

Searching for Optimal Off-Line Exploration Paths in Grid Environments for a Robot with Limited Visibility

Alberto Quattrini Li and Francesco Amigoni

Politecnico di Milano
Piazza Leonardo da Vinci 32
20133 Milano, Italy

Nicola Basilico

University of California, Merced
5200 North Lake Rd
Merced, CA 95343, USA

Abstract

Robotic exploration is an on-line problem in which autonomous mobile robots incrementally discover and map the physical structure of initially unknown environments. Usually, the performance of exploration strategies used to decide where to go next is not compared against the optimal performance obtainable in the test environments, because the latter is generally unknown. In this paper, we present a method to calculate an approximation of the optimal (shortest) exploration path in an arbitrary environment. We consider a mobile robot with limited visibility, discretize a two-dimensional environment with a regular grid, and formulate a search problem for finding the optimal exploration path in the grid, which is solved using A*. Experimental results show the viability of our approach for realistically large environments and its potential for better assessing the performance of on-line exploration strategies.

Robotic exploration for map building is a fundamental task in which autonomous mobile robots use their onboard sensors to incrementally discover the physical structure of initially unknown environments (Thrun 2002). The mainstream approach follows a Next Best View (NBV) process, i.e., a repeated greedy selection of the *next* best observation location, according to an *exploration strategy* (Stachniss and Burgard 2003; González-Baños and Latombe 2002; Tovar et al. 2006; Basilico and Amigoni 2011). At each step, a NBV system considers a number of candidate locations on the frontier between the known free space and the unexplored part of the environment, evaluates them using a utility function, and selects the best one. Current experimental evaluation of exploration strategies is almost exclusively based on relative comparisons between their performance in some test environments (Amigoni 2008; Lee and Recce 1997). As a consequence, it is difficult to assess how much room for improvement on-line exploration strategies have. A more complete evaluation should involve an *absolute* comparison between the performance of (on-line) exploration strategies and the optimal (off-line) performance in the test environments, based on the competitive ratio. In general, the *competitive ratio* of an on-line algorithm a is $\frac{P_a}{P_o}$, where P_a is the performance of a in an environment and P_o is the per-

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

formance of the optimal off-line algorithm that knows the environment in advance (Ghosh and Kleinl 2010). To allow calculating the competitive ratio for on-line exploration strategies used in autonomous mobile robotics, in this paper we address the problem of finding the optimal exploration path for test environments, when the optimality criterion is the travelled distance.

The problem of finding a shortest continuous exploration tour (a closed path starting and ending at the same point) for arbitrary two-dimensional polygonal environments has been shown to be NP-hard even in the case the robot has time-continuous perceptions (Arkin, Fekete, and Mitchell 2000). In this paper, we provide a method for calculating an approximation of the shortest continuous exploration path for mapping a given environment. More precisely, we consider a single robot with a limited range sensor moving in an arbitrary two-dimensional environment and performing time-discrete perceptions (i.e., at discrete points along a path). We assume to know the environment and we discretize it by using a two-dimensional fine-grained regular grid. Then, we formulate a search problem to calculate the shortest discrete exploration path in the grid, which can be solved using A*. Extensive experimental activities show the viability of our approach for realistically large environments. We also introduce some speedup techniques that reduce the computational time required to find the shortest exploration path in a grid, slightly penalizing the solution quality. With the availability of the optimal exploration paths, we show how to calculate the competitive ratio for the on-line exploration strategies considered in (Amigoni 2008).

Related Work

The problem of calculating an exploration path is an instance of the *area coverage problem*, in which a robot equipped with a covering tool with limited range has to completely cover an unknown planar environment (Choset 2001). As said, finding an optimal off-line area covering tour for the robot (i.e., a closed path that returns to the starting point such that every non-obstacle point of the environment is covered) is NP-hard for arbitrary polygonal environments (Arkin, Fekete, and Mitchell 2000). Hence, a number of approximation algorithms have been developed. For example, Arkin, Fekete, and Mitchell (2000) propose an algorithm that constructs a tour of length at most 2.5 times the length of the

optimal tour in a time $O(n \log n)$, where n is the number of edges of the polygonal environment. The provably complete coverage methods that approximate the environment with cells of same size and shape reported in (Choset 2001) are not guaranteed to produce *shortest* coverage paths, as for example the wavefront propagation method of (Zelinsky et al. 1993). The method in (Gabriely and Rimon 2001) uses an optimality criterion that is not related to the length of the path but to avoid repetitive coverage. Moreover, for the above methods, the cell size is the sensor footprint and the robot has time-continuous perceptions, so they cannot be applied to the problem we consider in this paper.

