Probabilistic localization with a blind robot

Lawrence H. Erickson[1], Joseph Knuth[1], Jason M. O'Kane[2], Steven M. LaValle[1]
Department of Computer Science
University of Illinois at Urbana-Champaign[1]
Urbana, IL 61801 USA
{lericks4, knuth, lavalle}@uiuc.edu

Department of Computer Science and Engineering
University of South Carolina[2]
Columbia, SC 29208 USA
{jokane}@cse.sc.edu

***Abstract*— Researchers have addressed the localization problem for mobile robots using many different kinds of sensors, including rangefinders, cameras, and odometers. In this paper, we consider localization using a robot that is virtually "blind", having only a clock and contact sensor at its disposal. This represents a drastic reduction in sensing requirements, even in light of existing work that considers localization with limited sensing. We present probabilistic techniques that represent and update the robot's position uncertainty and algorithms to reduce this uncertainty. We demonstrate the experimental effectiveness of these methods using a Roomba autonomous vacuum cleaner robot in laboratory environments.**

I. Introduction

Localization is one of the best-studied problems in mobile robotics. Accurate knowledge of the robot's position within its environment is widely considered to be essential for mobile robots to be useful. Although robots often obtain such knowledge by some combination of sensor measurements, motion estimates, and pre-supplied initial conditions, information from sensors is usually the primary means of eliminating position uncertainty. Understanding the role of sensing in the localization process is therefore essential to a complete understanding of localization problems.

In the mobile robotics literature, localization problems take many forms. In this paper, we consider active global localization, in which a robot has access to a complete map of its environment but is totally ignorant of its position. The robot must purposefully direct its motions to eliminate that uncertainty. Using probabilistic techniques, we demonstrate that a certain global active localization problem can be solved with very limited sensing capabilities. This work is distinguished from prior probabilistic localization techniques by the limitations on the robot's sensing. We consider a differential drive robot equipped with a contact sensor and a clock, but no other sensors.

This work is motivated by a desire to understand the *information requirements* of important robotic tasks. By finding very simple robots that are able to complete certain tasks, we begin to identify necessary conditions on the sensing and motion capabilities for completing that task. In a more directly practical sense, studying robots with very simple sensing schemes is profitable because such robots are better suited (for reasons of cost and complexity) for deployment in large cooperative teams.

Prior theoretical work showed that active global localization problems can be solved under severe sensor limitations [25]. However, that work relies in crucial ways on assumptions that the robot's sensing and control are perfect. For example, certain portions of [25] depend on the robot's ability to slide along the environment boundary without triggering its contact sensor. If the robot's control admits even the slightest directional errors, such motions are impossible. In practice, sensing is rarely fully accurate and control is rarely perfect. In this paper we present new algorithms that are robust to these errors and demonstrate experimentally that these algorithms are effective.

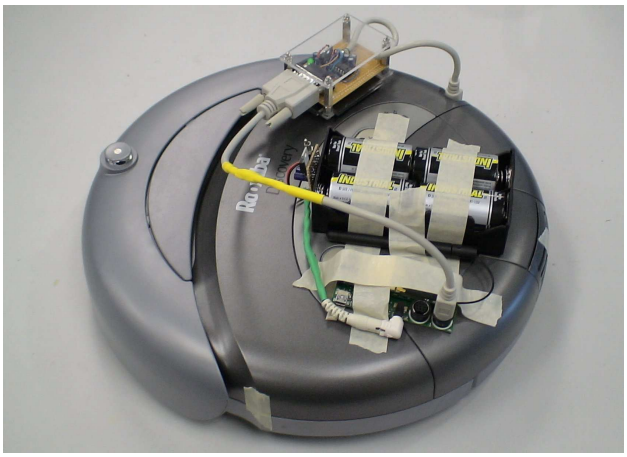

Fig. 1. A Roomba autonomous vacuum cleaner robot.

The most successful published approaches to localization are generally based on either particle filters [9], [12], [13], [19], extended Kalman filters [15], [20], or grid-based discretizations [3], [6], [7], [16]. The novelty of our work is that we use a robot model in which the sensing and motion capabilities are severely limited. These limitations introduce geometric issues, requiring new algorithms for both pose tracking and active localization. Our work also draws inspiration from theoretical results on localization that use geometrical reasoning and set-based representations of uncertainty [11], [25], [30].

Previous works on limited sensing exist, but even they use much more powerful robot models than we discuss in this paper. These include models that use short range (3-5 m) infrared sensors [4] and models that use a combination of sonar and fixed rangefinders [22]. These papers attempt to solve localization problems while delibrately avoiding

the use of omnidirectional laser rangefinders, devices that are used in a large portion of the papers on probabilistic localization, including [13] and [14]. Our research goes further in this direction by limiting the robot to absurdly simple sensors that are unable to detect obstacles the robot is not physically touching.

The minimalist approach we take has a long history in robotics. Researchers have studied the implications of sensing limitations for navigation [21], [27], exploration [1], [8], and manipulation [2], [23] tasks. The general problem of determining the information requirements of robotic tasks is taken up in [10], [24]. Our work contributes to this line of research by demonstrating that, with appropriate adaptations, such minimalist models are applicable in experimental contexts.

