Greedy Mapping of Terrain*

Sven Koenig Craig Tovey William Halliburton
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280, USA
{skoenig, ctovey }@cc.gatech.edu

Abstract

We study a greedy mapping method that always moves the robot from its current location to the closest location that it has not visited (or observed) yet, until the terrain is mapped. Although one does not expect such a simple mapping method to minimize the travel distance of the robot, we present analytical results that show (perhaps surprisingly) that the travel distance of the robot is reasonably small. This is interesting because greedy mapping has a number of desirable properties. It is simple to implement and integrate into complete robot architectures. It does not need to have control of the robot at all times, takes advantage of prior knowledge about parts of the terrain (if available), and can be used by several robots cooperatively.

1 Introduction

Mapping is an important task for mobile robots and a large number of mapping methods have been developed for them, both in robotics and in theoretical computer science [7, 18, 11, 15, 3, 6, 10, 19, 20, 24, 2, 8, 9, 16, 25, 23, 4, 21, 22]. A good overview is given in [26]. In this paper, we show that greedy mapping methods are easy to implement and easy to integrate into complete robot architectures. At the same time, planning is efficient and results in short travel distances of the robot. We study Greedy Mapping, a simple sensor-based planning method that always moves the robot from its current location to the closest location that it has not visited (or observed) yet, until the terrain is mapped. Greedy Mapping assumes that the location of the robot is always known, for example, from GPS data. It is greedy because its plans quickly gain information but do not take the long-term consequences of the movements into account. Yet, we will show that the travel distances of the robot are reasonably short. Greedy Mapping has the following desirable properties:

- **Theoretical Foundation:** Greedy Mapping has a solid theoretical foundation that allows one to characterize its behavior analytically. For example, it is guaranteed to map terrain under realistic assumptions and its plan-execution time can be analyzed formally, as we show in this paper.
- **Simple Integration into Robot Architectures:** Greedy Mapping is simple to implement and integrates well into complete robot architectures. It is robust with respect to the inevitable inaccuracies and malfunctions of other architecture components. For example, it does not need to have control of the robot at all times. This is important because search methods should only provide advice on how to act and work robustly even if that advice is ignored from time to time [1]. For example, if a robot has to re-charge its batteries during mapping, then it might have to preempt mapping and move to a known power outlet. Once restarted, the robot should be able to resume mapping from the power outlet, instead of having to return to the location where mapping was stopped (which could be far away) and resume its operation from there. Greedy Mapping exhibits this behavior automatically.
- **Prior Knowledge:** Greedy Mapping takes advantage of prior knowledge about parts of the terrain (if available) since it uses all of its knowledge about the terrain when determining which unvisited (or unobserved) location is closest to the robot and how to get there quickly. It does not matter whether this knowledge was previously acquired by the robot or provided to it.
- **Distributed Search:** Mapping tasks can be solved with several robots that each run Greedy Mapping and share their maps, thereby decreasing the mapping time. Cooperative mapping is a currently very active research area [5, 27].

These advantages explain why Greedy Mapping is an interesting mapping method to study. Greedy Mapping is probably one of the first mapping methods that come to mind when empirical robotics researchers quickly need to

*This work extends previous work by Sven Koenig and Yuri Smirnov. William Halliburton programmed the robot and performed the experiments. The Intelligent Decision-Making Group is partly supported by an NSF Award under contract IIS-9984827. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations and agencies or the U.S. government.

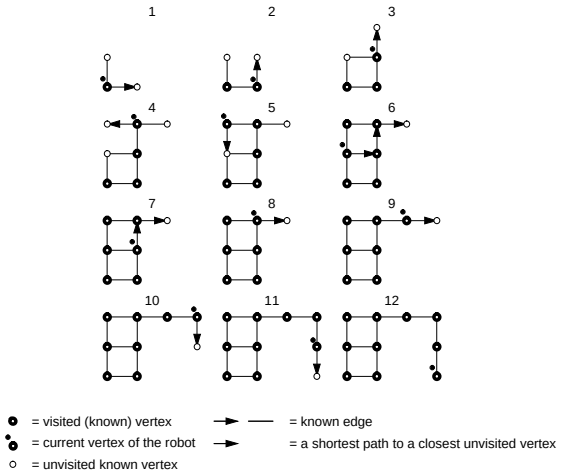

Figure 1: Possible Behavior of Greedy Mapping

implement a mapping method, and versions of it have been used on robots. For example, it appears that a version of Greedy Mapping has been used on a nomad-class tour-guide robot that offered tours to museum visitors [28].

