# Localization with Limited Sensing

Jason M. O'Kane and Steven M. LaValle

***Abstract*— Localization is a fundamental problem for many kinds of mobile robots. Sensor systems of varying ability have been proposed and successfully used to solve the problem. This paper probes the lower limits of this range by describing three extremely simple robot models and addressing the active localization problem for each. The robot, whose configuration is composed of its position and orientation, moves in a fully known simply connected polygonal environment. We pose the localization task as a planning problem in the robot's information space, which encapsulates the uncertainty in the robot's configuration. We consider robots equipped with (1) angular and linear odometers, (2) a compass and contact sensor, and (3) an angular odometer and contact sensor. We present localization algorithms for models 1 and 2 and show that no algorithm exists for model 3. An implementation with simulation examples is presented.**

***Index Terms*— information spaces, mobile robot localization, robots, robot sensing systems**

## I. Introduction

Localization, the task of systematically eliminating uncertainty in the pose of a robot, is widely regarded as a central problem in mobile robotics. A wide spectrum of sensor systems have been proposed for the localization problem, ranging from visibility sensors [1, 2, 3] to landmark detectors [4, 5, 6]. How complex a sensor system does localization truly demand? In this paper, we take a *minimalist* approach, describing two simple robots with which localization is still possible and a third for which localization is provably impossible.

Suppose a robot is given an accurate map of its environment, but has no knowledge of its configuration. The robot's goal is to move within the environment, gathering information about its configuration until the uncertainty is eliminated. We may consider this task as a planning problem with discrete stages, but this approach is complicated by the fact that the robot's configuration is unknown. This leads us to define the robot's *information space* and give methods for computing its *information state* within that space. Informally, the robot's information state is a set of candidate configurations in which the true configuration is known to lie. When the robot is finally localized, the information state contains only the robot's true configuration.

We consider the localization task for three distinct robot models:

- R1 – A robot equipped with angular and linear odometers. This robot can accurately rotate and translate through its environment, measuring each of these motions.
- R2 – A robot equipped with with a compass and contact sensor. This robot can, using its compass, orient itself with respect to a global reference frame, then move forward until its contact sensor detects the environment boundary.
- R3 – A robot equipped with an angular odometer and contact sensor. This robot can rotate with respect to a local frame and then move forward until reaching the environment boundary.

J. M. O'Kane (corresponding author) and S. M. LaValle are with the Department of Computer Science, University of Illinois at Urbana-Champaign, 201 North Goodwin Avenue, Urbana, IL 61801, USA. Email: {jokane, lavalle}@cs.uiuc.edu. Fax: +1-217-265-6591

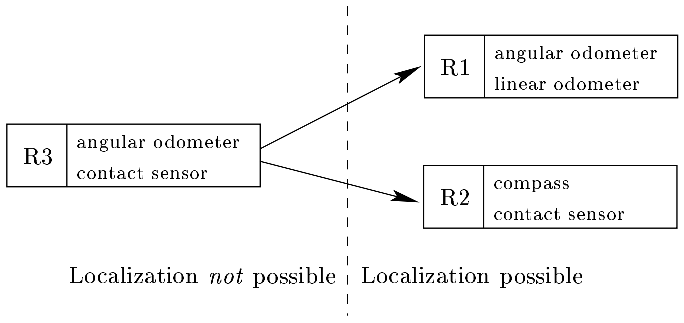

Fig. 1. Although R1 and R2 have only slightly stronger sensing than R3, they are capable of localization whereas R3 is not.

The main contribution of this paper is to classify these robots according to their ability to localize themselves. We show that R1 and R2 can localize themselves in polygonal environments, but R3 cannot.

The intention of this line of inquiry is to identify basic sensing requirements for robotic tasks. For a given task, some robot systems are capable of completing the task whereas others are not. Our goal is to search the space of robot systems for the boundary between the "can localize" and "cannot localize" regions. This boundary gives an indication of the *necessary* conditions on robot models for localization. In this paper we describe a very simple robot (R3) in the "cannot" set and show that small improvements to its angular sensing (R2) or linear sensing (R1) lead to models in the "can" set. See Figure 1.

The balance of this paper is organized as follows. We present related work in Section II. Section III formally defines our robot models and gives a problem definition. Sections IV and V describe localization algorithms for robot models R1 and R2, respectively. In Section VI we show that no localization algorithm exists for R3. Concluding remarks appear in Section VII.

Portions of this work appeared in preliminary form in [7] and [8].

## II. Related work

There are two primary lines of antecedent research. First, many works have studied the localization problem itself on theoretical and practical levels. Second, a recurring theme in robotics research has been the notion of *minimalism*, the idea that simple but carefully designed robotic systems can offer advantages in cost, efficiency and robustness over more complex systems that are richer in sensors and actuators.

### A. Localization

We can generally separate localization research into two flavors. *Passive localization* [2, 9, 10, 11] does not prescribe motions for the robot, but only provides methods for using sensor readings and externally-selected commands to estimate the robot's state. In this paper, we consider *active localization* problems, in which the goal is to prescribe motions for the robot in order to fully determine its position. Algorithms in this context are often expressed as online methods and evaluated in terms of their *competitive ratio* [12], which compares the lengths of paths generated by the algorithm to the length of the shortest possible path that could have been selected if the robot started with full information. In [13], the environment is constrained to an embedding of a bounded degree acyclic graph into $\mathbb{R}^n$ with sensing limited to the orientations of incident edges. This algorithm has competitive complexity $O(n^{2/3})$, in which $n$ is the number of leaves in the graph.

The problem of computing a localization strategy that minimizes the worst case distance traveled by a robot equipped with a visibility sensor was proved NP-hard in [14]. The optimal strategy can, however, be approximated (in the competitive ratio sense) and [14] gives an algorithm based on the visibility cell decomposition that does this. An important weakness of this algorithm is that it relies on motion commands that direct the robot into visibility cells that may be arbitrarily small. In [15], this difficulty is addressed by introducing randomization. Other work considers the problem in the framework of approximation algorithms [16].

Others have used probabilistic methods for active localization. In [17], the robot localizes itself with respect to a metric map by representing its knowledge as a probability distribution over its state space and selecting actions that reduce the entropy of this distribution. Jensfelt and Kristensen [18] address similar problems, but use a topological map.

### B. Minimalism

Both sensors and actuators are subject to significant errors in precision and accuracy. Effective robots must be robust to these errors. Starting, perhaps, with Whitney's critique of mid-1980's robotics research [19], an approach has arisen in which these difficulties are dealt with by designing extremely simple robots that exploit the compliant properties of the system in question to execute their assigned tasks. This approach has been called *minimalist robotics*. It has been applied to problems in manipulation for part orientation [20, 21, 22]. Bug algorithms [23, 24, 25, 26] are used for navigation by robots capable of moving toward obstacles and following walls. In [27], the robot has an extremely crude range sensor that can only detect discontinuities in depth information. As the robot explores its environment, this information is used to construct a data structure that allows for optimal navigation between previously visited locations. More explicit maps can be built with a range sensor by traversing the generalized Voronoi graph of the environment boundaries [28, 29].

## III. Problem Statement

In this section, we formally define an active, global localization problem for robot models R1, R2, and R3. We also define the robots' information space, which is the machinery we use to solve the problem.

### A. Actions, transitions, and observations

Allow a point robot with orientation to move in a compact simply connected polygonal environment $W \subset \mathbb{R}^2$. Assume that the rotational symmetry group of $W$ contains only the identity symmetry[1]. Let $\partial W$ denote the boundary of $W$, which is itself a subset of $W$. The robot has access to an accurate map of $W$, including its orientation in the plane. Since the robot's orientation is relevant, the configuration space is $\mathcal{C} = W \times S^1$, in which $S^1$ is the set of directions in the plane, represented as unit vectors in $\mathbb{R}^2$.

The space of available actions depends on the robot model. For each, we define an action set $U$ and a state transition function $f : \mathcal{C} \times U \to \mathcal{C}$.

- Robot R1 can, at each time step, issue either of two types of commands. First, the robot may rotate by a commanded amount. Since the robot has an angular odometer, we assume that rotation commands are executed precisely. Second, a translation command may be issued, instructing the robot to advance forward by a given distance. The actual distance traveled may be less than the commanded distance, if the robot reaches the boundary of the environment first. Formally, let $U = S^1 \sqcup [0, \infty)$ denote the robot's action space, in which elements of $S^1$ denote relative rotation commands and elements of $[0, \infty)$ denote translation commands. If $u \in S^1$, then $f(x, u)$ is the appropriate change of orientation of $x$. If $u \in [0, \infty)$, then $f(x, u)$ computes the appropriate forward translation of $x$ within $W$.
- The action space for R2 is the unit circle $U = S^1$. A single $u \in U$ represents a rotation to orient the robot in a given direction, followed by a motion forward to the environment boundary. The state transition function $f$ maps a configuration action pair $(x, u)$ to the opposite endpoint of the maximal segment in $W$ starting at $x$ and having direction $u$ in the global frame. Note that because the robot has a compass, we assume it can orient itself as it wishes; therefore the current orientation (specified as part of its configuration) is not relevant to R2.
- The model for R3 is similar to that of R2, but with the motion directions specified relative to the robot's *current* orientation, rather than with respect to a global reference frame. We still have $U = S^1$, but $f$ is modified to interpret $u$ as a motion direction relative to the robot's current heading.

[1]This assumption is important because if $W$ has a nontrivial symmetry group, the algorithm of Section IV is effective only up to symmetry. This technicality is addressed in greater detail in Section IV-F.

