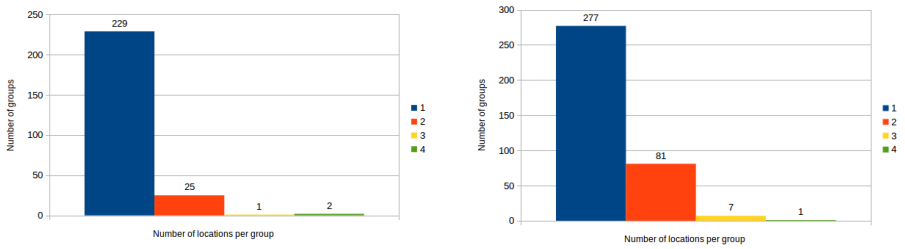
(e) Wrongly located group



(f) Wrongly located group

Figure 6.9: Sample of results of datasets A and B (left and right column respectively). The red dots are the EPC readings, the blue dots are the estimated position of the group and the green dot is the recorded location of the group.

Due to the clustering process, some groups are located in different places of the store. However, the fact that the group is in more than one place cannot be verified if it is not recorded in the ground-truth. Nevertheless, looking at the next two histograms, see Figure 6.10a and Figure 6.10b it can be seen that most of the groups of items are placed together and, therefore, the results obtained are not considered biased due to the creation of extra locations for the group, which could be a consequence of the clustering parameters. Nevertheless, it has been observed that dataset B presents more groups with 2 locations than dataset A but there are also more groups in general, therefore it is not considered an issue, notice that the vertical axis has a different scale for each figure.



(a) Test A

(b) Test B

Figure 6.10: Quantity of identified groups with respect to the quantity of locations found per group

Finally, out of curiosity, the test B lasted longer which means that the robot had more time to get more samples of EPCs but at the same time more operations done by the store associate took place, and also the store opened to customers. Due to this, some minor issues arose, for instance, in Figure 6.11 there are some EPCs found outside any area of the store, so in the backstore. This happened because the robot captured the moment in which the associates of the store were moving products from the backstore to the store floor. This is not critical for this study, but if the operation of the robot needs to be considered in a regular basis, it is required to properly organize the store operations accordingly.

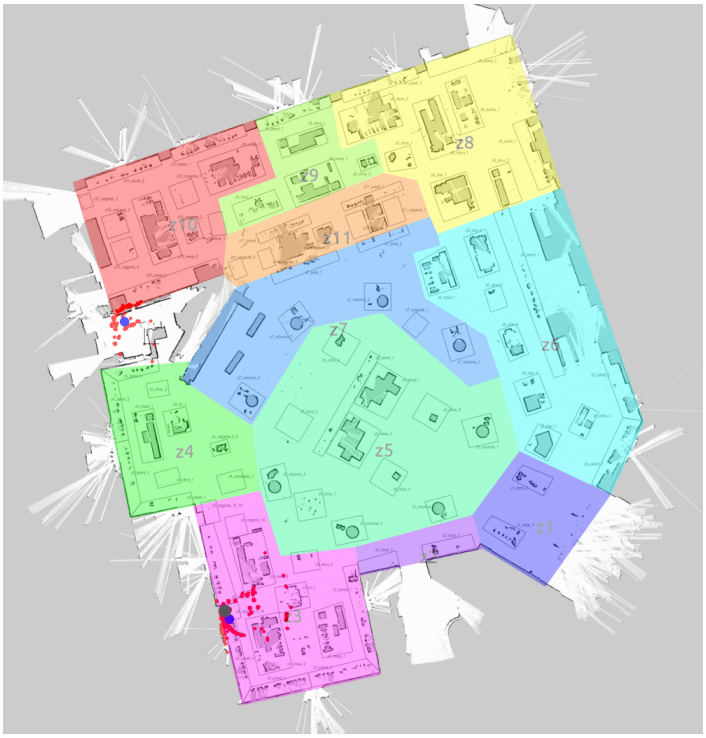


Figure 6.11: Group placed at backstore due to associates operations. The dots that are in an area without color are caused due to movement of products. The red dots represent the EPC readings of the same SKU, the blue dot is the estimated location of the group and the green dot represents the captured location of the SKU.