The contribution of this paper is to provide a theoretical foundation for Greedy Mapping in form of an analysis of the travel distance of the robot. Clearly, Greedy Mapping is too simple a mapping method to minimize the travel distance. However, we derive bounds on the travel distance in any terrain that show that the travel distance is reasonably small. The purpose of this paper is to show that even simple mapping methods can perform well and thus to provide some relief to empirical robotics researchers who want to keep their navigation systems simple and thus do not want to implement complex mapping methods on their robots.

2 Analysis of the Travel Distance

In the following, we analyze the travel distance of robots that use Greedy Mapping. To make the mapping task as hard as possible, we assume that the robot has no initial knowledge of the topology of the map and that its sensors provide information only about its close vicinity. We assume, for clarity, that the robot is omni-directional, point-sized, equipped with only a radial short-distance sensor, and capable of error-free motion and sensing. The sensors onboard the robot uniquely identify its location and the neighboring unobstructed locations. This assumption is realistic, for example, if the locations look sufficiently different or if the robot has GPS or a similar localization method available.

To analyze the mapping problem formally, we formulate it as a graph coverage problem similar to the one studied in [12]. The robot has to map an initially unknown, finite, undirected graph $G = (V, E)$. The robot begins at some designated start vertex. When the robot is at a vertex v, it learns the vertices adjacent to v (that is, the vertices connected to vertex v by an edge), and can identify these vertices when it observes them again at a later point in time. Greedy Mapping always moves the robot on a shortest path from its current vertex to a closest unvisited vertex. It terminates when it knows of no unvisited vertices. Greedy Mapping must terminate, since each time it moves from the current vertex to the closest unvisited vertex, it visits one more vertex, and there are only a finite number of them. At termination, Greedy Mapping has visited all vertices that can be reached from the start location and thus has mapped the connected component of the graph that contains the start location. In the following, we assume without loss of generality that the graph is strongly connected. In this case, Greedy Mapping maps all of the graph. Figure 1 shows a possible behavior of Greedy Mapping on a subset of a simple grid. Note that Greedy Mapping is not constrained to working on grids but can be used on arbitrary graphs, including Voronoi diagrams [17].

In the following, we analyze the worst-case travel distance of Greedy Mapping as a function of the number of vertices of the graph because a small worst-case travel distance provides a good performance guarantee in all terrains. We do not take the planning time into account because robots move so slowly that the total problem solving time is completely dominated by the travel distance.

2.1 Lower Bound on the Travel Distance

A lower bound on the worst-case travel distance of Greedy Mapping can be established by example. In this section, we present a graph $G = (V, E)$ for which the worst-case travel distance of Greedy Mapping is $\Omega(\frac{\log |V|}{\log \log |V|} |V|)$ steps [13]. The graph makes Greedy Mapping traverse the same path repeatedly forward and backward, and this travel distance is large compared to the number of edges that are necessary to mislead Greedy Mapping into this behavior. Our example graph is planar since maps and other kinds of graphs used in robotics often have this property.

Theorem 1 *The worst-case travel distance of Greedy Mapping is $\Omega(\frac{\log |V|}{\log \log |V|} |V|)$ steps on strongly connected undirected graphs $G = (V, E)$, even if they are planar.*

Proof: Consider the planar graph $G = (V, E)$ shown in Figure 2, which is a variation of a graph in [14]. It consists of a stem with several branches. Each branch consists of two parallel paths of the same length that connect the stem to a single edge. The leaves at the ends of these single edges are crucial to "fooling" Greedy Mapping. When the robot traverses one of the parallel paths, Greedy Mapping might choose to return to the stem along the other path without first exploring the leaf.

