SLAP: Simultaneous Localization and Planning Under Uncertainty via Dynamic Replanning in Belief Space

Ali-akbar Agha-mohammadi, Saurav Agarwal, Sung-Kyun Kim, Suman Chakravorty, and Nancy M. Amato

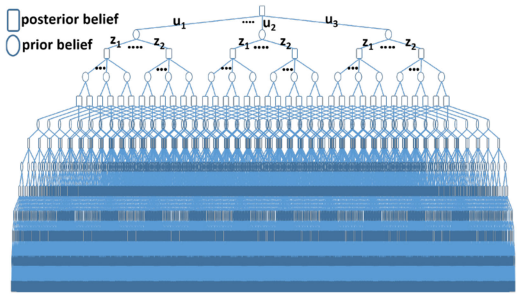
***Abstract*—Simultaneous localization and planning (SLAP) is a crucial ability for an autonomous robot operating under uncertainty. In its most general form, SLAP induces a continuous partially observable Markov decision process (POMDP), which needs to be repeatedly solved online. This paper addresses this problem and proposes a dynamic replanning scheme in belief space. The underlying POMDP, which is continuous in state, action, and observation space, is approximated offline via sampling-based methods, but operates in a replanning loop online to admit local improvements to the coarse offline policy. This construct enables the proposed method to combat changing environments and large localization errors, even when the change alters the homotopy class of the optimal trajectory. It further outperforms the state-of-the-art Feedback-based Information RoadMap (FIRM) method by eliminating unnecessary stabilization steps. Applying belief space planning to physical systems brings with it a plethora of challenges. A key focus of this paper is to implement the proposed planner on a physical robot and show the SLAP solution performance under uncertainty, in changing environments and in the presence of large disturbances, such as a kidnapped robot situation.**

***Index Terms*—Belief space, motion planning, mobile robots, partially observable Markov decision process (POMDP), robust, rollout, uncertainty.**

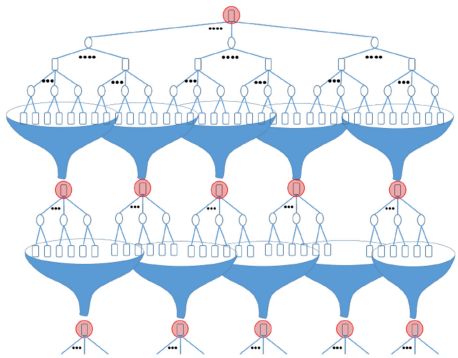
I. Introduction

SIMULTANEOUS localization and planning (SLAP) refers to the problem of (re)planning dynamically every time the localization module updates the probability distribution on the robot's state. For autonomous navigation, solving SLAP and enabling a robot to perform online (re)planning under uncertainty is a key step toward reliable operation in changing real-world environments with uncertainties. For example, consider a low-cost mobile robot operating in an office-like environment with a changing obstacle map (e.g., office doors switch state between open and closed), and responding to changing goals (tasks) assigned online based on user requests. Such changes in the obstacle map or in the goal location often call for *global replanning* to accommodate changes in the homotopy class of the optimal solution. What makes the problem more challenging is that such replanning has to happen online and fast in partially observable environments with motion and sensing uncertainties.

In general, decision making and control under uncertainty are ubiquitous challenges in many robotic applications. For an autonomous robot to operate reliably, it needs to perceive sensory measurements, infer its situation (state) in the environment,



(a) Belief tree: forward construction



(b)Belief graph: backward construction

Fig. 1. (a) This figure depicts a typical search tree in belief space, corresponding to a very small problem with three actions $\mathbb{U} = \{u_1, u_2, u_3\}$ and two observations $\mathbb{Z} = \{z_1, z_2\}$. Each posterior belief (probability distribution) branches into $|\mathbb{U}|$ number of priors and each prior belief branches into $|\mathbb{Z}|$ posteriors, and thus, the tree grows exponentially in the search depth (referred to as the *curse of history*). (b) This figure depicts the idea of using funnels (local feedback controllers) in belief space that can break this exponential growth by funneling a large set of posteriors into a precomputed beliefs (in red circles). Thus, a graph is formed in belief space with funnels as edges and the precomputed beliefs as nodes. This graph grows linearly with the search depth.

Manuscript received September 17, 2017; accepted January 7, 2018. Date of publication; date of current version October 2, 2018. This paper was recommended for publication by Associate Editor V. Kyrki and Editor C. Torras upon evaluation of the reviewers' comments. *(Corresponding author: Ali-akbar Agha-mohammadi).*

A. Agha-mohammadi is with NASA-JPL, California Institute of Technology, Pasadena, CA 91109 USA (e-mail: aliagha@jpl.nasa.gov).

S. Agarwal and S. Chakravorty are with the Department of Aerospace Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: sauravag@tamu.edu; schakrav@tamu.edu).

N. M. Amato is with the Department of CSE, Texas A&M University, College Station, TX 77843 USA (e-mail: amato@tamu.edu).

S.-K. Kim is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: kimsk@cs.cmu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2018.2838556

plan, and take actions accordingly. In partially observable environments, where the state of the system cannot be determined exactly (due to imperfect and noisy measurements), a filtering module (e.g., Kalman filter) can provide an estimate of the state, i.e., a probability distribution function (pdf) over all possible system states. This pdf describing the localization uncertainty is referred to as the *belief* or *information-state*. At each time step, actions are chosen based on the robot's belief. To find the optimal policy that maps beliefs to actions, we cast the problem in its principled form as a partially observable Markov decision process (POMDP) problem [1], [2].