6.5 Conclusions

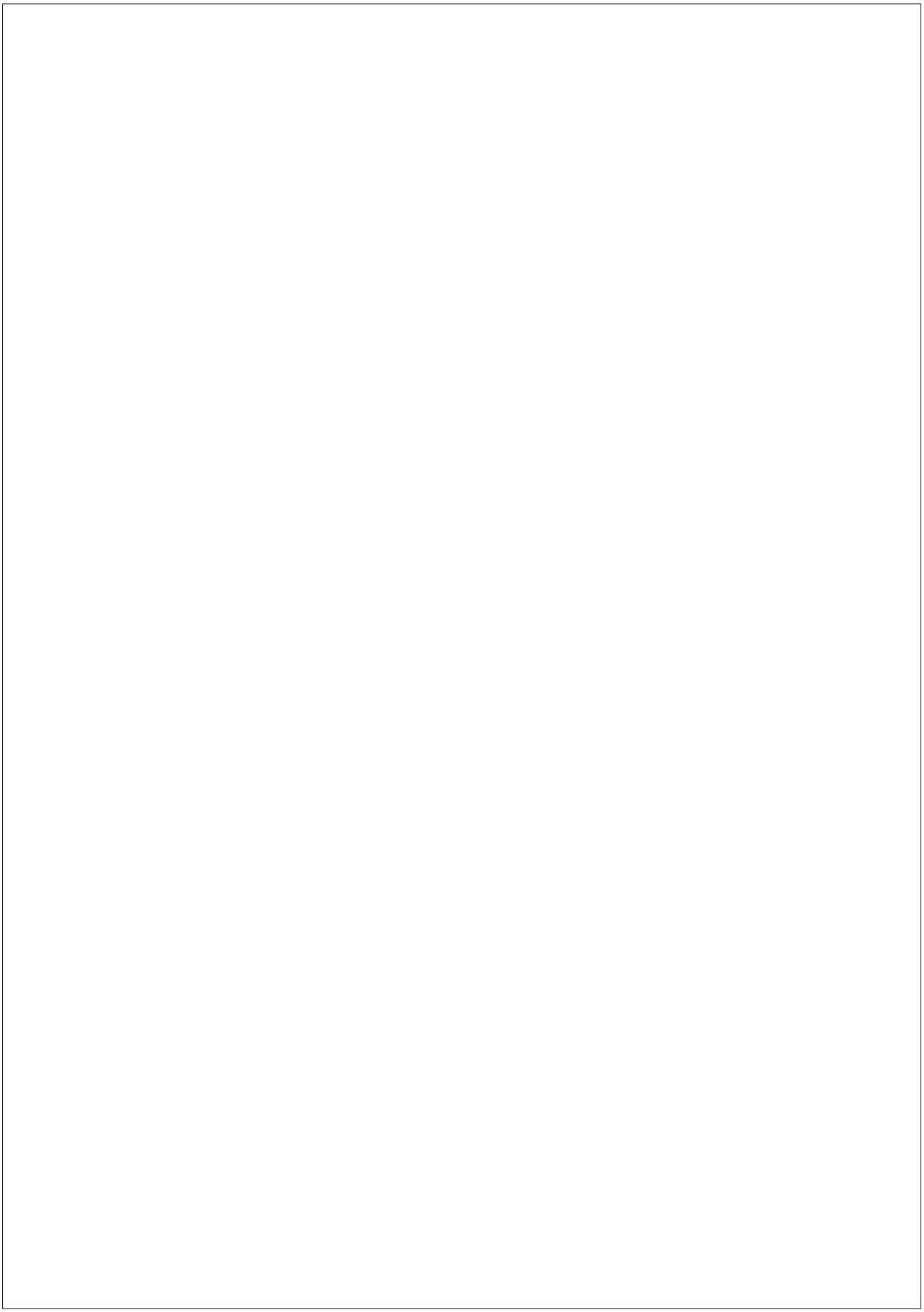
The main contribution of this chapter is showing that complexity does not always help in order to achieve significant results when you are trying to model a system that is dependent of an environment that can change. For this work, several simplifications have been adopted with respect to the location of RFID tags and the results obtained are valid and similar to those of the state of the art such as in [35].