Finding an optimal covering tour reduces to finding an *optimal watchman tour* when the sensors of the robot have infinite range. The optimal watchman tour is the shortest closed path inside a polygon P such that every point of P is visible from some point along the path. Also finding the optimal watchman tour is NP-hard for general polygons (Urrutia 2000). However, some algorithms can solve the optimal watchman tour problem in polynomial time for simple polygons (Chin and Ntafos 1991). In addition to infinite visibility, the above algorithms address the problem of finding optimal watchman *tours*, while in our problem we are looking for optimal exploration *paths*.

To the best of our knowledge, we are not aware of any algorithm that solves the problem of finding shortest off-line exploration paths in grid environments for a robot with limited and time-discrete visibility.

Optimal Exploration Problem

Assumptions and Problem Statement

We assume a single autonomous mobile robot moving in an arbitrary two-dimensional environment. The environment is represented by a finite grid, whose cells are identical squares. Each cell can be either free or occupied (by obstacles). Hence, the grid E representing the environment is partitioned in sets of cells E_f and E_o containing the free and the occupied cells, respectively. The free space E_f and the obstacles E_o can have any form. In the following, with a slight notation overload, we use the same symbol q to indicate both the cell $q \in E$ and the position of its center in a global reference frame. Since we are interested in calculating the optimal exploration path, we assume that the environment is static and completely known in advance, so we can formulate an off-line problem.

The robot is considered as a point (this assumption is without loss of generality if obstacles are “grown” to account for the real size of the robot, as usual in path planning (LaValle 2006)). The robot starts in the center of a free cell and its basic movements are from the center of its current cell to the center of another free cell. We assume that the grid is 8-connected. The robot is equipped with a 360° range sensor with a finite range r . With such a sensor, we can ignore the orientation of the robot and consider only its position on the grid. We consider a laser range finder sensor that perceives the state of any cell whose center can be connected to the position of the robot with a straight line segment of maximum length r and crossing only free

cells (without passing between occupied cells that share a vertex). We assume that the perception of the robot is discrete: the robot perceives the surrounding environment and updates the map only when in the next position and not continuously while moving (time-discrete perception is often assumed by on-line exploration algorithms (Amigoni 2008; Amigoni and Caglioti 2010; González-Baños and Latombe 2002; Tovar et al. 2006; Tovey and Koenig 2003)). More precisely, the robot operates according to the following steps: (a) it perceives the surrounding environment, (b) it integrates the perceived data within a map representing the environment known so far, (c) it reaches the next position and starts again from (a). Since we are interested in optimal exploration, we assume that the movements and the perceptions of the robot are error-free (i.e., deterministic). As a consequence, the robot perfectly knows its position in the environment.

The problem we address in this paper is the following. Given an environment represented by a grid E , given a robot with a laser range finder sensor with range r , and given an initial position q_0 for the robot in E , find an optimal sequence of positions (centers of cells) $Q = \langle q_0, q_1, \dots, q_n \rangle$ the robot should reach such that every free cell $q \in E_f$ is perceived by the robot from at least a position $q_i \in Q$. The *optimal exploration path* for the robot is the sequence Q that minimizes the *travelled distance*, namely the quantity $\sum_{i=0,1,\dots,n-1} d(q_i, q_{i+1})$, where $d(q_i, q_{i+1})$ is the length of the shortest path lying in E_f connecting q_i to q_{i+1} .

Formulation of the Search Problem

To solve the optimal exploration problem, we formulate a corresponding search problem, following a classical approach in Artificial Intelligence (Russell and Norvig 2010, Chapter 3). A state s in our search problem formulation is a pair (q, M) composed of the current position q of the robot in the environment and the map $M \subseteq E$ built so far during exploration.

- Initial state. The initial state $s_0 = (q_0, M_0)$ is represented by the initial position of the robot q_0 in the environment E and by the initial map M_0 , which contains the cells of E perceived from q_0 .
- Actions. From a state $s = (q, M)$, applicable actions for the robot are to move to free cells $q' \in M$ reachable from q and perceive the environment surrounding q' . A free cell q' is reachable from q when there is a safe path (within M and not colliding with any obstacle) between q and q' can be found. The path is calculated using a wavefront propagation algorithm on M (LaValle 2006, Chapter 8), considering cost 1 for vertical and horizontal movements and cost $\sqrt{2}$ for diagonal movements (recall that we consider 8-connected grids). In principle, from a state $s = (q, M)$, there are as many actions as many reachable free cells q' in the current map M . However, to limit the number of these actions (and the branching factor of the search tree used to calculate a solution), we consider only reachable free cells q' that are on the boundary between known and unknown parts of the map M . This assumption can affect optimality, for example at the end of a corridor, where

the robot does not need to reach the boundary to perceive the remaining part of the environment. An open issue is finding a non-trivial bound on the penalty on optimality introduced by this assumption.

