"Plug-and-Play" Inventory Robots: Autonomous Itinerary Planning through Autonomous Waypoint Generation

Sergio Lopez-Soriano

***Abstract*—Current robotic inventory systems rely on human interaction for installation, configuration, and reconfiguration tasks. However, this dependence on human involvement hampers the efficiency of the process chain in the industry and can lead to bottlenecks in the supply chain and the overall system. In this study, we present a "plug-and-play" methodology that enables the deployment of inventory robots and the autonomous reconfiguration of the map and inventory itineraries. This work introduces, for the first time, an autonomous waypoint generation method based on RFID exploration and the first fully autonomous solution for designing efficient itineraries for inventory robots. The proposed methodology is extensively detailed, and a series of experiments are conducted in a real environment with physical robots. The results demonstrate that the autonomously designed inventory itineraries achieved through the proposed method exhibit similar performances to those designed by humans.**

***Index Terms*—e-logistics, autonomous exploration, autonomous waypoint generation, autonomous planning, inventory robots.**

I. Introduction

NOWADAYS, radiofrequency identification (RFID) inventory robots are being commercialized and used in different environments such as libraries, stores, warehouses, etc. [1]–[3]. While current solutions promise reducing costs and improving the system efficiency [4], they are far from being fully efficient and their deployment entails complications that have not yet been addressed by the scientific community or the industry. In fact, the main barriers to achieving fully autonomous solutions are the lack of adaptability and scalability.

In terms of adaptability, fully autonomous inventory robots shouldn't require an installation phase in which a human operator performs the system configuration. Similarly, autonomous inventory robots should be able to self-reconfigure after environment variations. Indeed, inventory robots need a map and an itinerary, that depends on the environment layout, for proper operation [5]. Currently, the map and the itinerary waypoints are generated while a human operator drives the robot through the preferred itinerary [6]. Designing an itinerary for inventory applications is a relatively simple task for humans, but it reduces the robot autonomy and decreases the overall efficiency, since the human operator needs to leave its current task and the robot must wait for the operator to configure it, which, in addition, has a detrimental impact on the scalability of these solutions.

Sergio Lopez-Soriano, was with the Universitat Pompeu Fabra, Barcelona, 08018 Spain. He is now with the Interdisciplinary Institute (IN3) of the Open University of Catalonia, Barcelona, 08018 Spain.(e-mail: slopezsor@uoc.edu).

Manuscript received January 11, 2023; revised June 6, 2023.

By contrast, there is no current solution to obtain an itinerary adapted to a specific inventory layout autonomously. This is due to the fact that, at present, there aren't any methods that can generate, autonomously, a set of waypoints which are limited to the area of interest (AOI) of the inventory scenario, and that, at the same time, can represent with enough granularity the AOI. The AOI is defined as the minimum fraction of all the available space that includes the actual inventoried items, and that is bounded by a closed contour. Consequently, to date, either at the installation or reconfiguration phases, it is necessary for a human operator to carry out the mapping and the design of the itinerary.

A. *Inventory itinerary problem*

Currently, route planning for autonomous robots remains a hot topic of research [7]–[11]. However, the literature on RFID-assisted systems indicates that current approaches [12]–[14] perform poorly in terms of area coverage and inventory time. In addition, those approaches are based on selecting the shortest paths which, in a general unknown environment, is unlikely to optimize the inventory accuracy. Moreover, neither RFID-based approaches, nor any other techniques [15], [16], have shown to provide the essential autonomous generation of waypoints, specifically optimized for the inventory application, in a general inventory layout.

The inventory itinerary problem (IIP) is defined here as the task of creating an itinerary resembling an itinerary designed by a human expert. The classical approach for representing an inventory environment consists of structuring the AOI in aisles [6]. In fact, the IIP goal is to find an itinerary that traverses all the aisles efficiently, i.e., minimizing the path overlapping. Such itinerary must be composed of an ordered set of waypoints or intermediate goals that enables traversing all the aisles while optimizing the inventory accuracy. In addition, coping with dead ends and poorly connected areas requires that waypoints can be visited multiple times, and, therefore, the number of cycles (re-visiting waypoints) must be minimized to optimize the itinerary.

The IIP problem cannot be efficiently solved by any of the classical planning algorithms in their current form [17], such as coverage path planning (CPP) [18] or probabilistic roadmap (PRM) [19], due to the reasons mentioned next. First, only the starting point is known by the robot at the beginning of the installation stage, and all waypoints are potential final destinations. Second, waypoints can be visited more than one time. Third, in the IIP, the coverage is not related to the range sensors but to the RFID system. Therefore, shorter paths are not specially optimal and, in general, longer paths are preferred (as long as all waypoints are inside the AOI). In conclusion, solving the IIP is very different from solving the traveling salesman problem (TSP) [20]. Therefore, a new specific methodology must be formulated.

