# Randomized Belief-Space Replanning in Partially-Observable Continuous Spaces

Kris Hauser

School of Informatics and Computing, Indiana University, hauserk@indiana.edu

**Abstract:** We present a sample-based replanning strategy for driving partially-observable, high-dimensional robotic systems to a desired goal. At each time step, it uses forward simulation of randomly-sampled open-loop controls to construct a belief-space search tree rooted at its current belief state. Then, it executes the action at the root that leads to the best node in the tree. As a node quality metric we use Monte Carlo simulation to estimate the likelihood of success under the QMDP belief-space feedback policy, which encourages the robot to take information-gathering actions as needed to reach the goal. The technique is demonstrated on target-finding and localization examples in up to 5D state spacess.

## 1 Introduction

Many robotics problems involve planning in uncertain, partially-observable domains, which requires reasoning about how hypothetical state distributions, *belief states*, change over time as the robot acts upon the world and gathers information with its sensors. Although this has been studied heavily in discrete domains, most realistic robotics problems have continuous, high-dimensional state spaces with nonlinear dynamics, which places them far out of the reach of tractability for state-of-the-art planners built for discrete systems. Although recent techniques have made progress in addressing continuous systems assuming Gaussian process and observation noise [3, 19, 21], the more general case of nonlinear and multi-modal belief states have proven to be much more challenging, in large part because of the difficulty of representing policies over an infinite-dimensional belief space [20, 22].

We present a Randomized Belief-Space Replanning (RBSR) technique that addresses an undiscounted and cost-free version of the continuous partially-observable Markov decision process (POMDP) formulation, where the belief state must be driven to a goal region. Rather than solving the POMDP once and using the solution as a lookup table, RBSR repeatedly generates coarse plans, executes the first step, and uses sensor feedback to refine future plans

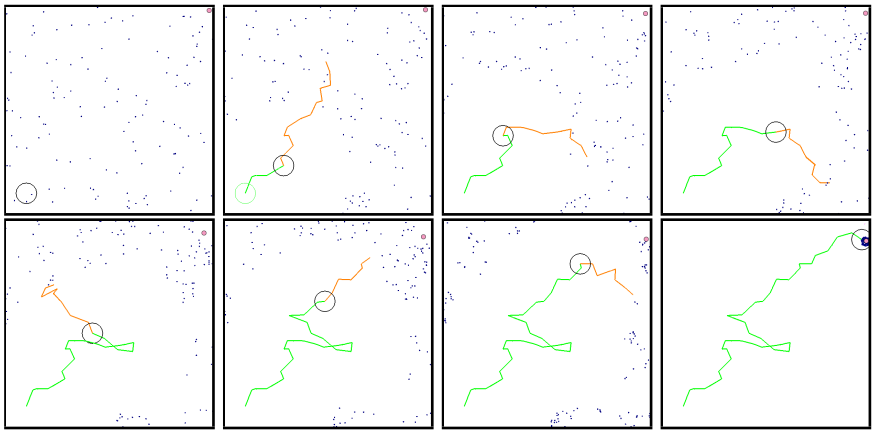


Fig. 1: An execution trace of a robot (large circle) searching for a wandering target (pink circle) in the unit square. The robot's sensor has a 0.25 unit range. The current belief state is represented by 100 particles (dots) and the current plan (orange) is updated by replanning.

(Figure 1). Much like a receding-horizon controller or model predictive controller, its success rate and computation time depend on the exploration strategy used to generate the search trees, and the evaluation function used to pick the "best" plan.

The RBSR exploration strategy performs a forward search in belief space by randomly sampling open-loop actions, and for an evaluation function we estimate the success rate of a QMDP-like policy. QMDP is a heuristic strategy that descends the cost-to-go function of the underlying MDP, averaged over the belief state [16], and it works well when state uncertainty is low, but with high uncertainty it may fall into local minima because it fails to perform active information-gathering. Hence, RBSR's random exploration strategy discourages information loss and encourages information-gathering actions because they improve the likelihood that QMDP succeeds.

RBSR employs random sampling approaches at multiple points in the procedure — random belief-space exploration strategies, particle filtering for state estimation, probabilistic roadmap-like approaches in the QMDP policy evaluation, and Monte-Carlo simulation in the evaluation function — making it highly parallelizable and applicable to high-dimensional spaces. In preliminary experiments, we applied RBSR to a simulated target pursuit problem with a 4-D state space and localization problems in up to 5-D (Figure 1). In our tests, RBSR computes each replanning step in seconds, and drives the belief state to a solution with high probability. Though our current implementation is promising, it is not as reliable in 6D or higher because it becomes much more difficult to maintain accurate belief estimates over time. Nevertheless,

we anticipate that future implementations of RBSR will be capable of solving many real-world robotics problems.

## 2 Related Work

Optimal planning in partially-observable problems is extremely computationally complex and is generally considered intractable even for small discrete state spaces [17]. Approximate planning in discrete spaces is a field of active research, yielding several techniques based on the point-based algorithms devised by Kearns et al [12] and Pineau et al (2003) [18]. For example, the SARSOP algorithm developed by Kurniawati et al (2008) has solved problems with thousands of discrete states in seconds [14].