- **Transition function.** The new state resulting from performing applicable action “move to q' ” in state $s = (q, M)$ is $s' = (q', M')$, where M' is the map M updated with the new perception in q' .
- **Goal test.** A state $s = (q, M)$ is a goal state when M is a complete map of the free space of the environment E , namely when all the free cells of E_f are present in M . (Interior of obstacles are invisible to the robot and cannot be considered to detect termination.)
- **Step cost.** The step cost for going from a state $s = (q, M)$ to a successor state $s' = (q', M')$ is $c_d = d(q, q')$.

A solution to the above search problem is a finite sequence of states $S = \langle s_0, s_1, \dots, s_n \rangle$ such that s_0 is the initial state and s_n is a state that satisfies the goal test. An optimal solution is a solution with minimum cost. From a solution $S = \langle s_0 = (q_0, M_0), s_1 = (q_1, M_1), \dots, s_n = (q_n, M_n) \rangle$ of the search problem it is trivial to extract a solution $Q = \langle q_0, q_1, \dots, q_n \rangle$ to the optimal exploration problem.

It is worth explicitly noting that searching for an optimal exploration path is different from searching for an optimal path between two points in an unknown environment (e.g., using algorithms like D* (LaValle 2006, Chapter 12), Learning Real-Time A* (Russell and Norvig 2010, Chapter 4), and PHA* (Felner et al. 2004)). First, in our problem, we don't know *a priori* the position of the robot at the end of exploration. Hence, we cannot operate in a state space in which each state is a position (cell) of the robot in the environment, but we need a more complex representation of state that accounts also for the portion of the environment discovered so far. Second, our approach is off-line and the robot should not physically move between positions (states).

Solution of the Search Problem

In principle, we can use any search algorithm to solve the problem formulated in the previous section. By some preliminary experiments, we obtained that A* performs slightly better than branch and bound on the environments of Figure 1 and so, although this issue deserves further investigation, we decided to use A* in our experimental activity. In order to apply informed strategies like A*, we need an heuristic function that, given a state s , returns the estimated cost of a solution from s . For the travelled distance, heuristic function $h_d(s)$, with $s = (q, M)$, is calculated as the difference between the distance of the farthest unexplored free cell of E and the range of the sensor (multiplied by $\sqrt{2}$ to account for the fact that all the area of a cell, including its diagonal, is perceived at once):

$$h_d(s) = \max_{q' \in E_f} d(q, q') - r\sqrt{2}. \quad (1)$$

The idea is that, in order to completely map the free space of environment E , the robot has (at least) to perceive the farthest cell of E_f . It is easy to show that the above heuristic

function is admissible and that, as a consequence, solving the search problem with A* guarantees to find an optimal solution (Russell and Norvig 2010, Chapter 3).

The worst-case computational complexity of our A*-based approach is exponential in the number of perceptions needed to completely map an environment (i.e., in n). However, as the results of the next section show, our approach can find optimal exploration paths for realistically large environments in reasonable time. In the attempt to improve the efficiency of our approach, we introduce a number of speedup techniques that are expected to reduce the computational effort, at the expense of worsening the quality (length) of solutions. Their effectiveness will be evaluated in the experimental activity.

Footprint sensor. The laser range finder sensor model can be substituted by a (less realistic) footprint sensor that perceives the state (free or occupied) of any cell whose center lies within the circle centred in the robot with radius r .

Weaker goal test. We can consider a weaker version of the goal test for which a state $s = (q, M)$ is a goal state when M contains a fraction G of the cells in E_f . In our experiments, we will use $G = 0.85, 0.90, 0.95$. (This is of interest for rescue applications for which knowing the general structure of the environment would suffice.) Of course, if $G = 1.00$, the weaker goal test is equivalent to the original goal test. With a weaker goal test, the heuristic function (1) becomes:

$$h_d(s) = \left(\max_{q' \in E_f} d(q, q') - r\sqrt{2} \right) \cdot \max \left(0, G - \frac{|M|}{|E_f|} \right) \quad (2)$$

Clustering boundary cells. Adjacent boundary cells in the current map M are grouped in clusters, according to the 8-adjacency map of the grid. Then, for each cluster, the cell belonging to the cluster that is closest to the centroid of the cluster is selected as representative of the cluster. More precisely, for a cluster C with h boundary cells p_1, p_2, \dots, p_h the representative cell c is selected as

$$c = \arg \min_{p_i \in C} d \left(p_i, \frac{\sum_{i=1,2,\dots,h} p_i}{h} \right),$$

where the sum is over the vector representation of positions p_i . In this way, only cells representative of clusters of boundary cells in M are considered for generating actions in a state $s = (q, M)$, drastically reducing the number of these actions and the branching factor of the search tree.

