Speeding-Up Robot Exploration by Exploiting Background Information

Stefan Oßwald, Maren Bennewitz, Wolfram Burgard, and Cyrill Stachniss

***Abstract*—The ability to autonomously learn a model of an environment is an important capability of a mobile robot. In this paper, we investigate the problem of exploring a scene given background information in form of a topo-metric graph of the environment. Our method is relevant for several real-world applications in which the rough structure of the environment is known beforehand. We present an approach that exploits such background information and enables a robot to cover the environment with its sensors faster compared to a greedy exploration system without this information. We implemented our exploration system in ROS and evaluated it in different environments. As the experimental results demonstrate, our proposed method significantly reduces the overall trajectory length needed to cover the environment with the robot's sensors and thus yields a more efficient exploration strategy compared to state-of-the-art greedy exploration, if the additional information is available.**

***Index Terms*—Mapping, motion and path planning, exploration.**

I. Introduction

ROBOTS that are able to acquire a map of their surroundings on their own fulfill a major precondition of truly autonomous mobile vehicles. Assuming that a robot has means for solving the underlying SLAM problem, i.e., they can learn a map and track their pose in this map given sensor data, the exploration task is to *generate motion commands* that guide the robot to build a complete model of the environment.

Typical approaches to mobile robot exploration assume zero prior knowledge about the world. Accordingly, the robot starts with an empty map and seeks to find a sequence of motion commands to cover the full environment with its sensors. Only few approaches consider additional background knowledge, such as semantic information [1] or information retrieved from dialog with a human user [2].

In this paper, we take a different approach to exploration than the majority of existing methods. We investigate means that allow a robot to *explore an environment faster if additional information is available*. We seek to answer the question: How much faster can a robot cover an environment with its sensors if it knows the rough layout of the environment beforehand? This problem is relevant for a large number of real-world exploration problems, as in several application scenarios the approximate layout of the environment is known beforehand. This holds, for example, when exploring underground structures such as abandoned mines [3], archaeologically relevant tunnels such as ancient catacombs [4], but also for inspection tasks such as mapping visually intact but possibly unstable buildings after an earthquake [5]. Thus, we do *not* address the problem of exploring a completely unknown environment in this paper. Instead, we provide an efficient exploration strategy given prior information about the layout of the environment.

Our work relies on a topo-metric map. This can be seen as a simplified Voronoi-style graph modeling the environment. Fig. 1a shows an example of such a user-provided graph. Using this information, our approach seeks to find an efficient exploration strategy (see Fig. 1b) to cover the scene with the robot's proximity sensor as fast as possible. Such a topo-metric graph can be provided by humans or can be automatically derived from floor plans, from previously built maps, or from hand-drawn maps. By knowing the topology of the environment including an estimate of the metric distances, the robot can generate more effective exploration trajectories that avoid redundant work. The robot typically explores dead ends, small loops, and similar structures first so that it will not have to return to these locations later during the mission. In the environment in Fig. 1, for example, it is advantageous to explore the rooms on the outside first, as the robot then has to traverse the corridor only once. We formulate the problem as a traveling salesman problem and use its solution to guide the robot through the environment. The length of the path that our approach generates (d) is significantly shorter than the one resulting from greedy exploration (c) because it avoids traversing the corridor multiple times. Finally, our approach combines the TSP solution with frontier-guided exploration. This combination avoids redundant work on a global scale and at the same time ensures that all areas are covered by the robot's sensors. In addition to that, exploiting such a graph structure also allows operators to easily exclude parts of the environment that the robot should not explore by simply labeling nodes or edges in the graph as "no go areas". As we will show in our experimental evaluation, our approach leads to significantly shorter overall exploration trajectories and thus significantly reduces the time needed to cover the terrain.

II. Related Work

Typical approaches to mobile robot exploration aim at selecting view points that minimize the time needed to cover the

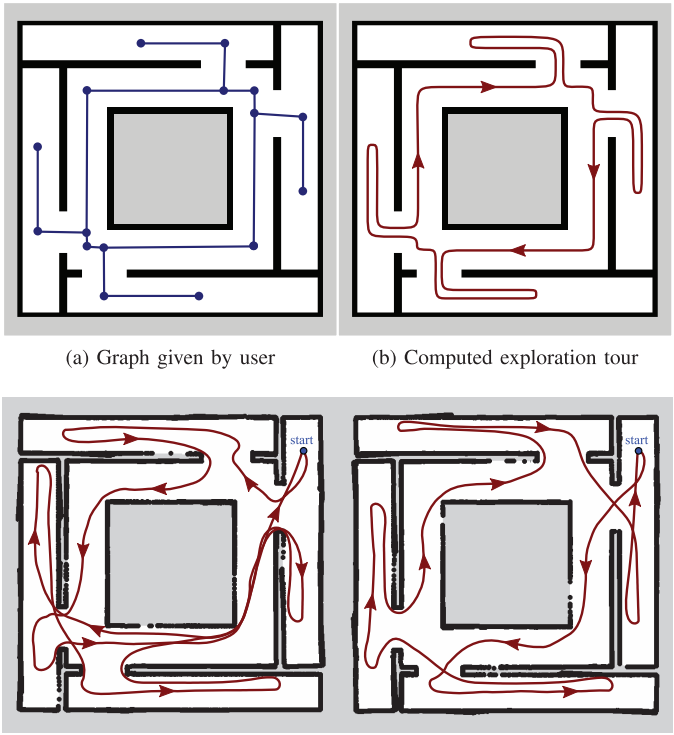
Manuscript received August 28, 2015; accepted January 6, 2016. Date of publication January 22, 2016; date of current version February 29, 2016. This paper was recommended for publication by Associate Editor J. Buchli and Editor D. Lee upon evaluation of the reviewers' comments. This work was supported in part by the German Research Foundation under contract number SFB/TR-8 and in part by the European Commission under contract number FP7-ICT-600890-ROVINA.

