Global Localization Using Odometry

Jason M. O'Kane
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
jokane@cs.uiuc.edu

***Abstract*—This paper presents a global localization technique for a robot with only linear and angular odometers. The robot, whose configuration is composed of its position and orientation, moves in a fully-known environment by alternating rotations and forward translations. We pose the problem as a discrete-time planning problem in the robot's *information space*, which encapsulates the uncertainty in the robot's configuration. Our contribution is to show that in any simply-connected, bounded polygonal environment, localization by odometry alone is possible, but only up to the symmetries in the environment.**

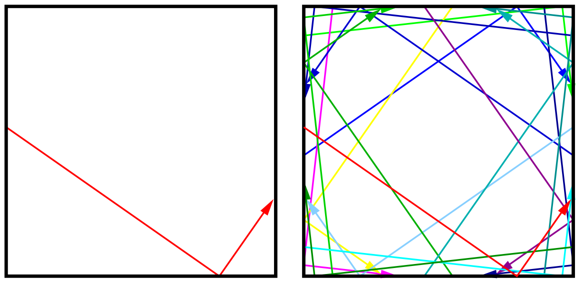
Fig. 1. [left] Two boundary-to-boundary motions in a square-shaped environment. [right] The 16 possibilities for motions of these lengths between boundary points in this environment.

I. Introduction

Localization is widely regarded as a central problem in mobile robotics. Many mobile robot tasks are more manageable for robots that know their location. A wide spectrum of sensor systems have been proposed for the localization problem, ranging from visibility sensors [11, 18, 31] to landmark detectors [5, 6, 13, 34]. How complex a sensor system does the localization problem truly demand? In this paper, we take a *minimalist* approach, describing a simple robot with which localization is still possible.

Consider a robot that has angular and linear odometers. Can such a robot localize itself? Under the model we describe, the robot can accurately rotate and translate through its environment, measuring each of these motions. Suppose the robot is given an accurate map of its environment, but has no knowledge of its configuration. This is the so-called "kidnapped robot" or "global localization" problem. Our contribution is to show that such a robot can localize itself, but only up to symmetries in the environment. Our choice to use an idealized geometric system is motivated by a desire to understand basic requirements for robotic tasks such as localization, independent of the idiosyncrasies of any hardware implementation.

We may consider this task as discrete-time planning problem, but this approach is complicated by the fact the robot's state is unknown. This leads us to define the robot's *information space* [24] and give methods for computing its *information state* within that space. Informally, the robot's information state is a set of configurations in which the true configuration is known to lie. As the robot moves, this set is updated to reflect the result of this motion for each candidate. As the robot receives sensor data, this set is pruned to eliminate candidates that are not consistent with these data. When the robot is finally localized, the information state will contain one element for each symmetry of the environment.

Space limitations prohibit a thorough survey of related work, but research in localization can broadly be divided into two groups. *Passive localization* [5, 6, 11, 12, 13, 18, 20, 25, 32, 34, 35, 36, 38] 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 contrast, *active localization* [14, 22, 27, 29, 30, 33] involves the selection of actions that eliminate position ambiguity. The problem we consider is an active localization problem. The minimalist approach has a long history in robotics, including many works in limited-sensing part orientation [3, 4, 7, 8, 15, 16] and simultaneous exploration and navigation [1, 2, 10, 21, 23, 28, 26, 37].

The remainder of this paper is organized as follows. Section II formally defines our robot model and problem definition. Section III describes our algorithm to solve this problem. We present an implementation of this algorithm in Section IV and discuss our results in Section V.

II. Problem Definition

We consider a point robot with orientation that can move in the plane by rotation and by forward translation. The robot is equipped with an accurate map of its environment, W, which we assume to be simply-connected, closed, bounded, and polygonal. Since the orientation is relevant, the robot's configuration space is $\mathcal{C} = W \times S^1$.

At each time step, the robot may issue either of two types of commands. First, the robot may rotate by a certain 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. For simplicity, we do not allow simultaneous rotation and translation.