An iterated version of $f$ that applies several actions in succession will also be useful:

$$f(x, u_1, \ldots, u_k) = f(\cdots f(f(x, u_1), u_2) \cdots), u_k). \tag{1}$$

Consider a sequence of commands $u_1, \ldots, u_K$. This, combined with an initial state $x_1$, defines a sequence of configurations $x_1, \ldots, x_{K+1}$ governed by $x_{k+1} = f(x_k, u_k)$.

After each action, the robot receives an *observation* from its sensors. One may regard this observation as a “hint” regarding the true configuration of the robot. Let $Y$ denote a space of possible observations and $h : \mathcal{C} \times U \to Y$ denote an *observation function* that gives the sensor reading that would result from choosing a particular action from a particular configuration. Both $Y$ and $h$ depend on the robot model.

- For R1, we must consider the feedback provided by the linear odometer. Choose $Y = [0, +\infty)$ as the observation space, in which an observation $y \in Y$ indicates that in executing the previous action, the robot's translation had magnitude $y$. Rotations always succeed without providing useful feedback, so $h(x, u) = 0$ when $u \in S^1$.
- Neither R2 nor R3 have sensors that provide useful feedback about the environment. For each, the capabilities of the sensors are instead modeled in the action sets. We assume that the compass (for R2) and the angular odometer (for R3) are used as part of a closed loop control system the correctly executes the desired rotation. Similarly, the contact sensor is used to stop the robot when it reaches the environment boundary, but does not provide sensor observations as such. Therefore, for both R2 and R3, we select a dummy observation space $Y = \{0\}$ and define $h(x, u) = 0$ for all configurations $x$ and all actions $u$.

Lastly, we can define a sequence of observations $y_1, \ldots, y_K$ so that $y_k = h(x_k, u_k)$.

### B. The information space

Since the robot's initial configuration is unknown, it can use only the actions it has selected and the observations it has received to draw conclusions about its configuration. To handle this complexity in a concrete way, we consider the problem as a search through a space we call the robot's *information space*. To begin, consider what information is available from the robot's action and observation sequences.

**Definition 1:** A configuration $x_k \in \mathcal{C}$ is *consistent* with an action sequence $u_1, \ldots, u_{k-1}$ and an observation sequence $y_1, \ldots, y_{k-1}$ if there exists some configuration $x_1 \in \mathcal{C}$ such that $x_k = f(x_1, u_1, \ldots, u_{k-1})$ and $y_j = h(f(x_1, u_1, \ldots, u_{j-1}), u_j)$ for each $j = 1, \ldots, k$.

The intuition is that the consistent configurations $x_k$ are those for which there is some starting configuration from which executing the given action sequence would produce the given observation sequence and leave the robot at $x_k$. The set of consistent configurations provides a concise way of describing the information available to the robot.

**Definition 2:** Suppose the robot has chosen actions $u_1, \ldots, u_k$ and received sensor readings $y_1, \ldots, y_k$. The *information state* $\eta_k$ is the set of all configurations consistent with these actions and observations. The *information space* $\mathcal{I}$ is the set of all information states, in this case the power set of $\mathcal{C}$.[2]

Transitions in information space are determined by the current information state, the selected action, and the observation from the sensors. The *information transition function* has the form $F : \mathcal{I} \times U \times Y \to \mathcal{I}$, and can be defined in terms of $f$ and $h$:

$$\begin{aligned} F(\eta_k, u_k, y_k) = & \bigcup_{x \in \eta_k} \{f(x, u_k)\} \\ & \cap \{f(x, u_k) \mid x \in \mathcal{C}, y_k = h(x, u_k)\} \end{aligned} \tag{2}$$

Thus, we have a sequence of information states $\eta_1, \ldots, \eta_K$ governed by $\eta_{k+1} = F(\eta_k, u_k, y_k)$.

We approach the task of localization as a planning problem in $\mathcal{I}$. Initially the robot has no knowledge of its configuration, so the initial information state $\eta_1 = \mathcal{C}$ contains the entire configuration space. The goal region is

$$\mathcal{I}_G = \{\eta \subset \mathcal{C} \mid |\eta| = 1\}. \tag{3}$$

A plan is a feedback strategy on $\mathcal{I}$: We want a function $\mathcal{I} \to U$ such that, regardless of the robot's initial configuration, repeatedly executing the actions chosen by this function leads in finite time to an information state in $\mathcal{I}_G$. For R1, we must specify a policy $\pi : \mathcal{I} \to U$. For R2 and R3, there is no meaningful feedback, so it is sufficient to choose a sequence $u_1, \ldots, u_K$ of actions that eliminates the state uncertainty. We call such a sequence a *localizing sequence*.

## IV. Localization with odometry

In this section we present an algorithm to solve the localization problem described in Section III, for robot model R1. Recall that R1 is equipped with linear and angular odometers. An overview appears in Algorithm 1. The algorithm is “online” in the sense that the commands it issues depend on the observations obtained as the robot is executing. Indeed, there is no external “plan” computed ahead of time; instead we may regard Algorithm 1 itself as a plan in the sense that it defines a feedback strategy on the information space.

### A. Algorithm overview

The algorithm tracks the robot's information state $\eta_k$ throughout the execution. The first step, InitialActions, issues several commands to move from the initial condition ($\eta_1 = \mathcal{C}$) to an information state of finite cardinality. This process is described in Section IV-B. For some degenerate but potentially interesting environments, InitialActions fails to generate a finite information state, instead possibly leaving one or more continua expressed as intervals on the boundary of $W$. The function EliminateSegments issues commands guaranteed to reach an information state devoid of such segments. This issue is dealt with in Section IV-C.

[2] In this context we consider only the *nondeterministic information space*, which is based on set membership. Other formulations use probabilistic reasoning or some other technique to manage the history data. See, for example, Chapters 11 and 12 of [30].

**Algorithm 1** LOCALIZER1($W$)

```
$(\eta_k, k) \leftarrow$ INITIALACTIONS($W$)
$(\eta_k, k) \leftarrow$ ELIMINATESEGMENTS($W, \eta_k$)

while $|\eta_k| > 1$ do
    Select $x_1, x_2$ from $\eta_k$.
    $W_{x_1} \leftarrow$ TRANSFORMTOLOCALFRAME($W, x_1$)
    $W_{x_2} \leftarrow$ TRANSFORMTOLOCALFRAME($W, x_2$)
    $p \leftarrow$ FINDPOINTINONLYONE($W_{x_1}, W_{x_2}$)
    $(u_k, \ldots, u_{k'}) \leftarrow$ PATHINPOLYGON($W_{x_1}, (0,0), p$)
    while $x_1, x_2 \in \eta_k$ do
        $y_k \leftarrow$ EXECUTECOMMAND($u_k$)
        $\eta_{k+1} \leftarrow F(\eta_k, u_k, y_k)$
        $x_1 \leftarrow f(x_1, u_k)$
        $x_2 \leftarrow f(x_2, u_k)$
        $k \leftarrow k + 1$
    end while
end while

return $\eta_{k-1}$
```

The final section of the algorithm, detailed in Section IV-D, systematically reduces $\eta_k$ until only a single configuration remains.

### B. Generating a finite set of candidates

This section describes a technique for reaching an information state of finite cardinality. The central idea is to make two motions between points on the boundary of the environment, separated by a 90° turn. We show that if the environment has no pair of parallel edges, only finitely many configurations are consistent with such a sequence of motions. Section IV-C addresses the more troublesome case when the environment violates this condition.

The robot, starting with no knowledge of its position, makes several motions:

1) Move forward until reaching the boundary.
2) Rotate 180°, then move forward until reaching the boundary. Let $d_1$ denote distance traveled on this motion.
3) Rotate 90°, then move forward until reaching the boundary. If the robot reaches the boundary immediately, rotate 180° and try again. Let $d_2$ denote distance traveled on this motion.

The commands to "move until reaching the boundary" can be realized by selecting a translation amount larger than the diameter of $W$. In order to continue in final step, the robot must make a net rotation of either 90° or −90°, depending on its angle of incidence with the boundary. Except when the robot reaches an environment vertex, at least one of these rotations allows the robot to continue. If the robot knows it has reached an environment vertex, then there are already only finitely many candidates. The use of 90° rotations is motivated by the simplifications it affords in Equation 5. In principle, rotations of other amounts would work equally well.

The problem remains to find the set of configurations consistent with these initial motions. For simplicity, we ignore the

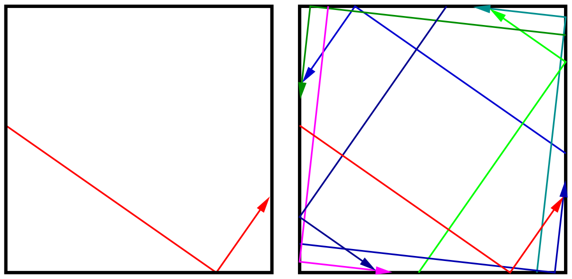

Fig. 2. [left] Two boundary-to-boundary motions in a square shaped environment, separated by a turn of 90°. [right] The 8 possibilities for these motions in this environment.

