Map Validation and Self-location in a Graph-like World *

G. Dudek[1], M. Jenkin[2], E. Milios[2], D. Wilkes[3]
[1]MCRCIM, McGill University, Montreal, Canada, H3A 2A7
[2]Dept. of Computer Science, York University, Toronto, Canada, M3J 1P3
[3]Dept. of Computer Science, University of Toronto, Toronto, Canada, M5S 1A4
dudek@mcrcim.mcgill.edu, {jenkin, eem}@cs.yorku.ca, wilkes@cs.toronto.edu
[1](514)398-4325, [2](416)736-5053, [3](416)978-7726

Abstract

We present algorithms for the discovery and use of topological maps of an environment by an active agent (such as a person or a mobile robot). We discuss several issues dealing with the use of pre-existing topological maps of graph-like worlds by an autonomous robot and present algorithms, worst cases complexity, and experimental results (for representative real-world examples) for two key problems. The first of these problems is to verify that a given input map is a correct description of the world (the VALIDATION PROBLEM). The second is to determine the robot's physical position on an input map (THE SELF-LOCATION PROBLEM). We present algorithms which require $O(N^2)$ and $O(N^3)$ steps to validate and locate the robot in a map (where N is the number of places in the map).

* *This paper appears in the* **Proceedings of IJCAI 1993**, *Chambery, France, August 1993.* Financial support from NSERC Canada, and the Ontario-Quebec interchange program is gratefully acknowledged. We also thank Xiaotie Deng and Andy Mirzaian for many helpful discussions.

1 Introduction

One of the problems confronting an autonomous mobile robot is that of maintaining an internal description of its environment. Without a useful internal representation (a map) and knowledge of the robot's position and orientation with respect to this map, many robotic tasks become difficult, if not impossible. Some tasks may be amenable to behaviour-based approaches, and for such tasks an internal description of the environment may not be required, for example [Brooks, 1986; Brooks, 1989; Connell, 1990; Arkin, 1990]. For other tasks, especially complex structured tasks, which are specific to particular landmarks in the environment, a map of the environment is crucial. If the robot is to have a map, what form would this map take, and how should it be acquired?

There are many possible ways in which a map can be built. Perhaps the simplest is to have the map input manually (offline). Although an attractive approach due to its ease of implementation, a manually crafted map of the environment may not be the best map for the robot to use; the features and structures most natural for humans may not correspond well to features and structures to which the robot has sensory access. It may be better to have a robot create the map itself.

Several map representations have been proposed in the literature. These include: *metric representations* [Ayache and Faugeras, 1989; Latombe, 1991; Lumelsky and Stepanov, 1987; Cox, 1989; Elfes, 1987], which explicitly model the two or three dimensional position of structure in the environment, *probabilistic representations*

[Meng, 1987; Smith and Cheeseman, 1986], which retain many metric properties in the representation, but augment the representation with uncertainty information, *graph-based representations* [Kuipers and Levitt, 1988; Davis, 1986; Basye and Dean, 1990], which represent locations of interest. Hybrid maps, which combine elements of each of these representational levels have also been proposed [Engelson and McDermott, 1992; Arkin, 1990]. Integration of a map within a reactive framework has been explored [Mataric, 1992].

When an autonomous agent explores its own environment, the fundamental problem that the robot has to address is the "have I been here before" problem. If the environment is modelled in a metric manner, this problem can be viewed as equivalent to the following: Given the current position and orientation of the robot (known with a particular covariance error matrix with respect to a world-centered coordinate system), has the robot been previously at that same position with the same orientation, but via a different path? If the environment is represented in a graph-like manner, the question becomes "have I visited this location before", and if so which entrance did I use last time I entered here?

In previous work [Dudek *et al.*, 1991; Dudek *et al.*, 1988] we described and analyzed an algorithm for building a graph-like map of an *a priori* unknown world (see also section 3). This work assumes that a topological, rather than metric, description of the world is desirable at some level in a descriptive hierarchy [Kuipers and Levitt, 1988]. A graph-like map was chosen because it represents the minimal information that a robot must be able to sense in order to distinguish one place from another. The exploration algorithm allows the robot to form a model of its world by exploring the world systematically with the use of one or more distinct markers that can be dropped and picked up at will. These markers can be recognized if they are found on the path of the robot. The exploration algorithm works by building up a known subgraph of the world, exploring unknown edges incident on the known subgraph, and thus incrementally adding to it. The algorithm requires the robot to make $O(N^3)$ moves between locations in the worst case, where N is the number of distinguished locations in its environment. These locations correspond to vertices in the graph-like map that is produced while the moves correspond the edge traversals. Note that it is not possible for the robot to explore its environment without some navigational aid. As an example, regular graphs of the same degree (i.e. graphs in which each vertex has the same number of incident edges, equal to the degree) are indistinguishable from each other without markers, because each vertex appears identical to every other vertex.