Eliminating small clusters. When applying clustering of boundary cells, we can avoid to consider small clusters, namely clusters that contain less than k boundary cells. The idea behind eliminating small clusters is that their contribution to the exploration of the environment is small. If small clusters are eliminated, then the search algorithm could be prevented from finding any solution when some free cells of E are visible only by reaching a small cluster.

Experimental Activity

Extensive simulated experiments have been conducted in the three environments, called indoor, openspace, and ob-

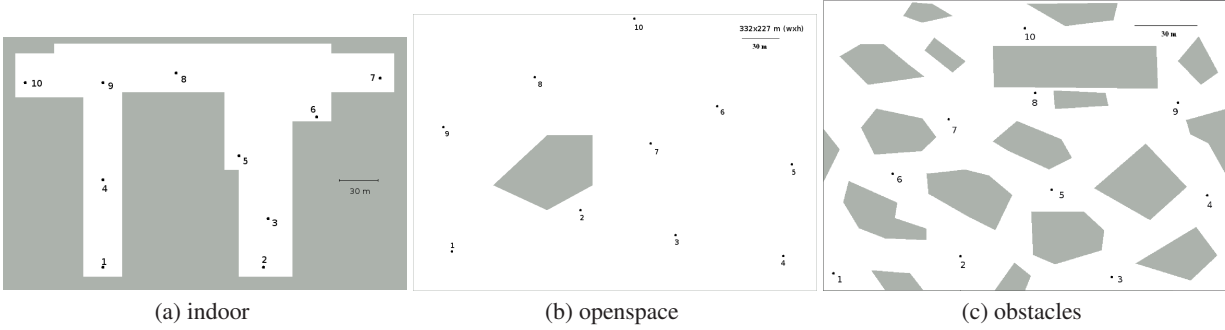


Figure 1: The three environments (points represent different initial positions for the robot)

	CELL SIZE	DISTANCE		
		$r = 20$	$r = 25$	$r = 30$
$G = 85\%$	1	674.8 (56.0) ^{*(2)}	575.4 (49.6)	503.1 (49.4)
	2	670.3 (50.4)	586.3 (44.6)	494.7 (43.6)
	4	657.9 (55.9)	571.3 (38.2)	522.8 (47.2)
$G = 90\%$	1	762.8 (45.2) ^{*(2)}	636.5 (53.4)	570.5 (57.2)
	2	751.2 (51.1) ^{*(1)}	666.0 (51.4)	560.4 (50.4)
	4	744.1 (48.8)	638.7 (44.0)	585.9 (46.1)
$G = 95\%$	1	831.4 (56.3) ^{*(6)}	715.3 (54.8)	648.5 (57.4)
	2	820.9 (47.7) ^{*(2)}	746.3 (65.1)	633.8 (59.3)
	4	861.7 (76.7) ^{*(2)}	729.6 (52.8)	663.1 (46.3)

Table 1: Results (average and standard deviation) for the indoor environment (^{*(#)}: # of runs terminated due to the timeout)

stacles, reported in Figure 1 (boundaries of environments are considered as obstacles), which are the same considered in (Amigoni 2008). The line segment reported in the figure measures 30 units, so the size of the environments is approximately 350 unit \times 250 unit. If we consider a unit equivalent to 0.1 m (which is reasonable, given the cell size and the sensor range values discussed below), we can say that the size of environments is realistically large. The environments have been discretized in square grids, using three different resolutions. We considered three cell sizes, corresponding to edges of 1, 2, and 4 units, from the highest to the lowest resolution. We considered three values for the sensor range r , namely 20, 25, and 30 units, and three values for the weaker goal test G , namely 0.85, 0.90, and 0.95. For an environment, we call setting a combination of cell size, r , and G . For each setting and for each initial position (shown as points in Figure 1), we ran our approach to find the optimal exploration path (implemented in C++) using c_d as cost function. We set a timeout of 5 hours for each run. For the runs that found a solution within the timeout, we measured the travelled distance (length) of the solution.

Table 1 reports experimental results for the indoor environment. In all experiments we considered the footprint sensor and the clustering of boundary cells. The values reported in each entry are the average and the standard deviation (in parentheses) over the 10 initial positions for the corresponding setting. We also report the number of runs that have not terminated within the timeout. An example of a generated optimal exploration path is shown in Figure 2.

