Locally-Optimal Navigation in Multiply-Connected Environments without Geometric Maps

Benjamin Tovar† Steven M. LaValle† Rafael Murrieta‡

†Dept. of Computer Science ‡Beckman Institue
University of Illinois
Urbana, IL 61801 USA
{btovar, lavalle, murrieta}@uiuc.edu

***Abstract*— In this paper we present an algorithm to build a sensor-based, dynamic data structure useful for robot navigation in an unknown, multiply-connected planar environment. This data structure offers a robust framework for robot navigation, avoiding the need of a complete geometric map or explicit localization, by building a minimal representation based entirely on critical events in online sensor measurements made by the robot. There are two sensing requirements for the robot: it must detect when it is close to the walls, to perform wall-following reliably, and it must be able to detect discontinuities in depth information. It is also assumed that the robot is able to drop, detect and recover a marker. The navigation paths generated are optimal up to the homotopy class to which the paths belong, even though no distance information is measured.**

I. Introduction

The goal of this work is to develop robotic algorithms and systems with minimal sensing requirements, which are able to perform sophisticated visibility-based tasks. Our motivation is to overcome some of the problems that classical approaches obtain, such as mapping uncertainty, registration, localization errors, and unpredictable control errors. Such problems arise because previous algorithmic efforts have often assumed the availability of perfect geometric models.

We believe that reliability can be increased by developing algorithms and mobile robots that minimize the information requirements. By constructing an algorithm and control law that use information directly from the robot sensors, it may be possible to solve the problem while eliminating the need to make potentially-flawed measurements. Focusing on the particular task at hand, many of the classical requirements are eliminated. This approach can provide low-cost solutions to challenging problems, while achieving greater reliability in the face of uncertainties.

The idea of using *minimal representations* was popularized in the context of manipulation planning in [7], [8]. Within mobile robotics, on-line models have been used for navigation [11], [12], [13], [15], target tracking [9], pursuit-evasion [17], and localization [3], [5], [10], [18], [19].

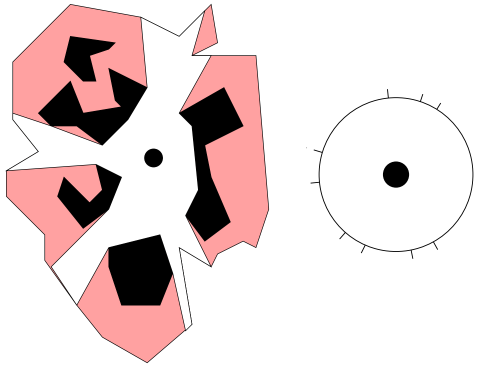
The work presented here is the continuation of a previous effort, in which only simply-connected environments were considered [20]. In the context of minimal representations, this work proposes new algorithms for environment exploration, navigation, and object location for environments with nonconvex obstacles. A dynamic tree data structure is proposed to serve as a topological map of the environment, in which geometric information (such as lengths, angles, distances, or segments) is not necessarily represented.

This tree is called *dynamic* because it is updated with visibility critical events as the robot moves in the environment. A path in this tree gives a sequence of critical events that the robot must follow to reach different places. We will show that once this dynamic tree is constructed, the path traversed by the robot between two locations is optimal up to a homotopy class with respect to distance, and that it is possible to find fixed objects in the environment.

II. Problem definition

The robot is required to explore an unknown environment, learning the location of certain *interesting objects*. When the exploration phase is finished, the robot must move efficiently and reliably between any two locations using sensor feedback. Under these conditions, the robot will be able to perform useful tasks, such as going where a certain object is located, moving an object from one place to another, or gathering all of the objects to one place.