This contribution introduces, for the first time till date, to the best of the knowledge of the author, an autonomous waypoint generation method based on RFID-driven exploration. The proposed method (Fig.1) is the first of its kind, and it enables fully autonomous design of itineraries for inventory robots. This is possible since the RFID-based exploration technique generates a set of robot poses which are naturally enclosed within the AOI, i.e., the RFID-based exploration never drives the robot away from the AOI. This document also presents a real life experiment consisting in the ”plug-and-play” installation of an inventory robot in a library containing 6000 tagged books. The dataset used in these experiments is a subset of a larger dataset that can be found at [21]. In other words, the robot knowledge about the environment before the installation starts is completely null. The robot doesn't have a map, nor a set of valid waypoints, nor a predefined itinerary. Instead, the proposed method enables the robot to autonomously produce a map of the AOI, a set of representative waypoints and an efficient itinerary. The results show that the itinerary, designed with the proposed method, can be recursively walked, back and forth, to perform inventory missions of an initially unknown environment. The results also show that the inventory accuracy reached during the experiments, using the proposed method, is comparable to the accuracy obtained using an itinerary designed by a human operator.

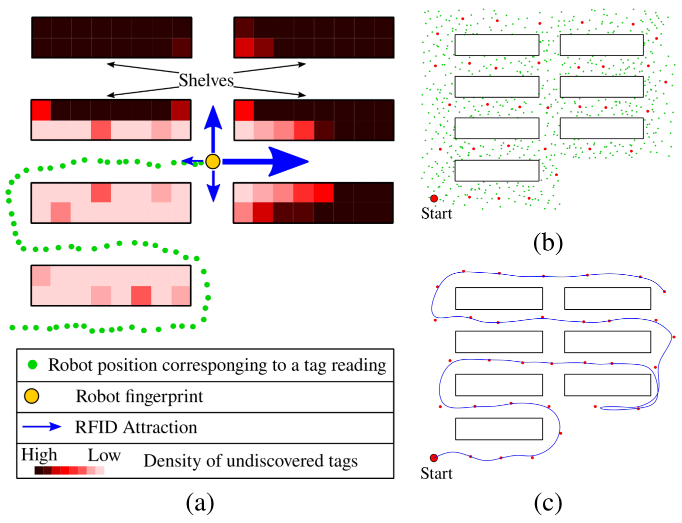

Shelves
Robot position correspongding to a tag reading
Robot fingerprint
RFID Attraction
High
Low
Density of undiscovered tags
(a)
Start
(b)
Start
(c)

Fig. 1: Different stages of the proposed methodology for autonomous itinerary planning. (a) This image illustrates the autonomous waypoint generation process. The schematic represents an inventory environment with seven shelves containing RFID-tagged items. The robot's position is depicted as a yellow circle, and the small green circles indicate the robot's location during tag readings. The density of undiscovered tags in the shelves is represented by squares of various colors, as shown in the color bar at the bottom of the image. The attraction felt by the robot towards the RFID tags in the front, back, left, and right directions is depicted by blue arrows, with their size corresponding to the magnitude of the attraction. (b) As time passes, the robot covers all the aisles multiple times, leading to a significant increase in the number of tag reading locations (unfiltered set of waypoints), as indicated by the green/red dots. Subsequently, a waypoint filtering process is carried out, selecting only the red dots as the final set of waypoints. (c) The final stage involves constructing a complete graph using the selected waypoints. Then, edges enabling the drawing of a planar connected graph are selected, while the rest are removed. Finally, the optimal itinerary is computed based on specific criteria (related to the area coverage).

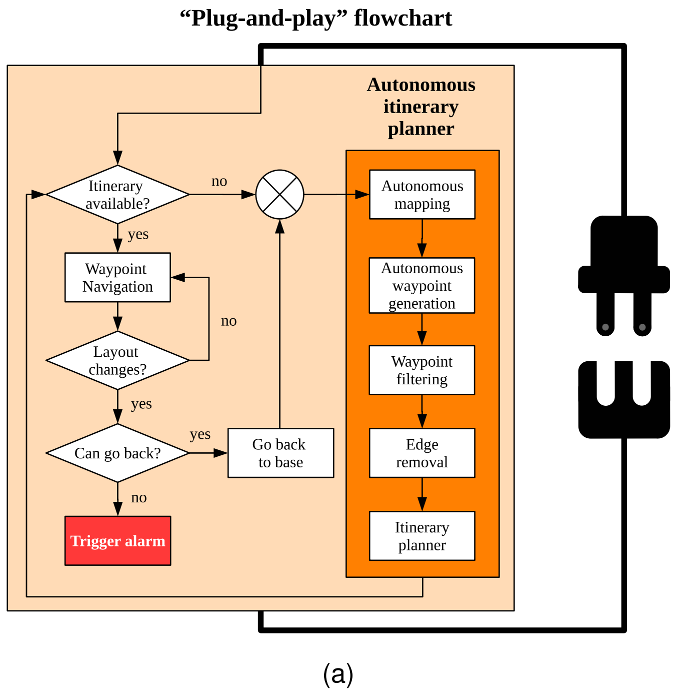
II. METHODS

The proposed methodology consists of five stages (Fig.2a): autonomous mapping, autonomous waypoint generation, waypoint filtering, edge removal and itinerary design. During the stages one to four, a set of waypoints is obtained from autonomously generated robot poses to obtain the vertices and edges of a graph. First, a map of the AOI is created autonomously, following the process described in [22], in order to provide accurate robot localization. Then, the robot executes an autonomous RFID-based navigation mission [23], through which the robot records a set of poses, scattered across the AOI (Fig.1(a)). Then, a set of candidate vertices is filtered out from all the possible poses depending on their distance to the selected ones (Fig.1(b)). Then, the graph edges are computed using the Dijkstra's algorithm [24]. At this point, the problem complexity of finding the longest itinerary with a limited number of revisited nodes is extremely high. Thus, the problem complexity is reduced by filtering the set of current edges, so that the complete graph is transformed into a planar graph [25]. Finally, the autonomous itinerary planning algorithm computes the itinerary that traverses all waypoints while maximizing the area coverage and minimizing the number of steps (Fig.1(c)). Tree search is used to find the optimal itinerary after applying a set of rules for the branch pruning.