Hypothetically, these algorithms can be applied to continuous problems by discretizing the space. But because of the "curse of dimensionality", any regular discretization of a high-dimensional space will requires an intractably large number of states. Porta et al (2006) has made progress in extending point-based value iteration to the continuous setting by representing belief states as particles or mixtures of Gaussians [20]. Thrun (2000) presented a technique that also works with continuous spaces by combining particle filtering with reinforcement learning on belief states [22]. For both of these methods, the need to approximate the value function over the infinite-dimensional belief space (either using alpha-vector or Q-value representations, respectively) comes at a high computational and memory expense. We use similar representations, but because we use replanning to avoid explicit policy representation, our approach sacrifices near-optimality for reduction in computational expense.

Several recently developed algorithms attempt to address continuous spaces by leveraging the success of probabilistic roadmaps (PRMs) in motion planning [11], which build a network of states sampled at random from the configuration space. Alterovitz et al (2007) present a Stochastic Motion Roadmap planner for continuous spaces with motion uncertainty, which solves an MDP using the discretization of state space induced by a PRM [1]. The techniques of Burns and Brock (2007) and Guibas et al (2008) augment roadmaps with edge costs for motions that have high probability of being in collision, and respectively address the problems of localization errors and environment sensing errors [4,7]. Huang and Gupta (2009) address planning for manipulators under base uncertainty by associating probabilistic roadmaps with particles representing state hypotheses and searching for a short path that is likely to be collision free [10].

Another set of related approaches use assumptions of Gaussian observation and process noise, which makes planning much faster because probabilistic inference can be performed in closed form. The Belief Roadmap technique of Prentice and Roy (2009) computes a roadmap of belief states under both motion and sensing uncertainty, under the assumptions of Gaussian uncertainty and linear transition and observation functions [21]. van den Berg et al

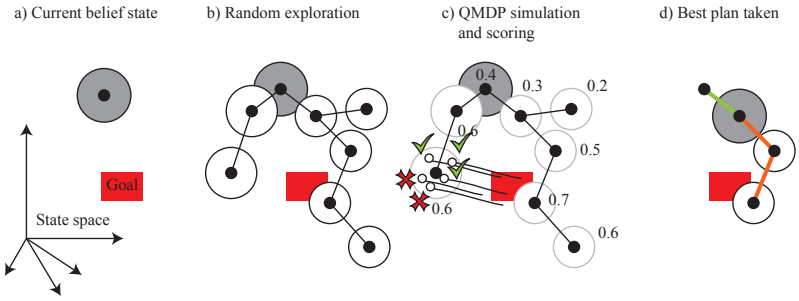



Fig. 2: Illustrating the replanning steps. (a) A belief-space search tree is initialized with the current belief state. (b) The tree is grown using random exploration of open-loop motions. (c) Nodes in the tree are scored with estimates of the likelihood of success under the QMDP policy. Traces of 5 belief-state particles under QMDP simulation are depicted. (d) The best plan is executed. (If the best node is the root, QMDP is executed by default).

(2010) consider path planning while optimizing the likelihood that a path is collision-free, under the assumption that a Linear-Quadratic-Gaussian feedback controller is used to follow the path. Platt et al (2010) and du Toit and Burdick (2010) construct plans using a maximum-likelihood observation assumption, and correcting for observation errors by replanning [6, 19]. RBSR also uses a replanning strategy, but uses a particle-based uncertainty representation that is better at handling nonlinear and multi-modal distributions, and makes no assumptions on the type of observations received.

The Randomized Path Planner (RPP) was an early approach in path planning in high-dimensional spaces that uses the principle that reactive policies often work well when the system is near the goal or when a space is relatively free of obstacles [2]. RPP plans by alternating steps of potential field descent and random walks to escape local minima, and was surprisingly effective at solving path planning problems that were previously considered intractable. RBSR shares a similar philosophy, but addresses problems with partial observablility.

## 3 Problem Definition

RBSR is given a POMDP-like model of the problem as input, and it interleaves planning and execution steps much like a receding-horizon controller. Each iteration performs the following steps:

1. The current sensor input is observed, and the robot's belief state is updated using a particle filter.
2. The planner generates a truncated search tree rooted at the current belief state. (Figure 2.a–b)

3. The robot executes the action associated with the "best" branch out of the root node. (Figure 2.c–d)

This section describes the POMDP formulation, particle filtering belief state update, and the QMDP policy that is used to evaluate the quality of nodes in the tree.

### 3.1 POMDP Modeling

The problem is formalized as an undiscounted partially-observable Markov decision process (POMDP) over a set of states $S$, actions $A$, and observations $O$. $S$, $A$, and $O$ are treated as subsets of Cartesian space, although this is not strictly necessary. A *belief state* is defined to be a probability distribution over $S$. We address the setting where the robot starts at an initial belief state $b_{init}$ and wishes to reach a goal set $\mathcal{G} \subseteq S$ with high probability. We treat obstacles by moving all colliding states to a special absorbing state. At discrete time steps the robot performs an action, which changes its (unobserved) state, and it receives an observation.