There are a number of challenges in dealing with POMDPs.

1) *Curse of dimensionality* [3] refers to the high dimensions of the belief space. If the underlying robotic system evolves in a discrete grid world with n cells, the corresponding belief space is an n-dimensional continuous space.
2) *Curse of history* [3], [4] refers to the exponential growth of the number of future outcomes in the search depth [see Fig. 1(a)].

Methods such as [3]–[13] alleviate these issues and take POMDPs to more challenging and realistic domains. In this paper, we consider a class of POMDPs that commonly arise in modeling the SLAP problem. The settings are similar to the ones used in the KF-based localization literature [14], [15], such as

1) the system model is given as differentiable nonlinear equations;
2) the state/action/observation spaces are continuous; and
3) belief is unimodal and well approximated by a Gaussian.

In addition to the above-mentioned challenges, when dealing with physical systems, POMDPs need to cope with discrepancies between real models and the models used for computation, e.g., discrepancies due to changes in the environment map or due to intermittent sensor/actuator failures. Dynamic replanning under uncertainty is a plausible solution to compensate for deviations caused by such discrepancies.

To enable an online replanning scheme for SLAP, we rely on multi-query methods in belief space and specifically the Feedback-based Information RoadMap (FIRM) method, discussed below. The main body of POMDP literature (sampling-based methods in particular) propose single-query solvers, i.e., the computed solution depends on the system's initial belief [8], [16], [17]. Therefore, in replanning (from a new initial belief), almost all the computations need to be reproduced, which limits their usage in solving SLAP where dynamic replanning is required. However, multi-query methods, such as FIRM, provide a construction mechanism, independent of the initial belief of the system (Figs. 1(b) and 4), making them suitable methods for SLAP. The original FIRM framework provides a reliable methodology for solving the problem of motion planning under uncertainty by reducing the intractable dynamic programming (DP) to a tractable DP over the nodes of the FIRM graph. In this paper, we extend our previous work on FIRM by proposing a dynamic replanning scheme in belief space that enables online replanning for real world applications in mobile robotics. This extension leads to intelligent robot behaviors that provably takes the solution closer to the optimal solution and guarantees that success probability only increases.

In addition to the proposed algorithms, an emphasis of this paper is on the implementation of the proposed SLAP solution on a physical robot. We investigate the performance of the proposed method and demonstrate its ability to cope with model discrepancies, such as changes in the environment and large deviations which can globally change the plan by changing the homotopy class of the optimal solution. We believe these results lay the ground work for advancing the theoretical POMDP framework toward practical SLAP applications and achieving long-term robotic autonomy.

A. Related Work

Online replanning in belief space is an important capability to solve the SLAP problem for two main reasons. First, belief dynamics are usually more random than the state dynamics because the belief is directly affected by the measurement noise. Therefore, a noisy measurement or spurious data association can cause large changes in the belief. Second, in practical applications, discrepancies between real and computational models or changes in the environment can cause the belief to occasionally show off-nominal behaviors. Online replanning while localizing equips the system with the ability to recover from such situations.

Active localization: Solving the planning problem alongside localization and mapping has been the topic of several recent works (e.g., [18]–[21]). The method in [22] presents an approach to uncertainty-constrained simultaneous planning, localization, and mapping for an unknown environment. In [23], Carlone and Lyons propose an approach to actively explore unknown maps while imposing a hard constraint on the localization uncertainty at the end of the planning horizon. Our work assumes the environment map is known, formulates the problem in its most general form (POMDP), and focuses on online replanning in the presence of obstacles which may possibly change over time.

General-purpose POMDP solvers: There is a strong body of literature on general purpose POMDP solvers (e.g., [24]–[28]), divided into offline and online solvers. Offline solvers [29] compute a policy over the belief space (e.g., [3], [5]–[7]) and online solvers [30] aim to compute the best action for the current belief by creating a forward search tree rooted in the current belief. In recent years, general-purpose online solvers have become faster and more efficient (e.g., AEMS [31], DESPOT [32], ABT [33], and POMCP [13]). However, direct application of these methods to SLAP-like problems is a challenge due to expensive simulation steps, continuous, high-dimensional spaces, and difficult tree pruning steps. We discuss these three challenges in the following paragraphs.

Forward search methods rely on simulating the POMDP model forward in time and creating a tree of possible scenarios in future. At each simulation step $(x', z, c) \sim \mathscr{G}(x, u)$, the simulator $\mathscr{G}$ simulates taking action u at state x and computes the next state x', observation z, and the cost c and constraints of this transition. When dealing with Games (e.g., Go) or traditional POMDP problems (e.g., Rock Sample [30]), the forward simulation step and cost computation are computationally very inexpensive. However, in SLAP-like problems, computing costs are typically orders of magnitude more expensive (e.g., computing collisions between the robot and obstacles).