We can model this system as a discrete-time planning problem. The robot begins at some unknown configuration $x_1 \in \mathcal{C}$. Let $U = \mathbb{R}^+ \sqcup S^1$ denote the robot's action space, in which elements in $\mathbb{R}^+$ denote translation commands and elements of S^1 denote rotation commands. Let $f : \mathcal{C} \times U \to \mathcal{C}$ denote the state transition function. If $u \in S^1$, then $f(x, u)$ is the appropriate change of orientation of x. If $u \in \mathbb{R}^+$, then $f(x, u)$ computes the appropriate forward translation of x within W, stopping short of the commanded distance only if the robot reaches the boundary of W first. An iterated version of this function that applies several actions in succession will also be useful:

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

Consider a sequence of commands $u_1, \ldots, u_K$. This 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 linear odometer. We may regard this observation as a "hint" regarding the true configuration of the robot. Let $Y = [0, \infty)$ denote the space of such observations, in which an observation $y \in Y$ indicates that in executing the previous action, the robot's translation had magnitude y. Since rotations always succeed without providing useful feedback, the robot receives the observation $y = 0$ after each rotation action. Let $h : \mathcal{C} \times U \to Y$ denote an *observation function* that gives the sensor reading that would result from choosing a particular action from a particular configuration. In this way we can define a sequence of observations $y_1, \ldots, y_K$ such that $y_k = h(x_k, u_k)$.

A. 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*

$$x_k = f^k(x_1, u_1, \ldots, u_{k-1}) \tag{2}$$

and

$$y_j = h(f^{j-1}(x_1, u_1, \ldots, u_{j-1}), u_j) \tag{3}$$

for each $j = 0, \ldots, k$.

The intuition is that 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. We may use the set of consistent configurations as 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 *η_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 $\mathcal{C}$.*

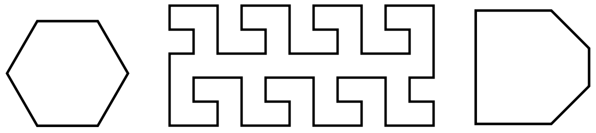
Fig. 2. Sample environments with, from left to right, 6, 2, and 1 rotational symmetries.

The information space is a sufficiently expressive structure that it can encode all of the robot's uncertainty about its configuration. As the robot moves, it can update its current information state, even in ignorance of its true configuration. 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 f and h:

$$F(\eta_k, u_k, y_k) = \{x \in \mathcal{C} \mid y_k = h(x, u_k)\} \cap \bigcup_{x \in \eta_k} \{f(x, u_k)\}. \tag{4}$$

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

B. Symmetries in the environment

Since the robot's movement and sensing are expressed in a frame local to the robot's (initially unknown) configuration, the notion of rotational symmetry in the environment is important.

Definition 3: A symmetry *is function composed of rigid translations and rotations mapping $\mathcal{C}$ onto itself. Two configurations $x_1, x_2 \in W$ are* symmetric *if there exists a symmetry $\mathcal{C} \to \mathcal{C}$ under which $x_1 \mapsto x_2$.*

Figure 2 shows several environments with varying numbers of symmetries. 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 Sketch: 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. □

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 η_k. For any $x \in \eta_k$ and $x' \in \mathcal{C}$ with $x \equiv x'$, $x' \in \eta_k$.

Proof Sketch: Since $x \in \eta_k$, there exists some initial configuration 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 τ under which $x' = \tau(x)$. Since f acts only locally, we know that a robot starting from x' and executing $u_1, \ldots, u_{k-1}$ will have configuration

$$f^j(x', u_1, \ldots, u_j) = \tau(f^j(x, u_1, \ldots, u_j)). \tag{5}$$

Moreover, the observation sequences are be identical, because the boundary edges of W are affected by τ in the same way as x' is. Consequently, $\tau(x_1)$ is an initial configuration 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 $x' \in \eta_k$. $\square$

The practical importance of this lemma is that 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. Therefore, we define the task of *localization up to symmetry*:

> *Given W, select actions to reduce the robot's information state to a set of symmetric configurations.*

Symmetry plays a similar role in some methods for part orientation [16].