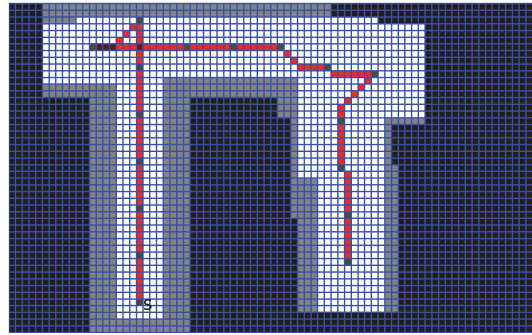


Figure 2: Optimal exploration path for the initial position 1 of the indoor environment (cell size = 4, $G = 0.85$, and $r = 30$), black cells are unknown, light grey cells are obstacles, white cells are free, dark grey cells are the positions from where the robot perceives the environment, and red cells are the path

From Table 1, it emerges that the travelled distance decreases when the sensor range r increases. Unsurprisingly, a robot with a wider sensor can explore the environment more efficiently. Another expected behavior is that the travelled distance increases when the robot is required to explore an increasingly larger fraction G of the environment. The variation of the quality of the solution with respect to the cell size does not show any strong pattern. This can be explained by noting that, when changing the cell size, the positions of the cell centers change in the environment, changing the possible discrete paths the robot can follow and the set of cells the robot perceives for a given r .

Computational time for finding solutions varies greatly with the setting. For example, finding the optimal exploration path requires an average time of 0.33 seconds for cell size = 4, $r = 30$, and $G = 0.85$, while 6 out of 10 runs do not terminate within 5 hours and the remaining 4 runs terminate in 4132.30 seconds on average for cell size = 1, $r = 20$, and $G = 0.95$ (on a computer equipped with a 1.60 GHz i7-720QM processor and 8 GB RAM). In general, computational time increases when cell size decreases, r decreases, and G increases (data are not shown here due to space constraints). Figure 3 shows that the computational

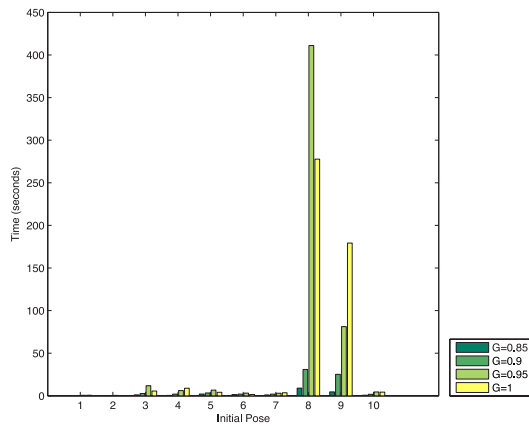


Figure 3: Computational time to find the optimal exploration path for different initial positions and G (cell size = 4 and $r = 25$)

time required for finding the optimal exploration path highly depends on the initial position in the environment (results are similar for other settings). In particular, position 8 is in the middle of the top corridor of the indoor environment and, from there, the search for the optimal exploration path is expensive because it has to follow two main branches, corresponding to going first right and then left, or vice versa. On the other hand, position 1 is at the bottom of the left vertical corridor and searching for an optimal exploration path from there basically amounts to perform a “focused” depth-first search, which is very fast.

Figure 3 reports also the time for $G = 1.00$. Note that the heuristic function (1) used for $G = 1.00$ dominates (is not smaller than) the heuristic function (2) used for $G < 1.00$. Being both admissible, a basic property of A* (namely, dominant heuristic functions never expand more nodes than the dominated ones) explains why the computation time for initial position 8 is larger for $G = 0.95$ than for $G = 1.00$. For the same setting of Figure 3, Figure 4 shows that the travelled distance of the optimal solution grows almost linearly with G , up to $G = 1.00$. Moreover, the travelled distance required to explore the indoor environment is about the same independently of initial positions (as also evidenced by the small standard deviations of Table 1).