We conducted experiments using Roomba autonomous vacuum cleaner robots, shown in Figure 1. The Roomba is attractive as a research platform because it is inexpensive and readily available, and because its sensors closely match the abstract models we use.

Like many approaches to the localization task, our basic method is probabilistic. In fact, our work can be considered as exploring a kind of probabilistic information space [18], which arises in partially observable Markov decision processes (POMDPs) [17], [28], [31]. As our algorithm progresses, the robot maintains an approximate, discrete probability distribution over positions along the environment boundary. This distribution is updated in response to motions by the robot. For active localization, we use an entropy-based heuristic to choose uncertainty-reducing motions. The algorithm constructs a localization plan consisting of several subplans, each of which results in a monotonic decrease in entropy, even if the entropy temporarily increases within a subplan. This willingness to tolerate temporary increases in uncertainty is crucial to effective handling of the multimodal distributions that arise in our problems.

The remainder of this paper is organized as follows. We present formal definitions for our robot model and for passive and active localization problems in Section II. Algorithms to solve the passive and active problems appear in Sections IV and III respectively. Details about our experiments are in Section V, followed by discussion and conclusion in Section VI.

II. Problem statement

In this section, we introduce our robot model and define the passive and active localization problems we solve.

A. Robot model

A point robot with orientation moves in an environment $W \subset \mathbb{R}^2$ that is planar, closed, bounded, and polygonal. The environment need not be simply connected. Let $\partial W \subset W$ represent the boundary of W and let n denote the number of vertices of ∂W. The robot knows its initial orientation, but not its initial position within W.

The robot is equipped with a contact sensor and a clock, but no other sensors. We consider two types of motions that these sensors enable.

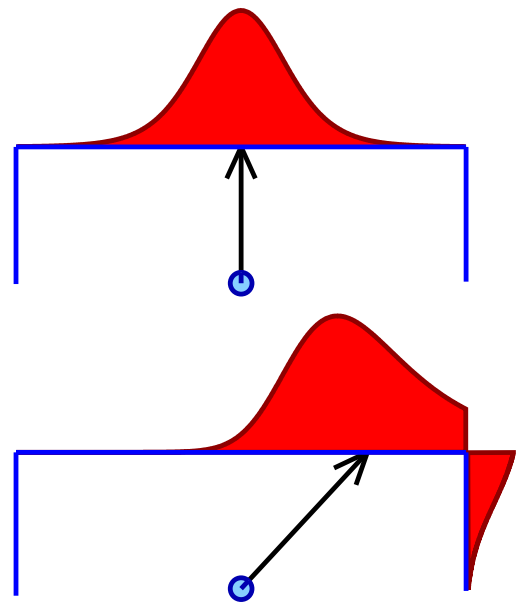

Fig. 2. A robot with Gaussian rotational error starts in the center of a rectangular environment and moves forward to the environment boundary. The probability density over points along the boundary is shown. For illustration purposes, the variance of the distribution is impractically large. [top] The robot moves upward. Although the resulting distribution appears superficially to be Gaussian, it is not. [bottom] The robot moves at an angle. Observe that, because of geometric effects, the mean of the resulting distribution need not be endpoint of the robot's nominal trajectory.

- *Rotations* – Using its clock, the robot can rotate in place by dead reckoning. Motion by dead reckoning is notoriously noisy, so we model the error in the amount of rotation by a zero-mean density p. We assume that p is strictly increasing below its mean and strictly decreasing above its mean, and that p has a continuous cumulative distribution function. Figure 2 shows an example, in which a robot starts in the center of a rectangular environment and moves upward.
- *Translations* – The robot can move forward, but since it lacks odometry, the only reliable translation it can make is to move forward until it reaches the environment boundary. We assume that the robot travels in a straight line, but because we explicitly model orientation errors, our algorithms are robust to small deviations from this assumption. Since we are interested in solving localization problems with a little sensor information as possible, we ignore the robot's clock during translations.

The robot's motions can be described as a sequence of discrete stages, in each of which the robot makes a single rotation, then moves forward until its contact sensor is triggered. We number these stages with consecutive integers $k = 1, 2, \ldots$. Since the robot can only move between points along ∂W, we need not consider the points in the interior of W as possible locations for the robot.

Note that rotational errors accumulate over time, and the robot's true heading will become more uncertain as more stages go by. Since the error in the robot's orientation at stage k will be the sum of the error at stage k plus all preceding error, we can say that the random variable R_k representing

the orientation error at step k is $R_k = \sum_{i=1}^{k} r_i$, in which r_i is a random variable describing the error resulting from step i. Note that the r_i's are independent and identically distributed according to p. Let p_k denote the distribution of R_k. The result of this sum will depend on the single-stage error distribution p. In the special case where p is Gaussian with variance σ^2, p_k is Gaussian with variance $k\sigma^2$.

B. Localization problems

We consider two related localization problems:

- *Passive localization* – The robot's motions are controlled by an external decision maker. The problem is to efficiently maintain and update the probability distribution of possible states of the robot.
- *Active localization* – The robot's primary task is to eliminate uncertainty in its position. The problem is to choose motions so that the robot will be certain it is in a disk of radius ϵ with a probability at least $1 - \delta$.

Observe that to solve the active localization problem typically requires as a "subroutine" a solution to the passive localization problem.