The simplifications adopted in this section are the following. First, this work assumes that for most applications the location of a single tag will not be as relevant as the location of a group of tags, so that, the information obtained can be deliberately mixed. Second, providing data on location based on the Cartesian coordinates in a map is not practical information, but a relative position can be much more informative, such as ”next to the door” or ”in the closet next to the elevators”.
Another simplification adopted is not using the value of the RSSI or the phase for the location model. Previous works show that this values cannot be used properly in general cases. Instead, the RF power has been decreased to penalize the readings from longer distances or with many bounces. As anticipated in [34] the closer the robot to the tag the better the precision. Therefore, with a fixed reading distance and many repeated readings from different positions of the robot, the obtained results are promising.

The parameters adjustment of the clustering process is not part of the intended simplification of the methodology, however, it is required once the groups of EPCs can be simultaneously in more than one location. The chosen method performs well and does not require the previous knowledge of the number of clusters required, which is an advantage for the application given that this information is not available. Nevertheless, if the separation between the two groups is lower than the reading distance of the robot it can be impossible to separate them.

Testing the solution in a real environment is also very significant because it shows that the solution presented in this chapter is relevant and that can be used in any scenario.
Regarding the results, the obtained accuracies in placing the groups in their correct zone or fixtures are satisfying. Over 95% of the identified groups are located in their right area and over the 92% of them also are located in their right fixture.

Finally, adding reference tags, for instance in every fixture, is seen as a simple modification of the environment but that could improve the results by adding a layer where similarity measures between the reference tags and the tags of the products can help to determine if a tag is in a fixture or not. Additionally it would avoid the tedious task of manually capturing the coordinates of all fixtures.



Chapter 7

CONCLUSIONS & FUTURE WORK

7.1 Conclusions

This research aimed to determine if simple robots, in terms of hardware and software are able to perform a complex tasks such as stock-counting RFID tagged items in a cooperative way and with no human intervention. In the course of this thesis this has been effectively proved, first through simulation and then by designing, developing and testing real robots in a library.
In fact, the investigation shows that by means of a stigmergic behavior these simple robots can operate with complete autonomy. This is achieved by the development of a map-less algorithm for stock-counting that does not require a map of the environment for navigation, and hence, no human intervention whatsoever.
Additionally, the development of the map-less algorithm has proved that the RFID labels placed on items in a given space can be used, without any prior knowledge of the environment, to guide robots and fulfill an inventory mission.

Specifically in simulation, the map-less algorithm has been compared

against area coverage algorithms. These methods have been adapted to solve the stock-counting problem, nevertheless, it has been observed that the developed algorithm has outperformed them. Additionally in simulation, the performance of the system with up to 50 robots has been studied showing that the scalability of the algorithm is very high. Due to a high level of parallelization of the task between robots.

In general terms, the design of the robots presented in this thesis shows that inventory robots can be simplified, and therefore, costs can be drastically reduced and reliability significantly increased. One of the objectives of this thesis is showing that features like efficiency or reliability do not need to be linked with complexity and therefore, by shifting to more suitable approaches, it is possible to obtain valuable performance enhancements without an increase of cost or complexity.
The design of a specific robot to leverage the map-less algorithm was required. There is not an RFID-based inventory robot in the market with the required specifications. By applying a design research methodology it has been possible to develop a payload for a robot base able to determine the relative orientation between the robot and the detected RFID labels. Only off the shelf hardware has been used facilitating the future commercialization of the designed robot.
Finally, the use of ROS, an open robotics platform, has allowed easier intercommunication between different hardware pieces and open software developed by the community has helped the development from scratch of a fully functional robot.