Although most POMDP formulations are concerned with optimizing rewards and action costs, we treat a somewhat simpler problem of simply maximizing the probability of reaching the goal. We also do not consider discounting. Discounting is numerically convenient and has a natural interpretation in economics, but in many respects is inappropriate for robotics problems because it gives preference to short term rewards.

The dynamics of the system are specified in the *transition model* $\mathcal{T} : s, a \rightarrow s'$ that generates a new state, given an existing state and an action. The sensor model is specified in the *sensor model* $\mathcal{O} : s \rightarrow o$ that generates an observation given a state. These models are stochastic, and we let the notation $s' \leftarrow \mathcal{T}(s, a)$ and $o \leftarrow \mathcal{O}(s)$ denote sampling at random from the posterior distributions of $\mathcal{T}$ and $\mathcal{O}$, respectively.

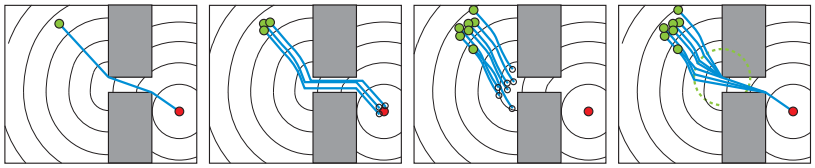
### 3.2 Simulation and Filtering with Belief Particles

To approximate the distribution over state hypotheses, we represent a belief state $b$ as a weighted set of $n$ particles $\{(w^{(1)}, s^{(1)}), \ldots, (w^{(n)}, s^{(n)})\}$, where $n$ is a parameter, and the weights sum to 1. Using such a representation, we can easily simulate an observation $o \leftarrow \mathcal{O}(s^{(k)})$ by sampling particle $(w^{(k)}, s^{(k)})$ proportional to its weight. We also define functions that compute the successor belief state after executing action $a$:

$$\mathcal{T}(b, a) = \{(w^{(i)}, \mathcal{T}(s^{(i)}, a))\}_{i=1}^{n} \tag{1}$$

And the posterior belief state after observing $o$:

$$\texttt{Filter}(b, o) = \{(\frac{1}{Z} w^{(i)} Pr(O(s^{(i)}) = o), s^{(i)})\}_{i=1}^{n} \tag{2}$$



(a) (b) (c) (d)

Fig. 3: The QMDP policy will succeed for well-localized belief states (a,b), but it may fall into local minima for a poorly localized belief state (c). On the other hand, QMDP allows the robot to incorporate new information during execution. So, if it could sense the corner of the obstacle as a landmark, then QMDP will also reach the goal (d).

where $Z$ is a normalization factor that ensures weights sum to 1.

We assume the robot performs state estimation using a particle filter, which have many variants that are beyond the scope of this work. We refer the reader to the survey in [5] for details. Most of these techniques address the problem of sample impoverishment that arises when few particles in $b$ are consistent with a given sequence of observations. For the remainder of this paper, we will assume that the chosen filter is robust enough to maintain sufficiently representative belief states.

### 3.3 QMDP Policy

As an endgame strategy, RBSR uses the incomplete QMDP policy that is quite successful in practice for highly-localized belief states or when information can be gathered quickly to localize the state (see Figure 3). QMDP is also used in RBSR to define a function $f(b)$ that measure the quality of hypothetical belief states by simulating how well QMDP makes progress toward the goal.

The QMDP policy essentially takes the optimal action assuming full observability is attained on the next step [16]. Suppose we are given a complete cost-to-go function $C_{fo}$ the fully-observable version of the problem. We will put aside the question of how to compute such a function until Section 3.4, and currently we describe how to use $C_{fo}$ to derive a QMDP controller for the partially-observable belief space.

The belief-space policy $\pi_{\text{QMDP}}(b)$ is defined to descend the expected value of $C_{fo}$ over the distribution of states in $b$. More precisely, we define

$$C(b) \equiv E_{s\sim b}[C_{fo}(s)] \approx \sum_{i=1}^{n} w^{(i)} C_{fo}(s^{(i)}), \tag{3}$$

and define $\pi_{\text{QMDP}}$ to pick the action that descends $C(b)$ as quickly as possible:

$$\pi_{\text{QMDP}}(b) \equiv \arg\min_{a} C(\mathcal{T}(b,a)). \tag{4}$$

If the expected value of the resulting belief state does not reduce $C(b)$, we define $\pi_{\text{QMDP}}(b)$ to return "terminate". Collision states are assigned infinite potential. In practice, we compute the arg min in (4) by sampling many actions and picking the one that minimizes the RHS.