Unfortunately, as the robot moves, the distribution of possible states along ∂W becomes increasingly difficult to represent analytically, even if both the prior and error distributions are well-behaved. Moreover, geometric features within the environment will cause discontinuities that complicate the analytical representation of the distribution even further. To combat this complexity, we approximate the true distribution by discretizing the boundary of the environment into small cells of size at most 2ϵ and recording the amount of probability mass in each of these cells. Such a nonparametric representation is well-suited for representing the complex, multimodal distributions that arise in global localization. This discretization can be viewed as a piecewise-constant approximation to the underlying density function. We divide each edge e of ∂W into $\lceil \text{length}(e)/(2\epsilon) \rceil$ equally-spaced cells. Let $S = \{s_1, \ldots, s_m\}$, in which each $s_i \in S$ is line segment in ∂W, denote the set of discrete cells generated in this way.

Under this discretization, we can represent the robot's uncertainty as an m-dimensional column vector

$$P_k = \begin{bmatrix} P_{k,1} & \cdots & P_{k,m} \end{bmatrix}^{\mathrm{T}}, \tag{1}$$

in which $P_{k,i}$ is the probability of the robot being in cell s_i at stage k. We assume a uniform prior, so that

$$P_{0,i} = \frac{\text{length}(s_i)}{\text{perimeter}(\partial W)}. \tag{2}$$

If additional information about the robot's starting position is available (for example, a known starting position), this initial condition can be changed accordingly.

In this context, the input to the passive localization problem is the environment W, a discretization S, a motion direction u, and a probability vector P_k; the output is a probability vector P_{k+1} the updated to reflect this motion. Similarly, note that if at least $1 - \delta$ of the probability mass is concentrated in a single cell, we can be certain (modulo any errors introduced by the discretization) that the active localization problem has been solved. Figure 3 shows a starting distribution and possible solution for a very simple environment.

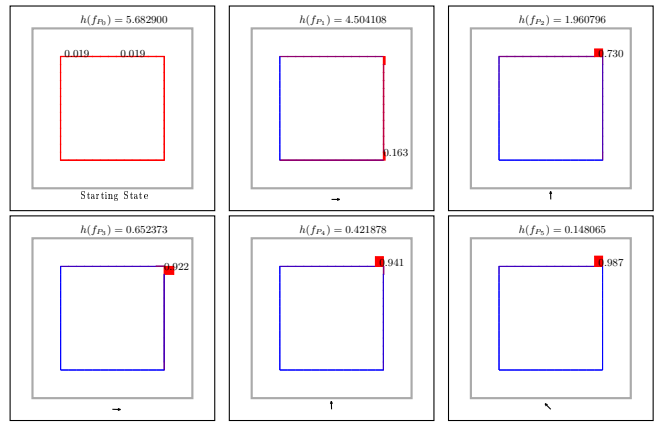

Fig. 3. A simple square environment, along with a 5-step plan that solves the active localization problem in that environment. The initial state is shown in the top left. The final state, which concentrates nearly all of the probability mass in one corner of the square, is shown in the lower right.

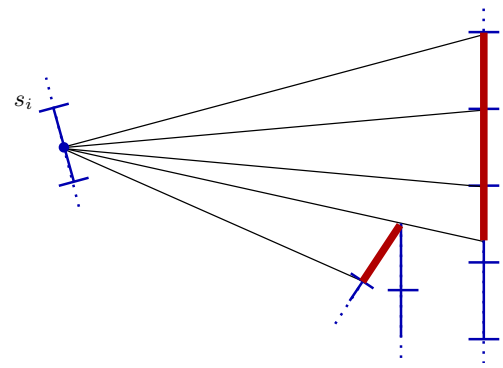

Fig. 4. Computing the set V_i of cells visible from a cell s_i. The diagram depicts three cells fully visible from the midpoint of s_i and one cell that, because of an obstruction, is only partially visible. All four cells are included in V_i.

III. Passive Localization

In this section, we solve the passive localization problem, in which the task is to observe the robot's execution and maintain a probability distribution of possible positions within the environment.

The robot receives as input a description of W, represented as a doubly-connected edge list. As a preprocessing step, we compute for each cell s_i in the environment discretization S a list $V_i \subseteq S$ of cells that are visible, either fully or partially, from the midpoint of s_i. See Figure 4. This step can be accomplished in time $O(mn \log n)$ by computing the visibility polygon in W of each such midpoint [26].

Given the environment W, a commanded motion direction u, and a belief distribution P_k, the problem of passive localization is to compute an updated distribution P_{k+1}. To accomplish this, we compute a transition matrix $R_{u,k}$ such that

$$P_{k+1} = R_{u,k} P_k \tag{3}$$

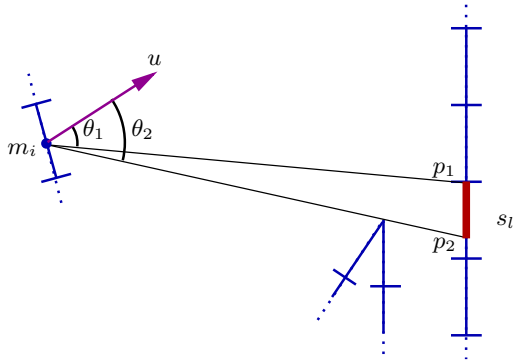

Fig. 5. Computing a single entry in $R_{u,k}$. Orientation errors between θ_1 and θ_2 will lead the robot from s_i to s_l.