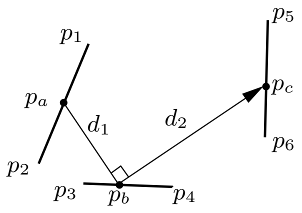

Fig. 3. Three fixed segments $\overline{p_1p_2}$, $\overline{p_3p_4}$, and $\overline{p_5p_6}$ and translations of length $d_1$ and $d_2$ between them.

first translation and instead consider only the two boundary-to-boundary translation with lengths $d_1$ and $d_2$. A geometric interpretation of the problem is perhaps helpful here:

> *Given $W$ and the two odometer readings $d_1$ and $d_2$, we want to find all ways to pack into $W$ a 2 link polygonal chain with edges having lengths $d_1$ and $d_2$ joined at a right angle, such that the initial and final endpoints rest on different boundary edges from the middle vertex.*

The set of final endpoints of these chains can be used directly to compute a set of candidate configurations of the robot. Figure 2 shows an example.

*1) Generating candidates for three fixed edges:* The robot's initial motions visit three environment edges. Suppose these three edges $\overline{p_1p_2}$, $\overline{p_3p_4}$, and $\overline{p_5p_6}$, and the order in which the robot visits them are fixed. Let $p_a \in \overline{p_1p_2}$, $p_b \in \overline{p_3p_4}$, and $p_c \in \overline{p_5p_6}$ denote the three boundary points visited by the robot. See Figure 3.

First, parameterize these three points as follows:

$$\begin{aligned} p_a &= (1-a)p_1 + ap_2 \\ p_b &= (1-b)p_3 + bp_4 \\ p_c &= (1-c)p_5 + cp_6. \end{aligned}$$

The first motion has length $d_1$, therefore $||p_a - p_b|| = d_1$. Expanding from the parameterization above gives a quadratic constraint in $a$ and $b$:

$$Aa^2 + Bab + Cb^2 + Da + Eb + F = 0 \tag{4}$$

with constant coefficients

$$\begin{aligned} A &= (x_2-x_1)^2+(y_2-y_1)^2 \\ B &= -2(x_2-x_1)(x_4-x_3)-2(y_2-y_1)(y_4-y_3) \\ C &= (x_4-x_3)^2+(y_4-y_3)^2 \\ D &= -2(x_3-x_1)(x_2-x_1)-2(y_3-y_1)(y_2-y_1) \\ E &= 2(x_3-x_1)(x_4-x_3)+2(y_3-y_1)(y_4-y_3) \\ F &= (x_3-x_1)^2+(y_3-y_1)^2-d_1^2, \end{aligned}$$

in which we use the convention that $p_i = (x_i, y_i)$. We also know that $p_c$ must be distance $d_2$ from $p_b$, and that $p_b - p_c$ must be perpendicular to $p_a - p_b$. These constraints are satisfied when

$$p_c - p_b = s_1 \frac{d_1}{d_2}(p_b - p_a)^\perp \qquad (5)$$

in which $s_1$ is either $-1$ or $+1$, depending on whether its net rotation was 90° or −90° in step 3 above. This vector equation can be separated into a pair of scalar linear equations in $a$, $b$, and $c$. Eliminating $c$ yields a single linear equation in $a$ and $b$:

$$Ga + Hb + I = 0 \qquad (6)$$

with constant coefficients

$$\begin{aligned} G &= \frac{\frac{s_1 d_2}{d_1}(y_2-y_1)}{x_5-x_6} + \frac{\frac{s_1 d_2}{d_1}(x_2-x_1)}{y_5-y_6} \\ H &= \frac{(x_4-x_3)-\frac{s_1 d_2}{d_1}(y_4-y_3)}{x_5-x_6} - \frac{(y_4-y_3)+\frac{s_1 d_2}{d_1}(x_4-x_3)}{y_5-y_6} \\ I &= \frac{(x_3-x_5)-\frac{s_1 d_2}{d_1}(y_3-y_1)}{x_5-x_6} - \frac{(y_3-y_5)+\frac{s_1 d_2}{d_1}(x_3-x_1)}{y_5-y_6}. \end{aligned}$$

Note that if either denominator is 0 (corresponding to $\overline{p_5 p_6}$ being horizontal or vertical), the system can be solved trivially. Equations 4 and 6 form a linear-quadratic system in $a$ and $b$. Barring degeneracies, this system has at most two solutions, which can be found analytically by standard methods.

The method described above gives candidate values for $a$, $b$, and $c$. Candidates for which any of $a$, $b$, or $c$ are outside the interval $[0, 1]$ should be discarded, because they correspond to endpoints outside of $\overline{p_1 p_2}$, $\overline{p_3 p_4}$, or $\overline{p_5 p_6}$ respectively. The final configuration (that is, position-orientation pair) of the robot resulting from such a candidate is $(p_c, \operatorname{atan}(y_c - y_b, x_c - x_b))$.

Lastly, note that if $d_1 = 0$ or $d_2 = 0$, then the robot knows that its position is at some convex vertex of $W$. This does not, however, eliminate the uncertainty in the robot's orientation. In order to determine its orientation, the robot must move away from the vertex. To do so, the robot must rotate and attempt translations, at most $360/\theta$ times, in which $\theta$ denotes the measure of the smallest interior angle in $W$, measured in degrees.

*2) Generating candidates over all of $W$:* The previous section showed how to find candidate solutions, given $d_1$, $d_2$ and three fixed environment edges to be visited in sequence. Candidate positions over the complete environment can be computed by iterating over each ordered triple of environment edges. Since we must admit the case where $\overline{p_1 p_2} = \overline{p_5 p_6}$, there are $n(n-1)(n-1)$ such triples. The at most 2 candidates for each can be computed in constant time. In practice, the performance of this process may possibly be improved by a preprocessing step which, for each pair of environment edges, computes the minimum and maximum distances between mutually visible points on these edges. This information can be used to filter some edge triples as infeasible without explicit consideration.

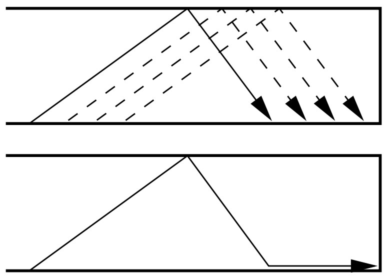
Fig. 4. [top] Parallel edges of the environment admit continua of candidate configurations. [bottom] A motion parallel to one of these segments leaves only a single candidate point.

As a final step, the candidate list must be pruned, retaining only those candidates that represent motions that lie entirely within $W$. In a simple polygon, data structures are known to answer such queries in $O(\log n)$ time, with $O(n)$ preprocessing time and $O(n)$ space [31]. This final candidate set becomes the robot's information state $\eta_k$.

## C. If some boundary edges are parallel

Although the preceding exposition made the assumption that the environment contains no pair of parallel edges, environments of practical interest often contain parallel edges. In particular, note the case where the environment contains a narrow strip bounded by two parallel edges. This situation would arise, for example, in a indoor corridor or narrow room. When parallel edges exist, continua of final configurations may be consistent with the robot's initial motions. See Figure 4. Each of these continua can be eliminated with a motion parallel to itself.

## D. Localization from a finite set

The previous sections showed how to select actions to guarantee that $\eta_k$ contains only finitely many configurations. How can we select additional actions to determine the robot's true position from among these candidates? The approach is to select two candidates and choose motions that are guaranteed to disambiguate them. More precisely, we want choose two configurations $x_1$ and $x_2$ from $\eta_k$ and choose actions $u_k, \ldots, u_{k+j}$ so that $\eta_{k+j+1}$ contains either $x_1$ or $x_2$ (or neither) but not both.

In Sections IV-B and IV-C, we described a method for reaching an information state representable by a finite union of single configurations. Given an information state $\eta_k$, an action $u_k$, and an observation $y_k$, how can we compute the resulting information state $\eta_{k+1} = F(\eta_k, u_k, y_k)$? Recall the definition of $F$, given in Equation 2. The definition suggests the algorithm should proceed in two stages: First, we find the forward projection of $\eta_k$ under action $u_k$, by ray shooting

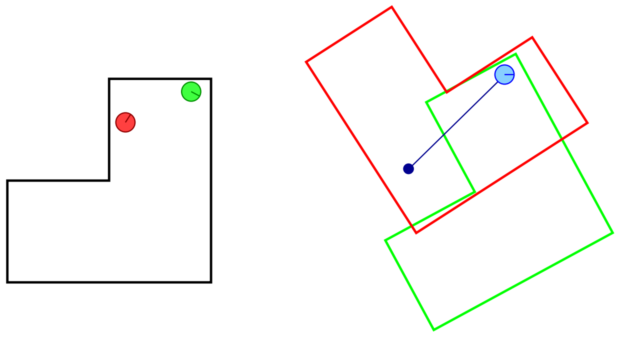
Fig. 5. [left] Two configurations in an L-shaped environment. [right] Two overlaid copies of the environment shown in the local frame of those configurations. Attempting to execute the path shown (which consists of one rotation and one translation) shown will result in different odometry readings for these two configurations.

in $W$. Then we prune from the result any configurations for which the distance traveled differs from $y_k$ using a simple constant time procedure.

For a given configuration $x$, let $W_x$ denote a transformation of the environment $W$ into the robot's local frame, such that the robot rests at the origin and faces the positive $x$-axis. Note that $(0, 0) \in W_x$ if and only if the position portion of $x$ is contained within $W$ in the global frame.

Select $x_1$ and $x_2$ arbitrarily from $\eta_k$. Compute $W_{x_1}$ and $W_{x_2}$ and overlay them. See Figure 5. In this overlay, rotation and translation commands affect both $x_1$ and $x_2$ in the same way; we can choose a destination position in this frame and command actions that to navigate both $x_1$ and $x_2$ to this point in their respective local frames.