This paper addresses two related problems: Given a map of the world, how can an active agent (robot) determine whether the map is correct in its description of the connectivity among locations. This is the MAP VALIDATION problem. A related problem is: given a map, how can an active agent determine its current location and orientation on the map (the SELF-LOCATION problem), where "orientation" is equivalent to the correct correspondence between the map edges and the world edges incident on the correct location.

2 The World Model

The algorithms presented in this paper operate on an augmented graph-like representation of the world which is defined below. Although the pros and cons of such a representation are beyond the scope of this paper, we view this abstraction as a lowest common denominator for real robotic systems, yet as nevertheless rich enough to represent location in a meaningful way.

2.1 Definitions

The robot's "purpose in life" is to use its ability to act and to perceive in the graph domain in order to build an augmented graph[1] which is isomorphic [Guibas and Stolfi, 1984] to the finite world it has been assigned to explore. The robot's inputs are its sensations and it can interact with the world only through its actions.

[1] By an augmented graph we mean a graph as well as constraints on how it is embedded.

The World The world is defined as an embedding of an undirected graph G:

$$G = (V, E)$$

with set of vertices V and set of edges E. Note that in practice such a graph could be defined within a continuous environment based of landmarks or other features [Kuipers and Levitt, 1988]. The vertices are denoted by:

$$V = \{v_1, ..., v_N\}$$

We will restrict the world model to graphs G that contain no cycles of length ≤ 2, i.e. the graph contains no degenerate or redundant paths. This restriction prohibits the world from having multiple edges between two vertices or an edge incident twice at the same vertex.

The definition of an edge is extended slightly to allow for the explicit specification of the order of edges incident upon each vertex of the graph embedding. This ordering is obtained by enumerating the edges in a systematic manner (e.g. clockwise if the graph is planar) from some standard starting direction. An edge $E_{i,j}$ incident upon v_i and v_j is assigned labels n and m, one for each of v_i and v_j respectively. n represents the ordering of the edge $E_{i,j}$ with respect to the consistent enumeration of edges at v_i, m represents the ordering of the edge $E_{i,j}$ with respect to the consistent enumeration of the edges at v_j. The labels m and n can be considered as general directions, e.g. from vertex v_i the nth exit takes edge $E_{i,j}$ to vertex v_j.

Movement and Action The robot can move from one vertex to another by traversing an edge (a *move*), it can pick up a marker that is located at the current vertex, and it can put down a marker it holds at the current vertex (a *marker operation*). The robot in general has K markers at its disposal.

Assume the robot is at a single vertex, v_i, having entered the vertex through edge $E_{i,l}$. In a single move, it leaves vertex v_i for vertex v_j by traversing the edge $E_{i,j}$, which is r edges after $E_{i,l}$ according to the edge order at vertex v_i (see Figure 1). This is given by the transition

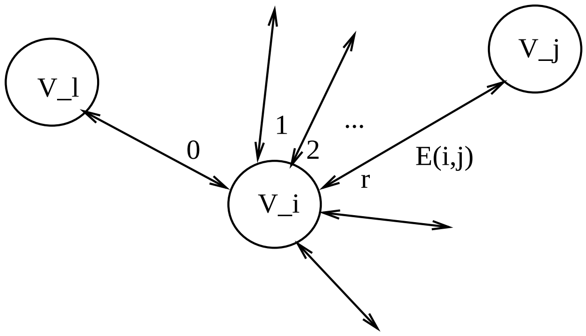

Figure 1: Edge ordering

function:

$$\delta(v_i, E_{i,l}, r) = v_j$$

We assume the following property about the transition function:
If $\delta(v_i, E_{i,l}, r) = v_j$ and $\delta(v_j, E_{i,j}, s) = v_k$, then $\delta(v_j, E_{j,k}, -s) = v_i$.
This implies that a sequence of moves is invertible, and can be retraced. We also assume that there does not exist a $t \neq -s$ such that $\delta(v_j, E_{j,k}, t) = v_i$ and that there does not exist a t such that $\delta(v_j, E_{j,k}, t) = v_j$, i.e. there are no redundant or degenerate paths.