We say that the length of a branch is the length of each of its two paths. The stem has length n^n for some integer $n \geq 3$ and

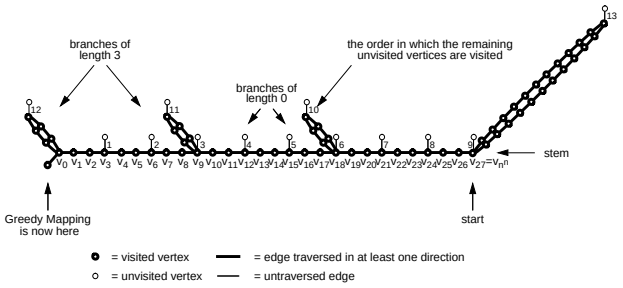

Figure 2: Planar Graph G for $n = 3$

n	travel distance	$\lvert V\rvert$	$\frac{\text{travel distance}}{\lvert V\rvert}$
3	207	80	2.587500
4	2279	778	2.929306
5	31253	9612	3.251457
6	515085	144014	3.576631
7	9928271	2542528	3.904882
8	219130987	51744018	4.234905
9	5448100629	1193201300	4.565953
10	150617283953	30753086422	4.897631
...	...	...	...

Table 1: Travel Distance of Greedy Mapping

consists of the vertices $v_0, v_1, \ldots, v_{n^n}$. For each integer i with $1 \leq i \leq n$ there are n^{n-i} branches of length $\sum_{j=1}^{i-1} n^j$ each (including branches of length zero). These branches attach to the stem at the vertices $v_{j\,n^i}$ for integers j; if i is even, then $0 \leq j \leq n^{n-i} - 1$, otherwise $1 \leq j \leq n^{n-i}$. There is one additional single edge that attaches to vertex v_0. The start vertex is v_{n^n}.

Greedy Mapping can choose to break ties so as to behave as follows: start at vertex v_{n^n}, traverse the whole stem and all branches, but bypass all the leaves at their ends, and then traverse the additional edge attached to vertex v_0, as shown in Figure 2. At this point, Greedy Mapping again traverses the whole stem, visiting the leaves of the branches of length 0. It then switches directions and travels along the whole stem in the opposite direction, this time visiting the leaves of the branches of length n, and so forth, switching directions repeatedly. It completes its exploration when it finally visits the leaf of the longest branch.

To summarize, the leaves at the ends of the branches are tried out in the order indicated in Figure 2. The total travel distance is $\Omega(n^{n+1})$ steps since the stem of length n^n is traversed $n+1$ times. To be precise, the total travel distance is $(n^{n+3}+3n^{n+2}-8n^{n+1}+2n^2-n+3)/(n^2-2n+1)$ steps. It holds that $|V| = \Theta(n^n)$ since $|V| = (3n^{n+2}-5n^{n+1}-n^n+n^{n-1}+2n^2-2n+2)/(n^2-2n+1)$. This implies that $n = \Omega(\frac{\log|V|}{\log\log|V|})$ since, for all sufficiently large n, it holds that

$$\frac{\log(n^n)}{\log\log(n^n)} = \frac{n\log n}{\log n + \log\log n} \leq \frac{n\log n}{\log n} = n.$$

It follows that the total travel distance is $\Omega(n^{n+1}) = \Omega(n\,|V|) = \Omega(\frac{\log|V|}{\log\log|V|}\,|V|)$ steps. ■

We also performed a simulation that confirmed our theoretical results for $n \leq 6$. Table 1 shows how the ratio of the travel distance and the number of vertices increases as n increases. Our graphs can be adapted to different assumptions as well. For example, we have assumed that the robot can identify only the vertices adjacent to its current vertex. The graph can easily be adapted to sensors with larger lookaheads, say of x vertices, by replacing each edge with x consecutive edges that are connected via $x-1$ intermediate vertices.

2.2 Discussion of the Lower Bound

No mapping method can be sure to omnisciently follow a best possible path in hindsight. To judge how good its travel distance is, we therefore need to compare it to other mapping methods. Depth-first search is such a method. It always moves the robot from its current vertex to an adjacent unvisited vertex. If such a vertex does not exist, it leaves the current vertex along the edge with which it was entered for the first time (backtracking). It terminates when it knows of no unvisited vertices. Its worst-case travel distance is at most twice the number of vertices since each step either visits a previously unvisited vertex (which can happen at most once for each vertex) or backtracks from a vertex (which can also happen at most once for each vertex). In the previous section, we have shown that the worst-case travel distance of Greedy Mapping is superlinear in the number of vertices. Thus, it is not a worst-case optimal mapping method. However, it has advantages over depth-first search. For example, depth-first search does not resume mapping from the power outlet after a robot has moved to recharge itself but rather requires the robot to return to the location where mapping was stopped. Second, depth-first search cannot be used by several robots cooperatively. Third, depth-first search does not take advantage of prior knowledge about the graph to direct the search, for example, if part of the terrain is already known and thus does not need to get mapped. While depth-first search can be modified to tackle some of these problems [31], these attempts have so far resulted in impractical robot navigation methods.