The proposed method is tested on a ground robot equiped with a range sensor, a sensor for collision avoidance, a single board computer, an RFID reader, and four RFID reader antennas with orientations at 0, 90, 180, and 270 degrees. The experiments presented in the following sections are performed in the library of the Univesritat Pompeu Fabra, campus Poblenou [26]. The rest of this section details the processes involved in the autonomous design of itineraries for inventory robots.

A. Autonomous mapping

As soon as the robot starts the inventory for the first time it will look for an available itinerary (Fig.2a). Not finding the itinerary will trigger the autonomous mapping process, also



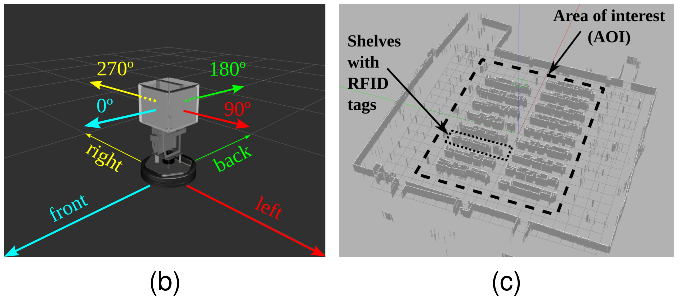


Fig. 2: Materials and methods: (a) The flowchart of the proposed methodology, (b) the relative position of the four antennas with respect to the robot frame, and (c) the representation of the map of the environment.

called RFID exploration. From this moment, the robot starts reading tags, i.e., it starts taking the inventory of the environment. The difference is that, during this process, instead of following a predefined itinerary, the robot explores the terrain driven by the attraction that the RFID reader feels from each of the connected antennas (Fig.2b). Thus, the robot selects the next exploration goal to be at one meter in the direction of the antenna with higher RFID attraction. Therefore, the exploration task is divided in steps, where each step starts at the moment when the robot computes the next goal until it reaches such goal, and the next step starts. The RFID attraction perceived from the antenna i at the step s is calculated as in (1).

$$Attraction_{i,s} = new\_tags_i + \frac{tags\_read_{i,s}}{read\_tags\_total\_count}, \quad (1)$$

where new_tags_i is the number of tags added to the inventory for the first time, which are read by the antenna i;

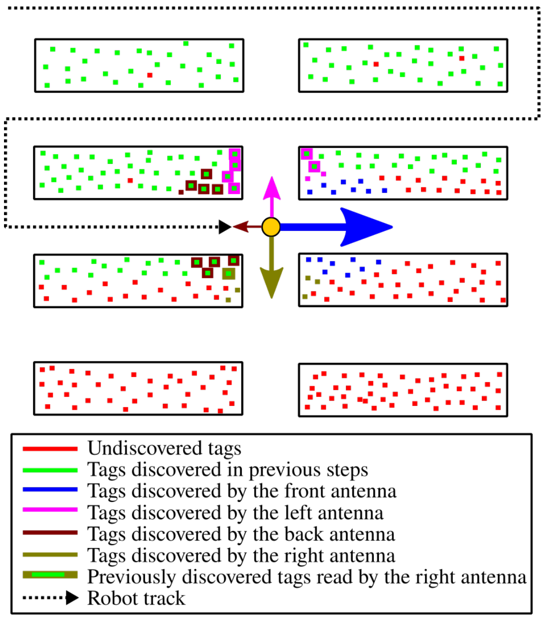

Undiscovered tags
Tags discovered in previous steps
Tags discovered by the front antenna
Tags discovered by the left antenna
Tags discovered by the back antenna
Tags discovered by the right antenna
Previously discovered tags read by the right antenna
Robot track

Fig. 3: Example of the calculation of the RFID attraction during an inventory mission, at the kth step.

$tags\_read_{i,s}$ is the number of the tags read from the antenna i at the step s, that were added in previous steps; and the $read\_tags\_total\_count$ is the total number of times that the tags, accounted for in the numerator, have been read since the beginning of the inventory mission. Therefore, the first term of the equation accounts for the attraction to areas where the inventory is still required, and the second term is intended to let the robot stay more time in tag-crowded areas, where a high number of tags can be more difficult to inventory.

The map of the AOI (Fig.2c) is built using RFID-based exploration [22] and gmapping, a laser-based simultaneous localization and mapping (SLAM) algorithm [27].