The robot is modeled as a point moving in a connected open set R in the plane. Let $O = \bigcup_{i=1}^{n} o_i$ be the set of pairwise disjoint nonconvex simply-connected obstacles, in which $o_i \subset R$ for $i = [1, 2, \ldots, n]$ is an open set. Let $F = R \setminus O$ be the free space. Assume that the boundary of F is piecewise smooth. The robot is only able to move in F. Let $G = \{g_1, g_2, \ldots, g_m\}$



(a) (b)

Fig. 1. The robot's view of the environment. The environment is shown on the left, with the white region denoting the visibility of the robot. On the right it is shown the angular position of the discontinuities detected in the visibility region.

be the collection of m interesting objects in the environment. It is assumed that each $o \in O$ and each $g \in G$ is uniquely identifiable. Elements in O are obstacles for movement and visibility, while elements in G are considered as points that the robot recognizes when it sees them. Also, during exploration, the robot uses a marker. The robot has no previous knowledge of R, O or G.

The robot is able to detect discontinuities in depth information. Each discontinuity corresponds to a portion of F that is not visible to the robot. As an example refer to Figure 1. Figure 1.b gives the location of discontinuities in depth information for the environment shown in Figure 1.a.

Although the precise distances to the walls may be unknown, it is assumed that the robot has a kind of edge detector that can detect each of the discontinuities, and return their direction relative to the robot's heading. Each discontinuity will be referred to as a *gap*, as in [17]. For polygonal environments these gaps are referred to as *spurious edges* of the visibility polygon in [10]. It is assumed that the robot can track the gaps at all times, and record any topological change, as we will discuss later.

Given the gaps that the robot detects at a given time, it is possible to command the robot to move toward a given gap. This sensor-feedback movement is defined as *chasing a gap*.

We make a general position assumption that no line is tangent to more than two points of the boundary of F.

III. The Dynamic Data Structure

In this section we present the dynamic data structure, the tree T_h, that will encode R, O, and G. First we will define the visibility events, and how they are encoded into T_h. Next, we will introduce the navigation algorithm, assuming that a partial construction of T_h is available. Finally, we will propose an algorithm to construct T_h for an unknown environment.

A. Encoding visibility critical events

There are four possible ways in which the gaps change: a new gap appears, an existing gap disappears, two or more gaps merge into one gap, and a single gap splits into two or more gaps. These changes are called *gap critical events*.

Appearance and disappearance events occur when the robot crosses generalized inflections of the boundary F. Merge and split events occur when the robot crosses the rays that extend generalized free bitangents of F (see Figure 2). An *inflection* is found by extending a ray outward from an inflection point of the boundary of F. A *bitangent* is a closed line segment whose supporting line is tangent at two points of the boundary of F. It is called *free* if lies entirely in F [16]. We use the term *generalized* as in [14] to extend the definition of inflections and bitangents to polygonal boundaries.

Because of the general position assumption, when a gap splits, it yields exactly two new gaps. Also, when gaps merge, exactly two gaps merge to yield a single gap.

The data structure T_h is a dynamic tree. The root of T_h moves along with the robot. Therefore, T_h is a local topological map, not a global one as in [2], [4], [6]. Each node in T_h encodes topological information about the environment. A child node of the root represents a gap that is detected in the robot's current position. Child nodes are maintained in circular order, as they are detected. As the robot moves in F, gap-critical events are triggered and encoded into T_h as follows:

1) **A gap appears.** A node corresponding to the gap is added as a child of the root of T_h, in a location that preserves the circular ordering of gaps.
2) **A gap disappears.** The corresponding node is removed from T_h.
3) **Two gaps merge.** The two corresponding children of the root become the children of a new node, n, and n becomes a child of the root (see Figure 3).
4) **A gap splits.** The corresponding node, child of the root, will be replaced with two children nodes.

Assume that the gaps are detected in counterclockwise order. A label of *left* or *right* can be assigned to each gap. Each label corresponds a transition in a discontinuity from "far to near" or "near to far",

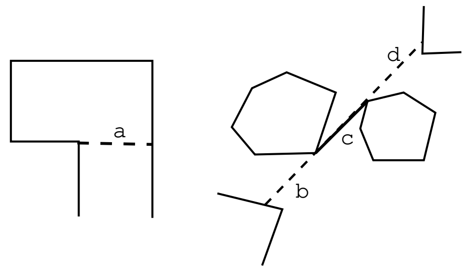


Fig. 2. Generalized inflections and bitangents of the boundary of F. Visibility critical events are triggered when the robot crosses inflections (a) and the rays that extend bitangents (b and d) of F.