The second challenge in applying tree-based methods to SLAP is that the tree-based methods require the simulator to revisit the same belief many times to learn its value. However, in SLAP-like problems with continuous state/action/observation spaces, the chances of visiting the same belief are almost zero. Even in the discretized version, the number of simulation steps $(x', z, c) \sim \mathscr{G}(x, u)$ along the tree is of the order $n_{\text{coll}} = O(n_b(|\mathbb{U}||\mathbb{Z}|)^d)$. For the problem in this paper, typical values are in the order of: $n_b > 10$ for the number of parti-

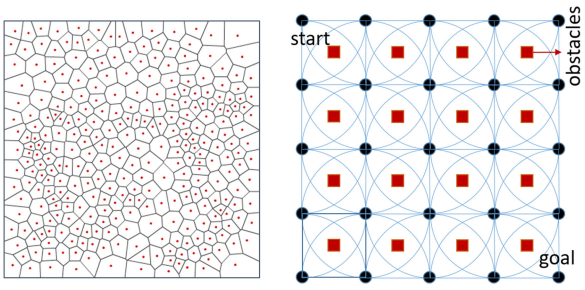


Fig. 2. Example environments with many homotopy classes. Left figure is a Voronoi graph around point obstacles. Right figure is a simple lattice around obstacles. The number of paths from start to goal is in the order of g^d, where g is the branching factor of the graph, and d is the search depth. In the right figure, even if we assume the robot can only go right and down directions in the lattice, the number of paths would be in the order of 4^8.

cles used to represent the belief for accurate collision checking; $d = 10^4$ steps for path length (search tree depth), $|\mathbb{U}| = 50^2$ and $|\mathbb{Z}| = 100^{10}$ for discretization of the 2-D, and 10-D control and observation spaces. Thus, the chances of revisiting the same belief in the discretized version of the problem are quite low.

Finally, unlike many traditional POMDP domains where the domain structure (e.g., game rules) prunes a lot of tree branches, pruning the search tree is much more challenging (if at all possible) in SLAP-like problems, where there is a terminal belief to achieve, no discount factor, and strong dependence between future costs and the past costs.

To handle these challenges, different methods limit the scope of POMDPs to smaller classes, such as POMDPs with differentiable models and Gaussian noise. The following include several examples of such methods.

Local optimization-based methods: In these methods, the optimization variable is typically an open-loop sequence of actions and the optimization converges to the local optimum around the initial solution. The challenge with these methods (e.g., [34], [35]) is that they require an initial trajectory. However, finding a good initial solution could be as difficult as the original problem depending on the environment and observation model. For example, Fig. 2 shows environments with thousands of homotopy classes and local minima. In contrast, the proposed method in this paper does not require an initial solution, and is not sensitive to local minima. Furthermore, typically the planning horizon in the local optimization-based methods is short since the computational complexity of the optimization grows (often superlinearly) in the planning horizon. Local optimization-based methods can be used in a Receding Horizon Control (RHC) scheme as follows: At every step, a sequence of optimal actions is computed within a limited horizon of T steps. Then, only the first action is executed and the rest are discarded. The executed action takes the system to a new point, from which a new sequence of optimal actions is recomputed within horizon T. This process is repeated until the system reaches the goal region. The RHC framework was originally designed for deterministic systems and its extension to stochastic systems and belief space planning is still an open problem. A direct approach is to replace the uncertain quantities (such as future observations) with their nominal values (e.g., most likely observations), and then treat the stochastic system as a deterministic one and use it in an RHC framework (e.g., [34], [36]–[39]). Due to this approximation and the limited optimization horizon, the system may myopically choose good local actions but after a while may find itself in a high-cost region.

Global sampling-based methods: To handle local minima, methods like [16], [17], [40] extend the traditional deterministic motion planning methods (e.g., PRM and RRT) to belief space. The main challenge is that these belief space planners are single query (the solution is valid for a given initial belief). Thus, when replanning from a new belief, most of the computation needs to be reproduced. In particular, when the planner needs to switch the plan from one homotopy class to another, replanning and finding the right homotopy class become challenging (see Fig. 2). The proposed method in this paper can inherently deal with changes in the homotopy class due to its multi-query graph structure.

Application of POMDPs to physical robots: From an experimental point of view, a few recent works have focused on applying belief space planning to real-world robots. Arne Sieverling and Brock [41] implement a belief planner on a mobile manipulator with time of traversal as a cost metric. An integrated task and motion planner, utilizing symbolic abstraction, whose performance is demonstrated on a PR2 robot tasked with picking and placing household objects is proposed in [42]. In [43], Bowen and Alterovitz develop a stochastic motion planner and show its performance on a physical manipulation task, where unknown obstacles are placed in the robot's operating space and the task-relevant objects are disturbed by external agents. Bai *et al.* [11] extend the application of POMDP methods to autonomous driving in a crowd by predicting pedestrian intentions. Marthi [44] applied a POMDP-based planner to navigate a PR2 robot in an office-like environment. This paper proposes an elegant way of incorporating environment changes into the planning framework and can cope with changes in the homotopy class. The main difference with our method is that in [44] Marthi addresses the uncertainty in obstacles and assume that the robot's position in the map is perfectly known. Compared to above methods, the work in this paper extends the application of POMDPs to continuous state/action/observation spaces with very long planning horizons. It further demonstrates the real-time replanning capability in belief space with changing environment while incorporating accurate risk and collision probabilities in the planning framework.