A single move is thus specified by the order r of the edge along which the robot exits the current vertex, where r is defined with respect to the edge along which the robot entered such vertex. Note that in the special case of a planar embedding of a graph, enumeration of edges in a clockwise fashion satisfies the above assumption.

Perception The robot's perception is of two kinds, marker-related and edge-related perception.

Marker-related Perception. Assume that the robot is at vertex v_i, having arrived via edge $E_{i,j}$. The marker-related perception of the robot is a K-tuple $B_s = (bs_1, bs_2, ..., bs_K)$, where bs_k has a value from the set $\{present, not-present\}$, according to whether marker k is present at vertex v_i.

Edge-related Perception. The robot can determine the relative positions of edges incident on the vertex v_i in a consistent manner, e.g. by a clockwise enumeration (in the planar case) starting with $E_{i,j}$. As a result, it can assign an integer label to each edge incident on v_i, representing the order of that edge with respect to the edge enumeration at v_i. The label 0 is assigned arbitrarily to the edge $E_{i,j}$, through which the robot entered vertex v_i. The ordering is local, because it depends on the edge $E_{i,j}$. Entering the same vertex from two different edges will lead to two local orderings, one of which is a permutation of the other. Note that if the graph is planar and a spatially consistent (e.g. clockwise) enumeration of edges is used, then two permutations will be simple circular translations of each other. But this will not hold in general, and in this paper we only assume that the edges can be ordered consistently. If the robot visits the same vertex twice, it must relate the two different local orderings produced and unify them into a single global ordering, for example by finding the label of the 0-th edge of the second ordering with respect to the first ordering. Determining when the same vertex has been visited twice and generating a global ordering for each vertex is part of the task of the following algorithms.

3 Map building: exploration

In earlier work ([Dudek *et al.*, 1988; Dudek *et al.*, 1991]) we demonstrated that in general it is not possible for a robot to explore and map an unknown environment with this sort of limited sensory perception. Additional information is required. This is not a particularly startling result: fairy tales, and mythology are full of stories of heros who avoided becoming lost within a maze by dropping markers or unwinding string as they went. The basic problem is that, when the explorer enters an unknown environment, he cannot always determine if he has returned to a previously visited location.

In these earlier papers it was also demonstrated that, as long as the explorer had a single unique marker which could be dropped and picked up at will, it was possible for the explorer (or robot) to fully map his environment (an arbitrary graph) within $O(N^3)$ steps (although complexity for most typical cases appears substantially better than this worst-case bound). The basis of the algorithm is the maintenance of an explored subgraph of the full graph. As new vertices are encountered, they are added to the explored subgraph, and their outgoing edges are added to the set of edges that lead to unknown places and therefore must be explored.

The cost of exploring the graph in terms of edges traversed by the robot (mechanical complexity) is $O(N^3)$, where N is the number of nodes. This follows from the need to go back and actually visit all of the locations in the known sub-graph to solve the "have I been here before" problem. Often it is not necessary to explore an environment completely from scratch. A map may already be available and it may suffice to determine if the existing map of the world corresponds to its current state and, if not, to determine where the two are inconsistent.

4 Map validation

The MAP VALIDATION problem is defined as follows. The robot is given a map of its environment, and it is told which map vertex is its current location and also the correspondence between one map edge incident on the current map vertex and a physical exit from the current physical vertex. This gives the robot a position and orientation with respect to the map. The problem is to determine whether the map is correct. One issue is whether this can be done more efficiently than full exploration. This paper gives an algorithm for validation that requires at most $O(N^2)$ moves of the robot.

The key idea underlying the validation algorithm is to construct a spanning tree rooted at the current vertex (a simple operation [Bondy and Murty, 1976]), to verify the presence in the world of that tree first, and then to verify the remaining edges of the graph-like world (which is akin to an exploration task). We now describe the algorithm, in the case in which the robot is equipped with a single movable marker.

1. Validate a spanning tree rooted at the start position.
   (a) Compute a spanning tree ST of the map, rooted at the initial vertex v_0.
   (b) Validate ST.
   This involves backup up the tree in the manner

of the exploration algorithm to verify that every vertex on the tree is unique and has a local signature (degree) matching the corresponding vertex on the map. This has complexity $(N^2 + N)/2$ or $O(N^2)$. If this validation fails, the entire validation fails.