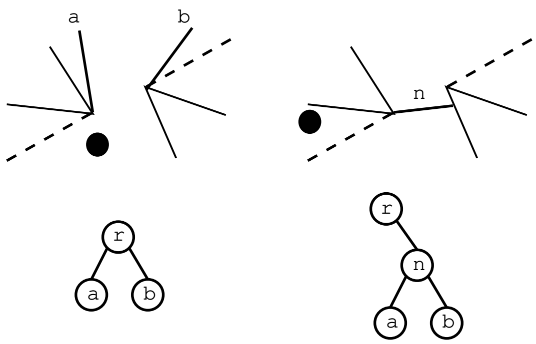


Fig. 3. Merge critical event. When the robot crosses one ray extending the bitangent, gaps a and b merge into gap n. In T_h, nodes encoding a and b become children of the node encoding n. The root of T_h is labeled with r, and the robot position is indicated with the black disk.

respectively, and indicates the side to which the region of F is hidden behind the gap. These transitions, together with the assumption that every obstacle $o \in O$ is uniquely identifiable, make it possible to associate gaps with obstacles. The "beginning" of the gap is the nearest obstacle (or R) in the transition, and the "end" is the farthest one (see Figure 4). For nonconvex obstacles these may be the same. When an obstacle o is visible, it is (partially) blocking the visibility of the robot of other obstacles or a portion of R. From the robot's perspective, o will be the beginning of some gaps. Thus, every obstacle is associated with at least one gap:

Observation 1: For every obstacle $o \in O$, there will be a gap beginning at o for some position of the robot. Also, if R is nonconvex, for some robot position there will be at least one gap beginning at R.

If an object $g \in G$, is in the visibility region of the robot, and its angular position is close to a gap η, then we say that η encodes a path to g. Following a path that takes the robot to η will make g visible. This

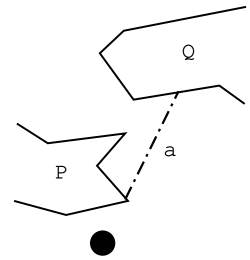


Fig. 4. Beginning and end of a gap. From the robot's perspective the obstacles P and Q are the beginning and end of the gap a, respectively.

idea will become clear in Section III-B. We also have a guarantee in the number of gaps associated with an object:

Observation 2: If the robot has seen the whole free space F, for each object $g \in G$, if there are no nodes in T_h associated with a gap that encodes a path to g, then g is visible from the current position of the robot.

This observation is true because at the instant when an object is not longer visible, a gap η refers to the invisible region containing the object. The object is associated with η.

Note that only the angular order, not the precise angular position of the gaps, is recorded in the encoding of T_h.

A similar data structure was presented in [1], in which a shortest path tree is updated when a point crosses *constraint lines*. However, in that work it is assumed that a simple polygon is given *a priori*, which represents a perfect map.

B. The navigation algorithm

The central idea of the navigation algorithm is to chase the gap associated with a given goal. For example, chasing a gap that begins in an obstacle will take the robot to the boundary of that obstacle, or will make a certain object visible at some point. As the robot moves, the node β encoding this information may not be a child of the root. Because of merging events, it may be in a lower depth in the tree. The robot must chase the child of the root that is an ancestor of β. When the corresponding child of the root splits, β will be one level closer to the root. This procedure is repeated until β is a child of the root, in which case following β will reach the goal.