B. Highlights and Contributions

This paper extends our previous work in [45]. Compared to [45], we discuss in detail the concept of rollout-based belief space planning, policy execution, and present extensive simulation and experimental results to demonstrate the performance improvements made by using the proposed method. We also present analyses that guarantee a lower execution cost and failure probability as compared to the nominal FIRM policy. The main highlights and contributions of this paper are as follows.

Online belief planner to enable SLAP: We propose an online planning method in belief space. The method lends itself to the class of rollout-based methods [46] and extends them to the belief space. Compared to belief space RHC methods, this method is not limited to a horizon, does not get stuck in local minima, and does not assume deterministic future observations.

Online switching between homotopy classes: In motion planning, a homotopy class of paths [47] refers to a set of paths that can be deformed into each other by continuous transformation (bending and stretching) without passing through obstacles (see Fig. 2). A unique feature of the presented method is that it is capable of updating the plan globally and online, even when the homotopy class of the optimal solution has changed. This feature

allows the proposed planner to work in changing environments and cope with large deviations.

Selective stabilization policy: The proposed method supercedes a state-of-the-art method, FIRM [4], in performance, success probability, and ability to cope with changing environments. It builds upon a FIRM and inherits the desired features of the FIRM framework, such as robustness, scalability, and the feedback nature of the solution. In addition, it significantly reduces the number of belief node stabilization (required in the original FIRM method) by eliminating the unnecessary ones. Thus, the proposed method can be viewed as FIRM with a selective stabilization policy. In the original FIRM framework, at every node the system needs to steer its sensory information to reach the belief node (each graph node is a belief, i.e., a particular localization uncertainty). In this paper, however, using an online local planning method, we achieve a locally optimal tradeoff between stabilization to a node (i.e., exploring the information space to reach the exact belief node) and moving toward the goal (exploiting the gradient of local cost function). As a result of this optimal exploration–exploitation tradeoff, interesting behaviors emerge out of the algorithm without encoding any heuristic. These behaviors (locally) optimally *trade-off information and energy*. For example, consider a case when the objective is to "reach a goal while minimizing the probability of colliding with obstacles." In that case, in the open areas where there are no narrow passages, the system bypasses the belief node stabilizations. It speeds up and does not waste time gathering information because reducing its uncertainty in obstacle-free regions does not have much benefit. However, once it is faced with obstacles and narrow passages, it automatically decides to perform stabilization until the uncertainty is small enough to safely traverse the narrow passage. Fig. 3, shows this phenomenon pictorially.

Performance guarantees: We provide lower bounds on the performance of the method and show that in stationary environments, the performance and success probability of the proposed method always exceed (or in the worst case are equivalent to) those of the FIRM method.

Applications to physical systems: Among the set of methods that cope with continuous state/action/observation POMDP, only a very small number of methods have been applied to physical systems due to their computational complexity when dealing with real-world robotics problems. An important contribution of this paper is to implement a continuous state/action/observation POMDP solver on a physical robot in a real-world office-like environment.

II. Problem Formulation

In motion planning under uncertainty, we are looking for a policy π that maps the available system information to an optimal action. Let $x_k \in \mathbb{X}$, $u_k \in \mathbb{U}$, and w_k denote the system's state, control action, and motion noise at the kth time step. Let us denote the state evolution model by $x_{k+1} = f(x_k, u_k, w_k)$. In a partially observable system, the system state is not perfectly known. Rather, the state needs to be inferred from noisy measurements. Let us denote the sensor measurement (or observation) vector by $z_k \in \mathbb{Z}$ at the kth time step and the measurement model by $z_k = h(x_k, v_k)$, where v_k denotes sensing noise. All the data that is available for decision making at the kth time step is the history of observations and controls: $\mathscr{H}_k = \{z_{0:k}, u_{0:k-1}\} = \{z_0, z_1, \ldots, z_k, u_0, \ldots, u_{k-1}\}$.

System *belief* or *information-state* $b_k \in \mathbb{B}$ compresses the data history $\mathscr{H}_k$ into a conditional probability distribution over all possible system states $b_k := p(x_k|\mathscr{H}_k)$. In Bayesian filtering, belief can be computed recursively based on the last action and current observation $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ [46], [14]:

$$\begin{aligned} b_{k+1} &= \alpha p(z_{k+1}|x_{k+1}) \int_{\mathbb{X}} p(x_{k+1}|x_k, u_k) b_k dx_k \\ &=: \tau(b_k, u_k, z_{k+1}) \end{aligned}$$

where $\alpha = p(z_{k+1}|\mathscr{H}_k, u_k)^{-1}$ is the normalization constant. Once the belief is formed, a policy π_k generates the next action, i.e., $u_k = \pi_k(b_k)$. The optimal policy π_k is the solution of a POMDP, which is intractable over continuous state/action/observation spaces.

SLAP is the problem of online planning under uncertain robot poses in a known environment with changing obstacles and goal locations. SLAP entails dynamic risk assessment and planning every time the localization module updates the probability distribution on the robot's state, or every time the environment map changes. We refer to the POMDP problem induced by SLAP, as SLAP-POMDP.