Once the robot has been developed in terms of hardware and software, the complete solution has been tested in a real environment. The experiments in a library have proved that the developed robots are able to cooperatively stock-count an environment without the need of a map. In fact, the accuracy figures obtained with this algorithm are as good as the state of the art RFID inventory accuracy figures, exceeding 99%. Additionally, this value has been achieved in a library, which is a challenging environment for RFID detection. However, due to the specific actions taken to

understand and help the process of taking inventory, the experiments have been very successfull.
In particular, it has been observed that two robots can take inventory of a library floor of around 280 m^2 with more than 4,000 books in less than two hours, which is very fast value for an inventory mission of this level of accuracy. Also, it is worth mentioning that if the space is reduced by half, but the number of robots of the system is the same, the time required for inventorying is also reduced by half. In other words, if the size of an environment is doubled, the time and performance for an inventory mission can be maintained by simply doubling the number of robots in the system.
The developed map-less algorithm, based on a stigmergic behavior, allows a system of multiple robots to coordinate their actions during an inventory task. This feature was already observed in simulation, however, this has been verified during the tests in a real environment.
Robustness is achieved at a system scale: the mission will no longer fail if one robot does fail, but only if all robots fail. During this thesis it has been observed that if a robot is not able to identify more RFID labels for whatever reason, the other robot of the system can continue identifying them until the mission goal is achieved. Therefore, the reliability of the system is increased proportionally to the number of robots in it.
Regarding the scalability of the system, both, simulation and experiments have shown that the system scales approximately linearly with the area of the environment. If a robot performs an inventory mission in a given area, when this area is doubled, time and accuracy can be maintained by doubling the number of robots.

The scalability of the system linked with the cost-efficiency design provides an inventory solution that adapts to any particular environment with great flexibility. This solution is particularly useful for retailers to allow them to take frequent and accurate inventories.

Aligned with the latter, this thesis also presents an algorithm for locating groups of RFID labels. It is mainly focused on simplicity, aiming to

obtain the maximum benefit from the system while optimizing its utility for retailers. The main hypothesis for the presented methodology is that the RFID parameters to achieve good results for inventory are not the same than for location. Therefore the results obtained are significantly improved only by adapting the RFID parameters to the goal of location.
The main variation of this method with respect to current methods is that it aims on determining the location of a group of RFID labels with respect to a landmark of the environment, like a piece of furniture or a precise area of the store, providing useful information for its user. Additionally, the system is developed with COTS hardware and requires no store intervention, which increases its convenience.
Testing the method in a real, non controlled environment, has yielded an outstanding location accuracy: above 92%.

7.2 Future work

This thesis also opened new research lines left for future investigation.

Regarding the map-less algorithm developed, this research presents an algorithm with an attraction function that is developed based on the knowledge on how RFID-based inventory robots perform their missions. However, it is possible than a better or, even an optimal attraction can be defined. For instance, this attraction could also depend on the local knowledge of the obstacles present in the surroundings of the robot and not only the RFID tags. Also, it would be interesting to test machine learning methods, such as those based on deep learning and reinforced learning, and develop a simulation environment, as realistic as possible, to train attraction functions and compare them against the ones that are knowledge based.

Regarding the designed robot, there are three possible new lines of work. Firstly, the robot has been developed under the ROS framework, but currently the ROS framework is being replaced by the newer version, ROS

2, which among other things, is completely distributed and does not have any central control. Therefore, it would be interesting to move all the developed software to ROS 2 as it fits much better with the stigmergic approach.
Secondly, it has been observed that by changing small policies on the algorithm, it could be possible to use it for continuous inventory missions. This along with a better navigation around people would also change how inventory robots are used in the future. So, given that they could be permanently navigating around the store, stopping only to recharge and providing a continuous inventory, many other features could be added, helping customers, measuring dirty or disinfecting surfaces with UV-light.
Thirdly, the developed package for collaborative mapping can also enhance features and functionalities for the inventory robots that have not been tested in this thesis. It could be possible to determine an automatic final condition for the method, such as all the aisles have been visited. Or, it could be possible to send a robot to a specific location if after a determined amount of time the stream of new tag identifications is very low. However, it will be required to assess if these benefits can still be available in a decentralized system.
Related to this, it would be very interesting to test location capabilities in a system with multiple robots that can share a map. This would dramatically increase the amount of detections per tag leading to larger sets of samples that would significantly increase the accuracy.