S. Oßwald and M. Bennewitz are with the Institute of Computer Science, University of Bonn, Bonn 53113, Germany (e-mail: sosswald@cs.uni-bonn.de; maren@cs.uni-bonn.de).

W. Burgard is with the Department of Computer Science, University of Freiburg, Freiburg 79110, Germany (e-mail: burgard@cs.uni-freiburg.de).

C. Stachniss is with the Institute of Geodesy and Geoinformation, University of Bonn, Bonn 53113, Germany (e-mail: cyrill.stachniss@igg.uni-bonn.de).

Digital Object Identifier 10.1109/LRA.2016.2520560



(a) Graph given by user (b) Computed exploration tour

(c) Greedy exploration (d) Our approach

Fig. 1. Based on the graph given by the user (a), our algorithm computes an exploration tour for visiting the nodes using a traveling salesman problem solver (b). While the greedy nearest-first exploration scheme has to revisit rooms (c), our approach exploits the background knowledge by using the computed exploration tour as a guideline for minimizing the overall traveled distance (d).

whole terrain. The most popular approach is probably frontier-based exploration by Yamauchi [6] in which the robots always move towards the closest unexplored location. Another option is using stochastic differential equations for goal-directed exploration [7]. The idea is to seed particles in the workspace and subject them to forces so that they move toward unexplored areas. The particles themselves follow molecular dynamics and eventually enter unknown areas and provide candidate goal positions for the exploring robot. A usual assumption in exploration systems is the knowledge about the pose of the robot during the mission, but there exist also exceptions such as information-theoretic methods that seek to minimize the uncertainty in the belief about the map and the trajectory [8], [9] simultaneously. In a similar way, Sim *et al.* [10] select vantage points to optimize the map accuracy by striving for loop closures.

The majority of exploration approaches start from scratch, i.e., they do not exploit background information or any assumptions about the structure of the environment. There are only a few techniques that incorporate background information into the decision-making process of where to move next. An interesting approach by Fox *et al.* [11] aims at incorporating knowledge about other environments into a cooperative mapping and exploration system for multiple robots. Related to that, Perea *et al.* [12] exploit already explored spaces to predict future loop-closures and to guide the exploration in this way. Others use semantic information [13], [1] or an environment segmentation [14] as background knowledge to optimize the target location assignment. Such approaches received considerable attention in multi-robot exploration. In addition to that, Zlot *et al.* [15] propose a multi-robot target assignment architecture that follows the ideas of a market economy. The assignment considers sequences of potential target locations and trades individual tasks among the robots using single-item first-price sealed-bid auctions. Similar approaches using auctions for task allocation processes have been applied by Gerkey and Matarić [16]. In contrast to that, Ko *et al.* [17] present an approach that uses the Hungarian method [18] to compute the optimal assignments of open frontier cells to robots in a given instance. The work of Ko *et al.* furthermore focuses on aligning the robot trajectories in case the start locations of the robots are unknown. Wurm *et al.* [14] proposes a coordinating technique for teams of robots using a segmentation of the environment to determine exploration targets for the individual robots. This is related to the spatial semantic hierarchy introduced by Kuipers and Byun [19]. Thus, semantic information [1] or segmentations [14] approaches show that by assigning robots to unexplored segments instead of frontier targets, a more balanced distribution of the robots over the environment is obtained. A related approach has been presented in the work by Holz *et al.* [20], which first analyses different exploration approaches and then proposes heuristics for improving the performance of the exploration strategies.

In this paper, we investigate means for allowing robots to explore an environment faster if additional information is available. We target application scenarios in which the approximate layout of the environment to explore is known beforehand. This is also related to coverage techniques [21], [22] but we do not assume a grid map or polygon to be given beforehand for planning view points. In contrast to purely graph-based coverage techniques as in [23], we also consider the surroundings around the graph nodes in a local exploration strategy. Patrolling approaches such as [24], [25] focus on multi-robot coordination and global exploration strategies, while our approach combines global with local strategies. Exploiting abstracted map information has also been investigated in other navigation problems. For example, Chronis and Skubic [26] exploit hand-drawn sketch maps for navigation, which enables robots to derive and use qualitative spatial relations to move along the given path. Related to that, Freksa *et al.* [27] use schematic maps to derive qualitative spatial relations for supporting navigation. Our work also exploits topo-metric information in form of a graph but the user is not expected to specify the path for the robot. Instead, the environment information is used for computing an optimal exploration strategy at a global scale once in the beginning, which is then used for visiting the individual areas in the environment. Our approach is also related to works by Brummit and Stentz [28] as well as Faigl *et al.* [29] as both approaches formulate the problem of coordinating multiple robots during exploration as a multiple traveling salesman problem (MTSP). Our approach also uses the TSP formulation but on the user-provided graph and combines the solution with local exploration at the individual locations.

III. Guiding Exploration Based on Background Knowledge

In this section, we describe our approach to generating navigation actions for acquiring sensor data about the complete environment. We hereby exploit background knowledge in form of a graph-structure provided to the robot. We model the global navigation problem as a traveling salesman problem in order to find a tour that covers the whole area and is as short as possible. Our approach solves the TSP once in the beginning and uses the TSP tour to guide a frontier-based exploration. In case the robot detects changes in the environment, it re-computes the TSP solution.

A. Representation of the Background Information About the Environment

We represent the background knowledge as a graph $G(V, E)$ consisting of nodes $V = \{v_1, \ldots, v_n\}$ and undirected edges $E \subseteq V \times V$. The nodes represent rooms or other convex regions in the environment and the edges indicate the connections in between, for example doors or passages. We hereby assume that each room to be explored contains at least one node and corridors contain a node in front of each door and intersection (note that the terms "room", "corridor", "intersection", etc. are only used for illustrative purposes in this paper, the algorithm itself does not contain any notion of these concepts). The lengths of the edges need to reflect only roughly the true metric distances (see Sec. IV-E for robustness against noise). The resulting graph is a topo-metric representation of the environment and is used to optimize the exploration tour with the starting position and orientation of the robot with respect to the graph given by the user.