SLAP-POMDP: In SLAP-POMDP, the system state is the robot pose (e.g., location). In SLAP-POMDP, the state, action, and observation spaces are continuous, and motion model f and sensor model h are locally differentiable nonlinear functions. The risk is critical throughout the entire plan, and hence, there is no discount factor in SLAP-POMDP, as opposed to traditional POMDP problems. Instead, there exists a termination set $B^{\text{goal}} \subset \mathbb{B}$ such that $c(b, u) = 0$ for all $b \in B^{\text{goal}}$. Risk typically represents the probability of violating constraints $X^{\text{obst}}, U^{\text{const}}$ on state (e.g., collision with obstacles) and actions (e.g., actuator saturation).

SLAP problem: At *each time-step*, re-solve the SLAP-POMDP from a new belief b_0 based on the updated constraint set X^{obst} and updated goal regions B^{goal}

$$\begin{aligned} \pi(\cdot) &= \arg\min_{\Pi} \mathbb{E}\left[\sum_{k=0}^{\infty} c(b_k, \pi_k(b_k)) \,|\, X^{\text{obst}}, B^{\text{goal}}\right] \\ \text{s.t. } b_{k+1} &= \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\ x_{k+1} &= f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)) \\ x_k &\notin X^{\text{obst}}, \quad u_k \notin U^{\text{const}} \\ c(b, \cdot) &= 0 \quad \forall b \in B^{\text{goal}}, \quad c(b, \cdot) > 0 \quad \forall b \notin B^{\text{goal}}. \end{aligned} \tag{1}$$

Following Kalman filter-based robot localization literature (which are widely applied to physical robots), in this paper, we restrict our scope to Gaussian beliefs. We also assume there is an upper bound on the magnitude of environment changes at each step. More precisely, when X^{obst} is updated, there is an upper bound on the number of affected graph edges [see Fig. 1(b)].

III. FIRM Overview

In this section, we briefly describe the abstract framework of feedback-based information RoadMap (FIRM) followed by a description of its concrete implementation in our experiments. We refer the reader to [4], [48] for more in-depth descriptions.

FIRM is a framework designed to reduce a class of intractable continuous POMDPs to tractable problems by generating a representative graph, Probabilistic RoadMap (PRM), in the belief

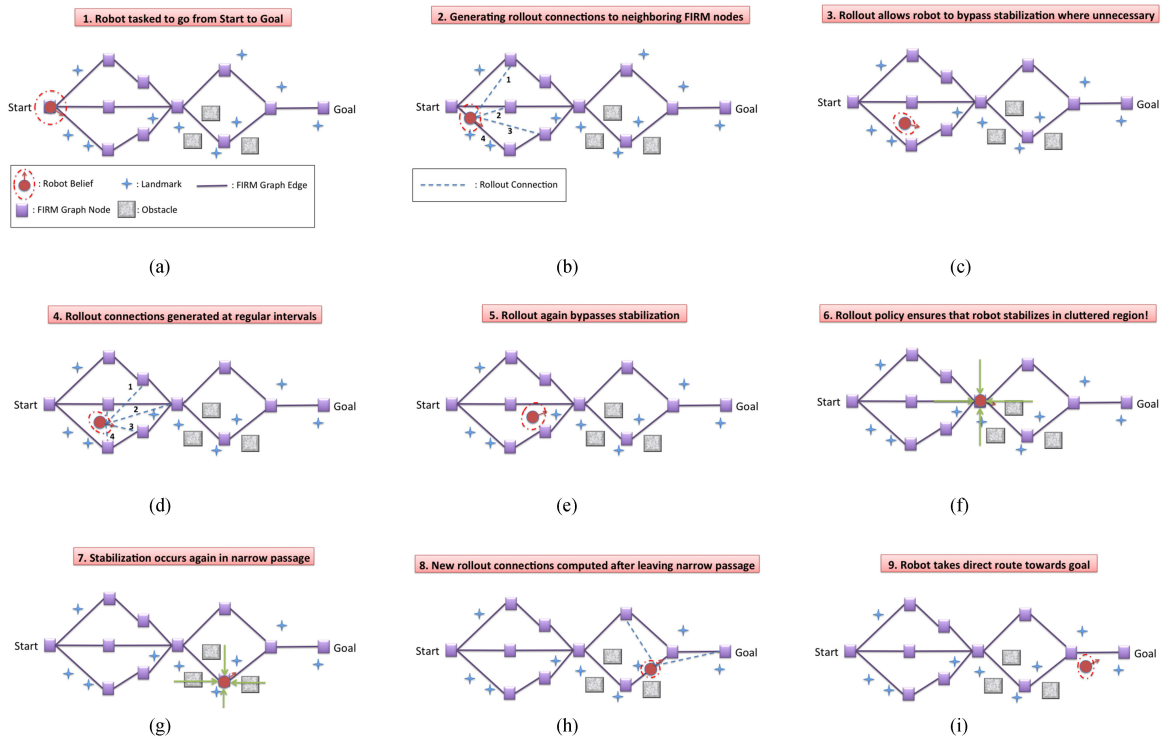


Fig. 3. A representative scenario depicting how rollout-based planner achieves higher performance compared to the standard FIRM algorithm while guaranteeing robustness. The nine scenes depict different stages of task execution as the robot moves from the start to goal location. (a) Simple scenario with a FIRM roadmap, robot, and environment as depicted. (b) Rollout policy is computed periodically. Four candidate edges (dashed lines), including the current edge, are compared. (c) Clutter-free regions, rollout takes a shorter route (edge 3), increasing performance and speed while loosing certainty (i.e., skipping node stabilization). (d) While completing edge 3, the new rollout further cuts down task execution time by taking shorter route through a newly computed rollout edge 2. (e) Robot is approaching the cluttered region. As needed, the planner will slow the robot down to trade performance with certainty. (f) Stabilization is performed because the reduced localization uncertainty (smaller covariance) leads to higher success probability in this case. (g) Stabilization occurs again as robot is still passing through the narrow passage. (h) New rollout connections allow bypassing stabilization. (i) Robot is approaching the goal.