Since $W$ has no nontrivial rotational symmetries, we have $W_{x_1} \neq W_{x_2}$. Therefore, there must exist some position $p$ in $W_{x_1}$ but not in $W_{x_2}$. Plan a path in $W_{x_1}$ from $(0, 0)$ to $p$. Since $(0, 0) \in W_{x_2}$ but $p \notin W_{x_2}$, this path must cross the boundary of $W_{x_2}$ at least once. The translation action corresponding to this crossing of the boundary of $W_{x_2}$ necessarily distinguishes between $x_1$ and $x_2$. If the robot began at $x_1$, its odometry reading at this step will be greater than if it had begun on $x_2$. One of the two can be pruned after this step. A third possibility is that both candidates are pruned before or during this step. This could happen if the robot's true configuration is neither $x_1$ nor $x_2$, but some third configuration in $\eta_k$. In this case, the remaining actions in the plan can be discarded, and new choices for $x_1$ and $x_2$ can be made from the reduced $\eta_{k+1}$.

Which path should the robot follow within $W_{x_1}$ to reach $p$ from $(0, 0)$? To disambiguate $x_1$ and $x_2$ requires only a path that stays within $W_{x_1}$ but leaves $W_{x_2}$. Our implementation uses the *shortest path* between $(0, 0)$ and $p$, which can be computed in time $O(n)$ [32, 33]. Also of potential interest is the minimum link path [34], which minimizes the number of robot commands. The minimum link path can also be computed in time $O(n)$. In any case, a piecewise linear path in $W_{x_1}$ can be trivially converted to a sequence of alternating translation and rotation commands.

## E. Complexity

Let $n$ denote the number of edges in $W$. In INITIALACTIONS, we execute fewer than $O(n^3)$ ray shooting queries, each taking time $O(\log n)$, so this step takes $O(n^3 \log n)$ time to generate $O(n^3)$ initial candidates. Let $r$ denote the number of such candidates. If $W$ has parallel edges, each segment returned by INITIALACTIONS takes time $O(n \log n)$ to compute.

The outer while loop in Algorithm 1 eliminates at least one candidate in each iteration, so there are at most $r-1$ iterations. There are fewer than $r - 1$ iterations if some candidates are pruned as a side-effect of distinguishing $x_1$ and $x_2$. The run time of each iteration is dominated by the time to compute $F$, which is $O(r \log n)$. This computation must be done at each of the $O(n)$ steps of the of the path generated at each iteration. Therefore, the total computation time for the algorithm is $O((n^3 + r^2 n) \log n) = O(n^7 \log n)$.

It is possible that these bounds can be improved. The question remains unanswered whether $r = \Theta(n^3)$. Our informal experiments suggest that in practical situations, both $r$ and the number of disambiguation iterations often fall far short of the upper bounds we present here.

## F. Dealing with Symmetries in the Environment

We have thus far assumed that $W$ has no nontrivial rotational symmetries. This is important in Algorithm 1 to ensure that there exists at least one point $p$ in $W_{x_1}$ but not in $W_{x_2}$. If this assumption does not hold, then we can still consider the problem of localization *up to symmetry*. This section makes the notion of localization up to symmetry more precise.

**Definition 3:** A *symmetry* is function composed of rigid translations and rotations mapping $W$ onto itself. Without ambiguity we can extend such a function to $\mathcal{C}$ by applying the appropriate change of orientation. Two configurations $x_1, x_2 \in \mathcal{C}$ are *symmetric* if there exists a symmetry under which $x_1 \mapsto x_2$.

The number of symmetries of $\mathcal{C}$ can be computed in $O(n)$ time [35]. The following lemma will be useful for showing the relevance of these symmetries to localization.

**Lemma 4:** The relation of symmetry between configurations is an equivalence relation, which we denote $\equiv$. Each equivalence class of $\equiv$ contains one configuration for each symmetry of the environment.

*Proof:* Observe that the symmetries of a polygon form a group under function composition. In particular the identity is always a symmetry, and the set of symmetries is closed under composition and inverse. The reflexivity, transitivity, and symmetry of the $\equiv$ relation all follow immediately. □

Now we show that R1 cannot distinguish between symmetric configurations.

**Lemma 5:** Consider an action sequence $u_1, \ldots, u_{k-1}$, an observation sequence $y_1, \ldots, y_{k-1}$ and the resulting information state $\eta_k$. For any $x \in \eta_k$ and $x' \in \mathcal{C}$ with $x \equiv x'$, $x' \in \eta_k$.

*Proof:* Since $x \in \eta_k$, there exists some initial state $x_1$ for which executing $u_1, \ldots, u_{k-1}$ leads to $x$ and generates $y_1, \ldots, y_{k-1}$. Since $x \equiv x'$, there exists a symmetry $\tau$ under which $x' = \tau(x)$. But $f$ acts only locally, so we know that a

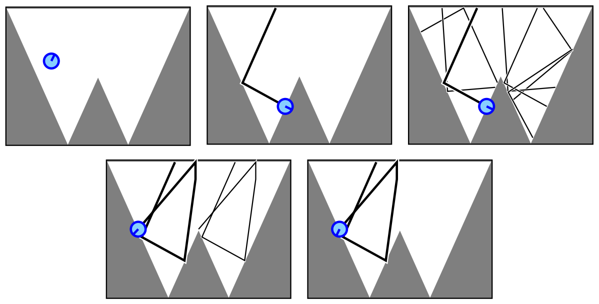

Fig. 6. A sample execution of Algorithm 1 generated by our implementation in approximately 0.03 seconds. Top row: (a) The robot in its initial configuration. (b) The motions generated by INITIALACTIONS. (c) There are 7 configuration consistent with these initial motions, so $|\eta_6| = 7$. Bottom row: (d) One disambiguation results in $|\eta_{12}| = 2$. (e) The robot is full localized after 13 commands, with final information state $|\eta_{14}| = 1$.

robot starting from $\tau(x_1)$ and executing $u_1, \ldots, u_{k-1}$ has state $f(\tau(x_1), u_1, \ldots, u_j) = \tau(f(x_1, u_1, \ldots, u_j)) = \tau(x) = x'$. Moreover, the observation sequences are identical, because the boundary edges of $W$ are affected by $\tau$ in the same way as $x_1$ is. Consequently, $\tau(x_1)$ is an initial state that leads to $x'$, thereby demonstrating that $x'$ is consistent with $u_1, \ldots, u_{k-1}$ and $y_1, \ldots, y_{k-1}$. Hence $x' \in \eta_k$. □

The practical importance of this lemma is that for R1, the localization task can only be accomplished modulo the symmetries in the environment. No sequence of actions and observations can distinguish between a pair of symmetric configurations. Note, however, that Algorithm 1 can be adapted to handle symmetries gracefully. The only modifications needed are to change the termination condition to stop when $|\eta_k|$ is equal to the number of symmetries, and to ensure that the configurations selected as $x_1$ and $x_2$ are not themselves symmetric. The rest of the algorithm remains unchanged.

### G. *Computed examples*

To illustrate its effectiveness, we have implemented Algorithm 1 in simulation, using simplified methods for many of the geometric computations. The implementation is in C++ on a 2.5GHz GNU/Linux system. Figure 6 shows a simple example in which the robot makes 13 motions to localize itself. In Figure 7, the environment is a regular pentagon, so the final information state contains one configuration for each of the 5 symmetries. The environment depicted in Figure 8 is serpentine and self similar, but has no symmetries.

## V. LOCALIZATION WITH A COMPASS AND CONTACT SENSOR

Having addressed the localization task for R1, we now consider R2, a robot equipped with only a compass and contact sensor. Once again we show constructively that the localization task can be completed. A simple example of our algorithm's execution appears in Fig. 9.

Recall that each action $u \in S^1$ represents a rotation to the given orientation, followed by a forward motion to the environment boundary. After its first action, the robot knows its true orientation. Also note that after the first motion, the robot's translations are all between points on the environment boundary. For these reasons, we can simplify the robot's state space to $\partial W$, ignoring orientation and the interior of $W$. In this context, the information states are subsets of $\partial W$. We use this simplification throughout Sections V and VI.

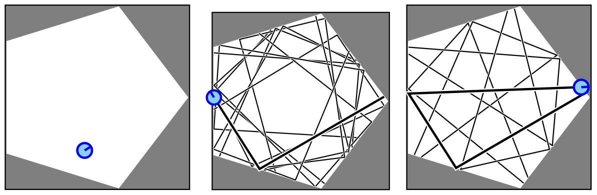

Fig. 7. A robot localizing itself in an environment with 5 symmetries. From top to bottom: (a) The robot's initial configuration. (b) Executing INITIALACTIONS results in an information state $\eta_8$ containing 15 configurations. (c) One disambiguation iteration fully localizes the robot, leaving 5 configurations in $\eta_{10}$. Our implementation took approximately 0.1 seconds to solve this problem.

### A. *Computing the information transition function*

This section presents an algorithm for computing $F(\eta, u)$ given $W$, $\eta$ and $u$. We restrict our attention to information states that can be reached from the initial state $\eta_1 = \partial W$.

Consider an information state $\eta$ that can be expressed as the union of a finite collection $s_1, \ldots, s_l$ of open segments and a finite set of points $p_1, \ldots, p_m$ on $\partial W$. To be precise, each $s_i$ is a linear subset of $\partial W$ not containing its endpoints. Each $s_i$ need not be a complete edge of $\partial W$ and since it is linear, cannot contain any vertex of $\partial W$. Without loss of generality, assume that the $s_i$'s are pairwise disjoint. The next lemma shows that every reachable information state can be expressed in this form.