Assume that a partial construction of T_h is available, and that a goal s is associated with some node of T_h. The goal s may be a gap, an object in G or an obstacle in O. The navigation algorithm to reach s is shown in Figure 5. A sequence of gaps, H, encoding a path to s is extracted from T_h. One by one the robot chases

NAVIGATION(T_h, s)
    $H \leftarrow$ FIND_SEQ(T_h, s)
    while $H \neq \emptyset$ **do**
     $h \leftarrow$ **pop** (H)
     until disappears(h) **or** splits(h) **or** in_goal(s) **do**
       CHASE(h)

Fig. 5. Navigation algorithm. A sequence of gaps that encode a path to the goal s is generated from T_h. The robot follows this sequence until reaching the goal.

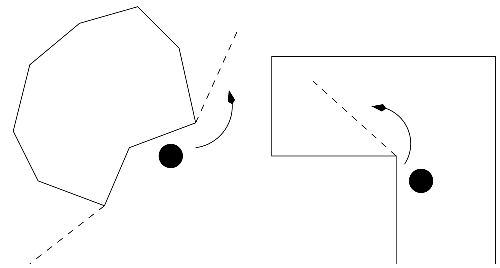


Fig. 6. Not all gaps disappear when chased. If the obstacles in which the gap begins and ends are different, the gap does not disappear (a). The gaps disappears if the gap begins and ends in the same obstacle (b).

the gaps in H, until the goal is reached. If s is a gap, the goal is reached when s splits or disappears (s will be the last gap in H). If s is an obstacle, the task ends when the robot *touches* s, and if s is an object, the task is completed when s is visible. The predicate *in_goal* handles these last two cases.

C. The exploration algorithm

For the exploration, the nodes of T_h are classified into four types:

- **Primitive.** Primitive nodes encode gaps that appear as the robot moves. A primitive node refers to some part of F which the robot has already seen (this part of F was visible before the corresponding gap appeared).
- **Branch.** Branch nodes are the parents of nodes corresponding to gaps which merge.
- **Non-primitive.** These correspond to nodes associated with gaps that the robot has not yet explored, or to gaps that did not disappear but are associated with some feature of the environment.
- **Block.** A block node encodes a gap that does not increase the robot's knowledge of the environment. The notion of the block nodes and why they are needed will be described later in this section.

When the robot is placed in a new environment, all of the leaves of T_h are marked non-primitive. This is because the robot has not yet seen what is behind the corresponding gaps. If $O = \emptyset$, when the robot chases a gap, the gap is guaranteed to split or to disappear. An appropriate exploration strategy commands the robot to chase every non-primitive gap, until all the leaves of the tree are primitive. Such a strategy is presented in [20].

In the general case, when $O \neq \emptyset$, one complication is presented because there are gaps that never disappear. As shown in Figure 6.a, chasing a gap until it disappears may result in the robot going around an obstacle forever. Compare this situation with Figure 6.b, in which a gap disappears.

Another strategy must be defined to construct T_h, which guarantees that the robot will see the whole environment. In our solution, the robot follows the boundary of R and of every obstacle $o \in O$. From Observation 1, there will be gaps beginning at each obstacle. Using the information in T_h, the robot can determine how to reach some of the obstacles in the environment, as described in Section III-B. The robot chooses arbitrarily to follow the boundary of an obstacle that has not been transversed before. When this boundary has been completely traversed, a new obstacle is selected. Incrementally, the robot will determine how to reach every obstacle.

To follow the border of an obstacle, or the border of R, the robot drops a marker when it first touches the obstacle, or when it first touches the border of R. The robot executes wall-following motions until it detects the marker. At this time the robot has completely followed the border, and it picks up the marker. Note that it would be very difficult, if not impossible, for the robot to detect when it had finished following a border, using only gap-sensing capabilities.

The exploration strategy is summarized in Figure 7. The tree is initialized with the non-primitive nodes corresponding to the gaps detected initially. During execution, a list Q is computed, consisting of all of the boundaries the robot knows how to reach, but has not explored. The exploration ends when Q is empty.