space. Similar to PRMs [49] where the *solution path* is a concatenation of local paths, in FIRM, the *solution policy* is a concatenation of local policies. Every node in a FIRM graph is a small region $B = \{b : \|b - \grave{b}\| \leq \epsilon\}$ around a sampled belief $\grave{b}$. We denote the ith node by B^i and the set of nodes by $\mathbb{V} = \{B^i\}$. Each edge in a FIRM graph is a closed-loop local controller whose goal is to steer the belief into the target node of the edge. An edge controller connecting nodes i and j is denoted by μ^{ij} and the set of edges by $\mathbb{M} = \{\mu^{ij}\}$. An analogy for each local controller is a "funnel in belief space." As depicted in Fig. 4, each funnel steers the set of beliefs to a milestone belief. Further, using the slide-funnel composition, we can create sparse graphs of funnels as shown in Fig. 4. A basic instantiation of the funnel in belief space is a stationary linear quadratic Gaussian (SLQG) controller (see [4, Appendix C]). A basic instantiation of the slide in belief space is a time-varying linear quadratic Gaussian (TV-LQG) controller (see [4, Appendix B]).

Given a graph of these local controllers [Fig. 4(e)], we can define policy π^g on the graph as a mapping from graph nodes to edges; i.e., $\pi^g : \mathbb{V} \rightarrow \mathbb{M}$. Π^g denotes the set of all such graph policies. Having such a graph in belief space, we can form a tractable POMDP on the FIRM graph (so-called FIRM MDP):

$$\pi^{g^*} = \arg\min_{\Pi^g} \mathbb{E} \sum_{n=0}^{\infty} C^g(B_n, \pi^g(B_n)) \tag{2}$$

where B_n is the nth visited node and μ_n is the edge taken at B_n. $C^g(B, \mu) := \sum_{k=0}^{\mathscr{T}} c(b_k, \mu(b_k))$ is the generalized cost of taking local controller μ at node B centered at b_0. We incorporate the failure (collision) set in planning by adding a hypothetical FIRM node B^0 to the list of FIRM nodes. Since the FIRM MDP in (2) is defined over a finite set of nodes, it can be solved by computing the cost-to-go for all graph nodes through the following DP problem:

$$J^g(B^i) = \min_{\mu} \left\{ C^g(B^i, \mu) + \sum_{\gamma=0}^{N} \mathbb{P}^g(B^\gamma | B^i, \mu) J^g(B^\gamma) \right\} \tag{3}$$

and $\pi^g(B^i) = \mu^*$, where μ^* is the argument of above minimization and $\mathbb{P}^g(B^\gamma | B^i, \mu)$ is the probability of reaching B^γ from B^i under μ. The failure and goal cost-to-go (i.e., $J^g(B^0)$ and $J^g(B^{\text{goal}})$) are set to a suitably high positive value and zero, respectively.

Collision (failure) probability of FIRM starting from a given node B^i can be computed [51] as

$$\mathbb{P}(\text{Failure} | B^i, \pi^g) = 1 - \Gamma_i^T (I - Q)^{-1} R_{\text{goal}} \tag{4}$$

where Γ_i is a column vector of zeros with only the ith element set to one, Q is a matrix, whose (i, j)th element is $Q[i, j] = \mathbb{P}(B^i | B^j, \pi^g(B^j))$, and R_{goal} is a column vector, whose jth entry is $R_{\text{goal}}[j] = \mathbb{P}(B^{\text{goal}} | B^j, \pi^g(B^j))$. It can be shown that FIRM is an anytime algorithm [51], i.e., in a given static environment, by increasing the number of nodes, the cost (e.g., the failure probability) will go down. As will be discussed Section III-A, FIRM's failure probability will be an upper bound for the failure probability of the proposed planner.