**Lemma 6:** Every information state $\eta$ reachable from $\partial W$ by an action sequence $u_1, \ldots, u_k$ can be expressed as a finite union of open segments and points on $\partial W$.

*Proof:* Use induction on $k$. When $k = 0$, $\eta = \partial W$, which is the union of the vertices and edges bounding $W$. Assume inductively that $\eta_{k-1}$ can be expressed as a finite union of open segments and points. Because $F$ maps each segment to a finite set of polygonal chains on $\partial W$ and each point to another single point, $\eta_k$ also has a representation as a finite set of points and segments. □

The intuition is that, given an action $u$ and an information state $\eta$ described as a finite union of points and segments, the resulting information state $F(\eta, u)$ is simply the projection of those points and segments onto $\partial W$ in direction $u$. For a point, this projection is a simple ray-shooting query. For a segment $\overline{ab}$, compute the projection by sweeping line parallel to $u$ from $a$ to $b$, generating a new segment each time the point on $\partial W$ intersecting $l$ closest to $\overline{ab}$ is a vertex of $W$. See Fig. 10. The time to perform this computation is $O((m + nl) \log n)$ for an information state described by $m$ points and $l$ segments in an environment with $n$ vertices.

### B. *Algorithm overview*

We now present the localization algorithm itself. The algorithm proceeds in two parts. First, actions are selected

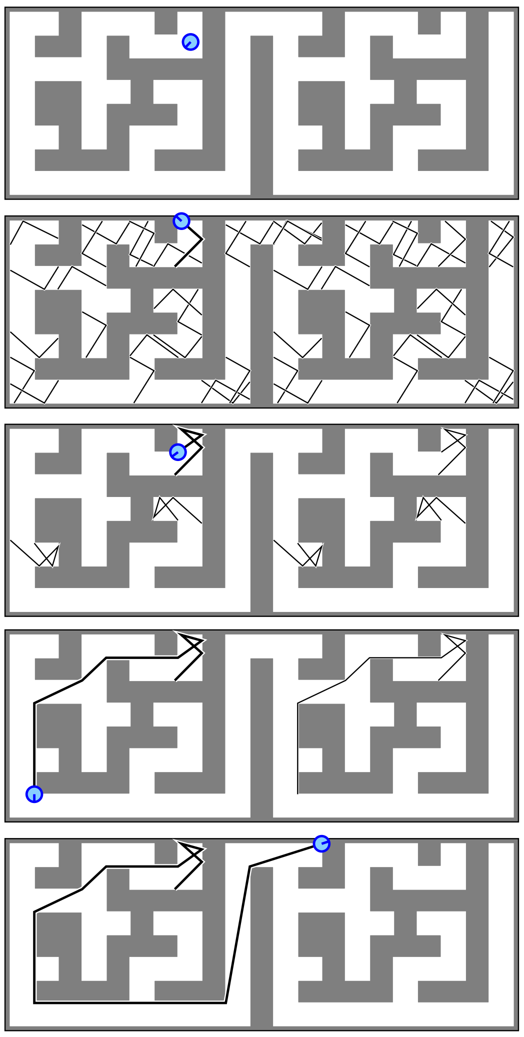

Fig. 8. A robot localizing itself in a serpentine environment. From top to bottom: (a) The robot's initial configuration. (b) Executing INITIALACTIONS results in an information state $\eta_6$ containing 48 configurations. (c) After 2 iterations of the disambiguation algorithm, only 6 configurations remain in $\eta_{10}$. (d) There are only two configurations in $\eta_{20}$. (e) The robot is fully localized after 25 motions. Our implementation took approximately 3.8 seconds to solve this problem.

which reduce the uncertainty in the robot's position to a finite set of possibilities. Second, additional actions are chosen to reduce the uncertainty from this finite set to a single point. The complete localizing sequence $u_1, \ldots, u_K$ is divided into two parts $u_1, \ldots, u_{K_1}$ and $u_{K_1+1}, \ldots, u_{K_2}$ generated by the respective parts of the algorithm. The complete algorithm is shown in Algorithm 2.

### C. From the entire boundary to a finite subset

This section presents a sweep line algorithm for computing a sequence of actions to reduce the robot's information state to a finite set of points. The following lemma, whose intent is illustrated in Figure 11, provides the basis for the algorithm.

**Lemma 7:** For any segment $s = \overline{ab} \subset W$, $F(s, u)$ is a single point if and only if $u = (a - b)/||a - b||$ or $u = (b - a)/||b - a||$.

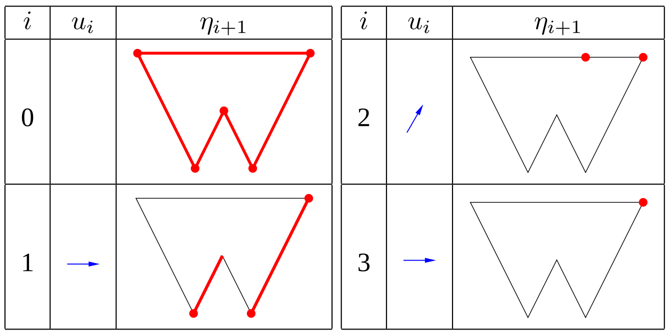

Fig. 9. A localizing sequence generated by Alg. 2 for R2 in a nonconvex polygon. The information state at each step is shaded. Compare to Fig. 6.

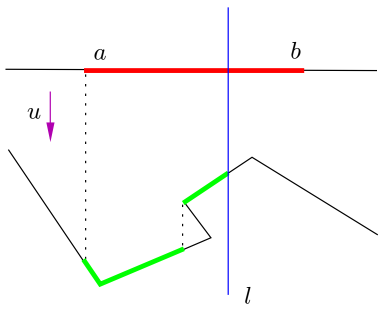

Fig. 10. Computing $F(\overline{ab}, u)$ by a line sweep algorithm. The diagram shows a snapshot of the algorithm as it runs. The sweep line $l$ moves from left to right.

*Proof:* For the forward part, note that since $\overline{ab}$ is contained in $W$ and is therefore itself collision free, the maximal collision free segment starting from each $x \in \overline{ab}$ is the same. Hence each $x \in \overline{ab}$ maps to the same point under $f$. For the backward part, suppose $u$ is not parallel to $\overline{ab}$ and $F(\overline{ab}, u)$ is a single point. Then $a$, $b$, and $F(\overline{ab}, u)$ form a nondegenerate triangle. This is a contradiction because by definition of $f$, we must have $\overline{ax}$ parallel to $\overline{bx}$. □

Starting with $\eta_1 = \partial W$, the algorithm maintains a "current" information state $\eta_k$ and a sequence of actions $u_1, \ldots, u_{k-1}$ mapping $\eta_1$ to $\eta_k$. Computation proceeds by sweeping a vertical line $l$ from left to right across $W$, maintaining the invariant that $\eta_k$ has no segments on the left side of $l$. Each time $l$ reaches the endpoint of a segment $\overline{ab}$ in $\eta_k$, the sweep line stops and the algorithm selects as $u_k$ whichever of $(a - b)/||a - b||$ and $(b - a)/||b - a||$ has nonnegative $x$ coordinate. The resulting $\eta_{k+1} = F(\eta_k, u_k)$ maintains the sweep invariant because the $x$-component of the motion of each segment in $\eta_k$ is nonnegative; hence, no segment can cross $l$. When $l$ passes the rightmost vertex of $W$, it is certain that no segments remain in $\eta_k$. It remains to show that this method generates a plan of finite length.

**Lemma 8:** The above algorithm generates $K_1 = O(n^3)$ actions for an environment with $n$ edges.

*Proof:* Let $e_1, \ldots, e_n$ denote the edges of $\partial W$ and let $v(e_i)$ denote a unit vector parallel to $e_i$ and oriented so that its $x$ component is nonnegative. For a fixed $i$ and $j$, $F(e_i, v(e_j))$ is a set of polygonal chains on $\partial W$ with total complexity $O(n)$. Let $R_{ij}$ denote the set of endpoints of segments in

**Algorithm 2** LOCALIZER2($W$)

```
$\eta_1 \leftarrow \partial W$
$k \leftarrow 1$
while $\eta_k$ contains at least one segment do
  $\overline{ab} \leftarrow$ LEFTMOSTSEGMENT($\eta_k$)
  if $(a - b).x > 0$ then
    $u_k \leftarrow (a - b)/||a - b||$
  else
    $u_k \leftarrow (b - a)/||b - a||$
  end if
  $\eta_{k+1} \leftarrow F(\eta_k, u_k)$
  $k \leftarrow k + 1$
end while

while $\eta_k$ contains at least two points do
  Select $p, q$ from $\eta_k$.
  $p_k \leftarrow p, q_k \leftarrow q$
  while $q_k \notin$ Vis$(p_k, W)$ do
    $t_k \leftarrow$ first vertex of shortest path from $p_k$ to $q_k$
    $u_k \leftarrow (t_k - p_k)/||t_k - p_k||$
    $\eta_{k+1} \leftarrow F(\eta_k, u_k)$
    $p_{k+1} \leftarrow$ SHOOTRAY($W, p_k, u_k$)
    $q_{k+1} \leftarrow$ SHOOTRAY($W, q_k, u_k$)
    $k \leftarrow k + 1$
  end while
  $u_k \leftarrow (q_k - p_k)/||q_k - p_k||$
  $\eta_{k+1} \leftarrow F(\eta_k, u_k)$
  $k \leftarrow k + 1$
end while

return $(u_1, \ldots, u_{k-1})$
```

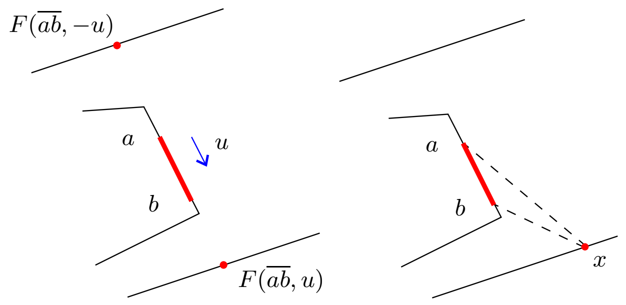

Fig. 11. [left] A motion along $\overline{ab}$ collapses $\overline{ab}$ to a single point. [right] No motion not parallel to $\overline{ab}$ can collapse $\overline{ab}$.