2.3 Upper Bound on the Travel Distance

Even though the travel distance of Greedy Mapping is not optimal in the worst case, it is our experience that it is reasonably small in practice. In the following, we show that the disadvantage of Greedy Mapping in terms of its worst-case travel distance is small compared to the optimal worst-case travel distance, which is linear in the number of vertices. This justifies its use on mobile robots.

It is easy to see that the worst-case travel distance of Greedy Mapping is at most $|V|^2$ steps on strongly connected

undirected graphs $G = (V, E)$. Since the robot always follows a shortest path to the closest unvisited vertex, it reaches another previously unvisited vertex after at most $|V|$ steps. Since there are only $|V|$ vertices, it can repeat this step at most $|V|$ times until it has visited all vertices, resulting in the upper bound. However, this quadratic upper bound on the worst-case travel distance of Greedy Mapping leaves a large gap with the linear optimal worst-case travel distance. We now narrow the gap by proving a tighter upper bound of only $O(|V|^{3/2})$ steps on strongly connected undirected graphs $G = (V, E)$.

Theorem 2 *The worst-case travel distance of Greedy Mapping is $O(|V|^{3/2})$ steps on strongly connected undirected graphs $G = (V, E)$.*

Proof: Let c^i ($0 \leq i \leq |V|-1$) denote the ith previously unvisited vertex that Greedy Mapping visits. c^0 is the start vertex of the robot. Once Greedy Mapping has visited $c^{|V|-1}$, it has visited all vertices at least once and stops. Let L_i ($1 \leq i \leq |V| - 1$) denote the number of steps when the robot moved from c^{i-1} to c^i after it visited c^{i-1} for the first time. Note that $0 \leq L_i \leq |V|$. Then, the travel distance of Greedy Mapping is $\sum_{i=1}^{|V|-1} L_i$ steps.

Let d_v^i ($0 \leq i < |V| - 1, v \in V$) denote the length of a shortest path from v to the closest unvisited vertex directly after the robot first visited c^i. Note that $0 \leq d_v^i \leq |V|$ and that d_v^i is nondecreasing in i. Furthermore, $d_{c^i}^i = L_{i+1}$ for $0 \leq i < |V|-1$. Define $\phi^i = \sum_{v \in V} d_v^i$ ($0 \leq i < |V| - 1$). Note that $0 \leq \phi^i \leq |V|^2$ and that ϕ^i is nondecreasing in i. The main idea behind our proof is that the travel distance of the robot is large only if many of the L_i are large. Each large L_i results in large increases from ϕ^{i-2} to ϕ^{i-1}. However, there is a limit on how large the ϕ_i can get, which forces the travel distance of the robot to be small.

For all $0 \leq z \leq L_{i+1}$ ($1 \leq i < |V| - 1$), there is at least one vertex x_z at a distance of z edges from vertex c^i when the robot visited c^i for the first time. This is so, since the robot then moved on a shortest path to c^{i+1} and the path had length L_{i+1}. Then, $L_{i+1} \leq z + d_{x_z}^i$ because an unvisited vertex was only $d_{x_z}^i$ steps away from x_z when the robot visited c^i for the first time, and x_z is z steps away from c^i. Furthermore, $d_{x_z}^{i-1} \leq z$ because c^i was still unvisited when the robot visited c^{i-1} for the first time, and c^i is z steps away from x_z (because the graph is undirected). Putting the two inequalities together, it holds that $d_{x_z}^i - d_{x_z}^{i-1} \geq L_{i+1} - 2z$.