There is an interesting complication when building T_h. Since the free space may be multiple connected, the homotopy class of paths between two locations may not be unique. As proved in [20], in absence of the obstacles the paths generated by T_h are optimal. With obstacles this is no longer true. From the robot's perspective, all paths through T_h are equivalent, since the robot lacks distance information. Therefore, paths through T_h that use the minimum sequence of gaps are selected. Suppose that the sequence of nodes $v_i, ..., v_m$ is selected to reach goal s. The association of s with any node $u \in T_h$ is removed if $u \neq v_m$. If an object is visible, then the association of the object with all

```
EXPLORATION()
    T_g ← INITIALIZE_TREE
    Q ← REMAINING_BNDR(T_g)
    while Q ≠ ∅ do
        o ← pop (Q)
        NAVIGATION(T_g, o)
        DROP_MARKER
        FOLLOW_BOUNDARY(o)
        RECOVER_MARKER
        Q ← REMAINING_BNDR(T_g)
```

Fig. 7. Exploration algorithm. The robot follows the boundary of each element of the environment. When the last obstacle boundary has been traversed, the robot has seen the whole environment and the construction of T_h is completed.

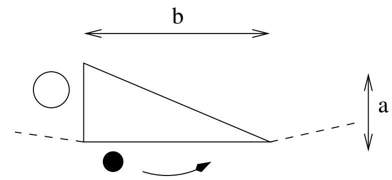


Fig. 8. A worst case navigation example. If $b \gg a$, and if the robot chooses to follow the gap on the right, practically the whole triangle boundary will be followed to reach the circle.

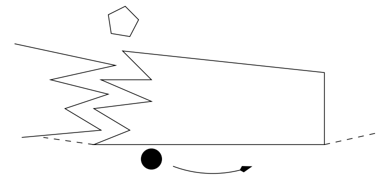


Fig. 9. Paths with shortest gaps sequence are preferred. The robot will follow the path on the right, because it offers less gaps to chase, although is not the shortest path. With only gaps information, the robot cannot do better.

the nodes is removed. If a leaf of T_h is not associated with a goal, it is marked as *block*. If a branch node is not associated with a goal, and all its children are marked as block, the branch node is marked as block, and all the children are eliminated. Two block nodes cannot merge (only one is kept). A block node splitting yields two block nodes. A block node returns to a non-primitive status if is associated with a new goal. We now show that this algorithm is complete:

Proposition 3: The tree T_h will provide a path from the current position of the robot to each obstacle $o \in O$ and to each object $g \in G$.

Proof: By Observations 1 and 2 it is known that if an object or obstacle is visible in the environment, it will be associated with a gap and the corresponding node in T_h. When the robot moves and the object or obstacle is no longer visible, the path information is preserved by nodes merging in T_h. Even when some nodes are eliminated or marked as block, path information is preserved. Elimination of nodes occurs unless T_h encodes more than one path to the same goal. The data structure T_h will provide a path from the current position of the robot to all the objects and obstacles in the environment. ∎

D. Some comments on performance

Since the robot has no distance information, good performance in the distance traveled may not be expected. As discussed in Section III-C, all paths with a common goal are equivalent. As a consequence the path length may compare poorly with the distance traversed having a complete geometric map.

Refer to Figure 8. After going around the triangular obstacle once in the exploration phase, the robot travels to the circular obstacle. If the robot follows the gap on the right, and if $b \gg a$, the robot will follow almost the entire triangle boundary again. A decision based only in our gap-chasing model cannot do better. In [11], a similar problem is considered. It is solved by changing the direction of the navigation if the robot moves in the opposite direction to the goal. Without a measurement of direction, this is not possible under our gap model.

Although global optimality cannot be guaranteed, the path that the robot follows is optimal in the homotopy class to which the path belongs. By following gaps, the robot follows the tangent lines between the obstacles, which in turn are the edges of the tangent visibility graph.

It is worth noting that the number of gaps in a path sequence is an indicator only of how "cluttered" a region is, but generally is not related with the distance to travel. Consider the example of Figure 9. The robot will choose to follow the gap on the right to reach a goal, instead of the path beginning on the gap of the left. The path on the left is longer in the number of gaps to chase, but is shorter in distance.