$F(e_i, v(e_j))$ and let $R = \bigcup_{i,j} R_{ij}$. Observe that $|R| = O(n^3)$. Clearly every segment $s$ reached by $l$ is in the initial condition $\eta_1$, or is a subset of some $F(e_i, v(e_j))$. There are $n$ segments in $\eta_1$ and $R$ is a set of earliest possible points at which an information state segment projected from another edge may begin. These events are sufficient to maintain the sweep invariant, so $K_1 = O(n) + O(n^3) = O(n^3)$. □

## D. From a finite subset to a single point

The previous section showed how to select actions $u_1, \ldots, u_{K_1}$ that map $\eta_1 = \partial W$ to a finite set $\eta_{K_1} = \{p_1, p_2, \ldots, p_m\}$ of points on $\partial W$. It remains to generate additional actions $u_{K_1+1}, \ldots, u_{K_2}$ mapping $\{p_1, p_2, \ldots, p_m\}$ to a single point. We derive this part of the algorithm by reduction to the special case when $m = 2$. The more general problem for $m$ points can be solved by iterating the algorithm

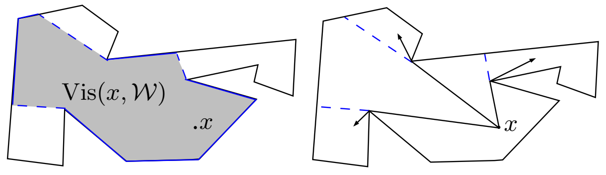

Fig. 12. [left] A visibility polygon. Spurious edges are dashed. [right] The shortest path to any point not in the visibility polygon begins with a motion in the direction of a spurious edge.

for two points.

Let $\eta = \{p, q\}$. The ordering of the points is arbitrary but must be fixed. Our goal is to design a sequence of actions $u_{K_1+1}, \ldots, u_{K_2}$ such that

$$f(p, u_{K_1+1}, \ldots, u_{K_2}) = f(q, u_{K_1+1}, \ldots, u_{K_2}). \quad (7)$$

That is, we want an action sequence mapping $p$ and $q$ to the same destination. For $K_1 < k \leq K_2$, let

$$p_k = f(p, u_{K_1+1}, \ldots, u_k)$$

and likewise

$$q_k = f(q, u_{K_1+1}, \ldots, u_k).$$

Our algorithm selects $u_k$ using only $p_k$ and $q_k$. We begin with the simple base case:

**Lemma 9:** If $\overline{p_k q_k} \subset W$, then the action $u = (q_k - p_k)/||q_k - p_k||$ is a localizing sequence for $\{p_k, q_k\}$.

*Proof:* Follows from Lemma 7 with $a = p_k$ and $b = q_k$. □

The intuition is that if $p_k$ can "see" $q_k$ in the sense that there is an unobstructed path between them, then a motion in the direction of this path maps both $p_k$ and $q_k$ to the same place.

Now suppose $\overline{p_k q_k} \not\subset W$. The following definition is useful in this case.

**Definition 10:** For any $x \in W$, let Vis$(x, W)$ denote the *visibility polygon* of $x$ in $W$, defined as

$$\text{Vis}(x, W) = \{x' \in W \mid \overline{xx'} \subset W\}. \quad (8)$$

We follow [2] in characterizing the boundaries visibility polygons in terms of non-spurious edges which are parts of $\partial W$ and spurious edges which are not. Observe that since $W$ is simply connected, the spurious edges subdivide $W$ in such a way that every point $x' \notin$ Vis$(x, W)$ can be associated with exactly one spurious edge such that the shortest path from $x$ to $x'$ crosses this spurious edge. Further, the first segment of the shortest path from $x$ to $x'$ is parallel to this spurious edge. See Figure 12. Let $\overline{t_k v_k}$ denote the spurious edge crossed by the shortest path from $p_k$ to $q_k$.

Assume momentarily that $\overline{t_k v_k}$ is not a bitangent of $W$. Choose $u_k = (t_k - p_k)/||t_k - p_k||$. That is, select a motion in the direction of the spurious edge that hides $q_k$ from $p_k$. Figure 13 illustrates this selection (and the intuition behind the proof of Lemma 11). This completes the definition of our action sequence $u_{K_1+1}, \ldots, u_{K_2}$:

$$u_i = \begin{cases} (q_i - p_i)/||q_i - p_i|| & \text{if } q_i \in \text{Vis}(p_i, W) \\ (t_i - p_i)/||t_i - p_i|| & \text{otherwise} \end{cases}, \quad (9)$$

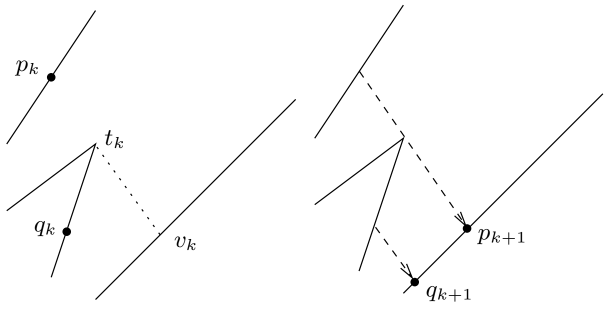

Fig. 13. [left] The spurious edge $\overline{t_k v_k}$ hides $p_k$ from $q_k$. [right] The point $q_{k+1}$ cannot cross $\overline{t_k v_k}$ because its motion is parallel to $\overline{t_k v_k}$.

in which $K_2$ is the minimal $i$ for which the first case applies. Clearly if $K_2$ exists, then this action sequence is a localizing sequence. It remains only for us to show that $K_2$ exists.

Let $Q_k = W - \bigcup_{i=K_1,\dots,k} \text{Vis}(p_i, W)$ and observe that $Q_{k+1} \subset Q_k$. Informally, $Q_k$ is the portion of $W$ that $p$ has never seen.

**Lemma 11:** For all $k > K_1$, $q_k \in Q_k$.

*Proof:* Use induction on $k$. The statement is true by construction when $k = K_1$. For the inductive step, note that $q_k$ moves parallel to $\overline{t_k v_k}$, so that $q_{k+1}$ is still behind this spurious edge. If $q_k \notin Q_k$, then $q_k$ must be in a region visible to some $p_i$, or in some region not seen by any $p_i$ but separated from $q_k$ by $\overline{t_k v_k}$. In either case, we can form a nontrivial loop in $W$, contradicting the simply connected property of $W$. □

One informal way to understand Lemma 11 is to imagine that $p$ is "chasing" $q$. With each motion, $p$ takes a step in pursuit of $q$ and eliminates a portion of the environment $Q_k$ in which $q$ could be "hiding". If $K_2$ exists, then $p$ eventually "catches" $q$.

Now we can prove the algorithm's correctness.

**Theorem 12:** The sequence $u_{K_1+1}, \dots, u_{K_2}$ is a localizing sequence for $\{p, q\}$.

*Proof:* If $K_2$ exists, it follows from Lemma 9 that $u_{K_1+1}, \dots, u_{K_2}$ is a localizing sequence for $\{p, q\}$. To show that $K_2$ exists, note that each $p_k$ is in a different cell of the visibility cell decomposition [2] of $W$. There are only $O(n^2)$ such cells on the boundary, so $K_2 = O(n^2)$. □

Finally, we must consider the special case when $\overline{t_k v_k}$ is a bitangent. This case is problematic because choosing $u_k = (t_k - p_k)/||t_k - p_k||$ is no longer sufficient to ensure that $Q_{k+1} \subset Q_k$. The algorithm as stated would alternate between the actions $t_k - v_k$ and $v_k - t_k$. This problem can be avoided by rotating $u_k$ by a sufficiently small $\epsilon$ ensuring that $\overline{q_k q_{k+1}}$ does not intersect $\overline{t_k v_k}$. Then select $u_{k+1} = (v_k - p_{k+1})/||v_k - p_{k+1}||$. Figure 14 illustrates this situation. This modification adds an additional action each time $p_k$ falls at the endpoint of a bitangent complement, but does not substantially change the analysis.

Now we can finally return to the general case with $m$ points. If $m > n$ (recall $n$ is the complexity of $\partial W$), then by the pigeonhole principle, at least two points must lie on the same edge of $\partial W$. This pair of points can see each other, and one motion collapses them to a single point. In this way, we can reduce the information state to a set of at most $n$ points using only $m - n$ actions. Then select an arbitrary pair of points $p$ and $q$ from the current information state $\eta_k$. We have shown how to merge $p$ and $q$ in $O(n^2)$ steps. Repeating this process at most $n$ times results in a plan of length $O(n^3)$ to map $\{p_1, \dots, p_m\}$ to a single point. Combining this with the $O(n^3)$ steps from the first part of the algorithm (Section V-C) yields a total plan length of $K = K_1 + K_2 = O(n^3)$.