We now experimentally evaluate the impact of the speedup techniques introduced in the previous section. In all the above experiments, we have used the footprint sensor. As expected, using the more realistic laser range finder sensor increases the computational time (see Figure 5 for an example relative to a setting). However, rather surprisingly, the solution quality obtained with the less realistic footprint sensor is very similar to that obtained with the more realistic sensor model, as shown in Figure 6. While, in principle, using the footprint sensor can cause a large difference in cost with respect to the laser range finder sensor (e.g., with two narrow parallel corridors, when moving along one of them, the footprint sensor could perceive also the other one), this does not happen in our test environments. These results pro-

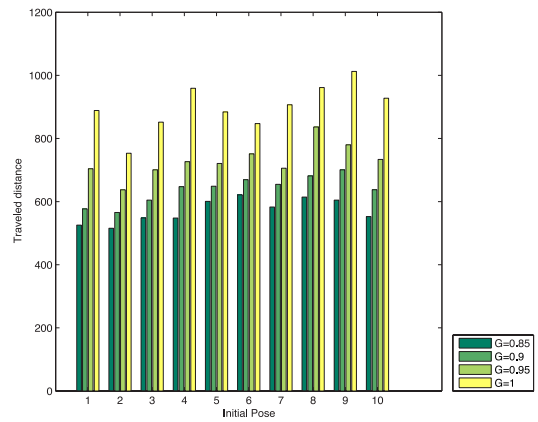


Figure 4: Optimal travelled distance for different initial positions and G (setting is the same of Figure 3)

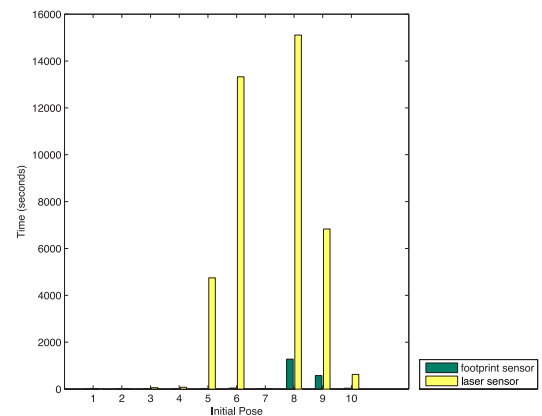


Figure 5: Computational time for finding the optimal exploration path for different initial positions and sensor models (cell size = 4, $G = 0.85$, and $r = 20$)

vide an *a posteriori* justification of the use of the footprint sensor in generating data in Table 1.

In the experiments of Table 1, we have also used the clustering of boundary cells. Without this clustering, the cost of computing a solution explodes and our algorithm does not find any solution within the timeout, even for simple settings. For example, for cell size = 4, $r = 30$, and $G = 0.85$, the algorithm with clustering terminates in an average time of 0.37 seconds, generating about 2,000 nodes, while the algorithm without clustering terminates at the timeout for every initial position, after having generated about 500,000 nodes.

Eliminating small clusters provides a consistent reduction of computational time, without affecting too much the solution quality (data are not shown here due to space constraints).

Tables 2 and 3 report experimental results for openspace and obstacles environments, respectively. Also in these experiments we have considered the footprint sensor and the clustering of boundary cells. Moreover, we have eliminated

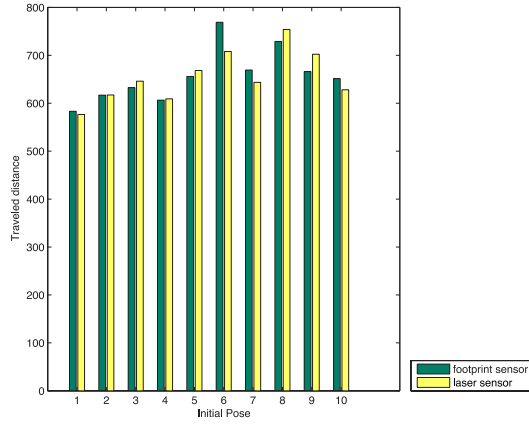


Figure 6: Optimal travelled distance for different initial positions and sensor models (setting is the same of Figure 5)

	CELL SIZE	DISTANCE		
		$r = 20$	$r = 25$	$r = 30$
G = 85%	2	3388.9 (249.8) ^{*(2)+(2)}	2351.1 (186.1) ⁺⁽¹⁾	1627.1 (112.6)
	4	3091.1 (217.2) ⁺⁽¹⁾	2328.6 (200.0) ⁺⁽¹⁾	1764.3 (102.4)
G = 90%	2	3732.1 (195.1) ^{*(3)+(2)}	2558.2 (181.0) ⁺⁽¹⁾	1808.4 (155.9)
	4	3384.0 (182.3) ⁺⁽¹⁾⁺⁽¹⁾	2530.4 (190.5) ⁺⁽¹⁾	1945.9 (107.3)
G = 95%	2	4856.9 (643.0) ^{*(3)+(5)}	2913.5 (133.1) ^{*(2)+(2)}	2119.8 (146.2)
	4	4103.9 (565.7) ⁺⁽²⁾⁺⁽²⁾	2893.5 (339.2) ^{*(1)+(2)}	2218.1 (105.6)