The QMDP policy alternates state estimation and potential field descent using the following feedback controller:

---

`QMDP`
**Input**: belief state $b_0$.
1. For $t = 1, 2, \ldots$, do:
2. Sense observation $o_t$
3. $b_o \leftarrow \texttt{Filter}(b_{t-1}, o_t)$
4. $a_t = \pi_{\text{QMDP}}(b_o)$
5. If $a_t =$"terminate," stop. Otherwise, execute $a_t$.
6. $b_t \leftarrow \mathcal{T}(b_o, a_t)$

---

QMDP is also used in RBSR to measure quality of future belief states. We define a belief-state evaluation function $f(b)$ that uses Monte-Carlo simulation of QMDP on a holdout set of $m$ particles $\{s^{(1)}, \ldots s^{(m)}\}$ from $b$ which are used to simulate "ground truth". The complement of the holdout set $b'$ is used as the initial belief state. For each test sample $s^{(i)}$, QMDP is invoked from the initial belief state $b_0 = b'$, and $s_0 = s^{(i)}$ is used for simulating the "true" observation $\mathcal{O}(s^{(i)})$ (Line 2). It is also propagated forward along with the belief state using the transition model $s_t \leftarrow \mathcal{T}(s_{t-1}, a_t)$ (Line 6). This continues until termination.

To enforce an ordering on $f(b)$ (with higher values better), we incorporate two results of the QMDP simulation: $s$, the fraction of terminal states $s_t$ that lie in the goal $\mathcal{G}$, and $c$, the average QMDP value function evaluated at the terminal belief states $C(b_t)$. We prioritize success rate $s$ over the value function $c$ by letting $f(b)$ return a tuple $(s, -c)$. To compare the tuples $f(b_1) = (s_1, -c_1)$ and $f(b_2) = (s_2, -c_2)$ we use lexicographical order; that is, the value function is used only break ties on success rate. (This usually occurs when all locally reachable belief states have zero success rate.)

### 3.4 Computing Value Functions for the Fully-Observable Problem

Let us now return to the question of how one might provide a potential field $C_{fo}$ for the fully-observable version of the input POMDP. The policy that descends $C_{fo}$ is assumed to be complete, that is, if state is fully observable, then a descent of $C_{fo}$ is guaranteed to reach the goal. Although in discrete POMDPs such a function can be computed using value iteration on the underlying MDP, the problem is more difficult in continuous POMDPs.

In problems with no motion uncertainty, $C_{fo}$ is simply a cost function of the underlying motion planning problem. This can sometimes be com-

puted analytically; e.g., for a robot with unit bounds on velocity in a convex workspace, $C_{fo}$ is the straight-line distance to the goal. For more complex problems with high-dimensional or complex state spaces, approximate methods may be needed. In our examples we use a Probabilistic Roadmap (PRM) [11] embedded in $\mathcal{S}$, where each point in space is identified with its closest vertex in the roadmap, and the shortest distance from each vertex to the goal is computed using Dijkstra's algorithm. We build the PRM with a sufficiently large number of samples such that shortest paths in the roadmap approximate shortest paths in $\mathcal{S}$. By caching the shortest distance for each PRM vertex, computing $C_{fo}(s)$ runs in logarithmic time using a K-D tree to lookup the vertex closest to $s$.

This PRM-based potential field assumes that velocities can be chosen in any direction and with unit cost, but can be adapted to handle differentially-constrained systems using other sample-based motion planners. If actions are stochastic, a potential based on the Stochastic Motion Roadmap [1] might yield better results. We leave such extensions to future work.

## 4 Randomized Belief Space Replanning

The replanning algorithm used by RBSR grows a *belief tree* $T$ whose nodes are belief states $b \in \mathcal{B}$, and the edges store open-loop actions.

---

`Randomized Belief Space Replanning`
**Input**: Current belief state $b_0$, current plan $a_1, \ldots, a_t$.
1. Initialize $T$ with the belief states in the plan starting from $b_0$.
2. For $i = 1, \ldots, N$, do:
3. Pick a node $b$ in $T$ and an action $a$.
4. If $b' = \mathcal{T}(b, a)$ is feasible, add $b'$ to $T$ as a child of $b$.
5. End
6. Sort the nodes in $T$ in order of decreasing $EIG(b)$.
7. For the $M$ best nodes in $T$, evaluate $f(b)$.
8. Return the plan leading to the node with the highest $f(b)$.

---

In Line 3 we use a Voronoi exploration strategy to quickly distribute nodes across belief space. Lines 6–7 are used to avoid evaluating $f$ on all nodes on the tree, because it is an relatively expensive operation. We use an *expected information gain* score $EIG(b)$ to restrict the evaluations of $f$ to a small subset of nodes $M << N$. Because $EIG(b)$ is less expensive than $f$ to compute, this strategy leads to major speed gains. These strategies are described in greater detail below.

### 4.1 Voronoi-Biased Exploration Strategy

The exploration strategy is designed to cover the space of reachable open-loop motions quickly, and we use a Voronoi-biasing strategy much like the Rapidly-Exploring Random Tree (RRT) motion planner [15]. To expand the tree, we sample a random target point $s_{tgt}$ from the state space $S$, and sample a set of representative particles from all belief states in the tree $R = \{s | b \in T, s \sim b\}$. Then, we find the closest point $s$ from $R$ to $s_{tgt}$. We then find a control $a$ action that brings $s$ closer to $s_{tgt}$.