TABLE I: Computation of the RFID attraction for the example of Fig.3

$Antenna\ direction$	Front	Left	Back	Right
new_tags	16	2	1	5
$tags\_read$	0	6	8	1
$read\_tags\_total\_count$	0	18	20	4
$Attraction$	16	2.33	1.2	5.25

For illustrative purposes, an example of a robot performing an inventory mission using RFID-based navigation is presented in Fig.3. The figure depicts a simple scenario with eight shelves arranged in two columns. The inventory items are represented by small colored squares, with unread tags shown in red and tags discovered in previous steps displayed in green. The tags discovered in the current step are colored based on the antenna that first read them (front = blue, left = pink, back = brown, right = golden). Tags that have been read during the current step but were previously discovered remain green and are surrounded by a larger square colored according to the specific antenna that read in this step. The robot's footprint is depicted as a yellow circle, the previous route is indicated by a dotted meandering line, and the attraction in the four directions is represented by arrows, with colors corresponding to the antenna colors and sizes reflecting the level of attraction in each direction. The numerical values for the attraction felt by the robot in the four inspected directions are calculated in Table I using Equation 1. The results demonstrate that the highest attraction value is sensed by the front antenna. Consequently, the robot will set its next goal in the forward direction (blue arrow).

B. Autonomous waypoint generation

This is the core concept of the proposed solution. Once the map has been created, the robot can localize itself on it using adaptive Montecarlo localization (AMCL) [28]. Then, the autonomous waypoint generation process starts. Again, the robot navigates the environment autonomously following the attraction (1) felt towards the RFID tags, while storing tag reading information.

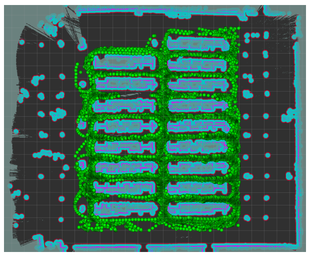
For each tag reading, the robot pose estimate is stored in a database. Then, after the navigation has finished, the algorithm uses the stored poses to generate NR waypoints. However, the number of initial waypoints ($|NR|$) can be huge (the RFID reader can record over a million tag readings per mission [29]), and therefore, the waypoints need to be filtered in order to produce a tractable problem. The results of the autonomous waypoint generation is illustrated in Fig.4a. The results clearly show that the RFID navigation is automatically bounded by the AOI ((Fig.2c) during all the inventory mission due to the attraction-based setting of goals.

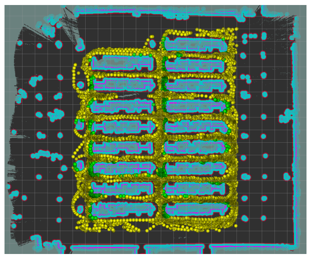
C. Waypoint filtering

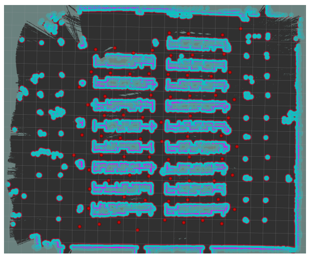
The waypoint filtering process is designed to efficiently remove unnecessary redundancy while tackling the inventory application specific requirements of granularity and coverage.

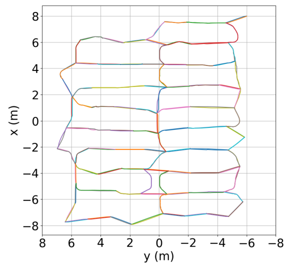
Three filters are applied in order to reduce the set size. Once the starting waypoint is defined, the first filter will select one sample each t_f seconds. Next, the waypoints close to obstacles will be filtered out. In the example shown in Fig.4b, the green circles represent the filtered waypoints while the yellow ones remain unfiltered. The last filter consists of dropping the waypoints within a circle of radius r_{in} = 1.5 m from the previous waypoint. Then, jumping to the next waypoint being the one with the closest distance inside a circle of radius r_{out} = 3 m. Then, it's appended to a list of final waypoints. The process is repeated until the number of waypoints in the final waypoints list is equal to the size of the list to be filtered (Fig.4c).

r_{in} and r_{out} are design parameters and must be tuned to a particular layout. They balance the density of waypoints. Since an inventory environment can be characterized by the number of aisles [6], the optimal density is the minimum number of waypoints required to generate tracks that traverse all the aisles in a particular layout. Accordingly, an aisle is marked as covered by a robot crossing it, if the width of the aisle is at most twice as large as the reading range of the system.


(a)


(b)


(c)



(d)

Fig. 4: Snapshots of the Rviz visualization software showing (a) the set of generated waypoints, (b) the set of waypoints after time and cost filtering (yellow circles) and the filtered ones (green circles), (c) the set of remaining waypoints after the whole filtering process (red circles) and, (d) the set of selected edges after the removal process. The map in the images has been obtained autonomously according using RFID-based exploration.

In the following experiments, r_{in} and r_{out} values will be fixed for the whole area. This is because the library can be seen as an organized grid consisting of 17 shelves arranged in two columns. However, other approaches might require using different configuration values at different regions to better represent an heterogeneous layout.

D. Edge removal

At this point, the problem can be described as a complete graph $G(V, E)$ [25], where V is the set of N vertices or nodes $\{v_i\}$ corresponding to the final set of filtered waypoints (Fig.4c), and i is the node index. E is the set of edges $\{e_{jk}\}$ connecting each node to the rest of nodes in the set, and j and k refer to the origin and goal nodes respectively. The set of all e_{jk} is a set of paths obtained using the Dijkstra' algorithm. Thus, all edges are computed to be the shortest paths from one node j to another node k.