Table 2: Results (average and standard deviation) for the openspace environment (*(#): # of runs terminated due to the timeout, +(#): # of runs terminated due to empty frontier)

clusters smaller than $k = 80$ cells and $k = 16$ cells for the openspace and the obstacles environments, respectively. These values have been empirically set considering that clusters are larger in the openspace environment. Employing this last method, some runs terminate without a solution because of empty frontier, namely because there are no available destination positions. These runs are reported in the tables together with runs that terminate due to the timeout. As expected, runs terminating because of empty frontier are more frequent in the openspace environment, where we used a larger k , and for larger values of G , for which a larger amount of environment has to be discovered. Note that, for the openspace and the obstacles environments, we do not report results for cell size = 1 because too many runs terminate at the timeout and corresponding averages are not significant. All the above considerations relative to the indoor environment hold also for the openspace and obstacles environments.

Discussion

In general, experimental results show that our approach can be applied to calculate the optimal exploration path in realistically large environments. Moreover, the approach offers the possibility of trading-off between solution quality and computational time by tuning cell size and G and by adopting the speedup techniques.

Since the environments of Figure 1 are the same used in (Amigoni 2008) to experimentally compare some on-

	CELL SIZE	DISTANCE		
		$r = 20$	$r = 25$	$r = 30$
G = 85%	2	633.9 (25.1)	564.0 (19.6) ^{*(1)}	451.1 (9.5)
	4	631.8 (25.1)	531.2 (19.9)	458.4 (14.9)
G = 90%	2	746.6 (48.1)	636.8 (38.9) ^{*(1)}	501.7 (19.3)
	4	721.1 (42.0)	601.1 (22.9)	518.7 (17.5)
G = 95%	2	885.6 (68.7) ^{*(3)+(3)}	735.5 (61.7) ^{*(1)}	581.1 (34.0)
	4	859.5 (66.3) ^{*(2)+(0)}	702.2 (24.4)	595.5 (17.6)

Table 3: Results (average and standard deviation) for the obstacles environment (*(#): # of runs terminated due to the timeout, +(#): # of runs terminated due to empty frontier)

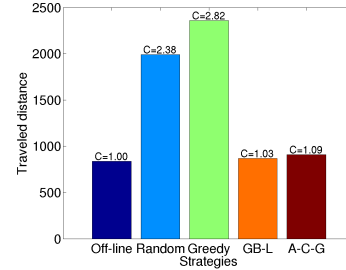


Figure 7: Performance (travelled distance) and competitive ratios C of some on-line exploration strategies

line exploration strategies for robots with limited and time-discrete visibility, we can calculate the competitive ratio for these strategies. Actually, we calculate an approximated competitive ratio using the approximation of the optimal exploration path returned by our approach. Figure 7 shows the results relative to the indoor environment with $r = 20$ and $G = 0.95$. It can be noted that on-line exploration strategies GB-L (González-Baños and Latombe 2002) and A-C-G (Amigoni and Caglioti 2010) show near-optimal performance. This finding is one of the original contributions that our approach enables. Note that, in real environments that are not known in advance, our approach can be applied to calculate the competitive ratio of on-line exploration strategies *a posteriori*, when the environments are mapped.

By calculating the competitive ratio for practical on-line exploration strategies employed in autonomous mobile robotics, we contribute to bridge the gap with theoretically-defined exploration strategies, see (Albers, Kursawe, and Schuierer 2008; Tovey and Koenig 2003) and the surveys (Ghosh and Kleinl 2010; Isler 2001). The former ones decide where to go on the basis of the current knowledge of the environment, while the latter ones are defined by behaviors independent of the specific environment. Theoretical results often present bounds on the competitiveness of on-line exploration strategies in classes of environments. For instance, while Tovey and Koenig (2003) consider generic graph environments, Albers, Kursawe, and Schuierer (2008) present bounds on the competitiveness of some on-line exploration strategies for grid environments. However, their results are not directly comparable with ours, since they consider a robot with visibility limited to the current node (cell).

Finally, one might wonder about the difference between the optimal discrete exploration paths found with our approach and the optimal continuous exploration paths for the

same environments. The fact that the travelled distance does not change much when cell size decreases (see Tables 1-3) makes us confident that our approximation is much smaller than the 2.5 bound of (Arkin, Fekete, and Mitchell 2000), but with a higher worst-case computational complexity (at least in the environments of Figure 1). However, the problem remains basically open and requires further investigation. Some hints could come from the results in (Nash, Koenig, and Tovey 2010), which show that the shortest discrete path between two points calculated in a two-dimensional grid is about 8% longer than the corresponding shortest continuous path.