### 4.2 Expected Information Gain Scoring Strategy

We use an expected information gain strategy to avoid running expensive evaluations of $f$ on belief states that are unlikely to yield improvements in $f$. The intuition is that information gain is a sort of proxy score for QMDP favorability because it measures the spread of a belief state distribution, and QMDP tends to succeed more when states are localized. We compute the expected information gain for a belief state $b$ as follows. The information gain of the observation $o$ is the Kullback-Leibler divergence between the posterior distribution $b_o \equiv Pr(s|o, b)$ and the prior $b \equiv Pr(s|b)$:

$$I(b_o||b) = \int_{s \in S} Pr(s|o, b) \log \frac{Pr(s|o, b)}{Pr(s|b)}. \tag{5}$$

Given a particle representation of belief states $b_o$ and $b$, we replace the distribution $Pr(s|b)$ using a kernel density estimator with Gaussian kernels centered on the particles in $b$, and approximate the integral by the weighted sum over the particles $s^{(i)}$ in $b_o$.

The expected information gain is simply the expectation of (5) over $o$:

$$EIG(b) = \int_{o \in O} Pr(o|b) I(b_o||b) \tag{6}$$

To compute this, we compute the observation $o^{(i)} \equiv \mathcal{O}(s^{(i)})$ for each particle in $b$, perform particle filtering $b_o = Filter(b, o^{(i)})$, and then compute the weighted average of (5) over all particles. Although $EIG$ is an $O(n^3)$ computation, in our experiments it is typically orders of magnitude faster than computing $f$, and this scoring stage leads to major speedups.

### 4.3 Complexity and Convergence

The time complexity of RBSR depends on several parameters: the number of belief space particles $n$, the number of holdout particles for QMDP simulation $m$, the number of exploration steps $N$, and the number of nodes retained for QMDP evaluation $M$. Assume that an evaluation of $\pi_{\text{QMDP}}(b)$ (4) takes time $O(n)$. Then, the exploration stage takes time $O(nN^2)$, the EIG scoring takes

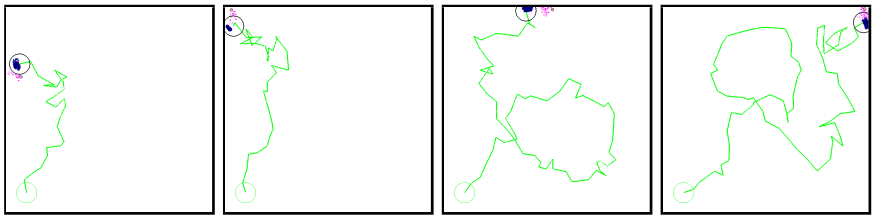


Fig. 4: Execution traces of the pursuit example for four different initial target locations (purple circles). The robot uses a distance sensor with maximum range 0.25.

time $O(n^3N)$, and the evaluation stage takes time $O(TnmM)$ where $T$ is the average number of steps taken by QMDP before it terminates. But the running time of the evaluation stage hides a higher constant factor because it uses more expensive operations such as state and path collision checking, and in our experiments it dominates running time. Space complexity is $O((n+m)N)$.

The parameter $n$ affects how accurately RBSP tracks and predicts belief states using the particle filter, and should be set high enough to attain a desired accuracy. In our experiments we do a small amount of tuning to find a reasonable parameter. $m$ affects how accurately RBSP predicts the success rate of the QMDP policy, and $f(b)$ may be quite noisy for low $m$. Specifically, the variance of the success rate estimate $p$ is bounded by $p(1-p)/m$, and $m$ should be chosen to achieve a desired accuracy. Parameters $N$ and $M$ affect the chance that RBSR makes progress toward the goal in a single time step. We used parameters $N = 100$ and $M = 10$ in our experiments.

## 5 Experimental Results

We performed experiments on two scenarios: a 2D pursuit scenario with a 4D state space, as well as a localization scenario that has tunable dimension. Although these problems are not difficult to solve using special-purpose strategies, they pose a challenge for general-purpose planners to solve in a reasonable amount of time and memory. For example, the SARSOP planner [14] can approximately solve a coarsely discretized version of the pursuit scenario in a few minutes, but it exhausts our test machine's 2Gb of memory once the resolution of the workspace grid exceeds 15x15.

### 5.1 Pursuit Scenario

Our first set of experiments consider a pursuit scenario in the unit square where the robot must reach a slower target that moves at random (Figure 1). The position of the robot is observable and controlled precisely, but it cannot sense the target outside a circle of radius 0.25. The target's position is a uniform distribution in the initial belief state, and the goal condition is to

achieve a distance of 0.05 to the target. We tested three sensor models: 1) a position sensor that reports the target's $x, y$ position relative to the robot, 2) a direction sensor that reports only direction and not distance, and 3) a distance sensor that does not report direction.