In most IIP problems, the number of edges can be significantly reduced without affecting the efficiency of the solution. The only condition is that the final set of edges must provide full coverage of the whole environment. In other words, the $\{e_{jk}\}$ need to connect the waypoints in a way that produces pathways across all the aisles in the layout.

First, the parameter r_{max} is set so that every edge separating two nodes at a distance higher than r_{max} is discarded, thus discarding edges that pass close to another node. The optimal

r_{max} depends on the specific geometry of the obstacles in the layout and it must be either configured according to a policy or learnt using a learning algorithm. For these experiments, r_{max} = 7 m which is slightly higher than the aisle length. The remaining edges must form a planar and generally irregular grid, without coincident paths, like the one shown in Fig.4d. Finally, the edges of distances lower than r_{max} are discarded according to the following policy: whenever three nodes $\{v_i, v_j, v_k\}$ are connected through edges $\{e_{ij}, e_{jk}, e_{ki}\}$, the two nodes separated by the longest edge will be disconnected.

As a result, only the neighboring nodes (n) are left connected by edges. However, in order to include all the casuistry of real environments, such as dead ends, cycles [25] are allowed. Therefore, once the starting node is set, the complexity of the problem is

$$\mathrm{O}\left(n \cdot (n-1)^{(N-2)} \cdot \prod_{k=0}^{K} n^{(k \cdot C_k)}\right), \tag{2}$$

where $\{C_k\} = \{C_0, C_1, C_2, ..., C_K\}$ is the set whose elements contain the number of nodes that are revisited k times, and C_0 = 1 (For instance, k = 1 and C_1 = 3 means revisiting 3 nodes one time each).

E. itinerary planning

The last process consists of finding an itinerary through a planar connected dynamically weighted graph (Fig.4d) that maximizes the inventory coverage, while minimizing the overlap. To that end, all edges must have a weight value (R_{ij}), or reward, assigned to them, and a number of times it has been traversed (num_trips). Then, the inventory coverage can be approximated by the $return$ of the itinerary, which is defined as the sum of rewards of the ordered set of edges it is composed of, including repetitions.

More specifically, the goal of the itinerary planner is to obtain an open walk w=$\{v_0, e_{0a}, v_a, e_{ab}, ..., v_x\}$ that contains all graph nodes, while minimizing the repetition of edges and nodes, and maximizing the $return$. The reward for traversing an edge is initially weighted by its length $|e_{ij}|$, and it is updated, once the edge has been traversed, according to the following function:

$$R_{ij} = \begin{cases} |e_{ij}|, & \text{if } num\_trips = 0 \\ -|e_{ij}| \cdot num\_trips, & \text{if } num\_trips > 0 \end{cases} \tag{3}$$

num_trips and R_{ij} are updated simultaneously for $|e_{ij}|$ and $|e_{ji}|$ when one of them is updated. Since n is much smaller than N, the problem can now be tackled using tree search with problem specific pruning conditions (Alg.1). For instance, branches are pruned if they don't fulfill the cycle_conditions (Alg.1-L7), i.e. a node has been revisited more than K times or more than C_k nodes have been revisited more than k times.

The proposed algorithm Alg.1 is a recursive algorithm (Alg.1-L12) intended to search through a tree composed by all the possible itineraries that can be formed from expanding the graph G according to each node edges (Alg.1-L11). The starting node is set to the pose where the robot was turned on. The algorithm first tries to find an optimal itinerary without node repetitions (k = 0), and if it doesn't find a feasible solution, it progressively increases k and C_k until a solution that visits all graph nodes is found.

Algorithm 1 Tree search with selective pruning

```
class State:
    attributes:
        branch: list
        Return: float
        graph[origin][destination]:
            {length: float, reward: float,
            num_trips: int}
        best_itinerary: list
    methods:
        update()
        rollback()
        update_best_itinerary()
```

```
1:  function CONSTRAINED_SEARCH($state$, $K$, $C_k$)
2:      if IS_ITINERARY_COMPLETE($state$) then
3:          $state$.UPDATE_BEST_ITINERARY()
4:          $state$.ROLLBACK()
5:          return
6:      end if
7:      if not CYCLE_CONDITIONS($state$, $K$, $C_k$) then
8:          $state$.ROLLBACK()
9:          return
10:     end if
11:     for $node$ in EDGES($state$) do
12:         $state$.UPDATE($node$)
13:         CONSTRAINED_SEARCH($state$, $K$, $C_k$)
14:     end for
15:     if IS_TREE_FULLY_INSPECTED($state$) then
16:         SAVE($state.best\_itinerary$)
17:     else
18:         $state$.ROLLBACK()
19:         return
20:     end if
21: end function
```

The information of the current state of the search is stored in an object of the class State Alg.1. It contains the information about the $branch$ that is currently being inspected, the branch $return$, the whole $graph$ and the $best\_itinerary$ found so far. The state object maintains an up to date snapshot of the graph features, namely the edges rewards, edges distances and number of times that all edges have been traversed. The update method modifies the state attributes given the next selected node and the current edge rewards and num_trips. The rollback method reverts the last state update. And the update_best_itinerary method compares the $best\_itinerary$ return to the $branch$ return and updates the $best\_itinerary$ according to the itinerary with highest return.