Since the d_v^i are nondecreasing in i, it holds for $1 \leq i < |V|-1$ that

$$\begin{aligned} \phi^i - \phi^{i-1} &\geq \sum_{z=0}^{L_{i+1}} (d_{x_z}^i - d_{x_z}^{i-1}) \\ &\geq \sum_{z=0}^{L_{i+1}} \max(L_{i+1} - 2z, 0) \geq \frac{(L_{i+1})^2}{4}. \end{aligned}$$

Summing over i results in

$$\sum_{i=1}^{|V|-2} \frac{(L_{i+1})^2}{4} \leq \phi^{|V|-2} - \phi^0 \leq |V|^2 - 0 = |V|^2.$$

We can now bound $\sum_{i=2}^{|V|-1} L_i$. The values L_i ($2 \leq i \leq |V| - 1$) are constrained by $L_i \geq 0$ and $\sum_{i=2}^{|V|-1} (L_i)^2 \leq 4|V|^2$. Calculus shows that maximizing $\sum_{i=2}^{|V|-1} L_i$ subject to these constraints is achieved by $L_i = 2|V|/\sqrt{|V| - 2} \leq 2\sqrt{|V|}+3$ for all $2 \leq i \leq |V| - 1$. Thus, $\sum_{i=2}^{|V|-1} L_i \leq (|V| - 2)(2\sqrt{|V|} + 3) \leq 2|V|^{3/2} + 3|V|$. Finally, the travel distance of Greedy Mapping is $\sum_{i=1}^{|V|-1} L_i = L_1 + \sum_{i=2}^{|V|-1} L_i \leq 2|V|^{3/2} + 4|V|$. ■

2.4 Discussion of the Upper Bound

Our analysis in the previous section showed that the worst-case travel distance of Greedy Mapping, although not optimal in the worst case, is reasonably small. It might be even smaller on graphs with restricted topologies. For example, on our robot, we use greedy mapping in conjunction with regular grids. In this case, all vertices have a small (bounded) degree. We currently do not know whether this decreases the worst-case travel distance of Greedy Mapping and, if so, by how much. Furthermore, the robot has some initial knowledge of the topology of the map since it knows that the graph is a subset of a regular grid. This allows the robot to move to the closest unobserved cell rather than the closest unvisited cell, which does not change our analytical results. Assume, for example, the following scenario. The robot operates on a graph whose vertices can be blocked. The robot always observes the status of its current vertex and all adjacent vertices, and can then move to any adjacent unblocked vertex. The robot knows which graph it operates on but initially does not know which vertices are blocked. It has to determine the status of each vertex (unless that is impossible from the start vertex of the robot). To do this, it uses a version of Greedy Mapping that always moves towards the closest unobserved vertex, that is, the closest vertex with unknown status. This is the version of Greedy Mapping that we implemented on a robot except that we used a sensor with a much larger range. Now consider graphs of the topology described in Section 2.1 except that each leaf vertex (including the vertex at the end of the edge attached to vertex v_0) is replaced by an edge. All vertices are unblocked. If $n \geq 6$, then the version of Greedy Mapping that always moves towards the closest unobserved vertex behaves exactly as described in Section 2.1.

3 A Simple Implementation

We have implemented Greedy Mapping an a Nomad 150 robot using a Sick LMS200 laser scanner. The purpose of the implementation was not to demonstrate a complete and realistic mapping scenario. Rather, it was to show that Greedy Mapping is easy to implement and easy to integrate into complete robot architectures.