In order to guide the exploration, the user may wish to explicitly exclude regions from the exploration. We represent the user's instructions by annotating the nodes of the graph with labels $A : V \rightarrow \{explore, skip\}$ indicating whether the region close to the node should be explored or not.

B. Representation of the Exploration Problem

The robot's objective is to optimize the tour length given the user-provided graph. The tour must visit each node at least once in order to cover the surrounding region. In most cases, there will be rooms that the robot has to pass through more than once, for example, the robot will have to enter the corridor multiple times to access the individual rooms. Hence, our goal is to find the shortest path T in the graph G that visits all nodes at least once and returns to the start node. This problem is closely related to the well-known traveling salesman problem (TSP): Given a list of cities and the distances between each pair of cities, find the shortest tour that visits all cities exactly once and finally returns to the starting point. A TSP solver is able to provide the optimal solution to this problem.

To represent this problem as a traveling salesman problem, we complete the graph G to a clique by adding edges between each pair of nodes. We define the length of the new edge (v_i, v_j) as the length of the shortest path between the positions of nodes v_i and v_j and use Johnson's algorithm [30], [31] to compute the shortest path distances between all pairs of nodes in time $\mathcal{O}(|V|^2 \log|V| + |V||E|)$.

The resulting graph G' is symmetric and the triangle inequality holds, as the length of newly inserted edges between two nodes is never longer than the distance of the shortest path between these nodes. G' is a clique, thus there exists a shortest tour T' in G' that visits each node exactly once and returns to the start node. The shortest tour T in the original graph G that visits all nodes at least once cannot be shorter than T' due to the triangle inequality. The tour T with revisiting nodes can be easily retrieved from T' by replacing the edges added using Johnson's algorithm by the shortest path between the edge's end nodes.

In some scenarios, it might not be required that the robot returns to the starting location after completing the exploration task. In this case, the robot only requires a shortest path from the given start node v_s that visits all nodes and leads to an arbitrary end node. This case can also be modeled as a TSP by setting the length of the edges from v_i to v_s to zero for all $i \in \{1, \ldots, n\}$, so the robot can "teleport" from an arbitrary node back to the start node. After determining the shortest tour, the zero-length edge (v_e, v_s) is removed so that the remaining path leads from v_s to the arbitrary end node v_e without returning. Note that the graph modified with the zero-cost edges is not symmetric and consequently not metric anymore.

C. Solving the TSP

Unfortunately, solving the TSP is NP-complete in the general case. For metric TSPs, however, there exist polynomial-time approximation schemes. For example, the Christofides algorithm [32] is able to find a tour that is guaranteed to be at most 1.5 times longer than the optimal solution and requires a run time of $\mathcal{O}(|V|^3)$.

The problem instances we consider, however, typically generate TSP instances of moderate complexity. State-of-the-art TSP solvers such as Concorde [33] can determine the exact solution to such problems within a few seconds. As our algorithm has to solve a TSP only once at the beginning of the mission, this step only adds a marginal amount to the overall run time in comparison to the time the robot needs to actually move through the environment and acquire sensor data. Thus, we use the Concorde solver in our implementation.

D. Frontier-Based Exploration Exploiting the TSP Solution

The tour determined by the TSP solver provides a *global strategy* for visiting all regions of the environment by ordering the corresponding nodes. Although the tour is the optimal solution to the TSP, the resulting trajectory is likely to be a suboptimal exploration trajectory for a real robot. The reason for that is the fact that the TSP formulation does not consider visibility information of the robot's sensor. For example, parts of the environment may be visible from multiple nodes, so that the robot only has to visit one of the nodes. At the same time, the exploration system must ensure that the surroundings of a node in the TSP is fully explored.

Therefore, we combine the TSP and the problem of exploring the *surroundings of each node* taking into account the sensing capabilities of the platform. To this end, we use a frontier-based exploration approach [6] for the local exploration but using the TSP solution on the global scale. In order to incorporate the sensor information into the world model, we rely on a standard graph-based SLAM system for 2D range sensing. It builds a grid map while moving through the environment and acquiring sensor data. The robot's grid map consisting of cells that are initially labeled as unknown. While traveling, the robot updates the cells within its field of view towards free or occupied with a standard occupancy model. Cells on the border between free and unknown space that have not been covered yet are called *frontier cells*. The robot can extend its knowledge about how the environment looks like by navigating to frontier cells in order to inspect the cells laying beyond the frontier. During this exploration, multiple frontiers appear and the robot has to decide to which frontier to move next. A common approach for choosing a frontier is to apply a cost function to the frontiers that takes into account cost factors such as the distance between the robot's current pose r and the frontier position f, the relative orientation, and the expected information gain (IG):

$$\begin{aligned}\text{cost}(f) := {} & \alpha_1 \cdot \text{distance}\,(r, f) \\ & + \alpha_2 \cdot \text{orientation}\,(r, f) \\ & - \alpha_3 \cdot \text{IG}\,(f)\end{aligned} \tag{1}$$

For applications which require additional or more complex cost terms, in particular probabilistically dependent or synergic cost terms, a multi-criteria decision making framework [34] could be used to design a suitable cost function. The robot updates the map based on the perceptions, computes the frontiers, and decides to navigate to the frontier with the lowest cost at a fixed frequency of 1 Hz.

In order to account for the global navigation strategy determined by the TSP tour, we add another cost factor to the cost function. Whenever a new frontier appears, we determine the nearest node $v^\star$ using Dijkstra's algorithm on the grid map, see Fig. 2 for an illustration of the assignment process.