Fig.5 shows the four itineraries used in these experiments, namely itinerary A corresponding to Fig.5a, itinerary B (Fig.5b), itinerary C (Fig.5c) and itinerary D (Fig.5d).

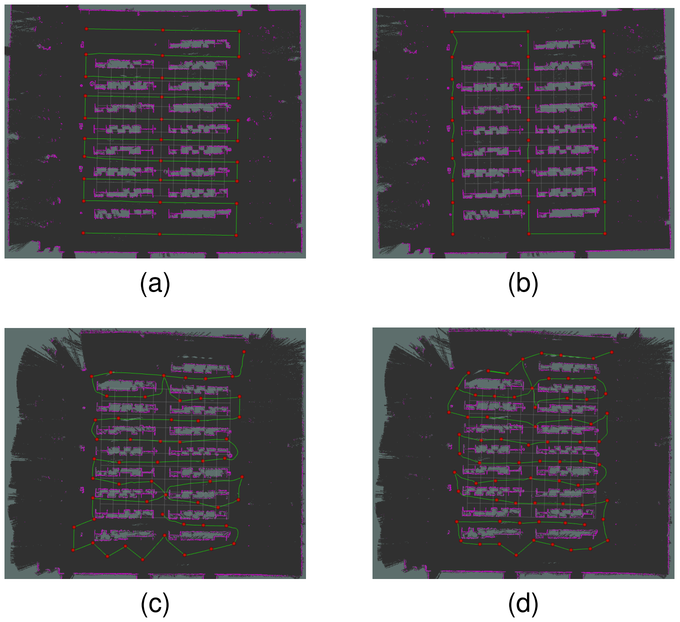


Fig. 5: Snapshots of Rviz visualization software of the tested itineraries: (a) itinerary A, (b) itinerary B, (c) itinerary C, (d) itinerary D.

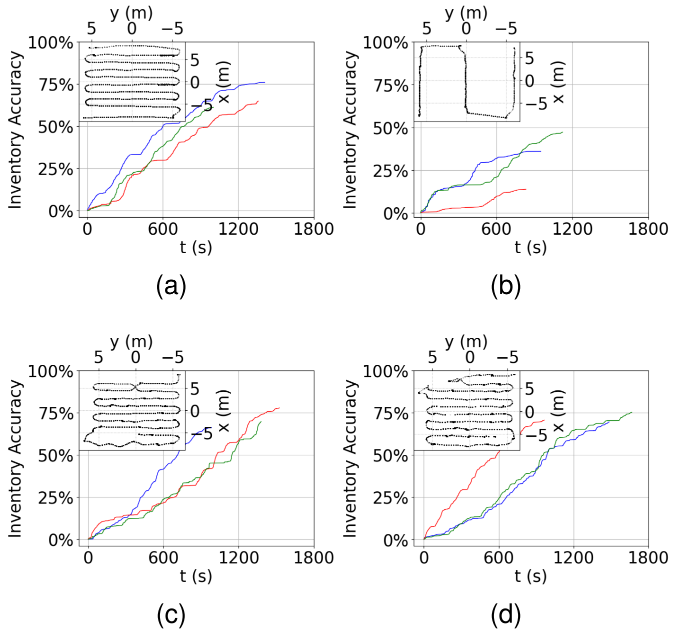


Fig. 6: Experimental results of three different robot walks through the (a) itinerary A, (b) itinerary B, (c) itinerary C, and (d) itinerary D. The robot trajectories are displayed in the top left corner of all figures.

Itineraries A and B are designed by a human operator. Itinerary A is designed with the intention of optimizing the inventory mission, i.e. maximizing the inventory accuracy and minimizing the mission time. Itinerary B is designed to achieve the shortest possible itinerary. Itinerary C and itinerary D show the results of the algorithm using two different datasets. In the case of itinerary C the autonomous navigation mission didn't travel across two aisles (left column second row and right column tenth row) during the waypoint generation stage, as we can appreciate from the image. Itinerary D traverses all aisles, while the order differs from the human-optimized itinerary.

III. Results

In this section, the performance of the proposed autonomous methodology is evaluated through real life experiments, and compared to manually designed itinerary plans. The ground truth of the library consists of 6000 books labeled with UHF RFID tags and spread around the library shelves.

The following experiments consist of unitary tests, i.e., the robot performs a one-way walk throughout the itineraries of Fig.5. This way, the accuracy results can be precisely compared while the effect of the time exposure to the RFID multi-path environment can be mitigated.

The results of the experiments are illustrated in Fig.6. These plots include the real time estimation of the robot pose at the top-left corner of the figures. The reader can appreciate that the experiments for the itinerary A (Fig.6a) and the autonomously designed itineraries C (Fig.6c) and D (Fig.6d) produce similar results in terms of mission time and inventory accuracy. As expected, the itinerary B (Fig.6b) produces lower inventory accuracy and shortest test times. However, it still obtains a considerable accuracy, especially if we take into account that the itinerary does not cross most of the aisles. The numerical data is summarized in the Table II.

The results show that, on average, the optimal itinerary (A) achieves slightly lower inventory accuracy (67.3%) in a shorter time than the autonomously designed itineraries (71.2% for itinerary C and 71.8% for itinerary D) for a one-way trip. The results are even closer for itineraries A and D, and with lower standard deviation, in the case of ten-trip realizations.