Using preliminary experiments we tuned the number of particles in the belief state needed for accurate particle filtering, and found that 100 particles were sufficient for the position and direction sensor, and 200 particles were needed for the proximity sensor. So, we used $m = 50$ particles as a holdout set, and $n = 150$ and $n = 250$, respectively, for the position/direction sensors and the proximity sensor. In 25 trials on each of these problems, with random target start states, RBSP never failed to reach the target. Several execution traces for different initial target positions are drawn in Figure 4. Average path length is approximately 1.7, which is close to the expected path length computed by SARSOP on a 15 x 15 grid, but is still suboptimal. Each replanning iteration took about 15 s on average, with standard deviation $\tilde{1}0$ s.

### 5.2 Localization Scenario

In our second scenario a robot is in an unknown configuration in a known $d$-dimensional environment and must localize itself and reach a small goal by measuring the distance to obstacles. The sensor has a limited range, which requires that the robot perform several steps of active sensing before reaching the goal. The optimal strategy is to proceed toward a wall until the sensor returns a reading, and then proceed to an adjacent wall until a closer reading is obtained, and so on until it achieves $d$ readings from $d$ linearly independent walls. Note that RBSR does not have a "proceed until" action in its action set, so instead it must approximate such a policy by a sequence of conditional movement actions and sensing actions.

In the first experiment (Figure 5), we set $d = 2$, $\mathcal{S}$ is the unit square, the initial belief state is a circular Gaussian distribution with standard deviation 0.1, and the goal radius and the sensing radius are both set to 0.05. To represent belief states we used 150 particles with a holdout set of 50. We also tested a space containing obstacles (Figure 6). In both examples, RBSR performs localization by moving close to obstacle boundaries, in somewhat random fashion, until it senses nearby walls. This continues until sufficient data is gathered to reach the goal.

We tested the effects of reducing the size of the holdout set $m$, and what we found was that the resulting executions tend to be much more noisy due to spurious noise in $f(b)$. In such cases, we found better results when replanning is initiated only when the current belief state experiences a large information gain due to an incoming observation (Figure 7). We hope to explore this strategy further in future work.

Our final set of experiments tested scalability with respect to dimension. Figure 8 plots the number of replanning steps taken by RBSR in problems from $d = 3$ to $d = 5$ in the unit hypercube. We increased the number of

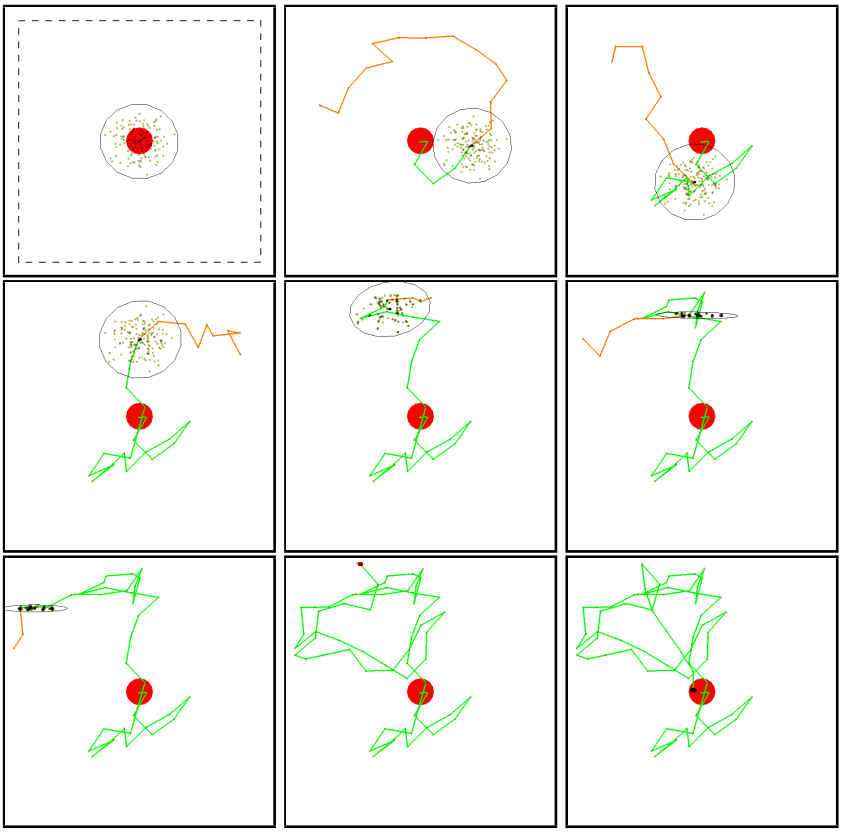


Fig. 5: An execution trace of a robot localizing itself to reach the red circle with high probability. Its sensor measures the distance to the walls, and has maximum range 0.05 (dashed lines). The current belief state is represented by 100 particles (dots) with a covariance ellipsoid, and the current plan (orange) is updated by replanning.

particles to 500, but kept all other parameters unchanged from the experiment in Figure 5. These experiments suggest that the number of replanning steps is roughly linear in dimension. Running time per timestep is roughly linear in dimension as well, ranging from approximately 6 s in the 3D case up to approximately 14 s in the 5D case. In higher dimensions, the accuracy of the particle filter dropped off sharply. In future work we hope to explore more sophisticated belief state representations, like Gaussian mixture models, that can maintain accuracy with a manageable number of particles.