The robot may explore the frontier at any time when passing by that node. When the robot passes by $v^\star$ for the last time while following the global tour computed by the TSP, however, our strategy enforces that the robot explores the frontier in order to avoid having to return to the node another time, as that would increase the tour length unnecessarily. Hence, we follow the TSP tour from the robot's original start node v_s up to the point when the TSP tour passes through $v^\star$ for the last time:

$$p := \left((v_s, v_{k_1}), (v_{k_1}, v_{k_2}), \ldots, (v_{k_m}, v^\star)\right) \tag{2}$$

We then sum up the lengths of the edges along the tour

$$d := \sum_{(v_i, v_j) \in p} \text{length}\left((v_i, v_j)\right) \tag{3}$$

and add this distance d to the cost function in Eq. (1) with a high weight α_4 so that it dominates the cost function. In this way, the robot explores the frontiers in the order determined by the solution of the TSP and applies the original cost function within each room, as d is the same for all frontiers within the neighborhood of the nearest node.

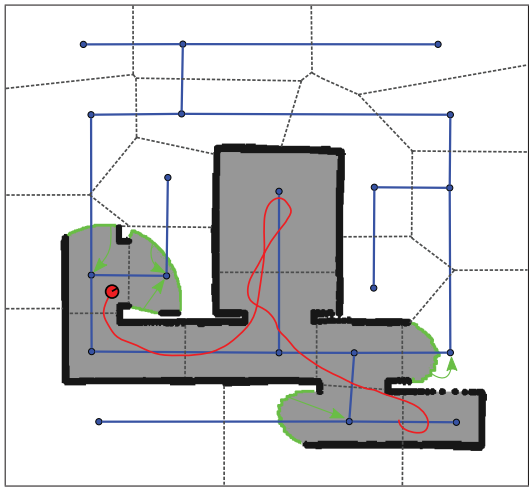


Fig. 2. The graph is used to guide the exploration and this requires an assignment of frontiers to graph nodes. The algorithm assigns each new frontier (green) to the nearest graph node (green arrow) by considering the length of the shortest path from the frontier to any node of the graph. The areas for the assignment are illustrated through the dashed lines and walls of the map. In unexplored areas, only the graph can be used to estimate the assignment. While the explored region grows, the assignment of a frontier to a node can change due to newly sensed walls. The thickness of the frontiers has been increased artificially in this figure to improve visibility.

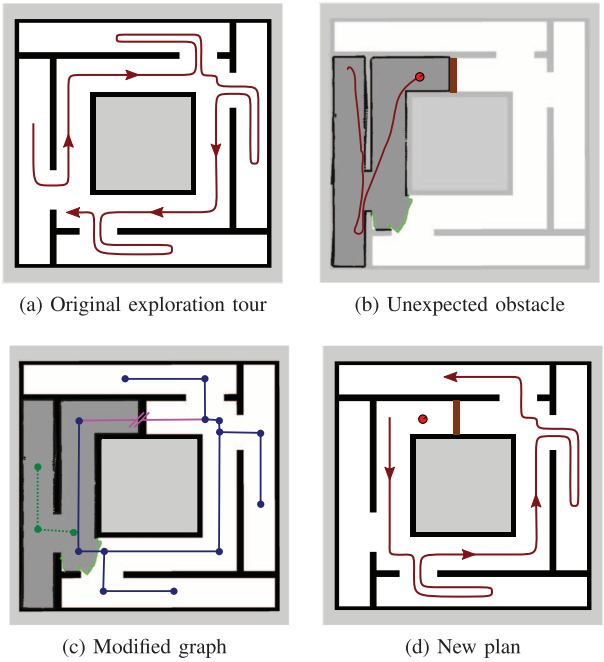
During exploration, the robot might encounter frontiers that are unreachable because of obstacles. If the robot's path planner determines that it cannot navigate to a frontier, the robots adds this frontier to a blacklist. If the user has annotated nodes to be skipped during exploration, the algorithm adds all frontiers associated with those nodes to the blacklist. When the association of frontiers to nodes changes, the algorithm modifies the blacklist accordingly.

E. Replanning

Our approach relies on the assumption that the user-provided graph correctly represents the topology of the environment. If the graph contains non-traversable edges, the resulting global strategy will be suboptimal as the robot would make a detour around the obstacle and continue to explore the frontiers in the order as planned originally. To avoid such detours, the robot has to replan the global strategy once it detects that the planned path is obstructed:

1) Correct the user's graph according to the observations.
2) Complete the graph to a clique G' as explained above.
3) Remove the nodes that have already been visited and do not have any frontiers associated with them, except for node v_s to which the robot has to return after exploring (if applicable) and the robot's current node v_r.

4a) If the robot has to return to the original start node v_s: After exploring the last remaining node, the robot has to return to v_s instead of the current node v_r. Hence, we replace the edge lengths (v_i, v_r) for all $i \in \{1, \ldots, n\}$ by the shortest path costs between the positions of nodes v_i and v_s, making the TSP asymmetric.



(a) Original exploration tour (b) Unexpected obstacle

(c) Modified graph (d) New plan

Fig. 3. Based on the graph given by the user, the algorithm computes an exploration tour, in this case without returning to the start location (a). While traveling, the robot encounters an unexpected obstacle blocking the planned path (b). The algorithm modifies the graph by removing the blocked edge (magenta) and already visited nodes without open frontiers (green) (c). Solving the TSP on the new graph then yields a global strategy for exploring the remaining parts of the environment (d).

4b) If the robot does not have to return to the start node: Set the edge lengths of (v_i, v_r) to zero for all $i \in \{1, \ldots, n\}$ to allow the path to end at an arbitrary node as explained in Sec. III-B.

5) Solve the TSP to get a tour T' for the remaining nodes.