2. Validate the edges outside the spanning tree, by checking that they connect the correct two vertices. The efficient way to do this is to check all edges that are incident on a single vertex with a single physical traversal of the graph.

   For each vertex v_i of the graph (connected to at most $N-1$ vertices)

   (a) leave the marker there

   (b) visit all neighbours of v_i, v_j of the map via ST ($O(N)$ effort)

   (c) for each v_j (having an edge (v_i, v_j) according to the map) check physically whether edge (v_i, v_j) exists by traversing it and looking for the marker at the other end.

Step 1 is based on traversing the spanning tree for the graph. A graph's spanning tree has $O(N)$ nodes and $O(N)$ edges and thus this step requires $O(N^2)$ moves of the robot. Note that a graph's spanning tree can be computed efficiently [Bondy and Murty, 1976].

Step 2 requires, in the worst case, $O(N^2)$ moves.

Thus, the entire algorithm requires $O(N^2)$ moves in the worst case.

In the case in which we have at least N movable, distinguishable markers available, we can save a substantial number of moves by making a single re-traversal of the spanning tree to validate all non-tree edges. In this case, $O(M)$ moves may be required in the worst case, for M the number of edges in the graph.

For the case of k markers, where $k < N$, we cut the number of traversals of the spanning tree by a factor k by placing all k markers at different vertices, and then traversing the spanning tree ST once for this marker placement, thereby validating the edges incident on all k vertices.

(a) (b)

Figure 2: Stages in validating a graph. The graph was validated with itself starting with the correct location and orientation. Edges that have been validated are drawn with a thicker line. Note that non-spanning tree edges are validated in one direction at a time and these are drawn half thick/half thin with the thicker end corresponding to the validated end of the edge. The location of the robot is indicated with a square and the location of the marker is indicated by a diamond. (a) shows the validation algorithm after the validation of the spanning tree, while (b) shows the validation algorithm after validating half of each of the graph edges. This graph was correctly validated.

Figure 2 shows the state of the validation process at two stages in the validation of a simple graph. Figure 2a shows the validation process after the validation of the spanning tree of the graph, while Figure 2b shows the same graph after half of each of the graph edges have been validated. Treating this graph as a map, two corrupted versions and the point of failure of the validation process are shown in Figure 3. Figure 3a is a graph having an extra edge. The validation process (indicated by the heavy lines) fails in the spanning tree validation stage as the spanning trees do not agree (the degree of one of the nodes on the spanning tree is incorrect). Figure 3b shows a corrupted world in which the spanning trees agree but in which the graph edges are incorrect. Here the spanning tree is verifiable, but the graph edges disagree.

(a) (b)

Figure 3: Two graphs which fail when being validated by the map given in Figure 2. (5 a) fails as the spanning tree cannot be validated, while (b) fails because a non-tree edge is incorrect.

5 Self-location and validation if the current position and orientation is unknown

Now consider the more general problem in which the robot is given a map of its environment, but is not told its location and orientation with respect to the map. In this section we show an algorithm for performing two tasks at once: validating the correctness of the map, and, in case the map is correct, finding the correspondence between the map and the world. The latter is the SELF-LOCATION PROBLEM, which in our case involves identifying the initial vertex on the map and/or the correct "orientation" (i.e. the mapping between physical exits from that vertex and map edges incident on that vertex).

Two major issues are involved here:

1. Under what conditions does the problem have a unique solution? In graphs with symmetries it may not be possible to uniquely identify the robot position.
2. Can the problem can be solved efficiently? This paper gives an algorithm for SELF-LOCATION problem that uses $O(N^3)$ moves.

To solve the general problem, we first form all possible hypotheses using the given map, corresponding to all possible initial vertices and orientations (i.e. their reference edges) in the map. The number of such hypotheses is

$$\sum_{i=1..N} degree(v_i) \ \ \forall \ \ vertices \ \ v_i \in V \qquad (1)$$

If D is the maximum degree of any vertex, the number of hypotheses is $O(ND)$. Assuming that $D = O(N)$, there are $O(N^2)$ hypotheses in total (in typical real environments $D << N$).

The idea of the algorithm is to carry out exploration but, at the same time that we form the explored subgraph S and the list of unexplored edges U, we also establish a correspondence between their elements (vertices and edges) and elements of each hypothesis. Such a correspondence creates expectations about the vertices and edges that are being explored, which, when violated, cause the rejection of a hypothesis. In other words, the robot "imagines" that each hypotheses is correct until this is contradicted by what is found in the real world.