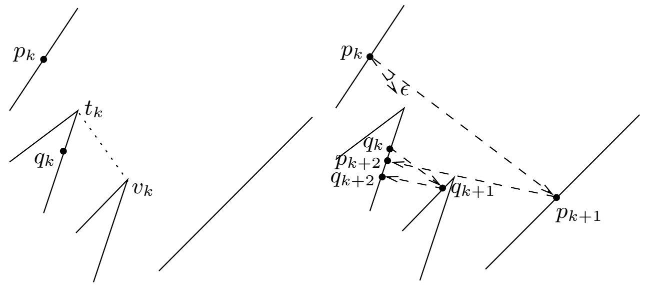

Fig. 14. The special case when $\overline{t_k v_k}$ is a bitangent.

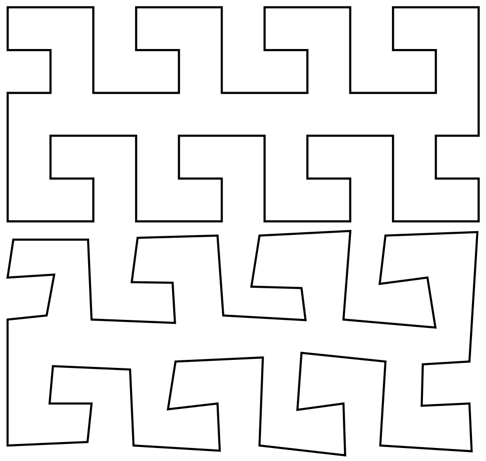

Fig. 15. [top] An environment with many regularities. Algorithm 2 generates a 5-step localizing sequence for this environment, running in approximately 0.4 seconds. [bottom] A modified version of this environment in which the regularities have been broken. Our algorithm generates a 26 step localizing sequence for this environment, running in approximately 1.0 seconds.

## E. Computed examples

We have implemented this algorithm in simulation. The top portion of Figure 15 shows an environment with many regularities for which Algorithm 2 generates a 5-step localizing sequence. In contrast, our algorithm needs 28 steps for the similar but irregular environment in the bottom portion of Figure 15. This is in sharp contrast to visibility based localization, in which such regularities are precisely what make localization problems difficult. Figure 16 shows a very irregular environment for which our algorithm generates a 30 step localizing sequence. This sequence is executed from six different initial positions. The robot's final position is in the lower right.

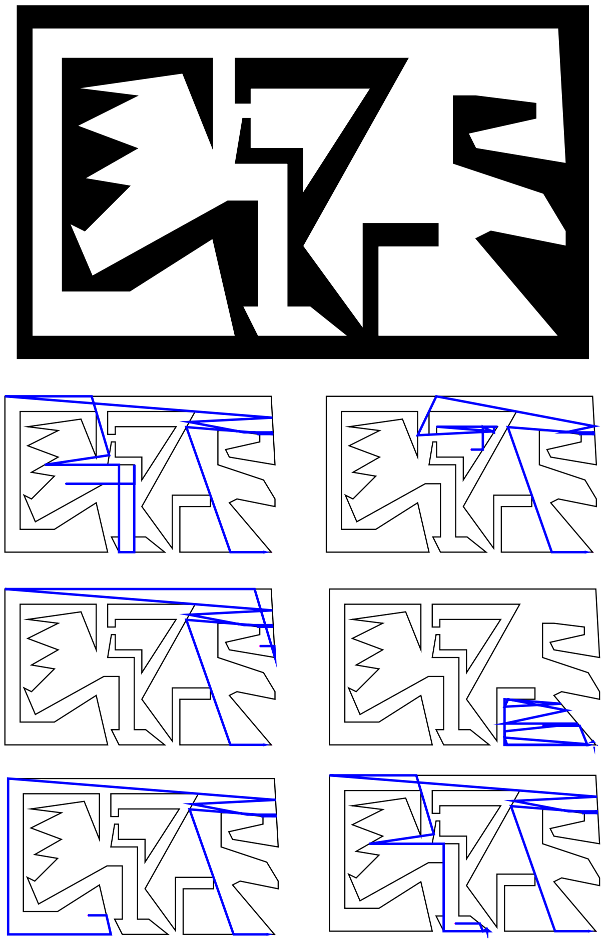

Fig. 16. [top] An irregular environment for which the localizing sequence computed by our algorithm requires 30 steps. The computation took about 1.9 seconds. [bottom] Execution traces of this localization sequence for 6 different starting positions. For each starting position, the final position is the lower right corner of the environment.

## VI. Localization with an Angular Odometer and Contact Sensor

In Section V, we showed that robot model R2, a robot with only a compass and a contact sensor, is capable of localizing itself within its environment. In this section we consider R3, a weaker version of R2 in which the compass has been replaced by an angular odometer. This model is identical to that of Section V, except that we now consider actions specified relative to an unknown initial orientation, rather than a global reference direction. Equivalently, we can consider the environment to have been rotated through an unknown angle $\theta$, representing the difference between the global reference direction and the robot's initial orientation. A localizing sequence must map every $x \in W$ to the same $x_f$, regardless of $\theta$. We show that, under this model, every sequence of actions fails.

**Definition 13:** An information state-action pair $(\eta, u)$ is a *collapsing transition* if $u$ is parallel to some segment in $\eta$.

**Lemma 14:** Every localizing sequence contains at least one collapsing transition.

*Proof:* Suppose there exists some localizing sequence $u_1, \ldots, u_K$ with no collapsing transitions. Arbitrarily pick a segment $s_1 \subseteq \eta_1 = \partial W$. Because of Lemma 7, at every step $1 \le k \le K$, $F(s_k, u_k)$ contains at least one segment $s_{k+1}$. We have constructed a segment $s_K \subseteq \eta_K$. Therefore $|\eta_K|$ is infinite, a contradiction. □

**Theorem 15:** For a robot with only angular odometry and a contact sensor in any polygonal environment $W$, no localizing sequence exists.

*Proof:* Suppose such a sequence $u_1, \ldots, u_K$ exists. Let $e_1, \ldots e_n$ denote the set of edges of $\partial W$, and let $\mathrm{Rot}(v, \phi)$ denote the rotation of $v \subseteq \mathbb{R}^2$ by angle $\phi$. If there exists no action-edge pair $(u_i, e_j)$ with $u_i$ and $\mathrm{Rot}(e_j, \theta)$ parallel, then $u_1, \ldots, u_K$ contains no collapsing transitions. The sequence is required to work for all $\theta \in S^1$ but the subset of $S^1$ in which some $u_i$ coincides with some $\mathrm{Rot}(e_j, \theta)$ has measure 0. Therefore $u_1, \ldots, u_K$ fails for almost every $\theta$. □

The intuition is that reaching a finite cardinality information state requires at least one motion parallel to some environment wall. No finite length localizing sequence can achieve this for all possible initial orientations.

## VII. Discussion and Conclusions

This paper presented a localization techniques for several robots with severely limited sensing capabilities. In this final section, we discuss these results and mention several problems we have left open.

### A. Comparison of results

There are also some subtle but perhaps illustrative differences with the results we have presented for R1 and R2. The algorithm for R1 is effective only up to symmetry, whereas symmetries are not relevant to R2. This difference can be directly attributed to the fact that, for R1, angular information is only local, rather than global. Likewise, the algorithm for R2 can only guarantee a known *final* configuration. For R1, each motion is precisely measured. This provides sufficient information to determine the initial configuration and indeed the robot's entire path.

### B. Comparison between sensing models

Perhaps the most closely related localization model is that of [14], in which the robot uses an omnidirectional range sensor. The two phase approach described in that work – that of finding a finite set of candidates (*hypothesis generation*) followed by determination of the true configuration from among these candidates (*hypothesis elimination*) – is similar to the approach of both Algorithm 1 and Algorithm 2.

Model R1 is strictly weaker than the visibility based model used in [14]. The visibility polygon available to the robot in that work can be viewed as an omnidirectional measure of the distance to the environment boundary. By ignoring all of these distances except the distance to the boundary directly forward, their robot can accurately simulate R1. Moreover, the work of [14] is mainly concerned with *minimum distance* localization, a problem we have not addressed.

Observe also that R1 and R2 are not directly comparable. Comparing R1 to R2, we exchange the compass for an angular odometer and the contact sensor for a linear odometer. In doing so we have strengthened the linear (distance) sensing while reducing the robot's angular sensing. More broadly, we can imagine a partial ordering on robot systems, in which a comparison relation is defined by the ability of one robot to simulate another. In this context, the minimalist approach can be described as a search for minima in this partial order. We address comparisons of this type more formally and more generally in [36].

## C. Relationship to probabilistic methods

There is a large body of research on Bayesian methods for mobile robot localization (for example, [9, 17, 18, 37, 38]). One way to interpret the our results is as a special case of techniques based on POMDPs (for example, [38]) in which sensing is perfect. However, our use of set-based uncertainty allows us to treat the continuous state space exactly, but existing POMDP methods generally require discretization to a finite state space. This sort of Bayesian approach is a very natural way of extending our robot models to account for errors in sensing and motion. Progress has already been made on probabilistic models for the some sensing capabilities considered here. For example, [37] presents probabilistic models for local odometry information. Our algorithms themselves, however, would require substantial adaptation. There is no clear analog to Lemma 7, so R2 could not "collapse" intervals of probability mass to single points in the same way. Another consideration is that, because we would be forced to settle for accumulating a sufficiently large portion of the probability mass in a sufficiently small region, the basic argument of Theorem 15 fails.