In other words, the autonomously designed itineraries obtained by the here-proposed planner achieve very similar performance when compared to the human-optimized itinerary, where the inventory mission performance is defined as the number of unique tag readings per second.

The results highlight the effect of the time that the tags are exposed to the reader signal in the inventory accuracy. In some environments, some tags can be more difficult to read than others. In these situations, the exposure of such tags to the reader interrogation signal, for a longer period of time, can favor their activation. In addition, the environment contains highly reflective objects such as the book shelves. Therefore, the multi-path effect can't be underestimated [30].

IV. Discussion

This section presents some arguments regarding the performance of the proposed method, its main advantages and drawbacks, the future research lines, and a few ideas on how to address them.

In the previous section, we saw how this method provides results comparable to the optimal itinerary that a human operator can intuitively design. This can be a complex task

TABLE II: Numerical results

itinerary	One-way accuracy (%)		One-way Itinerary time (s)		Ten-trip accuracy (%)		Average performance
	avg	std	avg	std	avg	std	($unique\_tags/s$)
A	67.3	7.7	1288	250	97.2	4.4	3.1
B	32.3	16.9	1006	126	68.1	8.9	1.8
C	71.2	6	1312	275	92.4	3.9	3.3
D	71.8	3	1427	333	97.0	3.2	3.1

for a robot and it depends strongly on the environment or the layout of the application scenario. For example, a grocery store will differ from a sports store, a library or a warehouse. The articles in grocery stores tend to be placed in shelves organized forming a grid across the available area. However, in clothes shops, while they keep some geometrical organization, the items are organized in a more artistic way and using furniture of different sizes and shapes.

The proposed solution handles this heterogeneity of the environments using the configuration parameters r_{in}, r_{out} and r_{max}. A conservative configuration of the parameter values will try to generate a slightly high number of waypoints in order to cover all the AOI, including corners and surrounding relatively small objects, by controlling the values of all three parameters. Nevertheless, these configurations can lead, in some cases, to large graphs, which in turn can increase the computation time of the best solution. The approach used in the presented experiments involves setting r_{in} to the minimum distance between aisles, setting r_{out} to be twice r_{in}, and letting r_{out} to be slightly larger than the longest aisle length. There are a number of artificial intelligence algorithms [31] that could optimize these parameters using the number of aisles and the maximum and minimum dimensions of the aisles in the environment as inputs. However, it would require a large number of realizations in many different layouts until acquiring a dataset large enough to achieve good generalization.

Another issue that requires special attention is the appearance of unexplored areas. In previous sections, we investigated the case, illustrated in Fig.6c, of a itinerary that didn't navigate all the aisles of the AOI. While this fact didn't affect the performance of the presented experiments, unexplored areas lead to suboptimal itinerary solutions, which in turn can decrease the inventory accuracy (see itinerary C in Table II). The main reasons leading to this issue are: the incomplete navigation of the AOI during the autonomous RFID-based navigation and the configuration of the waypoint filtering and the edge selection algorithms. In the last paragraph, we have already discussed the problem of finding the proper configuration of the waypoint selection stages. As for the incomplete navigation of the AOI, the RFID attraction based navigation can get stuck in local maxima or just not get enough attraction to reach areas of local or global minima. While this problem is meant to be addressed in future works, adding the data from several missions tends to solve this issue in the vast majority of cases.

Providing full scalability to a fully autonomous solution for robotic inventory systems will require the development of multi-robot itinerary planners. Although a methodology like the one presented in this work can be applied, this problem requires an additional step that considers the distribution of the AOI among the members of the robot fleet. The areas could be divided using tag information matched to a database with the item details, or using a geometrical approach to sectorize the area in clusters.

V. Conclusions

This paper has presented the first autonomous waypoint generation method based on RFID exploration. The proposed method provides a set of wapypoints that represents the AOI layout to efficiently draw aisles. All in all, the presented methodology enables fully autonomous itinerary planning for inventory robots. It has been proved that, the presented methodology solves the adaptability and scalability issues of the current commercial solutions by enabling the autonomous installation and re-configuration of robotic inventory systems. The presented solution has been tested in the installation of an autonomous inventory robot in a real library. The results have shown that autonomously designed itineraries reach similar inventory accuracy compared to those designed by humans.

Acknowledgment

This project has been co-financed by the Spanish Government trough the Ministry of Science, Innovation and Universities under RF-VOLUTION project (PID2021-122247OB-I00).

References

[1] V. Garg and R. Agrawal, *Transforming Management Using Artificial Intelligence Techniques.* (Eds.). CRC Press, 1st ed., 2020.

[2] M. Gareis, A. Parr, J. Trabert, T. Mehner, M. Vossiek, and C. Carlowitz, "Stocktaking robots, automatic inventory, and 3d product maps: The smart warehouse enabled by uhf-rfid synthetic aperture localization techniques," *IEEE Microwave Magazine*, vol. 22, no. 3, pp. 57–68, 2021.

[3] A. Motroni, F. Bernardini, S. Vaiani, A. Buffi, and P. Nepa, "Performance assessment of a uhf-rfid robotic inventory system for industry 4.0," in *2022 16th European Conference on Antennas and Propagation (EuCAP)*, pp. 1–5, 2022.