The above algorithm is identical to the exploration algorithm described in [Dudek *et al.*, 1991], except that additional data structures are added to it. We generate one map for each hypothesis, and we establish links from the edges and vertices of each hypothesis to the edges and vertices of the explored subgraph. With each step of the exploration algorithm, we scan all of the hypotheses and then check whether their expectations are satisfied, rejecting the hypotheses for which they are not. It should be noted that in realistic situations, pruning of most hypotheses would occur extremely quickly. For example, all hypotheses that have an incorrect degree for the starting node are immediately falsified. The algorithm that performs self location has three major stages, each of which consists of an exploration action and result followed by management of the set of map hypotheses. These stages are as follows.

Exploration action: Vertex Validation. A step of the algorithm consists of the selection of an unexplored edge e from U, and validation of the vertex v_2 at the other end of edge $e = (v_1, v_2)$. Validation of the vertex consists of placing the marker there and visiting all vertices of S, staying inside S, looking for the marker. If the marker is found at vertex v_i of the explored subgraph, then vertex v_2 is identical to the already known vertex v_i.

Hypothesis testing action: Vertex validation. Each hypothesis expects a specific v_i, and if the true v_i is not the expected one, the hypothesis is rejected. For the remaining hypotheses, v_i is linked with the appropriate vertex, as well as all the incident edges.

Exploration action: Vertex in S and Edge ordering. In case vertex v_2 is in S, edge $e = (v_1, v_2)$ must be assigned an index with respect to the edge ordering of vertex v_2. To determine this, the robot drops the marker at v_1 and goes back to v_2 along the shortest path in the explored subgraph S. At v_2, it tries going out of the vertex along each of its incident edges. One of them will take the robot back to v_1, which the robot will immediately recognize by the existence of the marker. Note that the index of e with respect to the edge ordering of v_1 is known by construction. Edge e is added to the subgraph S and removed from U.

Hypothesis testing action: Vertex in S and Edge ordering. Each hypothesis expects a specific index for e with respect to vertex v_2, and therefore if the true index is not the same as expected, the hypothesis is rejected.

Exploration action: Vertex not in S. If the marker is not found anywhere in S, then vertex v_2 is not in the subgraph S and therefore must be added to it. The unexplored edge e is also added to S, which has now been augmented by one edge and one vertex. Edge e is the reference edge for vertex v_2. All other edges incident on v_2 are assigned an index with respect to it and added to U.

Hypothesis testing action: Vertex not in S. Each hypothesis expects a specific degree for v_2. Therefore, if the true degree is not the same as expected, the hypothesis is rejected.

The above algorithm performs validation and self-location simultaneously, since only the following cases are possible:

1. The map is correct, in which case a single hypothesis will not get rejected until the whole graph has been explored. This hypothesis establishes the proper mapping between the map and the world.
2. The map is correct and contains symmetries, in which case more than one hypothesis may remain valid until the end. In this case we can safely discard all valid hypotheses but one (they are identical anyway), and use it for solving path planning problems in that world.
3. The map is incorrect, in which case all hypotheses get rejected sooner or later.

In the case of a correct map with symmetries, the self-location problem is unsolvable without additional information. Examples of additional information which may be brought to bear include: metric information on distances and orientation, the ability to check whether a region with a given cycle as its boundary contains a vertex not belonging to the cycle, or the ability to decide which side of an edge a vertex is on - treating edges as infinite lines that divide 2D space into two [Levitt and Lawton, 1990].

6 Discussion and Conclusions

We have considered issues relating to the use of *a priori* maps of an environment. A realistic problem is that of how to deal with changes to the environment or errors in the map. We call this the MAP RECOVERY PROBLEM: given a partially incorrect map of an environment, how can one (optimally or near-optimally) determine where the errors in the map are and correct them. This could arise, for example, either is a map was constructed incorrectly or if the environment changed. Assumptions are the same as those for the validation algorithm. Of course, if the map is totally different from the world, one has to discard it and run the full exploration algorithm. In general, there are various ways in which one map can be modified to produce another; the precise definition of optimality is crucial. However, if we limit the allow-able set of transformations, it may be possible to devise efficient algorithms for fixing a "defective" map. For example, possible constraints could be: to assume damage is local, only node deletion might be allowed but no node addition, only edge deletion might be allowed, etc. Many such constraints have real world analogues (for example, only edge deletions might correspond to navigating in an office where some of the doorways had been closed). We leave this as a future research problem.