6) Replace the edges added by Johnson's algorithm by the shortest paths, passing previously visited nodes if necessary.

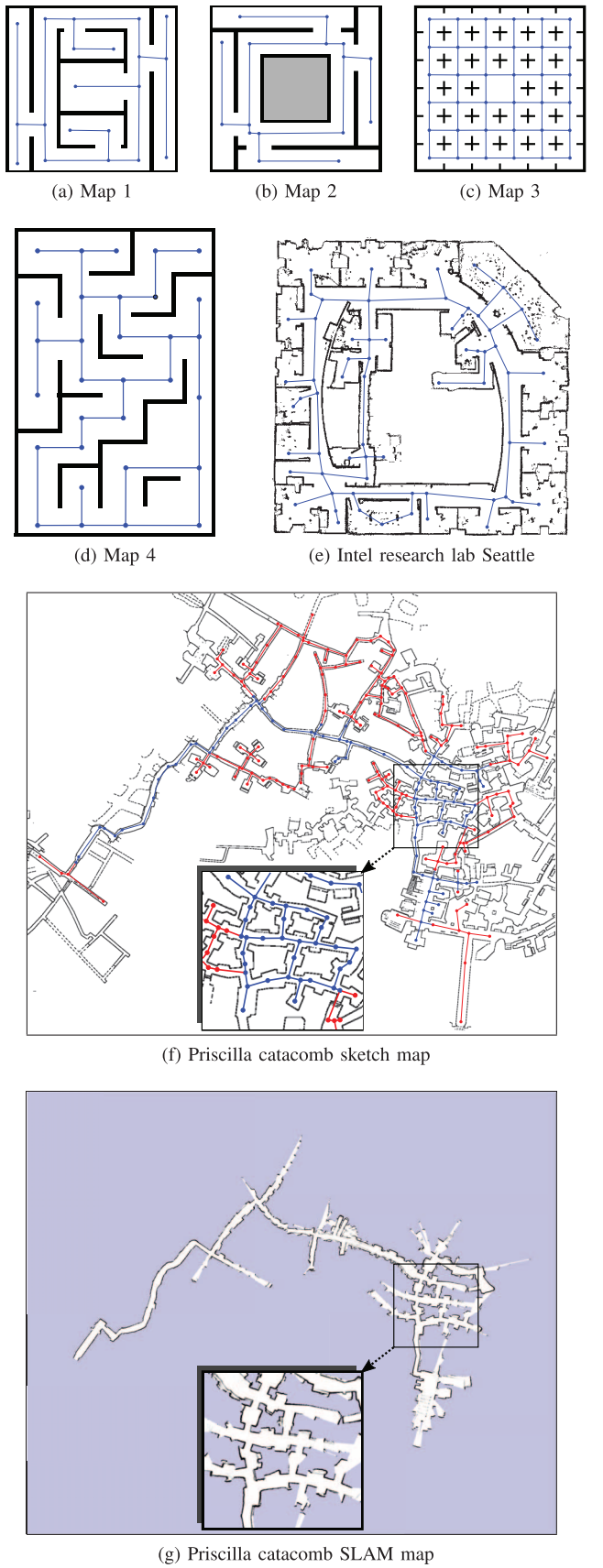
The resulting tour lets the robot explore the remaining areas on the shortest path assuming that the rest of the topology is correct. Fig. 3 shows an example of a replanning step.

IV. Experimental Evaluation

For testing our approach, we implemented our system in ROS [35] and simulated the exploration in Player/Stage [36]. We based our exploration implementation on the ROS exploration stack [37] that already implements frontier-based exploration. We keep the default cost function, except for setting the weight for the expected information gain to zero (see Sec. IV-F for a discussion). This ROS greedy exploration strategy serves as a baseline for the evaluation of our approach.

A. Environments

For our evaluation, we use a set of different maps, containing artificial maps, real maps recorded with mobile robots, and maps created from hand drawings of archaeological sites. See Fig. 4 for an overview of the maps used in the experiments.



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

(d) Map 4 (e) Intel research lab Seattle

(f) Priscilla catacomb sketch map

(g) Priscilla catacomb SLAM map

Fig. 4. Maps and graphs used in the experiments. (a)–(d): Artificial maps. (e): Interior of the Intel research lab in Seattle, courtesy of D. Fox. (f): Map drawn by archaeologists of the Priscilla Catacomb in Rome, Italy. (g): Corresponding map built by a real robot.

Maps 1–4 are artificial maps for studying the behavior of our approach in environments with different levels of connectedness. We furthermore used a map of the Intel research lab in Seattle (Fig. 4e). This map contains a high amount of clutter, which poses a challenge to exploration algorithms in general.

Finally, we evaluated our approach using the data from a real-world application for digitizing cultural heritage sites. Within the ROVINA project, we plan to digitize the Catacomb of Priscilla in Rome, Italy [4]. During previous expeditions, archaeologists created hand-drawn maps of the catacombs. Based on these maps, we created a graph of the catacomb's topology (see Fig. 4f).

We then used this information as background knowledge for our exploration system. As the access to the catacomb is quite restrictive, we used a map (see Fig. 4g) built in the Catacomb of Priscilla during a previous robotic mission with the SLAM system. In this previous mission, the robot was joysticked through the environment and we used the map from this run for the simulation, so that multiple exploration runs can be simulated and evaluated with statistical tests. Please note that the SLAM map deviates from the sketch map in some parts. For example, some corridors that are straight on the hand-drawn map appear curved on the SLAM map. Our experiments show that our approach compensates for such inaccuracies as long as the frontiers can be assigned correctly to graph nodes.

B. Background Information

We created the graphs resembling the background information manually by placing one node per room and connecting the nodes according to the map topology. In general, one node per convex region is sufficient, as the greedy exploration then dominates the exploration strategy in the convex regions, for which it is well-suited. If the user places more than one node in a convex region, the global tour strategy gains influence, giving the user more control on the exploration strategy.