Conclusions

In this paper we have presented a method for finding the optimal (shortest) off-line exploration path in a grid environment for a single robot with limited and time-discrete visibility. Our approach models the optimal exploration problem as a search problem and finds the solution using A*. We have also presented a number of speedup techniques to find a lower quality solution in a shorter time. Extensive experimental activities showed that the proposed approach can find optimal exploration paths in environments of realistic size. Although the main motivation of this work is the attempt of providing a better experimental evaluation of on-line exploration strategies for autonomous mobile robots, our results could be useful also for rescue applications. Assume that an unknown number of victims with no *a priori* information on their distribution are spread in a known environment. The problem of calculating the shortest path to find all of them reduces to the problem of finding the shortest path covering the free space with the robot's sensor, that basically is the problem we study.

Further work could develop better heuristic functions, for example using the solutions obtained with larger cell sizes as heuristics for finding solutions with smaller cell sizes. Moreover, the use of search algorithms different from A*, starting with branch and bound, could be investigated and a broader set of test environments could be considered. Possible extensions of our approach include multiple robots and moving from grid environments to more general graph environments. Finally, using our approach to improve on-line exploration strategies is a long-term goal.

References

- Albers, S.; Kursawe, K.; and Schuierer, S. 2008. Exploring unknown environments with obstacles. *Algorithmica* 32:123–143.
- Amigoni, F., and Caglioti, V. 2010. An information-based exploration strategy for environment mapping with mobile robots. *Robotics and Autonomous Systems* 5(58):684–699.
- Amigoni, F. 2008. Experimental evaluation of some exploration strategies for mobile robots. In *Proc. ICRA*, 2818–2823.
- Arkin, E.; Fekete, S.; and Mitchell, J. 2000. Approximation algorithms for lawn mowing and milling. *Computational Geometry - Theory and Applications* 17(1-2):25–50.
- Basilico, N., and Amigoni, F. 2011. Exploration strategies based on multi-criteria decision making for searching environments in rescue operations. *Autonomous Robots* 31(4):401–417.
- Chin, W., and Ntafos, S. 1991. Shortest watchman routes in simple polygons. *Discrete and Computational Geometry* 6:9–31.
- Choset, H. 2001. Coverage for robotics: A survey of recent results. *Annals of Mathematics and Artificial Intelligence* 31(1-4):113–126.
- Felner, A.; Stern, R.; Ben-Yair, A.; Kraus, S.; and Netanyahu, N. 2004. PHA*: Finding the shortest path with A* in an unknown physical environment. *Journal of Artificial Intelligence Research* 21:631–670.
- Gabriely, Y., and Rimon, E. 2001. Spanning-tree based coverage of continuous areas by a mobile robot. *Annals of Mathematics and Artificial Intelligence* 31:77–98.
- Ghosh, S., and Kleinl, R. 2010. Online algorithms for searching and exploration in the plane. *Computer Science Review* 4(4):189–201.
- González-Baños, H., and Latombe, J.-C. 2002. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research* 21(10-11):829–848.
- Isler, V. 2001. Theoretical robot exploration. Technical report, Computer and Information Science, University of Pennsylvania.
- LaValle, S. 2006. *Planning Algorithms*. Cambridge University Press.
- Lee, D., and Recce, M. 1997. Quantitative evaluation of the exploration strategies of a mobile robot. *International Journal of Robotics Research* 16(4):413–447.
- Nash, A.; Koenig, S.; and Tovey, C. 2010. Lazy Theta*: Any-angle path planning and path length analysis in 3D. In *Proc. AAAI*.
- Russell, S., and Norvig, P. 2010. *Artificial Intelligence: A Modern Approach*. Pearson.
- Stachniss, C., and Burgard, W. 2003. Exploring unknown environments with mobile robots using coverage maps. In *Proc. IJCAI*, 1127–1134.
- Thrun, S. 2002. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millennium*. Morgan Kaufmann. 1–35.
- Tovar, B.; Munoz, L.; Murrieta-Cid, R.; Alencastre, M.; Monroy, R.; and Hutchinson, S. 2006. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems* 54(4):314–331.
- Tovey, C., and Koenig, S. 2003. Improved analysis of greedy mapping. In *Proc. IROS*, 3251–3257.
- Urrutia, J. 2000. Art gallery and illumination problems. In Sack, J.-R., and Urrutia, J., eds., *Handbook of Computational Geometry*. North-Holland. 973–1027.
- Zelinsky, A.; Jarvis, R.; Byrne, J.; and Yuta, S. 1993. Planning paths of complete coverage for an unstructured environment by a mobile robot. In *Proc. ICAR*, 533–538.