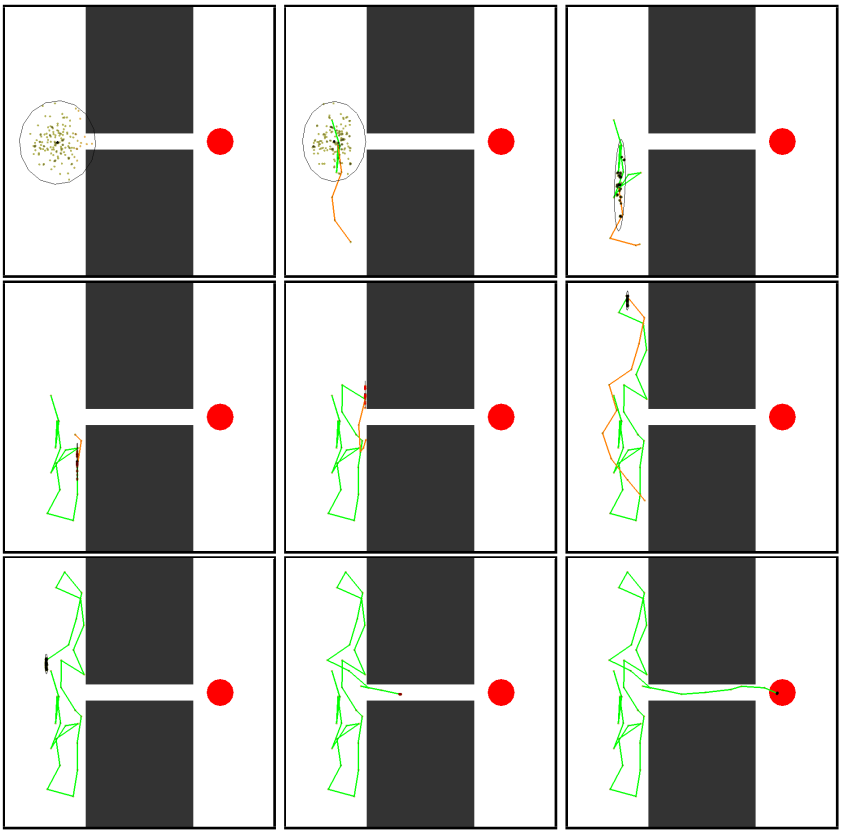


Fig. 6: A robot localizing itself using a proximity sensor in a space with obstacles.

## 6 Discussion: Exploration Strategies

The experiments above are preliminary but promising, and in future work we would like to study RBSR's theoretical performance in the face of approximate belief states and randomization in the exploration strategy. In this section we argue why we expect that RBSR will work well in a broader class of problems; particularly those in which 1) random walks in belief space have a significant probability of finding useful information, and 2) in the process of information-gathering, uncertainty is not significantly increased.

Under these assumptions, RBSR is roughly a belief-space analogue to the Randomized Path Planner (RPP) [2], which addresses path planning in a deterministic, fully-observable environment by alternating potential field descent with random walks to escape local minima. RBSR is, however, better than RPP for two reasons: 1) it perform many walks in simulation only and then picks the best one for execution, and 2) it performs many walks in parallel using the Voronoi bias heuristic, which is more efficient at exploring belief space

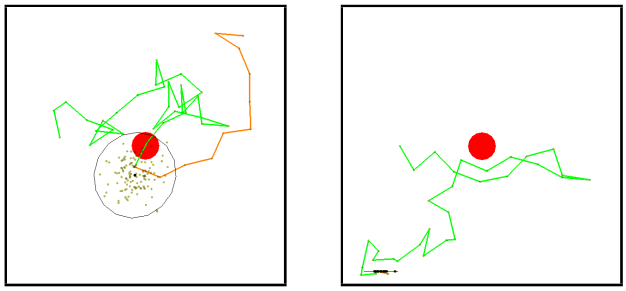


Fig. 7: Left: a partial localization execution using only 5 holdout particles. Because the evaluation function is noisy, the plan is often drastically revised and the walls have not yet been sensed after 30 steps. Right: by initiating replanning only when information gain exceeds a threshold, the path is smoother and two walls have been sensed within 30 steps.

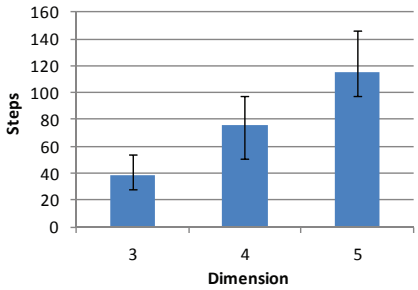



Fig. 8: In localization problems up to 5 dimensions the number of replanning steps scales roughly linearly. Columns report average, minimum, and maximum steps over 10 trials.

than a random walk. So, we should be able to show that RBSR performs at least as well as RPP, which is probabilistically complete.