The interpretation of $R_{u,k}$ is that the entry at row i, column j contains the fraction of probability mass that moves from cell s_i to cell s_j, under a motion in direction u. We use the subscript k on the transition matrix R to emphasize the dependence on time-varying the orientation error distribution p_k.

It remains to describe how to compute $R_{u,k}$. Column i of this matrix describes how probability mass moves from s_i to each other cell of the discretization. Since the robot moves in a straight line, this fraction is nonzero only for cells visible from s_i, that is, the cells in V_i. For each of these, we compute two angles θ_1 and θ_2 that bound the interval[1] of orientation errors that, given commanded motion direction u, leads the robot from the midpoint of s_i into a visible portion of s_j. By integrating p_k over the interval between θ_1 and θ_2, we obtain the transition probability from cell s_i to cell s_j. Since p_k has a continuous cumulative distribution function, the integral is guaranteed to exist. After all of the visible cells in V_i are accounted for, the remaining probability remains concentrated in s_i. This procedure is pictured Figure 5 and summarized in Algorithm 1. The algorithm runs in time $O(m^2)$, which (holding $\text{perimeter}(\partial W)$ constant), is $O(\epsilon^{-2})$.

IV. Active localization

Now we turn to the problem of active localization. We present an algorithm that chooses motions for the robot in order to eliminate uncertainty in its position. The intuition is to chain together a sequence of subplans, each of which "merges" the probability mass from two cells into a single destination. This basic structure is inspired by the algorithm of [25], but because we admit errors in control, the algorithm requires significant modifications. Our algorithm is greedy in the sense that it selects, from a group of candidate subplans, the subplan that makes the most "progress" toward localization.

[1]Note that, if the visibility of s_j by the midpoint of s_i is obstructed by an obstacle small relative to ϵ, then visible portion of s_j need not be a segment. Although, in general, this indicates the ϵ is too large for localization in W, this case can be handled in Algorithm 1 by integrating over each connected component of the visible portion of s_j and summing the results.

Algorithm 1 PassiveLocalization(W, S, P_k, u)

$R_{u,k} \leftarrow m \times m$ matrix of zeros
for $i \in \{1, \ldots, m\}$ **do**
  $m_i \leftarrow$ midpoint of s_i
  for $s_l \in V_j$ **do**
    if $s_l \neq s_i$ **then**
      $\overline{p_1 p_2} \leftarrow$ maximal subset of s_l visible from m_i
      $\theta_1 \leftarrow u - \textsc{Angle}(p_1 - m_i)$
      $\theta_2 \leftarrow u - \textsc{Angle}(p_2 - m_i)$
      $R_{u,k,l,i} \leftarrow \left| \int_{\theta_1}^{\theta_2} p_k(\phi) \mathrm{d}\phi \right|$
    end if
  end for
  $R_{u,k,j,j} \leftarrow 1 - \sum_{1 \leq j \leq m, j \neq i} R_{u,k,j,i}$
end for
$P_{k+1} \leftarrow R_{u,k} P_k$
return P_{k+1}

We propose a certain form of entropy as a progress measure in Section IV-A, describe how we generate subplans in Section IV-B, and combine these two elements to form a complete active localization plan in Section IV-C.

A. Progress measure

We follow [6] and others in using entropy as a heuristic for measuring the progress of the algorithm. Recall that P_k approximates the density of a continuous random variable representing the robot's true position. Let $f_{P_k} : \partial W \to [0, 1]$ denote the (piecewise constant) approximation to the true density function induced by P_k. This density has differential entropy

$$h(f_{P_k}) = -\int_{\partial W} f_{P_k}(x) \log f_{P_k}(x) \mathrm{d}x \quad (4)$$

$$= -\sum_{i=1}^{m} \int_{s_i} f_{P_k}(x) \log f_{P_k}(x) \mathrm{d}x \quad (5)$$

$$= -\sum_{i=1}^{m} \text{length}(s_i) P_{k,i} \log P_{k,i}. \quad (6)$$

Note in particular that this formulation differs from the discrete entropy of P_k, because the contribution of each cell to $h(f_{P_k})$ is weighted by the size of that cell. Our algorithm selects a series of motions $u_1, \ldots, u_k$ intended to minimize $h(f_{P_k})$.

B. Candidate subplans

Suppose two cells s_i and s_j each have nonzero probability in P_k. What actions by the robot will transfer (most of) the probability mass in these cells into a single common destination? A solution to a similar problem for a robot without errors appears in [25]. The approach is based on pursuit-evasion, in which one point (representing a possible position of the robot) chases another, repeatedly moving in the direction of the first step of the shortest path in W between the two points, until they finally merge.