## D. Open questions

This work is based on an idealization in which the robot's internal map is perfect. If the robot's map is imperfect or absent, we may consider a space $\mathcal{E}$ of potential environments. The robot's state would be defined by its environment $W \in \mathcal{E}$ and its configuration $x \in W \times S^1$. The complete information space is the power set of $\mathcal{E} \times \mathcal{C}$. If $|\mathcal{E}|$ is finite, we can compute candidates within each possible environment and continue until only one environment-configuration pair remains. If $\mathcal{E}$ is a richer set, perhaps defined by allowing tolerances in the positions of vertices, the extension is not as straightforward.

We have also assumed that the robot moves in a simply-connected environment. This assumption is not needed for R1. For R2, it is needed primarily in Section V-D to ensure that $p$ eventually "catches" $q$. However, because the motions of $p$ and $q$ have the same directions, it seems plausible that a very similar method would apply when $W$ has holes.

Lastly, in this paper we have only considered the question of existence of localization strategies. It remains an open problem to generate optimal localization strategies for our sensing-limited models. One relevant optimality criterion is the maximum distance travelled over all initial states in $W$. For R1, it may be possible to adapt the techniques used in [14] to show, by reduction from the Abstract Decision Tree problem, that computing an optimal localization strategy is NP-hard. For R2, it is less clear how to proceed, because R2 does not admit branching in the localizing sequence.

## ACKNOWLEDGMENTS

Portions of this work have been supported by ONR grant N00014-02-1-0488, by NSF-CONACyT award 0296126 and by DARPA grant #HR0011-05-1-0008.

## REFERENCES

[1] I. J. Cox, "Blanche – an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Trans. Robot. and Autom.*, vol. 7, no. 2, pp. 193–204, 1991.
[2] L. J. Guibas, R. Motwani, and P. Raghavan, "The robot localization problem," in *Proc. Workshop on Alg. Found. of Robot.*, K. Goldberg, D. Halperin, J.-C. Latombe, and R. Wilson, Eds. Wellesley, MA: A.K. Peters, 1995, pp. 269–282.
[3] G. Dudek, K. Romanik, and S. Whitesides, "Localizing a robot with minimum travel," in *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1995.
[4] D. Avis and H. Imai, "Locating a robot with angle measurements," *J. Symb. Comput.*, vol. 10, no. 3-4, pp. 311–326, 1990.
[5] E. D. Demaine, A. López-Ortiz, and J. I. Munro, "Robot localization without depth perception," in *Scandinavian Workshop on Algorithm Theory*, 2002.
[6] K. Sugihara, "Some location problems for robot navigation using a simple camera," *Comp. Vis., Graphics, & Image Proc.*, vol. 42, no. 1, pp. 112–129, 1988.
[7] J. M. O'Kane, "Global localization using odometry," in *Proc. IEEE Int. Conf. Robot. and Autom.*, 2006.
[8] J. M. O'Kane and S. M. LaValle, "Almost-sensorless localization," in *Proc. IEEE Int. Conf. Robot. and Autom.*, 2005.
[9] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proc. IEEE Int. Conf. Robot. and Autom.*, 1999.
[10] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Trans. Robot. and Autom.*, vol. 7, no. 3, pp. 376–382, June 1991.
[11] K. T. Sutherland and W. B. Thompson, "Inexact navigation," in *Proc. IEEE Int. Conf. Robot. and Autom.*, 1993, pp. 1–7.
[12] D. D. Sleator and R. E. Tarjan, "Amortized efficiency of list update and paging rules," *Comm. of the ACM*, vol. 28, pp. 202–208, 1985.
[13] J. M. Kleinberg, "The localization problem for mobile robots," in *IEEE Symposium on Foundations of Computer Science*, 1994, pp. 521–531.
[14] G. Dudek, K. Romanik, and S. Whitesides, "Localizing a robot with minimum travel," *SIAM J. Comput.*, vol. 27, no. 2, pp. 583–604, 1998.
[15] M. Rao, G. Dudek, and S. Whitesides, "Randomized algorithms for minimum distance localization," in *Proc. Workshop on Algorithmic Foundations of Robotics*, 2004, pp. 265–280.
[16] S. Koenig, A. Mudgal, and C. Tovey, "An approximation algorithm for the robot localization problem," in *Proc. ACM-SIAM Symposium on Discrete Algorithms*, 2006.
[17] D. Fox, W. Burgard, and S. Thrun, "Active markov localization for mobile robots," *Robotics and Autonomous Systems*, vol. 25, pp. 195–207, 1998.
[18] P. Jensfelt and S. Kristensen, "Active global localisation for a mobile robot using multiple hypothesis tracking," *IEEE Trans. Robot. and Autom.*, vol. 17, no. 5, pp. 748–760, Oct. 2001.
[19] D. E. Whitney, "Real robots don't need jigs," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1986.
[20] M. A. Erdmann and M. T. Mason, "An exploration of sensorless manipulation," *IEEE Trans. Robot. and Autom.*, vol. 4, no. 4, pp. 369–379, Aug. 1988.
[21] K. Y. Goldberg, "Orienting polygonal parts without sensors," *Algorithmica*, vol. 10, pp. 201–225, 1993.
[22] S. Akella, W. Huang, K. M. Lynch, and M. T. Mason, "Parts feeding on a conveyor with a one joint robot," *Algorithmica*, vol. 26, no. 3, pp. 313–344, Mar. 2000.
[23] I. Kamon and E. Rivlin, "Sensory-based motion planning with global proofs," *IEEE Trans. Robot. and Autom.*, vol. 13, no. 6, pp. 814–822, Dec. 1997.

[24] I. Kamon, E. Rivlin, and E. Rimon, “Range-sensor based navigation in three dimensions,” in *Proc. IEEE Int. Conf. Robot. and Autom.*, 1999.
[25] V. Lumelsky and S. Tiwari, “An algorithm for maze searching with azimuth input,” in *Proc. IEEE Int. Conf. Robot. and Autom.*, 1994, pp. 111–116.
[26] V. J. Lumelsky and A. A. Stepanov, “Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape,” *Algorithmica*, vol. 2, pp. 403–430, 1987.
[27] B. Tovar, L. Guilamo, and S. M. LaValle, “Gap Navigation Trees: Minimal representation for visibility-based tasks,” in *Proc. Workshop on Alg. Found. of Robot.*, 2004.
[28] E. U. Acar and H. Choset, “Complete sensor-based coverage with extended-range detectors: A hierarchical decomposition in terms of critical points and voronoi diagrams,” in *Proc. of IEEE IROS, Int’l Conference on Intelligent Robots and Systems*, 2001.
[29] H. Choset and J. Burdick, “Sensor based planning, part I: The generalized Voronoi graph,” in *Proc. IEEE Int. Conf. Robot. and Autom.*, 1995, pp. 1649–1655.
[30] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/), 2006.
[31] B. Chazelle and L. G. Guibas, “Visibility and intersection problems in plane geometry,” *Disc. and Comp. Geom.*, vol. 4, pp. 551–589, 1989.
[32] L. J. Guibas and J. Hershberger, “Optimal shortest path queries in a simple polygon,” *J. Comput. Syst. Sci.*, vol. 39, no. 2, pp. 126–152, 1989.
[33] J. Hershberger, “A new data structure for shortest path queries in a simple polygon,” *Inform. Process. Lett.*, vol. 38, pp. 231–235, 1991.
[34] J. S. B. Mitchell, G. Rote, and G. Woeginger, “Minimum-link paths among obstacles in the plane,” *Algorithmica*, vol. 8, pp. 431–459, 1992.
[35] J. D. Wolter, T. C. Woo, and R. A. Volz, “Optimal algorithms for symmetry detection in two and three dimensions,” *The Visual Computer*, vol. 1, pp. 37–48, July 1985.
[36] J. M. O’Kane and S. M. LaValle, “On comparing the power of robots,” Submitted to *International Journal of Robotics Research*, 2007, under review.
[37] H. Shatkay and L. P. Kaelbling, “Learning topological maps with weak local odometric information,” in *Proc. International Joint Conference on Artificial Intelligence*, 1997, pp. 920–927.
[38] R. Simmons and S. Koenig, “Probabilistic robot navigation in partially observable environments,” in *Proceedings International Joint Conference on Artificial Intelligence*, 1995, pp. 1080–1087.

**Jason M. O’Kane** earned the B.S. degree in computer science from Taylor University in 2001 and the M.S. degree in computer science from the University of Illinois at Urbana-Champaign in 2005. He is currently a Ph.D. candidate at the University of Illinois at Urbana-Champaign. His research interests include algorithmic robotics, planning under uncertainty, artificial intelligence, computational geometry, and motion planning.

**Steven M. LaValle** Steven M. LaValle received the B.S. degree in computer engineering, and the M.S. and Ph.D. degrees in electrical engineering, from the University of Illinois at Urbana-Champaign in 1990, 1993, and 1995, respectively. From 1995-1997 he was a postdoctoral researcher and lecturer in the Department of Computer Science at Stanford University. From 1997-2001 he was an Assistant Professor in the Department of Computer Science at Iowa State University. He is currently an Associate Professor in the Department of Computer Science at the University of Illinois. His research interests include planning algorithms, motion planning, computational geometry, and control theory. He authored the book Planning Algorithms, Cambridge University Press, 2006 (which is available online for free).