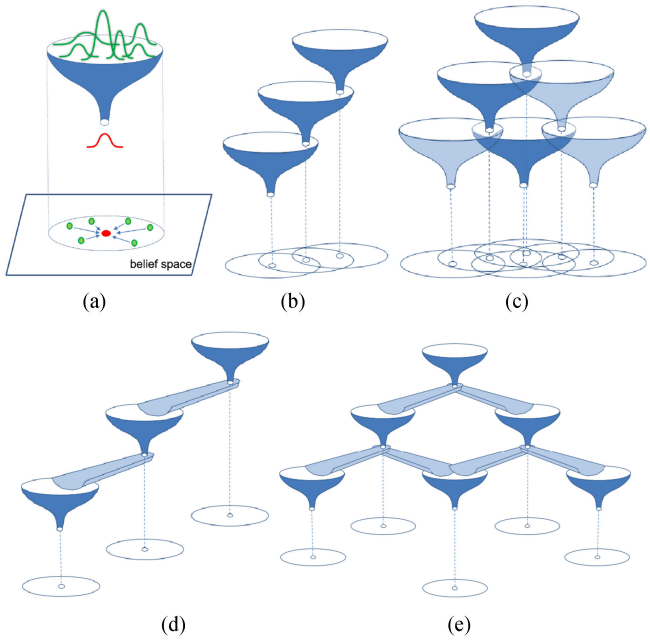


Fig. 4. Extension of sequential composition methods [50] to belief space. (a) Funnel in belief space that collapses a set of Gaussian distribution to a particular Gaussian distribution, referred to as the graph node or milestone. The 2-D projection denotes the belief space, where each point represents a full Gaussian distribution. The projection of the mouth of funnel is analogous to its region of attraction in belief space. (b) Chain of funnels to guide the belief toward a goal. (c) Graph of funnels, where the tip of multiple funnels can fall into the region of attraction of a single funnel. (d) For a sparse set of funnels, one can use tracking controllers (slides) to create the funnel-slide-funnel structure. (e) Graph of funnel-slide-funnel: a FIRM graph. (a) Belief funnel. (b) Funnel chain. (c) Funnel graph. (d) Funnel-slide-funnel. (e) FIRM: graph of funnel-slide-funnel.

A. Concrete FIRM Instance in Our Implementation

Here, we discuss the concrete realization of the FIRM graph constructed for conducting the experiments in this paper.

One-step-cost: Although the objective function can be general, the cost function we use in our experiments includes the localization uncertainty, control effort, and elapsed time

$$c(b_k, u_k) = \zeta_p \mathrm{tr}(P_k) + \zeta_u \|u_k\| + \zeta_T \tag{5}$$

where $\mathrm{tr}(P_k)$ is the trace of estimation covariance as a measure of localization uncertainty. The norm of the control signal $\|u_k\|$ denotes the control effort and ζ_T is present in the cost to penalize each time lapse. Coefficients ζ_p, ζ_u, and ζ_T are user-defined task-dependent positive scalars to combine these costs to achieve a desirable behavior. In the presence of constraints (such as obstacles in the environment), we assume the task fails if the robot violates these constraints (e.g., collides with obstacles). Therefore, collision and goal belief are terminal states such that $J^g(B^0) = J^g(\text{Failure})$ is set to a suitably high cost-to-go and $J^g(B^{\text{goal}}) = 0$. Note that typically the one-step-cost in (5) is defined in the state space (i.e., cost of taking action u at state x). While our cost can be written as a state space cost, writing it directly in belief space better demonstrates the active localization aspect of the work (in the sense of minimizing the uncertainty in the localization belief) along the plan.

Steering localization covariance: To construct a FIRM graph, we first need to sample a set of stabilizers (belief steering functions). Each stabilizer is a closed-loop controller, whose role is to drive the localization uncertainty (or belief) to a FIRM node. A stabilizer consists of a filter and a separated controller [52]. The filter governs the belief evolution and the separated-controller generates control signals based on the available belief at each time step [52]. To design these steering laws, we first sample a set of points $\mathcal{V} = \{\mathbf{v}^i\}$ in the robot's state space. Then, for each point, we construct a stabilizer (i.e., funnel) [4]. In the vicinity of each node $\mathbf{v}^j$, we rely on the Stationary Kalman Filter (SKF) as the steering filter (which is constructed by linearizing the system about the target point $\mathbf{v}^j$). It can be shown that for an observable system, the covariance under the jth SKF approaches to covariance P_s^{+j}, which can be efficiently computed by solving a corresponding discrete algebraic Riccati equation [53].

Steering localization mean: While steering belief toward node B^j, separated-controller μ^{ij} is responsible for generating the control signals based on the available belief, i.e., $u_k = \mu^{ij}(b_k)$. The iRobot Create (used in our experiments) is a nonholonomic robot and is modeled as a unicycle (see Section VI-A2). Thus, to steer the estimation mean toward the target node $\mathbf{v}^j$, one needs to use controllers designed for stabilizing nonholonomic systems (e.g., [54]–[56]). However, the randomness of the estimation mean (resulting from randomness of observations) calls for a controller that can perform such stabilization under uncertainty. To this end, we implemented different controllers including polar coordinate-based controller [57] and dynamic feedback linearization-based controller [58]. Observing the behavior of different controllers, we adopted a variant of the Open-Loop Feedback Control (OLFC) scheme [46] for stabilization purposes. In this variant of OLFC, for a given $\mathbf{v}^j$, we compute an open-loop control sequence from the current estimation mean to $\mathbf{v}^j$. Then, we apply a truncated sequence of the first l controls ($l = 5$ in our experiments).[1] This process repeats every l steps until we reach the vicinity of $\mathbf{v}^j$.

FIRM graph: Associated with each sample $\mathbf{v}^j$, we can define the belief node $\grave{b}^j \equiv (\mathbf{v}^j, P_s^{+j})$. Defining a FIRM node as a ball around this point $B^j = \{b : \|b - \grave{b}^j\| \leq \epsilon\}$, we can steer the Gaussian localization uncertainty to this ball with combination of OLFC and SKF. Accordingly, we sample N FIRM nodes $\{B^j\}_{j=1}^N$.