More precisely, we approach the task of localization as a planning problem in $\mathcal{I}$ with initial state $\eta_1 = \mathcal{C}$ and goal region

$$\mathcal{I}_G = \{\eta \subset \mathcal{C} \mid \forall x_1, x_2 \in \eta, x_1 \equiv x_2\}, \tag{6}$$

or equivalently

$$\mathcal{I}_G = \{\eta \subset \mathcal{C} \mid |\eta/{\equiv}| = 1\}\,. \tag{7}$$

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 applying the action recommended by this function will lead in finite time to an information state in $\mathcal{I}_G$.

Note that the limitations arising from symmetry are no longer relevant if we are concerned only with determining the robot's *position* and are not interested in its final orientation. In this case we can, after reaching an information state consisting of symmetric points, issue additional commands to navigate to a point fixed by the environment's symmetries. We will not revisit this variant problem.

III. An Algorithm for Localization via Odometry

Now we turn to an algorithm for the task described in Section II. 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 η_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 III-B. For some degenerate but potentially interesting environments, InitialActions will fail 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. Section III-C candidates, the final section of the algorithm, detailed in Section III-E, systematically reduces η_k until only a set of mutually symmetric configurations remain.

Algorithm 1 Localize(W)

```
(η_k, k) ← InitialActions(W)
(η_k, k) ← EliminateSegments(W, η_k)

while |η_k| > NumberOfSymmetries(W) do
  repeat
    (x_1, x_2) ← SelectTwo(η_k)
    W_{x_1} ← TransformToLocalFrame(W, x_1)
    W_{x_2} ← TransformToLocalFrame(W, x_2)
  until W_{x_1} ≠ W_{x_2}
  p ← FindPointInOnlyOne(W_{x_1}, W_{x_2})
  (u_k, ..., u_{k'}) ← PathInPolygon(W_{x_1}, (0,0), p)
  while x_1, x_2 ∈ η_k do
    y_k ← ExecuteCommand(u_k)
    η_{k+1} ← F(η_k, u_k, y_k)
    x_1 ← f(x_1, u_k)
    x_2 ← f(x_2, u_k)
    k ← k + 1
  end while
end while

return η_{k-1}
```

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. For each of these motions, the linear odometer will report the distance moved. We will show that for environments in general position, only finitely many configurations are consistent with such a sequence of motions. The precise general position assumption we must make is that W has no pair of parallel boundary segments. Section III-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. This positions the robot to begin the first of two boundary-to-boundary motions.
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.

Figure 4 illustrates these initial motions. The commands to "move until reaching the boundary" can be realized by selecting an 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 will allow the robot to continue, as shown in Figure 3. If the robot knows it has reached an environment vertex, then there are already only finitely many candidates. The use of 90° rotations is

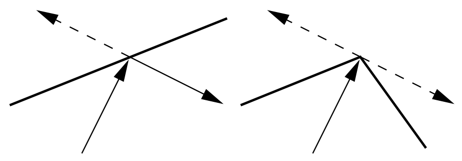

Fig. 3. [left] After reaching a wall, the robot can continue by rotating either 90° or $-90°$. [right] If the robot reaches a vertex, neither of these turns will allow non-zero translation.

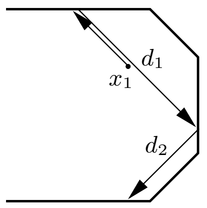

Fig. 4. Three initial motions intended to move the robot to an information state of finite cardinality.

motivated by the simplifications it affords in equation 9. In principle, rotations of other amounts would work equally well.

The problem remains to find the set of configurations of the robot that are consistent with these initial motions. For simplicity, we ignore the 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 1 shows an example environment and sample values for d_1 and d_2.

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 5.

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:

$$Aa^2 + Bab + Cb^2 + Da + Eb + F = 0 \tag{8}$$

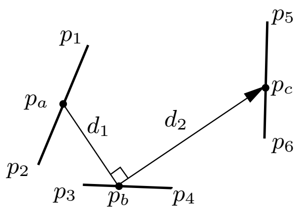

Fig. 5. Three fixed segments $\overline{p_1p_2}$, $\overline{p_3p_4}$, and $\overline{p_5p_6}$ and translations of length d_1 and d_2 between them.

with constant coefficients

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

where 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

$$p_c - p_b = s_1 \frac{d_2}{d_1}(p_b - p_a)^{\perp} \tag{9}$$

in which s_1 is either -1 or $+1$, depending on the "handedness" of the robot's motion, that is, 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:

$$Ga + Hb + I = 0 \tag{10}$$

with constant coefficients

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

Note that if either denominator is 0 (corresponding to $\overline{p_5p_6}$ being horizontal or vertical), the system can be solved trivially. Equations 8 and 10 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 $\overline{p_1p_2}$, $\overline{p_3p_4}$, or $\overline{p_5p_6}$ respectively. The final configuration (that is, position-orientation pair) of the robot resulting from such a candidate is simply $(p_c, \operatorname{atan}(y_c - y_b, x_c - x_b))$.