Unfortunately, this solution is not directly applicable, because it often generates motions that require the robot

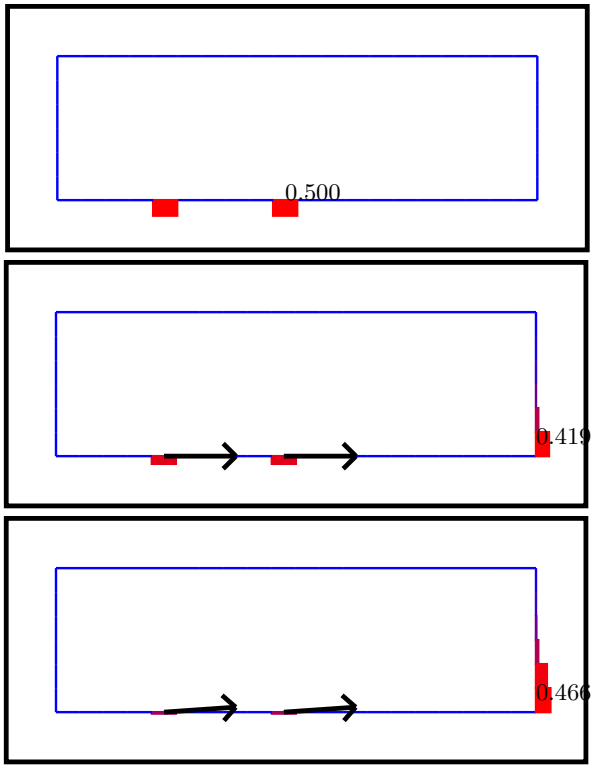

Fig. 6. Adjusting robot's motion angle. [top] An initial, bimodal distribution. [middle] Because of orientation errors, a motion parallel to the wall leaves a large fraction of the probability mass unaffected. [bottom] A small adjustment to the motion direction, computed by Algorithm 2 with $\alpha = 0.1$, corrects the problem.

Algorithm 2 AdjustAction(u, s_i, k)

$(x_1, x_2) \leftarrow$ endpoints of s_i, ordered so that the interior of W is on the clockwise side.
$m \leftarrow (x_1 + x_2)/2$
$\psi_1 \leftarrow \textsc{Angle}(x_1 - m)$
$\psi_2 \leftarrow \textsc{Angle}(x_2 - m)$
$u_{min} \leftarrow \min(u, \textsc{Angle}((x_2 - x_1)^{\perp}))$
$u_{max} \leftarrow \max(u, \textsc{Angle}((x_2 - x_1)^{\perp}))$
$u \leftarrow (u_{max} + u_{min})/2$
$b \leftarrow \int_{[\psi_2, \psi_1]} p_k(\theta) \mathrm{d}\theta$
if $b \leq \alpha$ **then**
  return u
end if
while $|b - \alpha| > \zeta$ **do**
  if $b < \alpha$ **then**
    $u_{min} \leftarrow u$
  else
    $u_{max} \leftarrow u$
  end if
  $u \leftarrow (u_{max} + u_{min})/2$
  $b \leftarrow \int_{[\psi_2, \psi_1]} p_k(\theta) \mathrm{d}\theta$
end while
return u

to move very close to ∂W without triggering its contact sensor. In extreme cases, the robot makes a "collapsing transition" by sliding along an edge of ∂W. Such motions are not reliable for the robot model in this work, because so much probability mass lies between angles occupied by the wall the robot is currently touching. This probability mass corresponds to the possibility that the robot may collide with the wall immediately. Figure 6 illustrates the phenomenon.

This behavior can be minimized by adjusting the robot's motion direction away from the boundary, thereby increasing the probability that the robot will move. Select an algorithm parameter $\alpha \in (0, 1/2]$, representing the maximum allowable chance of the robot failing to make a move. Then, given a desired motion direction u and a cell s_i, we can compute a new motion direction u', defined as the direction closest to u that leaves at most α probability of the robot not moving from s_i. The resulting angle u' will always be between u and the angle normal to the s_i in the interior direction. This u' can be approximated to within a tolerance ζ using binary search. Algorithm 2 summarizes this method, which runs in time $O(\log(1/\zeta))$.

This adjustment technique allows us to construct a subplan that attempts to unify the probability mass in a given pair of cells s_i and s_j. We use the midpoints of the cells as representatives, and apply the pursuit-evasion technique introduced in [25] to unify those representatives. At each step, however, we apply Algorithm 2 to adjust those motions to allow for orientation errors. This process continues until the two representatives are merged. Since, in contrast to [25], some motions may actually increase the robot's uncertainty, we truncate the subplan at the stage at which entropy is lowest. The complete algorithm to generate subplans appears in Algorithm 3.

C. A complete localization plan

Now we can state the complete active localization algorithm, which appears in Algorithm 4. At each step, it considers a set of candidate subplans generated by Algorithm 3 and appends to its master plan the candidate that improves the entropy the most. This process continues until no entropy-reducing candidate can be found.

Which candidate subplans should be considered? Ideally, all $m^2 - m$ possibilities should each be evaluated, but because generating each subplan is computationally expensive, this approach is impractical. Instead, we choose an algorithm parameter N and we assign a score $P_{k,i}P_{k,j}$ to each pair (s_i, s_j) of cells. The algorithm considers only the N pairs with the highest scores. This had the effect of excluding low-probability unions that would more likely scatter the probability mass around the environment rather than lower the entropy. Note that the pairs (s_i, s_j) and (s_j, s_i) are distinct (and therefore considered separately) but always have identical scores. In practice, we obtained acceptable results setting N as low as 10.