[4] P. Donepudi, "Robots in retail marketing: A timely opportunity," *Global Disclosure of Economics and Business*, vol. 9, pp. 97–106, 11 2020.

[5] V. Casamayor-Pujol, B. Gaston, S. Lopez-Soriano, A. A. Alajami, and R. Pous, "A simple solution to locate groups of items in large retail stores using an rfid robot," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 2, pp. 767–775, 2022.

[6] B. Gaston, V. Casamayor-Pujol, S. Lopez-Soriano, and R. Pous, "A metric for assessing, comparing, and predicting the performance of autonomous rfid-based inventory robots for retail," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 10, pp. 10354–10362, 2022.

[7] G. Fragapane, R. de Koster, F. Sgarbossa, and J. O. Strandhagen, “Planning and control of autonomous mobile robots for intralogistics: Literature review and research agenda,” *European Journal of Operational Research*, vol. 294, no. 2, pp. 405–426, 2021.

[8] B. Patle, G. Babu L, A. Pandey, D. Parhi, and A. Jagadeesh, “A review: On path planning strategies for navigation of mobile robot,” *Defence Technology*, vol. 15, no. 4, pp. 582–606, 2019.

[9] J. R. Sánchez-Ibáñez, C. J. Pérez-del Pulgar, and A. García-Cerezo, “Path planning for autonomous mobile robots: A review,” *Sensors*, vol. 21, no. 23, 2021.

[10] M. Hank and M. Haddad, “A hybrid approach for autonomous navigation of mobile robots in partially-known environments,” *Robotics and Autonomous Systems*, vol. 86, pp. 113–127, 2016.

[11] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, “Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring,” *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1453–1468, 2021.

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

[13] S. Lee, C.-Y. Lee, W. Jo, and D.-H. Kim, “An efficient area coverage algorithm using passive rfid system,” in *2014 IEEE Sensors Applications Symposium (SAS)*, pp. 366–371, 2014.

[14] J. Zhang, Y. Lyu, T. Roppel, J. Patton, and C. P. Senthilkumar, “Mobile robot for retail inventory using rfid,” in *2016 IEEE International Conference on Industrial Technology (ICIT)*, pp. 101–106, 2016.

[15] F.-A. Moreno, J. Monroy, J.-R. Ruiz-Sarmiento, C. Galindo, and J. Gonzalez-Jimenez, “Automatic waypoint generation to improve robot navigation through narrow spaces,” *Sensors*, vol. 20, no. 1, 2020.

[16] A. Kamalova, K. D. Kim, and S. G. Lee, “Waypoint mobile robot exploration based on biologically inspired algorithms,” *IEEE Access*, vol. 8, pp. 190342–190355, 2020.

[17] S. Lin, A. Liu, J. Wang, and X. Kong, “A review of path-planning approaches for multiple mobile robots,” *Machines*, vol. 10, no. 9, 2022.

[18] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, “A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms,” *IEEE Access*, vol. 9, pp. 119310–119342, 2021.

[19] A. A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, “Hpprm: Hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots,” *IEEE Access*, vol. 8, pp. 221743–221766, 2020.

[20] J. Janoš, V. Vonásek, and R. Pěnička, “Multi-goal path planning using multiple random trees,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4201–4208, 2021.

[21] M. Morenza-Cinos and V. Casamayor-Pujol, “Rfid location dataset,” Nov. 2018.

[22] S. López-Soriano and R. Pous, “Inventory robots: Performance evaluation of an rfid-based navigation strategy,” *IEEE Sensors Journal*, pp. 1–1, 2023.

[23] V. Casamayor-Pujol, M. Morenza-Cinos, B. Gaston, and R. Pous, “Autonomous stock counting based on a stigmergic algorithm for multi-robot systems,” *Computers in Industry*, vol. 122, p. 103259, 2020.

[24] E. Dijkstra, “Performance assessment of a novel miniaturized rfid tag for inventorying and tracking metallic tools,” *A note on two problems in connexion with graphs.*, vol. 1, p. 269–271, 1959.

[25] R. Diestel, *Graph Theory*, vol. 173 of *Graduate Texts in Mathematics*. Heidelberg; New York: Springer, fourth ed., 2010.

[26] “Library of the universitat pompeu fabra,” https://www.upf.edu/es/web/biblioteca-informatica/poblenou, Last Accessed: 2022-08-23.

[27] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[28] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005.

[29] “Advanreader 160,” Available at https://wiki.keonn.com/rfid-components/advanreader-160, Last Accessed: 2022-08-23.

[30] 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, no. 10, pp. 8250–8259, 2018.

[31] K. P. Murphy, *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013.

Lopez-Soriano S. received the M.Sc. degree in Micro and Nanoelectronics Engineering and the Ph.D. degree in Electronics and Telecommunication Engineering from the Universitat Autonoma de Barcelona (UAB), Bellaterra, Spain, in 2013 and 2018 respectively. From 2019 to 2022, he enrolled the Ubiquitous computing Lab at the Universitat Pompeu Fabra as an associate researcher in robotics. He is currently an Associate Researcher with the WiNe group at the Universitat Oberta de Catalunya, and he holds the position of Associate Lecturer at the Universitat Autonoma de Catalunya. His current research interests include autonomous robotic inventory systems, RFID sensing systems and RFID tag antenna design.