With our approach, the human operator can easily limit the exploration area by marking nodes that the robot should not explore. In the Priscilla catacomb shown in Fig. 4f, for example, we marked the nodes that should be explored in blue and the nodes to be skipped in red. The robot then deliberately leaves frontiers open that get assigned to the red parts of the graph, so the robot only explores the desired region.

Depending on the application scenario, the robot may be required to return to its starting position after completing the exploration. For the Priscilla map, we run experiments both with returning to the starting location and without.

C. Traveled Distance

The distance that a robot has to travel to explore the whole environment depends on the start location. Hence, we chose a set of start locations for each map and executed our approach and the baseline approach once for each start location.

Fig. 5 visualizes the mean length of the exploration trajectory together with the corresponding 95% confidence intervals. Tab. I shows the results of a paired t-test at the 0.05 level for the exploration tour length for 10 respectively 5 runs in the

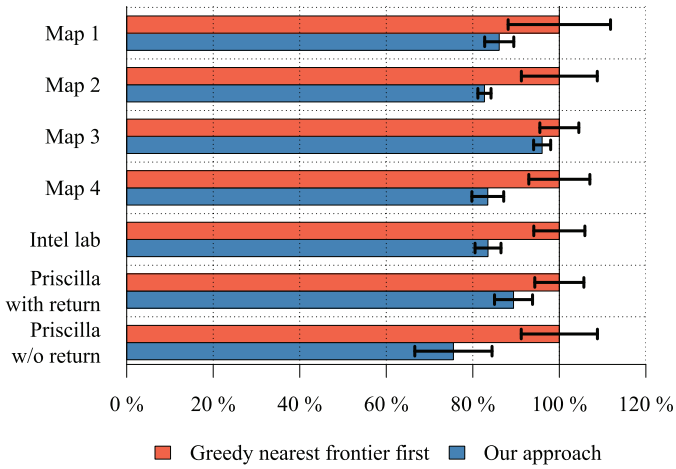


Fig. 5. Mean and 95% confidence interval of the traveled distance. The distances are normalized so that the greedy approach equals 100%.

TABLE I
COMPARISON OF TRAVELED DISTANCE

Map	Runs	Mean difference	Gain	p value
Map 1	10	40.6 m	13.9%	0.019
Map 2	10	50.1 m	17.3%	0.002
Map 3	10	9.0 m	4.0%	0.051
Map 4	10	30.6 m	15.2%	0.004
Intel lab	10	164.5 m	16.5%	0.001
Priscilla with return	5	49.3 m	10.6%	0.020
Priscilla without return	5	101.8 m	24.5%	0.004

environments. The trajectories generated by our approach are significantly shorter than the trajectories generated by the baseline approach on all maps except Map 3. On Map 3, the greedy strategy is already near-optimal in many cases, as the map is similar to a large freespace area.

For the Priscilla map, we run experiments both with returning to the starting location and without. Our approach uses the knowledge about whether or not to return to the starting location to optimize the exploration tour as described in Sec. III-B. In case the robot has to return to the starting location, the TSP tour generated by our approach is independent of the starting location of the robot. The robot explores the rooms always in the same sequence, hence the variance of the exploration tour length is small. In the baseline approach of greedy exploration, by contrast, the order in which the robot explores the rooms strongly depends on the starting location, leading to a higher variance in the tour length. In case the robot does not have to return to the start, the overall tour length depends on the start location, leading to a higher variance in the tour length for both approaches when averaged over different starting locations.

As can be seen in Tab. I, our approach outperforms frontier-based exploration in all settings. The gain in exploration time is achieved by better planning the exploration. For example, on the Priscilla map when the robot does not have to return to the starting location, our approach always explores the corridor system in the left half of the map last (or first in case the robot starts already in the corridor) so that the robot does not have to traverse the corridor system for a second time. On a local scale, our approach enforces that the robot finishes exploring a room

TABLE II
TIME REQUIRED FOR SOLVING THE TSP

Map	Nodes to explore	Time required to solve TSP
Maps 1–4	16–36	(0.013 ± 0.006) s
Intel lab	65	(0.84 ± 0.43) s
Priscilla catacomb with return	94	(5.92 ± 2.57) s
Priscilla catacomb without return	94	(0.38 ± 0.11) s

completely before moving on to the next room, such that the robot will not have to enter the room again.

D. Runtime

For computing the global exploration strategy, our approach has to solve a traveling salesman problem once at the beginning of the exploration. In our experiments, we use the Concorde solver [33], which attempts to find the optimal solution of the TSP using branch-and-cut strategies. Tab. II shows the time required to solve the TSP for the individual maps. As the TSP has to be solved only once in the beginning (or when changing the graph leads to replanning), the required time adds only marginally to the total exploration time.

During exploration, our algorithm needs to maintain a distance information for assigning each frontier to the nearest graph node (see the illustration in Fig. 2). For this purpose, a low resolution map is sufficient. As this shortest path information changes whenever new walls appear during the exploration, we recompute it at a frequency of 1 Hz at a grid resolution of 15 cm, which can easily be done online during the exploration.

E. Robustness

To evaluate the robustness of our approach, we added Gaussian noise with increasing standard deviation to the node positions of the user-provided graphs shown in Fig. 4, which consequently also changes the edge lengths, and evaluated the effect on the overall tour length. Fig. 6 shows that our approach still works even if the graph provided by the user is considerably inaccurate. For Gaussian noise with a standard deviation of $\sigma = 1$ m as shown in the top of the figure, our approach still performs significantly better than the greedy approach according to a paired t-test at the 0.05 level. This robustness is mainly achieved by dynamically updating the assignment between frontiers and graph nodes using Dijkstra's algorithm as explained in Sec. III-D.