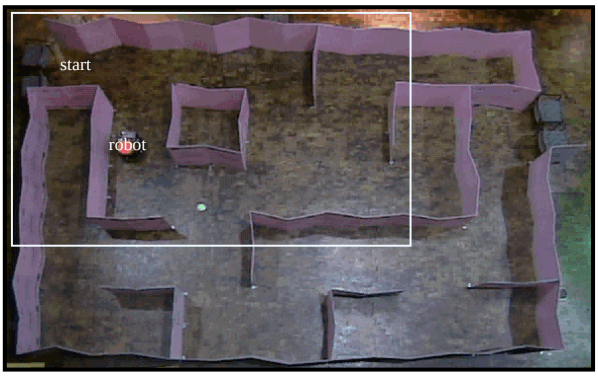

Figure 3: The Maze

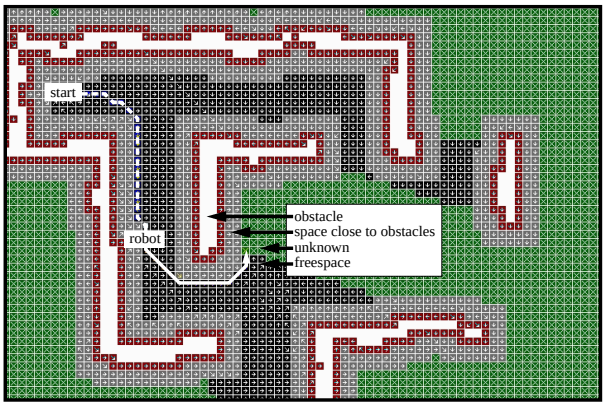

Figure 4: Screen Shot of Part of the Map

We used Greedy Mapping in conjunction with a simple 8-connected grid, the cells of which had a size of 10 centimeters by 10 centimeters. All processing was performed on-board the robot on a Toshiba Pentium MMX 233 MHz laptop running Redhat 6.2 Linux. The robot interleaved sensing, planning, and movement. Sensing consisted of a full 180 degree scan with the laser scanner. Initially, all cells of the grid were marked as unobserved. The cells that corresponded to detected obstacles were marked as observed and untraversable. Obstacles were surrounded by traversable cells with a large cost, to bias the robot away from them. The other cells swept by the sensor were marked as observed and traversable. Cells at distance one from obstacles had traversal cost ten, cells at distances two or three from obstacles had traversal cost five, and cells at larger distances from obstacles had traversal cost one. Planning found a shortest path from the current location of the robot to the closest unobserved cell, the first action of which was executed. Then, the cycle repeated until all cells had been observed or the shortest path to the closest unobserved cell had infinite cost. The whole system was implemented by one graduate student from scratch in a couple of days, which demonstrated that Greedy Mapping is really easy to implement and integrate into complete robot architectures.

We used the robot to map a maze of size 28 by 20 feet that we constructed out of polystyrene insulation on the ground floor of our building. We let the robot map the maze five times. All five experiments were successful. Figure 3 shows a top view of the maze. Figure 4 shows a snapshot of the map during map building, together with the shortest path from the current location of the robot to a closest unobserved cell. The part of the maze that corresponds to the part of the map shown in the screen shot is outlined in Figure 3.

The simple version of Greedy Mapping analyzed in this paper assumes that there is neither position nor sensor uncertainty. The assumption that there is no position uncertainty makes Greedy Mapping well suited for outdoor navigation in conjunction with GPS. This assumption was not justified in our experiments since the location of the robot was estimated using a simple dead-reckoning technique. The map shown in Figure 4 shows some of the resulting inaccuracies. However, the runs were not long enough for this to become a problem. The assumption that there is no sensor uncertainty was justified. The Sick laser scanner is highly accurate and has sufficient resolution and range accuracy.

4 Conclusions

In this paper, we studied Greedy Mapping, a simple mapping method that always moves the robot to the closest location that it has not visited (or observed) yet, until the terrain is mapped. We analyzed the worst-case travel distance of Greedy Mapping using a graph-theoretic framework. We showed that the worst-case travel distance of Greedy Mapping is not optimal since it is superlinear in the number of vertices while the worst-case travel distance of depth-first search is only linear in the number of vertices $|V|$. However, we also showed that the worst-case travel distance of Greedy Mapping is at most on the order of $|V|^{3/2}$ steps in general and thus that its performance disadvantage is small. This upper bound provides a theoretical justification for using Greedy Mapping in practice especially since it has a variety of advantages over alternative mapping methods. For example, it is simple to implement and integrate into complete robot architectures, resumes mapping from the power outlet after the robot has moved to recharge itself instead of having to return to the location where mapping was stopped (which could be far away), takes advantage of prior knowledge about parts of the terrain (if available), and can be used by several robots cooperatively. In future work, we intend to close the gap between the upper and lower bounds of the worst-case travel distance that the results presented in this paper were able to reduce but could not yet eliminate. We

also intend to analyze more complicated mapping methods including those that can deal with sensor uncertainty and the resulting position uncertainty [29, 30]. We believe that the analysis presented here provides a good start for this venue.

References

[1] P. Agre and D. Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of the National Conference on Artificial Intelligence*, pages 268–271, 1987.

[2] S. Albers and M. Henzinger. Exploring unknown environments. *SIAM Journal on Computing*, 29(4):1164–1188, 2000.

[3] R. De Almeida and C. Melin. Exploration of unknown environments by a mobile robot. *Intelligent Autonomous Systems*, 2:715–725, 1989.

[4] B. Awerbuch, M. Betke, R. Rivest, and M. Singh. Piecemeal graph exploration by a mobile robot. *Information and Computation*, 152(2):155 –172, 1999.