IV. Simulations with results

We have implemented the algorithms shown in computer simulation. Figure 10 shows the changes in T_h before and after going around an obstacle. It is interesting to see that the root's children are the same, but at the end one of them has encoded paths to reach other places in the environment.

For the figures shown, the root of T_h is represented with the bigger black circle. The different shapes inside the nodes represent the objects associated with that node. The shapes inside the root node are the objects visible from the current robot's position. The color of the circle in the top of the nodes indicates the obstacle

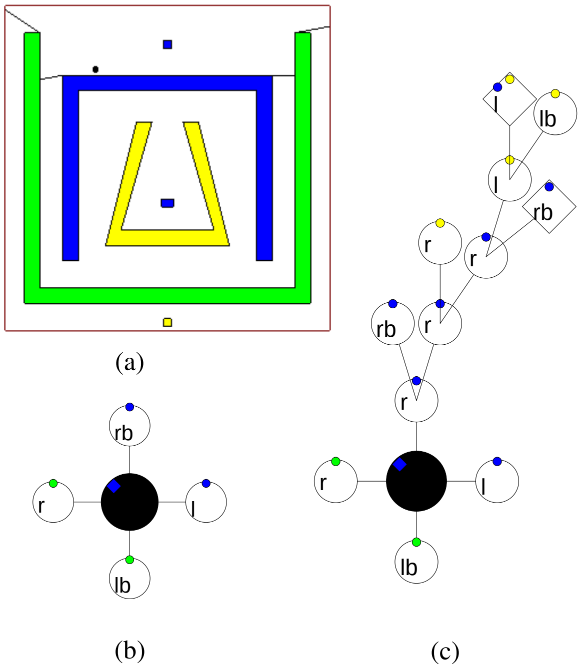


Fig. 10. The robot surrounds the darker obstacle in (a). In (b) it is shown the initial state of T_h, and in (c) the paths information gained after surrounding the obstacle completely. The robot is the black point, and the small polygons are the interesting objects.

in which the associated gap begins. Primitive nodes are drawn as squares. The “b” label indicates that the node is a block node. Finally, the labels “r” and “l” indicate that the associated gap hides the environment to the right or to the left, respectively.

In the example shown in Figure 11, after the exploration, the robot moves all objects to a single place, previously determined. The trees shown correspond to T_h after the exploration phase and during the delivery task.

The model we assume has been successfully demonstrated in a real robot. The robot setup used is shown in Figure 12. The reader is referred to [20] for a description of the implementation and some of the issues faced.

V. Conclusions

We have presented a dynamic data structure useful for robot navigation in a multiply-connected environment. Algorithms for construction and exploitation of this data structure have been described. These algorithms are based on the robot’s capability of detecting visibility critical events, more precisely, in changes of depth information. The navigation paths generated by the data structure are optimal, up to a homotopy class.

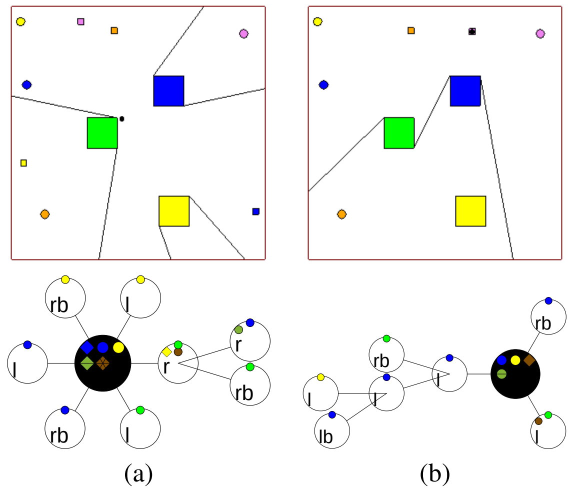


Fig. 11. Environment exploration and object delivery. After exploring the environment, the robot built the tree shown in (a). Using this tree, square objects are moved to a designated circular object. While moving in this task, the tree keeps updating, as shown in (b).