V. Experimental results

We have implemented this algorithm and evaluated its effectiveness in localizing a Roomba autonomous vacuum

Algorithm 3 CandidateSubplan(s_i, s_j, P_k)

$q_1 \leftarrow$ midpoint of s_i
$q_2 \leftarrow$ midpoint of s_j
$P_{min} \leftarrow P_k$
$\pi \leftarrow$ empty list of actions
while $|q_1 - q_2| > \epsilon/2$ **do**
  $u \leftarrow$ first step of the shortest path from q_1 to q_2
  $s \leftarrow$ cell in S containing q_1
  $u \leftarrow$ ADJUSTACTION(u, s, k)
  append u to π
  $R_{u,k} \leftarrow$ PASSIVELOCALIZATION(W, S, P_k, u)
  $P_{k+1} \leftarrow R_{u,k} P_k$
  $k \leftarrow k + 1$
  if $h(f_{P_k}) < h(f_{P_{min}})$ **then**
    $\pi_{min} \leftarrow \pi$
    $P_{min} \leftarrow P_k$
  end if
  $q_1 \leftarrow$ SHOOTRAY(W, q_1, u)
  $q_2 \leftarrow$ SHOOTRAY(W, q_2, u)
end while
return (π_{min}, P_{min})

Algorithm 4 ActiveLocalization(W)

$S \leftarrow$ discretization of ∂W into cells no larger than 2ϵ
$\pi \leftarrow$ empty list of actions
$P_0 \leftarrow \begin{bmatrix} \frac{\text{length}(s_1)}{\text{perimeter}(\partial W)} & \cdots & \frac{\text{length}(s_m)}{\text{perimeter}(\partial W)} \end{bmatrix}^{\mathrm{T}}$
$k \leftarrow 0$
loop
  $\boldsymbol{\mathcal{S}} \leftarrow S \times S - \{(s, s) \mid s \in S\}$
  sort $\boldsymbol{\mathcal{S}}$ by decreasing values of $P_{k,i} P_{k,j}$
  delete all but the first N elements from $\boldsymbol{\mathcal{S}}$
  $P_{min} \leftarrow P_k$
  $\pi_{min} \leftarrow$ empty list of actions
  for (s_i, s_j) in $\boldsymbol{\mathcal{S}}$ **do**
    $(\pi_{can}, P_{can}) \leftarrow$ CANDIDATESUBPLAN(s_i, s_j, P_k)
    if $h(f_{P_{can}}) < h(f_{P_{min}})$ **then**
      $\pi_{min} \leftarrow \pi_{can}$
      $P_{min} \leftarrow P_{can}$
    end if
  end for
  if π_{min} is empty **then**
    return π
  else
    append π_{min} to π
    $k \leftarrow k + \text{length}(\pi_{min})$
    $P_k \leftarrow P_{min}$
  end if
end loop

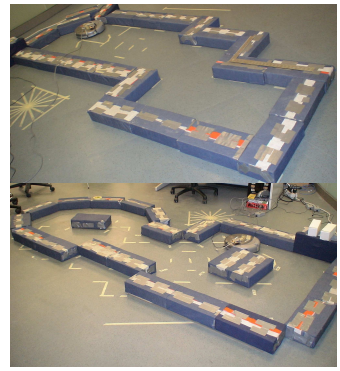

Fig. 7. Two laboratory environments we used to test Algorithm 4. The floor is a pitted vinyl surface, and the walls are covered cinderblocks.

cleaner robot. In this section, we describe those experiments.

Since the Roomba is a disk rather than a point, we perform computations using the configuration space of the robot in W, rather than W itself. Where the boundary of the configuration space is a circular arc, we use a piecewise-linear approximation by segments of length less than ϵ.

Note that even as simple a robot as the Roomba is equipped with several sensors that we ignore in our models. In particular, the robot has an infrared wall sensor with range approximately 5cm, a more powerful infrared sensor for receiving remote control commands, encoders for each wheel, and several sensors that report on the internal conditions of the robot (battery voltage, battery current, etc.). Although some of these sensors provide information that might be helpful for localization, we ignore them in this work because our intention is to find minimal sensor configurations that enable localization solutions.

A. Error modeling

Since the robot uses dead reckoning to rotate, calibration was required to minimize the rotational error. The robot was commanded to rotate a certain amount of time in an effort to get it to rotate a certain number of radians. Our results in this calibration indicate that the manufacturer-specified rotational speeds were off by small amounts, though large enough that they had to be considered. We also confirmed that the same calibration works well for two other Roombas not used for collecting the calibration data. The calibrations were made on a floor with pitted vinyl tiles, and it is possible that different flooring materials would require different calibrations.

At each stage, the rotation error is relatively small. In our experiments, we modelled this error by using for p a Gaussian distribution with variance $\sigma^2 = 0.0001$. Recall, however, that at stage k, the accumulated orientation error p_k has variance $\sigma_k^2 = k\sigma^2$.