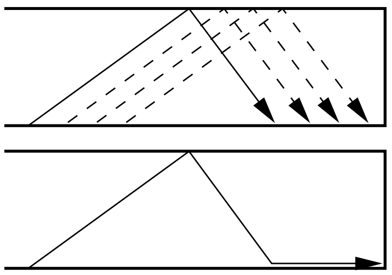

Fig. 6. [left] Parallel edges of the environment admit continua of candidate configurations. [right] A motion parallel to one of these segments leaves only a single candidate point.

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 $\overline{p_1p_2} = \overline{p_5p_6}$, 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.

As a final step, the candidate list must be pruned, retaining only those that represent motions that lie entirely within W. For each, it is sufficient to ensure that $\overline{p_ap_b}$ and $\overline{p_bp_c}$ are fully contained in W. Such queries can be answered in time $O(\log n)$ [9].

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 6. This case can be handled while iterating over the environment edge triples as described above.

1) Finding segments in the information state: Triples for which $\overline{p_1p_2} \parallel \overline{p_3p_4} \parallel \overline{p_5p_6}$ result a degeneracy in the system of Equations 8 and 10. Since we may have $\overline{p_1p_2} = \overline{p_5p_6}$, this may occur even if only two distinct boundary edges are parallel. These edge triples must be handled specially. Fixing these three segments, the set of consistent configurations is a (possibly empty) interval of $\overline{p_5p_6}$, with a single constant orientation across the interval. To compute this interval involves two steps: finding feasible motion directions, then using these directions to determine the extent of the interval.

The first step is to find a single motion of length d_1 between $\overline{p_1p_2}$ and $\overline{p_3p_4}$. This can be accomplished by segment-circle intersections, centering circles with radius d_1 at each of p_1, p_2, p_3, and p_4. If none of these circles intersects the opposite segment, then there are no feasible motions and the triple can be safely discarded. If there is an intersection point, it can be used to compute the direction of motion v_1 between $\overline{p_1p_2}$ and $\overline{p_3p_4}$ resulting in a distance of d_1. The appropriate direction of motion between $\overline{p_3p_4}$ and $\overline{p_5p_6}$ is $v_2 = s_1v_1^{\perp}$, using the same s_1 as in Equation 9.

Knowing the two directions v_1 and v_2, we use the forward projection algorithm for segments described in [29]. Given W, a segment on the boundary of W, and a motion direction v, this algorithm computes the set of possible final configurations for a robot that moves in direction v until reaching the environment boundary. This set is itself a union of segments. This added complexity is needed to account for other environment edges that may interfere; it is analogous to the ray-shooting queries used to prune point candidates in Section III-B.2. The algorithm must be applied twice, with intermediate pruning steps:

1) Project $\overline{p_1p_2}$ forward in direction v_1.
2) Discard any segments from the result that are not contained in $\overline{p_3p_4}$. or have distance other than d_1 from $\overline{p_1p_2}$.
3) Project the remaining results in direction v_2.
4) Discard any segments from the result that are not contained in $\overline{p_5p_6}$.

Since W is simply-connected, the result will be a single segment. The remaining segment $S \subset \partial W$ represents the set of final positions for feasible motions from $\overline{p_1p_2}$ to $\overline{p_3p_4}$ to $\overline{p_5p_6}$ with the appropriate distances and right-angle rotation. Therefore we must add the continuum $S \times \{\text{angle}(v_2)\}$ to η_k.