Regarding the developed location method there are, at least, three new research lines. First, studying the performance of the new location algorithm with the latest robot design. This design has four antennas pointing in orthogonal directions, while the previous design used in testing has only two different directions, therefore, the developed robot could better identify the relative orientation of the items detected.
Second, the retailers have expressed their willingness to add reference tags to the fixtures in their stores to enhance location missions. In this sense, it would be very interesting to modify this method accordingly. For instance, similarity measurements between tag detections could be

used in order to determine with respect to which reference tag the group of tags is placed.
Third, it would be interesting to tackle the location of RFID labels with deep learning algorithms. These type of algorithms require large amounts of data and this has usually stopped them for being used in this field. However, systems with multiple robots provide much larger datasets from more simultaneous detections and, by understanding the robots as data streamers, deep learning algorithms could use them to produce faster real-time location results.

Bibliography

[1] K. Chawla, C. McFarland, G. Robins, and W. Thomason, "An accurate real-time RFID-based location system," *International Journal of Radio Frequency Identification Technology and Applications*, vol. 5, no. 1, p. 48, 2018.

[2] E. DiGiampaolo and F. Martinelli, "Mobile Robot Localization Using the Phase of Passive UHF RFID Signals," *IEEE Transactions on Industrial Electronics*, vol. 61, pp. 365–376, jan 2014.

[3] Y. Liu and Y. Qiu, "An indoor localization of UHF RFID using a hybrid approach," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pp. 1653–1656, IEEE, apr 2012.

[4] M. Morenza Cinos, *RFID autonomous robot for product inventory and location*. PhD thesis, Universitat Pompeu Fabra, nov 2018.

[5] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 99–110, jun 2006.

[6] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): part II," *IEEE Robotics & Automation Magazine*, vol. 13, pp. 108–117, sep 2006.

[7] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem," *Proc. AAAI Nat. Conf. Artif. Intell.*, pp. 593–598, 2002.

[8] G. Grisetti, C. Stachniss, and W. Burgard, "Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pp. 2432–2437, IEEE, 2005.

[9] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, feb 2007.

[10] B. P. Gerkey and K. Konolige, "Planning and control in unstructured terrain," *Workshop on path planning on costmaps. ICRA Proceedings*, 2008.

[11] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, pp. 23–33, mar 1997.

[12] M. Morenza-Cinos, V. Casamayor-Pujol, J. Soler-Busquets, J. L. Sanz, R. Guzmán, and R. Pous, "Development of an RFID inventory robot (AdvanRobot)," in *Robot Operating System (ROS)* (A. E. Koubaa, ed.), vol. 2, ch. 12, pp. 387–417, Springer, 2017.

[13] M. Morenza-Cinos, V. Casamayor-Pujol, and R. Pous, "Stock visibility for retail using an RFID robot," *International Journal of Physical Distribution and Logistics Management*, vol. 49, pp. 1020–1042, dec 2019.

[14] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pp. 146–151, IEEE Comput. Soc. Press, 1997.

[15] C. Stachniss, G. Grisetti, and W. Burgard, "Information Gain-based Exploration Using Rao-Blackwellized Particle Filters.," *Robotics: Science and Systems*, 2005.

[16] P. Senarathne and D. Wang, "Incremental algorithms for Safe and Reachable Frontier Detection for robot exploration," *Robotics and Autonomous Systems*, vol. 72, pp. 189–206, oct 2015.

[17] J. M. Santos, T. Krajnik, J. P. Fentanes, and T. Duckett, "Lifelong Information-Driven Exploration to Complete and Refine 4-D Spatio-Temporal Maps," *IEEE Robotics and Automation Letters*, vol. 1, pp. 684–691, jul 2016.

[18] A. Kleiner, J. Prediger, and B. Nebel, "RFID Technology-based Exploration and SLAM for Search And Rescue," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4054–4059, IEEE, oct 2006.

[19] V. A. Ziparo, A. Kleiner, B. Nebel, and D. Nardi, "RFID-based exploration for large robot teams," in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 4606–4613, IEEE, apr 2007.

[20] P.-P. Grassé, "La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs," *Insectes Sociaux*, vol. 6, pp. 41–80, mar 1959.

[21] A. A. Khaliq, "Stigmergy at Work : Planning and navigation for a service robot on an RFID floor," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1085–1092, may 2015.