Fig. 12. Setup used to validated the gaps model. A robot was provided with two SICK lasers.

Given that the robot does not use geometric information, it is interesting how the data structure is able to capture paths between locations in the environment. We believe that the assumption in which every obstacle is uniquely identifiable is strong, and is not easily implementable in real robots. Adding another sensing capability, or detecting patterns of nodes in the data structure, may lead to a relaxation of this assumption, and we consider them as improvements in future work. The use of a marker may also be relaxed if the robot is provided with basic image processing capabilities.

VI. Acknowledgments

We thank Boris Simov and Shai Sachs for helpful discussions and suggestions. This work is supported in part under the NSF-CONACyT Grant 0296126 and the ONR Grant N00014-02-1-0488. The gaps model validation in real robots was done with equipment of ITESM Campus Ciudad de México, México.

VII. REFERENCES

[1] B. Aronov, L. Guibas, M. Teichmann, and Li Zhang. Visibility queries in simple polygons and applications. *Algorithms and Computation, 9th International Symposium, ISAAC '98, Taejon, Korea, December 14-16, 1998, Proceedings*, 1533, 1998.

[2] H. Bulata and Michel Devy. Incremental construction of a landmark-based and topological model of indoor environments by a mobile robot. In *IEEE Int. Conf. Robot. & Autom.*, 1996.

[3] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Int. Conf. Robot. & Autom.*, 17(2), April 2001.

[4] G. Dedeoglu, M. J. Mataric, and G. S. Sukhatme. Incremental, on-line topological map building with a mobile robot". In *Proceedings of Mobile Robots XIV - SPIE*, 1999.

[5] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *IEEE Int. Conf. Robot. & Autom.*, pages 190–197, Sacramento, CA, April 1991.

[6] Gregory Dudek, Paul Freedman, and Souad Hadjres. Using local information in a non-local way for mapping graph-like worlds. In *Proc. Int. Joint Conf. on Artif. Intell.*, pages 1639–1645, 1993.

[7] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Trans. Robot. & Autom.*, 4(4):369–379, August 1988.

[8] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[9] H. González-Baños, Cheng-Yu Lee, and J.C. Latombe. Real-time combinatorial tracking of a target moving unpredictable among obstacles. In *IEEE Int. Conf. Robot. & Autom.*, 2002.

[10] L. J. Guibas, R. Motwani, and P. Raghavan. The robot localization problem. In K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, editors, *Proc. 1st Workshop on Algorithmic Foundations of Robotics*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.

[11] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Trans. Robot. & Autom.*, 13(6):814–822, December 1997.

[12] I. Kamon, E. Rivlin, and E. Rimon. A new range-sensor based globally convergent navigation algorithm for mobile robots. In *IEEE Int. Conf. Robot. & Autom.*, 1996.

[13] K. N. Kutulakos, C. R. Dyer, and V. J. Lumelsky. Provable strategies for vision-guided exploration in three dimensions. In *IEEE Int. Conf. Robot. & Autom.*, pages 1365–1371, 1994.

[14] S. M. LaValle and J. Hinrichsen. Visibility-based pursuit-evasion: The case of curved environments. *IEEE Transactions on Robotics and Automation*, 17(2):196–201, April 2001.

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

[16] M. Pocchiola and G. Vegter. Minimal tangent visibility graphs. *CGTA: Computational Geometry: Theory and Applications*, 6, 1996.

[17] S. Rajko and S. M. LaValle. A pursuit-evasion bug algorithm. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 1954–1960, 2001.

[18] K. Romanik and S. Schuierer. Optimal robot localization in trees. In *Symposium on Computational Geometry*, pages 264–273, 1996.

[19] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence Journal*, 2001.

[20] B. Tovar, S. M. LaValle, and R. Murrieta. Optimal navigation and object finding without geometric maps or localization. In *IEEE Int. Conf. Robot. & Autom.*, 2003.