2) Eliminating segments: We have shown how, when W contains parallel boundary edges, to compute an initial information state that may contain configurations along intervals of W. Unfortunately, the methods of Section III-E to complete the localization process require that $|\eta_k|$ be finite. How can we command the robot in a way that eliminates these segments?

Let $S \times \{\theta\}$ denote such a segment. Informally, we want to issue commands so that if the robot's true configuration is in $S \times \{\theta\}$, the robot will move parallel to S. The resulting configuration will be known regardless of the robot's initial position within S, thereby collapsing the continuum $S \times \{\theta\}$ in η_k to (at most) a single configuration in η_{k+1}. This elimination can be accomplished with two actions. To align the robot's heading with S, command a rotation by difference between θ and the angle formed by S with the horizontal axis. Next, move forward until reaching a wall. These motions are illustrated in the bottom portion of Figure 6. Since the information state may contain multiple segments, this process may need to be repeated several times.

D. Computing the information transition function

In Sections III-B and III-C, we described a method for reaching an information state representable by a finite union of single configurations and (if the environment contains some parallel edges) boundary intervals. From this point forward, the robot must update its information state as it issues additional commands. Given an information state η_k, an action u_k, and an

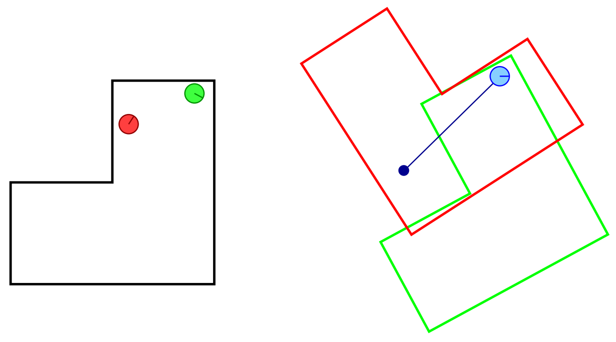

Fig. 7. [left] Two configurations in an L-shaped environment. [right] Two overlaid copies of the environment shown in the local frame of those configurations. Attempting to execute the path shown (which consists of one rotation and one translation) shown will result in different odometry readings for these two configurations.

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 4. The definition suggests the algorithm should proceed in two stages: First to find to forward projection of η_k under action u_k, then to prune from the result any configurations for which the distance traveled differs from y_k. For rotation actions, there is nothing to compute except to add or subtract the appropriate amount from the orientation part of each configuration. Translation of single configurations is easily accomplished by ray shooting in W. To translate segments requires the algorithm of [29]. In either case, to verify that candidates have traveled distance y_k requires a simple constant-time procedure.

E. Localization from a finite set

The previous sections showed how to select actions to guarantee that η_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 η_k and choose actions $u_k, \ldots, u_{k+j}$ so that η_{k+j+1} contains either x_1 or x_2 (or neither) but not both. This method is similar in spirit to that of [14].

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 η_k. Compute W_{x_1} and W_{x_2} and overlay them. See Figure 7. In this overlay, rotation and translation commands will 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. There are two cases to consider.

- If $W_{x_1} = W_{x_2}$, then $x_1 \equiv x_2$. These configurations cannot be be distinguished from one another.
- If $W_{x_1} \neq W_{x_2}$, then 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} will necessarily distinguish 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 η_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 η_{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 $O(n)$ [17, 19]. Also of potential interest is the *minimum-link path*, 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.

F. 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 will be 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 $O((n^3 + r^2 n) \log n)$.

Is is worth emphasizing that these bounds can likely be improved. The question remains unanswered whether any initial configuration in any environment achieves $r = \Omega(n^3)$. The experiments in Section IV suggest that in practical situations, both r and the number of disambiguation iterations will often fall far short of the upper bounds we present here. Also, in practice it is not unreasonable to assume that the computation time is dominated by the execution time for physical motions of the robot.