[22] M. Batalin and G. Sukhatme, "Efficient exploration without localization," *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, pp. 2714–2719, 2003.

[23] J. Svennebring and S. Koenig, "Building Terrain-Covering Ant Robots: A Feasibility Study," *Autonomous Robots*, vol. 16, pp. 313–332, may 2004.

[24] I. A. Wagner, Y. Altshuler, V. Yanovski, and A. M. Bruckstein, "Cooperative Cleaners: A Study in Ant Robotics," *The International Journal of Robotics Research*, vol. 27, pp. 127–151, jan 2008.

[25] M. Mamei and F. Zambonelli, "Pervasive pheromone-based interaction with RFID tags," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 2, pp. 4–es, jun 2007.

[26] E. Ferranti, N. Trigoni, and M. Levene, "Rapid exploration of unknown areas through dynamic deployment of mobile and stationary sensor nodes," *Autonomous Agents and Multi-Agent Systems*, vol. 19, pp. 210–243, oct 2009.

[27] A. A. Khaliq, M. Di Rocco, and A. Saffiotti, "Stigmergic algorithms for multiple minimalistic robots on an RFID floor," *Swarm Intelligence*, vol. 8, pp. 199–225, sep 2014.

[28] B. A. Dembo, Y. Peres, J. Rosen, and O. Zeitouni, "Cover times for Brownian motion," *Annals of Mathematics*, vol. 160, pp. 433–464, 2004.

[29] A. Schroeder, S. Ramakrishnan, M. Kumar, and B. Trease, "Efficient spatial coverage by a robot swarm based on an ant foraging model and the Lévy distribution," *Swarm Intelligence*, vol. 11, pp. 39–69, mar 2017.

[30] S. Koenig, B. Szymanski, and Y. Liu, "Efficient and inefficient ant coverage methods," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1-4, pp. 41–76, 2001.

[31] S. Koenig and Y. Liu, "Terrain coverage with ant robots: a simulation study," *Proceedings of the fifth international conference on Autonomous agents*, pp. 600–607, 2001.

[32] H.-Y. Kung, S. Chaisit, and N. T. M. Phuong, "Optimization of an RFID location identification scheme based on the neural network," *International Journal of Communication Systems*, vol. 28, pp. 625–644, mar 2015.

[33] D. Hahnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with RFID technology," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, pp. 1015–1020 Vol.1, IEEE, 2004.

[34] G. Cicirelli, A. Milella, and D. Di Paola, "RFID tag localization by using adaptive neuro-fuzzy inference for mobile robot applications," *Industrial Robot: An International Journal*, vol. 39, pp. 340–348, jun 2012.

[35] J. Zhang, Y. Lyu, J. Patton, S. C. G. Periaswamy, and T. Roppel, "BFVP: A Probabilistic UHF RFID Tag Localization Algorithm Using Bayesian Filter and a Variable Power RFID Model," *IEEE Transactions on Industrial Electronics*, vol. 65, pp. 8250–8259, oct 2018.

[36] M. Morenza-Cinos and V. Casamayor-Pujol, "RFID Location dataset." https://zenodo.org/record/1215660#.XjxIqXX0lhE, nov 2018. doi: 10.5281/ZENODO.1215660".

[37] M. Wooldridge and N. R. Jennings, "Agent theories, architectures, and languages: A survey," in *Proceedings of the Workshop on Agent Theories, Architectures, and Languages on Intelligent Agents*, ECAI-94, (Berlin, Heidelberg), pp. 1–39, Springer-Verlag, 1995.

[38] S. Vaishnavi, V., Kuechler, W., and Petter, "Design Science Research in Information Systems," 2004.

[39] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly: Management Information Systems*, vol. 28, no. 1, pp. 75–105, 2004.

[40] H.-C. Lee, S.-H. Lee, M. H. Choi, and B.-H. Lee, "Probabilistic map merging for multi-robot RBPF-SLAM with unknown initial poses," *Robotica*, vol. 30, pp. 205–220, mar 2012.

[41] M. Ester, M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," pp. 226—-231, 1996.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.