[5] W. Burgard, D. Fox, M. Moors, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *Proceedings of the International Conference on Robotics and Automation*, pages 476–481, 2000.

[6] C. Choo, J. Smith, and N. Nasrabadi. An efficient terrain acquisition algorithm for a mobile robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 306–311, 1991.

[7] H. Choset. *Sensor-Based Motion Planning: The Hierarchical Generalized Voronoi Graph*. PhD thesis, California Institute of Technology, Pasadena (California), 1996.

[8] X. Deng, T. Kameda, and C. Papadimitriou. How to learn an unknown environment; I: The rectilinear case. *Journal of the ACM*, 45(2):215 –245, 1998.

[9] X. Deng and C. Papadimitriou. Exploring an unknown graph. In *Proceedings of the Symposium on Foundations of Computer Science*, pages 355–361, 1990.

[10] G. Dudek, P. Freedman, and S. Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *Proceedings of the International Conference on Artificial Intelligence*, pages 1639–1647, 1993.

[11] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. *IEEE Transactions on Robotics and Automation*, 7(6):859–865, 1991.

[12] B. Kalyanasundaram and K. Pruhs. Constructing competitive tours from local information. *Theoretical Computer Science*, 130:125–138, 1994.

[13] S. Koenig. Exploring unknown environments with real-time search or reinforcement learning. In *Proceedings of the Neural Information Processing Systems*, pages 1003–1009, 1999.

[14] S. Koenig and Y. Smirnov. Graph learning with a nearest neighbor approach. In *Proceedings of the Conference on Computational Learning Theory*, pages 19–28, 1996.

[15] B. Kuipers and Y. Byun. A robust, qualitative method for robot spatial learning. In *Proceedings of the National Conference on Artificial Intelligence*, pages 774–779, 1988.

[16] S. Kwek. On a simple depth-first search strategy for exploring unknown graphs. In *Proceedings of the Workshop on Algorithms and Data Structures*, volume 1272 of *Lecture Notes in Computer Science*, pages 345 –353. Springer, 1997.

[17] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[18] J. Leonard, H. Durrant-Whyte, and I. Cox. Dynamic map building for an autonomous mobile robot. *International Journal of Robotics Research*, 11(4):286–298, 1992.

[19] V. Lumelsky and A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

[20] G. Oriolo, G. Ulivi, and M. Vendittelli. Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(3):316–333, 1998.

[21] P. Panaite and A. Pelc. Exploring unknown undirected graphs. *Journal of Algorithms*, 33(2):281 –295, 1999.

[22] C. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84(1):127–150, 1991.

[23] L. Prasad and S. Iyengar. A note on the combinatorial structure of the visibility graph in simple polygons. *Theoretical Computer Science*, 140(2):249–263, 1995.

[24] N. Rao. Algorithmic framework for learned robot navigation in unknown terrains. *IEEE Computer*, 22(6):37–43, 1989.

[25] N. Rao. Robot navigation in unknown generalized polygonal terrains using vision sensors. *IEEE Transactions on Systems, Man, and Cybernetics*, 25(6):947–962, 1995.

[26] N. Rao, S. Hareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM–12410, Oak Ridge National Laboratory, Oak Ridge (Tennessee), 1993.

[27] K. Singh and K. Fujimura. Map making by cooperating mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, pages 254–259, 1993.

[28] S. Thrun, A. Bücken, W. Burgard, D. Fox, T. Fröhlinghaus, D. Hennig, T. Hofmann, M. Krell, and T. Schmidt. Map learning and high-speed navigation in RHINO. In D. Kortenkamp, R. Bonasso, and R. Murphy, editors, *Artificial Intelligence Based Mobile Robotics: Case Studies of Successful Robot Systems*, pages 21–52. MIT Press, 1998.

[29] S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. In *Proceedings of the International Conference on Robotics and Automation*, pages 321–328, 2000.

[30] S. Thrun, D. Fox, and W. Burgard. Probabilistic mapping of an environment by a mobile robot. In *Proceedings of the International Conference on Robotics and Automation*, pages 1546–1551, 1998.

[31] I. Wagner, M. Lindenbaum, and A. Bruckstein. Distributed covering by ant-robots using evaporating traces. *IEEE Transactions on Robotics and Automation*, 15(5):918–933, 1999.