IV. IMPLEMENTATION

In order to demonstrate its feasibility, we have implemented Algorithm 1 in simulation, using simplified algorithms for many of the geometric computations. The implementation is in C++ on a GNU/Linux platform. Figure 8 shows a simple example in which the robot makes 13 motions to localize

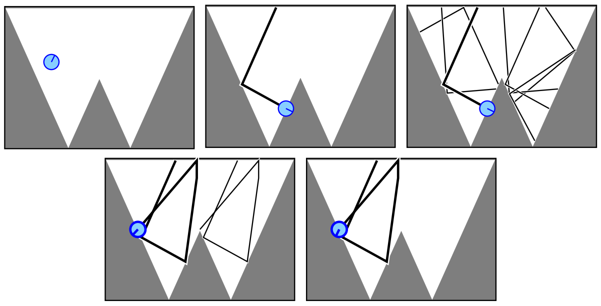

Fig. 8. Sample execution of Algorithm 1 generated by our implementation. Top row: (a) The robot in its initial configuration. (b) The motions generated by INITIALACTIONS. (c) There are 7 configuration consistent with these initial motions, so $|\eta_6| = 7$. Bottom row: (d) One disambiguation results in $|\eta_{12}| = 2$. (e) The robot is full localized after 13 commands, with final information state $|\eta_{14}| = 1$.

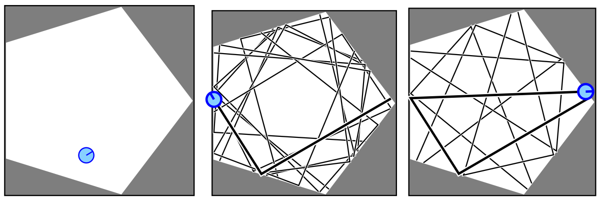

Fig. 9. A robot localizing itself in an environment with 5 symmetries. From top to bottom: (a) The robot's initial configuration. (b) Executing INITIALACTIONS results in an information state η_8 containing 15 configurations. (c) One disambiguation iteration fully localizes the robot, leaving 5 configurations in η_{10}.

itself. In Figure 9 the environment is a regular pentagon, so the final information state contains 5 configurations. The environment depicted in Figure 10 is serpentine and self-similar, but has no symmetries. This environment has 88 edges and geodesic diameter 65 meters. To gauge the efficiency of our implementation, we selected at random 100 configurations an executed the localization algorithm starting at each. The results of these runs are summarized in Figure 11. These experiments suggest that in at least some non-pathological situations, the algorithm's performance is significantly better than the upper bounds in Section III-F might indicate.

V. Discussion and Conclusions

A. Comparison to other models

The two most closely related lines of work are those of Dudek et. al. [14] and O'Kane and LaValle [29]. The two-phase approach described here – that of finding a finite set of candidates (*hypothesis generation*) followed by determination of the true configuration from among these candidates (*hypothesis elimination*) – echoes the approaches of both of these methods.

The robot model used here 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 the one presented here. Moreover, the work of [14] is mainly

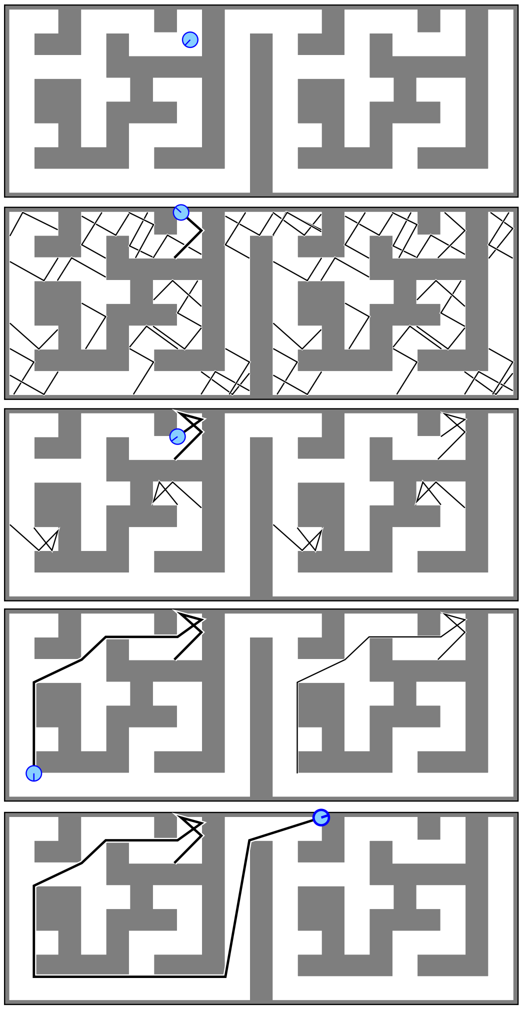

Fig. 10. A robot localizing itself in a serpentine environment. From top to bottom: (a) The robot's initial configuration. (b) Executing INITIALACTIONS results in an information state η_6 containing 48 configurations. (c) After 2 iterations of the disambiguation algorithm, only 6 configurations remain in η_{10}. (d) There are only two configurations in η_{20}. (e) The robot is fully localized after 25 motions.