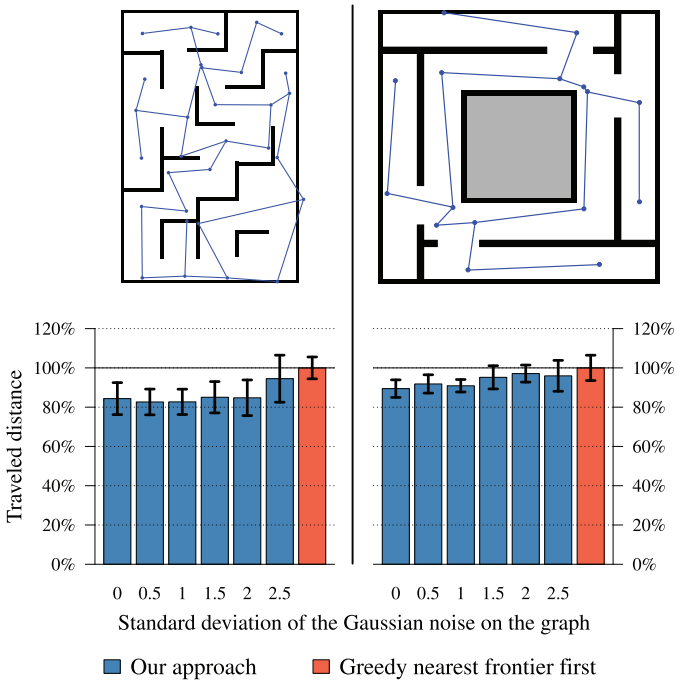


Fig. 6. Robustness of the exploration strategy when Gaussian noise is added to the human-provided graph. The figure shows the experiments for Map 4 on the left and Map 2 on the right (see Fig. 4 for the original graphs provided by the human user). Top: Gaussian noise of $\sigma = 1$\;m added to the original graph. Bottom: Mean and 95% confidence interval of the traveled distance for different amounts of Gaussian noise ($n = 10$ for each condition). The distances are normalized so that the greedy approach equals 100% for the respective map.

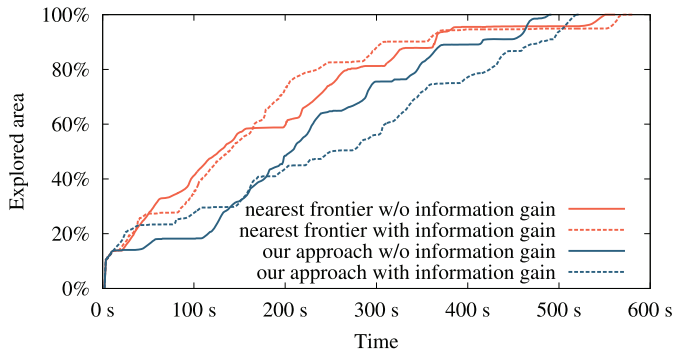


Fig. 7. Comparison of the exploration progress for different strategies for a run on map 4 shown in Fig. 4d. While the nearest frontier approach makes faster progress at first, our approach explores more thoroughly and avoids having to backtrack, thus our approach finishes in a shorter total time.

F. Influence of Information Gain

Many exploration approaches use the estimated information gain in the cost function for selecting the next goal. The rationale of this approach is to greedily explore the largest frontiers first in order to cover largest possible area within a given time limit. However, this strategy tends to leave smaller frontiers unexplored, which would be counterproductive in our scenario as the robot would have to come back later to complete the exploration.

Fig. 7 illustrates how fast the presented approaches make progress during exploration. The nearest frontier strategy explores greedily, while our approach explores thoroughly. Consequently, the greedy approach covers more area per time frame than our approach in the beginning. After exploring the biggest frontiers, however, the greedy approach has to backtrack to explore the smaller areas and rooms that it left out, which is time-consuming and causes the total exploration time to be longer than with our approach. Considering the information gain in the cost function increases this effect even further, leading to longer overall exploration times.

Our goal is to explore a map completely without having a fixed time limit after which the mission is stopped. Hence, our

approach without considering the information gain is the best choice for covering the complete environment.

V. Conclusions

We presented a novel approach to autonomously learning a model of the environment under the assumption that the layout of the environment is known beforehand in form of a topo-metric graph. Such a graph can be provided by humans or automatically derived from existing floor plans or other maps. Our system exploits the given topo-metric graph to generate shorter navigation trajectories that cover the terrain. We represent the problem as a traveling salesman problem and derive a global exploration strategy from its solution. Locally, a frontier-based approach is used that exploits the TSP solution and fully covers the environment with the robot's sensors. Accordingly, the overall trajectory length is reduced as the robot will move to dead ends, small loops, and similar structures first and thus does not need to return to these locations later on. We implemented and thoroughly tested our approach in different environments. The results show that our approach significantly reduces the time needed to cover the terrain with the robot's sensors requiring a negligible amount of additional computational resources. Our approach is valuable for several real-world exploration problems, for example, when exploring abandoned mines or archaeological sites, or for inspection tasks where the layout of the environment is not completely unknown.

References

[1] C. Stachniss, O. Martinez Mozos, and W. Burgard, "Efficient exploration of unknown indoor environments using a team of mobile robots," *Ann. Math. Artif. Intell.*, vol. 52, pp. 205–227, 2009.

[2] T. Fong, C. Thorpe, and C. Baur, "Collaboration, dialogue, human-robot interaction," in *Robotics Research*, R. A. Jarvis and A. Zelinsky, Eds.New York, NY, USA: Springer, 2003, vol. 6.

[3] S. Thrun *et al.*, "Autonomous exploration and mapping of abandoned mines," *IEEE Robot. Autom. Mag.*, vol. 11, no. 4, pp. 79–91, Dec. 2004.