Another interpretation is that RBSR uses *macro-actions* to make planning more efficient. The idea of macro-actions have existed for some time in the discrete POMDP literature as a way to reduce the exploration breadth and depth in large robotics problems [16]. For example, Hsiao et. al. addressed a robot grasping problem using specially constructed macro-actions that either provide information or seek the goal [9]. They demonstrate that if uncertainty grows slowly during information-gathering, then forward planning can be limited to depth one. RBSR can also be interpreted as depth-one forward planning, using the QMDP policy as a goal-seeking macro-action and belief-space sampling to produce information-gathering macro-actions on the fly. Two other recent works have also tackled the problem of constructing macro-actions automatically and with increasing granularity during forward planning [8, 13]. These approaches are limited to macro-actions that

reach various states as subgoals, and we suspect that RBSR constructs better information-gathering macro-actions using belief space criteria; on the other hand we also suspect that the approaches in [8, 13] construct more optimal plans by searching to a greater depth. (Note that our current presentation of RBSR does not incorporate action costs; future implementations may incorporate path cost during the selection of information-gathering paths.) It remains an open question whether these varied approaches will yield problem-independent principles for generating and exploiting macro-actions in both discrete and continuous POMDPs.

## 7 Conclusion

This paper presented preliminary work in a Randomized Belief-Space Replanning (RBSR) technique for partially-observable problems in continuous state spaces. It constructs partial plans by sampling open-loop actions at random, and by evaluating the quality of future belief states by simulating a QMDP-like policy that performs well when the state is well-localized. By iteratively incorporating sensor feedback from plan execution and replanning, RBSR avoids having to compute a policy over large belief spaces. Experiments show that it solves a target pursuit problem with a 4D state space and a localization problem in 2D–5D state spaces relatively efficiently.

We argued informally that RBSR works well given certain assumptions on the problem space, and future work should attempt to formally characterize convergence rates and compare RBSR to established techniques for discrete POMDPs. Future benchmark development for partially observable continuous problems would aid the empirical study of planner sensitivity to dimensionality and other belief space properties. We also intend to address improving path optimality and using sensing more efficiently in the RBSR framework, because randomization yields somewhat jerky plans. With additional refinements, RBSR-like approaches may lead to breakthroughs in planning under partial observability in realistic robotic systems.

## References

1. R. Alterovitz, T. Simeon, and K. Goldberg. The stochastic motion roadmap: A sampling framework for planning with markov motion uncertainty. In *Robotics: Science and Systems*, June 2007.
2. J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Rob. Res.*, 10(6):628–649, 1991.
3. J. V. D. Berg, P. Abbeel, and K. Goldberg. Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Proc. Robotics: Science and Systems*, 2010.
4. B. Burns and O. Brock. Sampling-based motion planning with sensing uncertainty. In *Proc. IEEE Int. Conf. on Robotics and Automation*, 2007.

5. A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.
6. N. du Toit and J. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *IEEE Int. Conf. on Robotics and Automation*, 2010.
7. L. J. Guibas, D. Hsu, H. Kurniawati, and E. Rehman. Bounded uncertainty roadmaps for path planning. In *Workshop on the Algorithmic Foundations of Robotics*, Guanajuato, Mexico, 2008.
8. R. He, E. Brunskill, and N. Roy. Puma: Planning under uncertainty with macro-actions. In *Proc. Twenty-Fourth Conf. on Artificial Intelligence (AAAI)*, 2010.
9. K. Hsiao, T. Lozano-Perez, and L. P. Kaelbling. Robust belief-based execution of manipulation programs. In *Eighth International Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2008.
10. Y. Huang and K. Gupta. Collision-probability constrained prm for a manipulator with base pose uncertainty. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1426–1432, Piscataway, NJ, USA, 2009. IEEE Press.
11. L. E. Kavraki, P. Svetska, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. and Autom.*, 12(4):566–580, 1996.
12. M. Kearns, Y. Mansour, and A. Y. Ng. Approximate planning in large pomdps via reusable trajectories. In *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.
13. H. Kurniawati, Y. Du, D. Hsu, and W. Lee. Motion planning under uncertainty for robotic tasks with long time horizons. In *Proc. Int. Symp. on Robotics Research*, 2009.
14. H. Kurniawati, D. Hsu, , and W. Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.
15. S. M. LaValle and J. J. Kuffner, Jr. Rapidly-exploring random trees: progress and prospects. In *WAFR*, 2000.
16. M. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In *Proc. 12th Int. Conf. on Machine Learning*, pages 362–370. Morgan Kaufmann, 1995.
17. M. L. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. In *Journal of Artificial Intelligence Research*, volume 9, pages 1–36, 1998.
18. J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for pomdps. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025–1032, Acapulco, Mexico, Aug 2003.
19. R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proc. Robotics: Science and Systems*, 2010.
20. J. M. Porta, N. Vlassis, M. T. J. Spaan, and P. Poupart. Point-based value iteration for continuous pomdps. *J. of Machine Learning Research*, 7:2329–2367, 2006.
21. S. Prentice and N. Roy. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465, 2009.
22. S. Thrun. Monte carlo pomdps. In *Advances in Neural Information Processing Systems 12 (NIPS-1999)*, page 10641070. MIT Press, 2000.