concerned with *minimum-distance* localization, a problem we have not addressed. A central result of [14] is that minimum distance localization with a visibility sensor is NP-complete. The hardness question remains unanswered for sensing-limited robots such as ours, although it seems a reasonable conjecture that similar results would hold.

The model used in this work is not directly comparable to that of [29], which describes a robot equipped with only a compass and a contact sensor. In exchanging the compass for an angular odometer and the contact sensor for a linear odometer, we have strengthened the linear (distance) sensing while reducing the robot's angular sensing. In this sense, 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. The minimalist approach can be described as searching for minima in this partial order. The robot model of [14] dominates those of [29] and the present work, which are themselves incomparable.

There are also some subtle but perhaps illustrative differ-

	minimum	mean	maximum
Distance Traveled (m)	11.45	40.97	64.09
Initial Candidates	3	42.11	103
Actions Executed	9	21.84	45
Computation Time (s)	2.59	5.12	9.26

Fig. 11. One hundred initial configurations were randomly selected from the configuration space of the environment depicted in Figure 10.

ences with [29]. The present results are only up to symmetry, while symmetries are not relevant to the robot in [29]. This difference can be directly attributed to the fact that, in the present work, angular information is only local, rather than global. Likewise, the algorithm of [29] can only guarantee a known *final* configuration. In the present work, each motion is precisely measured. This provides sufficient information to determine the initial configuration and indeed the robot's entire path.

B. Future research

This work is based on an idealization in which the robot's control, sensing and internal map are all perfect. What if these assumptions are relaxed?

If the robot's control or sensing are noisy, we may model this noise with Gaussian distributions with known mean and variance. An information state is a probability distribution over $\mathcal{C}$. Such distributions would be periodic with the symmetries of the environment. The goal is to reach a distribution having at least $1-\delta$ of its mass concentrated in configurations within a ϵ-ball, for appropriately small choices of δ and ϵ.

If the robot's map is imperfect or absent, we may consider a space $\mathcal{E}$ of potential environments. The robot's state 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 $\mathcal{E} \times \mathcal{C} \times S^1$. If $|\mathcal{E}|$ is finite, we can compute candidates in each possible environment and continue until all remaining environment-configuration pairs are symmetric. If $\mathcal{E}$ is a richer set, perhaps defined by allowing tolerances in the positions of vertices, the extension is not as straightforward.

Finally, there remain a number of open questions regarding the hardness of the localization problem we have proposed. We have not attempted to optimize the distance traveled. Is minimum distance localization using odometry NP-hard? How large a distance must the robot travel at most? Can the bound of $O(n^3)$ initial candidates be improved?

Acknowledgments

The author gratefully acknowledges the mentoring of Steve LaValle. The author also thanks Ken Goldberg and Gregory Dudek for brief but helpful discussions in the initial stages of this work.

References

[1] 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.

[2] E. U. Acar and H. Choset. Robust sensor-based coverage of unstructured environments. In *Proc. of IEEE IROS, Int'l Conference on Intelligent Robots and Systems*, 2001.

[3] P.K. Agarwal, A.D. Collins, and J.L. Harer. Minimal trap design. In *Proc. IEEE Int. Conf. Robot. and Autom.*, volume 3, pages 2243–2248, 2001.

[4] S. Akella, W. Huang, K. M. Lynch, and M. T. Mason. Parts feeding on a conveyor with a one joint robot. *Algorithmica*, 26(3):313–344, March-April 2000.

[5] D. Avis and H. Imai. Locating a robot with angle measurements. *J. Symb. Comput.*, 10(3-4):311–326, 1990.