[4] G. Grisetti, L. Iocchi, B. Leibe, V. Ziparo, and C. Stachniss, "Digitization of inaccessible archeological sites with autonomous mobile robots," in *Proc. Conf. Robot. Innov. Cultural Heritage*, 2012, p. 8.

[5] G.-J. Kruijff *et al.*, "Rescue robots at earthquake-hit Mirandola, Italy: A field report," in *Proc. IEEE Int. Symp. Safety Sec. Rescue Robot.*, 2012, pp. 1–8.

[6] B. Yamauchi, "Frontier-based exploration using multiple robots," in *Proc. Int. Conf. Auton. Agents*, 1998, pp. 47–53.

[7] S. Shen, N. Michael, and V. Kumar, "3D indoor exploration with a computationally constrained MAV," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2012, pp. 4976–4982.

[8] C. Stachniss, G. Grisetti, and W. Burgard, "Information gain-based exploration using Rao-Blackwellized particle filters," in *Proc. Robot. Sci. Syst. (RSS)*, 2005, pp. 65–72.

[9] A. Makarenko, S. Williams, F. Bourgoult, and F. Durrant-Whyte, "An experiment in integrated exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2002, pp. 534–539.

[10] R. Sim and N. Roy, "Global A-optimal robot exploration in SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Barcelona, Spain, 2005, pp. 661–666.

[11] D. Fox, J. Ko, K. Konolige, and B. Stewart, "A hierarchical Bayesian approach to the revisiting problem in mobile robot map building," in *Proc. Int. Symp. Robot. Res. (ISRR)*, 2005, vol. 15, pp. 60–69.

[12] D. Perea Ström,, F. Nenci, and C. Stachniss, "Predictive exploration considering previously mapped environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2015, pp. 2761–2766.

[13] C. Stachniss, O. Martínez-Mozos, and W. Burgard, "Speeding-up multi-robot exploration by considering semantic place information," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Orlando, FL, USA, 2006, pp. 1692–1697.

[14] K. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2008, 1160–1165.

[15] R. Zlot, A. Stenz, M. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Washington, DC, USA, 2002, vol. 3, pp. 3016–3023.

[16] B. Gerkey and M. Matarić, "Sold! Auction methods for multirobot coordination," *IEEE Trans. Robot. Autom.*, vol. 18, no. 5, pp. 758–768, Oct. 2002.

[17] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, "A practical, decision-theoretic approach to multi-robot mapping and exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Las Vegas, NV, USA, 2003, vol. 4, pp. 3232–3238.

[18] H. Kuhn, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart.*, vol. 2, no. 1, pp. 83–97, 1955.

[19] B. Kuipers and Y.-T. Byun, "A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations," *J. Robot. Auton. Syst.*, vol. 8, pp. 47–63, 1991.

[20] D. Holz, N. Basilico, F. Amigoni, and S. Behnke, "A comparative evaluation of exploration strategies and heuristics to improve them," in *Proc. Eur. Conf. Mobile Robots (ECMR)*, 2011, pp. 25–30.

[21] A. Xu, C. Viriyasuthee, and I. Rekleitis, "Efficient complete coverage of a known arbitrary environment with applications to aerial operations," *Auton. Robots*, vol. 36, no. 4, pp. 365–381, 2014.

[22] R. Mannadiar and I. Rekleitis, "Optimal coverage of a known arbitrary environment," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 5525–5530.

[23] L. Xu and A. Stentz, "A fast traversal heuristic and optimal algorithm for effective environmental coverage," in *Proc. Robot. Sci. Syst. (RSS)*, 2010, pp. 161–168.

[24] F. Pasqualetti, A. Franchi, and F. Bullo, "On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms," *IEEE Trans. Robot.*, vol. 28, no. 3, pp. 592–606, Jun. 2012.

[25] D. Portugal, R. Rocha, "A survey on multi-robot patrolling algorithms," in *Technological Innovation for Sustainability*, L. M. Camarinha-Matos, Ed. New york, NY, USA: Springer, 2011, vol. 349.

[26] G. Chronis and M. Skubic, "Sketch-based navigation for mobile robots," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ)*, 2003, pp. 284–289.

[27] C. Freksa, R. Moratz, and T. Barkowsky, "Schematic maps for robot navigation," in *Spatial Cognition II, Integrating Abstract Theories, Empirical Studies, Formal Methods, and Practical Applications*. Berlin, Germany: Springer-Verlag, 2000, pp. 100–114.

[28] B. Brummit and A. Stentz, "GRAMMPS: A generalized mission planner for multiple mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1998, pp. 1564–1571.

[29] J. Faigl, M. Kulich, and L. Preucil, "Goal assignment using distance cost in multi-robot exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2012, pp. 3741–3746.

[30] J. G. Siek, L.-Q. Lee, and A. Lumsdaine, *The Boost Graph Library: User Guide and Reference Manual*. Reading, MA, USA: Addison-Wesley, 2002.

[31] D. B. Johnson, "Efficient algorithms for shortest paths in sparse networks," *ACM*, vol. 24, no. 1, pp. 1–13, 1977.

[32] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Graduate School of Industrial Administration, Carnegie Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. 388, 1976.

[33] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, "TSP cuts which do not conform to the template paradigm," in *Computational Combinatorial Optimization*, M. Jünger and D. Naddef, Eds. New York, NY, USA: Springer, 2001, vol. 2241, pp. 261–303.

[34] N. Basilico and F. Amigoni, "Exploration strategies based on multi-criteria decision making for searching environments in rescue operations," *Auton. Robots*, vol. 31, no. 4, pp. 401–417, 2011.

[35] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, 2009, pp. 1–6.

[36] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. Int. Conf. Adv. Robot. (ICAR)*, 2003, pp. 317–323.

[37] C. DuHadway (2012). *The ROS Exploration Stack* [Online]. Available: http://wiki.ros.org/exploration