B. Execution examples

We tested Algorithm 4 on two synthetic environments, depicted in Figure 7. Figure 8 shows a localization plan for the environment in the top portion of Figure 7. We used $\epsilon = 50$mm, $\alpha = .05$, and $N = 10$. The perimeter of the boundary of the free space is 5.6m, resulting in

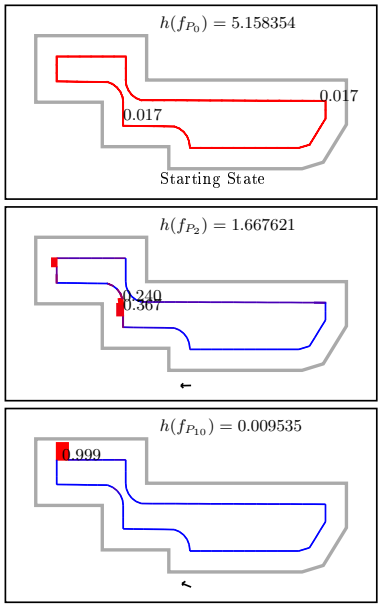

Fig. 8. Localization in a simple environment. [top] The initial probability distribution around the environment boundary. [middle] An intermediate step resulting from the first two steps of the plan. [bottom] The resulting distribution after the localization plan is run.

71 discretized cells. Our implementation took 18 seconds to compute a 10-stage plan that concentrated essentially all of the probability mass in a single cell. The resulting plan consists primarily of movements approximately upwards alternating with movements approximately left. The Roomba was able to repeatedly localize itself in this environment using this plan, without any failures. Our algorithm computes a similar plan even if the variance in the rotational error is increased by a factor of five. This occurs largely because the environment is relatively simple, having few concavities and no holes.

Figure 9 shows a significantly more difficult environment. There are two holes, and there is a narrow corridor in the center of the environment around which it is difficult for a Roomba to navigate. It also contains a segment in the upper left corner where the environment width is only larger than the robot by a few centimeters. We used $\epsilon = 50$mm, $\alpha = .1$, and $N = 10$. The perimeter of the boundary of the free space is 12.8m, resulting in 175 discretized cells. Our implementation took 4 minutes and 49 seconds to compute a 34-stage plan that concentrated almost 95% of the probability mass in a single cell. The Roomba was also able to consistently localize itself in this environment.

We also tested the algorithm in simulation (but not with physical experiments) on about 12 other environments. We were able to solve most of them without any parameter tuning using a standard set of parameters. The left portion of Figure 10 shows one of these, along with the final state after executing a 52-step localization plan. The environments requiring special tuning of α, N, and ϵ shared the feature of having portions between which it is difficult to move without stopping in the interior and turning, an action that the robot is incapable of performing. The right portion of Figure 10 shows an extreme example of this feature, where movements by our robot between arms of the environment are difficult to execute reliably.

VI. Discussion and Conclusions

In this paper, we presented algorithms for both passive and active localization problems a robot equipped with only a contact sensor and a clock, and demonstrated its usefulness experimentally. We have left several important areas unexplored.

First, the active localization algorithm depends on several constants that must be hand-tuned. The discretization resolution ϵ can be eliminated using dynamic discretization methods [5]. We found that the performance of Algorithm 4 depends on only weakly on N, the number of candidate subplans considered. It remains a challenging problem to choose α in an automated way that balances overly long localization plans (if α is too small) against the inability to deal with multimodal distributions (if α is too large).

Second, the environments we used for our experiments are relatively small and artificial. We are actively working to implement our techniques in a much larger, more realistic office environment. We expect larger environments to significantly increase the computation requirements and magnify the noise of the robot's control.

Finally, we are also interested in solving navigation problems with similar sensor limitations. This problem is related to both path planning and active localization, because the robot must carefully plan paths that keep uncertainty at manageable levels throughout the robot's motion. Prior work on so-called "coastal navigation" with longer-range sensors [29] is also relevant. More detailed tasks, such as mapping or delivery, might require even more informative sensors.

Acknowledgment

This work was supported in part by the DARPA SToMP program (DSO HR0011-07-1-0002).

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. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001.

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

[3] D. J. Austin and P. Jensfelt, "Using multiple gaussian hypotheses to represent probability distributions for mobile robot localization," in *Proc. IEEE International Conference on Robotics and Automation*, 2000, pp. 1036–1041.

[4] K. R. Beevers and W. H. Huang, "SLAM with sparse sensing," in *Proc. IEEE International Conference on Robotics and Automation*, pp. 2285–2290.

[5] W. Burgard, A. Den, D. Fox, and A. B. Cremers, "Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998.

[6] W. Burgard, D. Fox, and S. Thrun, "Active mobile robot localization," in *Proc. International Joint Conference on Artificial Intelligence*, 1997.

[7] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, "Acting under uncertainty: Discrete bayesian models for mobile robot navigation," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.