[6] K. Basye and T. Dean. Map learning with indistinguishable locations. In *Proc. Conf. Uncert. Artif. Intell.*, pages 331–342. North-Holland, 1990.

[7] R-P. Berretty, K. Goldberg, M. Overmars, and F. Van der Stappen. Trap design for vibratory part feeders. *Int. J. Robot. Res.*, 20(11), November 2001.

[8] M. Brokowski, M. A. Peshkin, and K. Goldberg. Curved fences for part alignment on a belt. *ASME Journal of Mechanical Design*, 117(1), March 1995.

[9] B. Chazelle and L. G. Guibas. Visibility and intersection problems in plane geometry. *Disc. and Comp. Geom.*, 4:551–589, 1989.

[10] H. Choset and J. Burdick. Sensor based planning, part I: The generalized Voronoi graph. In *Proc. IEEE Int. Conf. Robot. and Autom.*, pages 1649–1655, 1995.

[11] I. J. Cox. Blanche – an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Trans. Robot. and Autom.*, 7:2:193–204, 1991.

[12] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *Proc. IEEE Int. Conf. Robot. and Autom.*, 1999.

[13] E. D. Demaine, A. López-Ortiz, and J. I. Munro. Robot localization without depth perception. In *Scandinavian Workshop on Algorithm Theory*, 2002.

[14] G. Dudek, K. Romanik, and S. Whitesides. Localizing a robot with minimum travel. *SIAM J. Comput.*, 27(2):583–604, 1998.

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

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

[17] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. *J. Comput. Syst. Sci.*, 39(2):126–152, 1989.

[18] 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. Workshop on Alg. Found. of Robot.*, pages 269–282. A.K. Peters, Wellesley, MA, 1995.

[19] J. Hershberger. A new data structure for shortest path queries in a simple polygon. *Inform. Process. Lett.*, 38:231–235, 1991.

[20] R. Hinkel and T. Knieriemen. Environment perception with a laser radar in a fast moving robot. In *Proceedings of Symposium on Robot Control*, 1988.

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

[22] J. M. Kleinberg. The localization problem for mobile robots. In *IEEE Symposium on Foundations of Computer Science*, pages 521–531, 1994.

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

[24] S. M. LaValle. *Planning Algorithms*. Cambridge University Press (also available at http://msl.cs.uiuc.edu/planning/). To be published in 2006.

[25] J. Leonard, H. Durrant-Whyte, and I. Cox. Dynamic map building for an autonomous mobile robot. *Int. J. Robot. Res.*, 11(4):89–96, 1992.

[26] V. Lumelsky and S. Tiwari. An algorithm for maze searching with azimuth input. In *Proc. IEEE Int. Conf. Robot. and Autom.*, pages 111–116, 1994.

[27] M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for on-line problems. In *Proc. ACM Symp. Theory Comput.*, pages 322–333, 1988.

[28] C. Ó. Dúnlaing and C. K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[29] J. M. O'Kane and S. M. LaValle. Almost-sensorless localization. In *Proc. IEEE Int. Conf. Robot. and Autom.*, 2005.

[30] M. Rao, G. Dudek, and S. Whitesides. Randomized algorithms for minimum distance localization. In *Proc. Workshop on Algorithmic Foundations of Robotics*, pages 265–280, 2004.

[31] M. Rao, G. Dudek, and S. Whitesides. Minimum distance localization for a robot with limited visibility. In *Proc. IEEE Int. Conf. Robot. and Autom.*, 2005.

[32] W. Renken. Concurrent localisation and map building for mobile robots using ultrasonic sensors. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 1993.

[33] K. Romanik and S. Schuierer. Optimal robot localization in trees. In *Proc. Symp. Comp. Geom.*, pages 264–273, 1996.

[34] K. Sugihara. Some location problems for robot navigation using a simple camera. *Comp. Vis., Graphics, & Image Proc.*, 42(1):112–129, 1988.

[35] S. Thrun. Probabilisitic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[36] S. Thrun, W. Burgard, and D. Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Machine Learning*, pages 1–25, April 1998.

[37] 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.

[38] G. Weiss, C. Wetzler, and E. von Puttkamer. Keeping track of position and orientation of moving indoor systems by correlation of range-finder scans. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, 1994.