We have presented new algorithms that suggest how a mobile robot might make use of *a priori* maps of a known environment. The SELF-LOCATION and MAP VALIDATION problems suggest how a robot with a map could initially begin using it once out of the packing crate it was shipped in. The algorithms presented here have been implemented and a number of experiments has been conducted on several types of random graphs [Bollobas, 1985]. Comparisons between the actual performance of the exploration and the validation algorithm, as well as three-dimensional examples, are presented in [Dudek *et al.*, 1993]. In another paper of this proceedings [Dudek *et al.*,], the case of exploration without markers is being examined.

References

[Arkin, 1990] R. Arkin. Integrating behavioural, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[Ayache and Faugeras, 1989] Nicholas Ayache and Olivier D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Journal of Robotics and Automation*, Vol. 5, NO.6:804–819, 1989.

[Basye and Dean, 1990] Kenneth Basye and Thomas Dean. Map learning with indistinguishable locations. In M. Henrion L. N. Kanal J. F. Lemmer, editor, *Uncertainty in Artificial Intelligence 5*, pages 331–340. Elsevier Science Publishers, 1990.

[Bollobas, 1985] B. Bollobas. *Random Graphs.* Academic Press, London, 1985.

[Bondy and Murty, 1976] J. A. Bondy and U. S. R. Murty. *Graph Theory With Applications.* North-Holland, New York, 1976.

[Brooks, 1986] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, March 1986.

[Brooks, 1989] R. Brooks. A robot that walks: Emergent behaviours from a carefully evolved network. *Neural Computation*, 1(2):253–262, Summer 1989.

[Connell, 1990] J. Connell. *Minimalist Mobile Robotics: A Colony-style Architecture for an Artificial Creature.* Academic Press, Boston, MA, 1990.

[Cox, 1989] I. Cox. Blanche: Position estimation for an autonomous robot vehicle. In *Proceedings of the IEEE/RSJ International Workshop on Robots and Systems (IROS)*, pages 432–439, 1989.

[Davis, 1986] Ernest Davis. *Representing and Acquiring Geographic Knowledge.* Pitman and Morgan Kaufmann Publishers, Inc., London and Los Altos, California, 1986.

[Dudek *et al.*,] G. Dudek, P. Freedman, and S. Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *this proceedings*.

[Dudek *et al.*, 1988] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Robotic exploration as graph construction. Technical Report RBCV-TR-88-23, Research in Biological and Computational Vision, Department of Computer Science, University of Toronto, 1988.

[Dudek *et al.*, 1991] G. Dudek, M. Jenkin, E. Milios, and David Wilkes. Robotic exploration as graph construction. *IEEE Trans. on Robotics and Automation*, 7(6):859–864, 1991.

[Dudek *et al.*, 1993] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes. Map exploration, validation and self-location in a graph-like world. Technical report, Department of Computer Science, York University, 1993.

[Elfes, 1987] A. Elfes. Sonar-based real-world mapping and navigation. *IEEE Journal of Robotics and Automation*, 3(3):249–265, 1987.

[Engelson and McDermott, 1992] S. P. Engelson and D. V. McDermott. Maps considered as adaptive planning resources. In *AAAI Fall Symposium on Applications of Artificial Intelligence to Real-World Autonomous Mobile Robots*, Cambridge, MA, 1992.

[Guibas and Stolfi, 1984] L. Guibas and J. Stolfi. Primitives for the manipulation of general subdivisions and the computation of Vornoi diagrams. *ACM Transactions on Graphics*, 4, 1984.

[Kuipers and Levitt, 1988] B. Kuipers and T. Levitt. Navigation and mapping in large-scale space. In *AI Magazine*, pages 25–43, 1988.

[Latombe, 1991] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Standford University, 1991.

[Levitt and Lawton, 1990] T. Levitt and D. Lawton. Qualitative navigation for mobile robots. *Artificial Intelligence*, 44(3):305–360, August 1990.

[Lumelsky and Stepanov, 1987] V. Lumelsky and A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(4):403–440, 1987.

[Mataric, 1992] M. Mataric. Integration of representation into goal-driven behaviour-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[Meng, 1987] A. Meng. Free space modelling and geometric motion planning under location uncertainty. In *Workshop on Spatial Reasoning and Multisensor Fusion*, pages 430–440. Morgan Kaufmann, 1987.

[Smith and Cheeseman, 1986] R. Smith and P. Cheeseman. On the representation and estimation of spatial uncertainty. *International Journal of Robotics Research*, 5(4):56–68, Winter 1986.