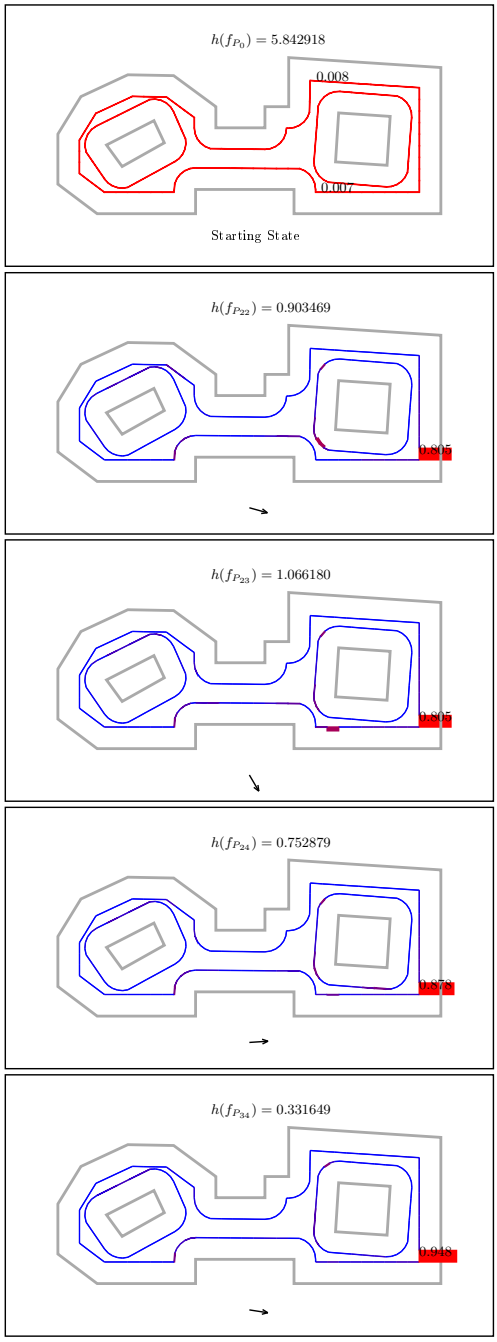

Fig. 9. Localization in a non-simple environment. [top] The initial probability distribution. [middle 3 pictures] A cell with intermediate probability merges with a high probability cell. [bottom] The final probability distribution.

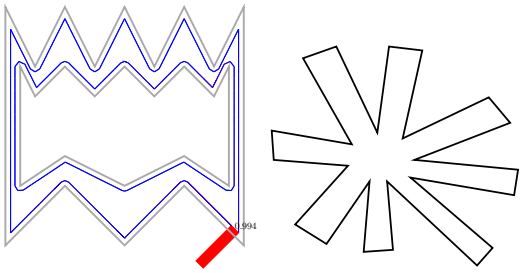

Fig. 10. [left] The final probability distribution after the algorithm was run on a complicated, jagged environment. [right] A problematic environment for the algorithm. The algorithm could not solve this environment with reasonable parameters.

[8] H. Choset and J. Burdick, "Sensor based planning, part I: The generalized Voronoi graph," in *Proc. IEEE International Conference on Robotics and Automation*, 1995, pp. 1649–1655.

[9] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *Proc. IEEE International Conference on Robotics and Automation*, 1999.

[10] B. R. Donald, "On information invariants in robotics," *Artificial Intelligence*, vol. 72, no. 1-2, pp. 217–304, 1995.

[11] G. Dudek, K. Romanik, and S. Whitesides, "Localizing a robot with minimum travel," *SIAM Journal on Computing*, vol. 27, no. 2, pp. 583–604, 1998.

[12] D. Fox, "Adapting the sample size in particle filters through kld-sampling",," *International Journal of Robotics Research*, vol. 22, pp. 985–1003, 2003.

[13] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proc. National Conference on Artificial Intelligence (AAAI)*, 1999.

[14] ——, "A monte carlo algorithm for multi-robot localization," in *AAAI/IAAI*, 1999, pp. 343–349.

[15] J.-S. Gutman and C. Schlegel, "AMOS: Comparison of scan matching approaches for self-localizing in indoor environments," in *Euromicro Workshop on Advanced Mobile Robots*, 1996.

[16] P. Jensfelt and S. Kristensen, "Active global localisation for a mobile robot using multiple hypothesis tracking," *IEEE Transations on Robotics and Automation*, vol. 17, no. 5, pp. 748–760, Oct. 2001.

[17] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, 1998.

[18] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 2006, also available at http://msl.cs.uiuc.edu/planning/.

[19] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in *Proc. IEEE International Conference on Robotics and Automation*, 2000.

[20] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transations on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991.

[21] V. J. Lumelsky and A. A. Stepanov, "Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape," *Algorthmica*, vol. 2, pp. 403–430, 1987.

[22] C. C. Mike Peasgood and J. McPhee, "Localization of multiple robots with simple sensors."

[23] M. Moll and M. Erdmann, "Manipulation of pose distributions," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 277–292, 2002.

[24] J. M. O'Kane and S. M. LaValle, "On comparing the power of mobile robots," in *Robotics: Science and Systems*, 2006.

[25] ——, "Localization with limited sensing," *IEEE Transactions on Robotics*, vol. 23, pp. 704–716, 2007.

[26] J. O'Rourke, *Art Gallery Theorems and Algorithms*. Cambridge, UK: Oxford University Press, 1987.

[27] C. H. Papadimitriou and M. Yannakakis, "Shortest paths without a map," *Theoretical Computer Science*, vol. 84, pp. 127–150, 1991.

[28] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration," in *Proc. International Joint Conference on Artificial Intelligence*, 2003, pp. 1025–1032.

[29] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation - mobile robot navigation with uncertainty in dynamic environments," in *Proc. IEEE International Conference on Robotics and Automation*, 1999.

[30] 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.

[31] N. L. Zhang and W. Zhang, "Speeding up the convergence of value iteration in partially observable Markov decision processes," *Journal of Artificial Intelligence Research*, vol. 14, pp. 29–51, 2001.