The SKF/OLFC combination between nodes i and j forms the FIRM edge (local controller) and is denoted by μ^{ij}. We connect each node to its k-nearest neighbors. The set of constructed edges is denoted by $\mathbb{M} = \{\mu^{ij}\}$.

Then, we compute and store costs and transition probabilities associated with each edge using offline simulations. Finally, we solve the DP in (3) to get the optimal graph cost-to-go values $J^g(B^i)$ and policy $\pi^g(B^i)$ for all i.

IV. SLAP via Rollout-Based Dynamic Replanning in Belief Space

As discussed in Section II, SLAP in this paper refers to the problem of (re)planning dynamically every time the localization module updates the probability distribution on the robot's state. In this section, we develop an online replanning method in belief space by extending *rollout policy methods* [46] to the stochastic partially observable setting. In particular, we discuss replanning in the presence of changes in the environment and goal location, large deviations in the robot's location, and discrepancies between real and computational models. We show that the proposed method increases the performance of FIRM by enabling selective stabilization.

[1] Only one control (i.e., $l = 1$) is not enough due to the nonholonomicity of the system.

To make the connection with the rollout policy, we restate the POMDP problem in a more general setting of the time-varying policy

$$\pi_{0:\infty}(\cdot) = \arg\min_{\Pi_{0:\infty}} \sum_{k=0}^{\infty} \mathbb{E}\left[c(b_k, \pi_k(b_k))\right]$$
$$\text{s.t. } b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k)$$
$$x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)). \quad (6)$$

In (6), we seek a sequence of policies $\pi_{0:\infty} = \{\pi_0(\cdot), \pi_1(\cdot), \pi_2(\cdot), \ldots\}$, where π_k maps belief b_k to the optimal action u_k. Π_k is the space of all possible policies at time step k, i.e., $\pi_k \in \Pi_k$. In the infinite horizon case, it can be shown that the solution is a stationary policy π_s, i.e., $\pi_1 = \pi_2 = \cdots = \pi_s$ and the problem is reduced to (1). However, we keep the time-varying format for reasons that will be made clear below.

As discussed earlier, in the SLAP problem, one needs to re-solve this POMDP "online" every time the localization pdf is updated. To handle the computational intractability of the continuous POMDP in (6), we reuse computations in an efficient way, as will be explained in the following section. Here, we first start by discussing the general form of repeated online solutions as an RHC scheme.

RHC in belief space: Receding horizon control (also referred to as rolling horizon or model predictive control) was originally designed for deterministic systems [59] to cope with model discrepancies. For stochastic systems, where the closed-loop (feedback) control law is needed, formulation of the RHC scheme is up for debate [37], [60]–[62]. In the most common form of RHC for stochastic systems [46], the system is approximated with a deterministic one by replacing the uncertain quantities with their typical values (e.g., maximum likelihood value.) In belief space planning, the quantity that injects randomness in belief dynamics is the observation. Thus, one can replace the random observations z_k with their deterministic maximum likelihood value z^{ml}, where $z_k^{ml} := \arg\max_z p(z_k|x_k^d)$ in which x^d is the nominal deterministic value for the state that results from replacing the motion noise w by zero, i.e., $x_{k+1}^d = f(x_k^d, \pi_k(b_k^d), 0)$. The deterministic belief b^d is then used for planning in the receding horizon window. At every time step, the RHC scheme performs a two-stage computation. At the first stage, the RHC scheme for deterministic systems solves an open-loop control problem (i.e., returns a sequence of actions $u_{0:T}$) over a fixed finite horizon T as follows:

$$u_{0:T} = \arg\min_{\mathbb{U}_{0:T}} \sum_{k=0}^{T} c(b_k^d, u_k)$$
$$\text{s.t. } b_{k+1}^d = \tau(b_k^d, u_k, z_{k+1}^{ml})$$
$$z_{k+1}^{ml} = \arg\max_z p(z|x_{k+1}^d) \text{ and } x_{k+1}^d = f(x_k^d, u_k, 0). \quad (7)$$

In the second stage, RHC executes only the first action u_0 and discards the remaining actions in the sequence $u_{0:T}$. However, since the actual observation is noisy and is not equal to the z^{ml}, the belief b_{k+1} will be different from b_{k+1}^d. Subsequently, RHC performs these two stages from the new belief b_{k+1}. In other words, RHC computes an open-loop sequence $u_{0:T}$ from this new belief, and this process continues until the belief reaches the desired belief location. Algorithm 1 recaps this procedure. State-of-the-art methods, such as [63] and [39], utilize RHC in belief space. Toit and Burdick [39] refer to the method as partially closed-loop RHC (PCLRHC) as it exploits partial information about future observations (i.e., z^{ml}) and does not ignore them.

Algorithm 1: RHC with most likely observations for partially-observable stochastic systems.

```
input : Initial belief b_current ∈ 𝕏, B^goal ⊂ 𝔹
while b_current ∉ B^goal do
    u_{0:T} = Solve the optimization in Eq.(7) starting from
      b_0^d = b_current;
    Apply the action u_0 to the system;
    Observe the actual z;
    Compute the belief b_current ← τ(b_current, u_0, z);
```

Algorithm 2: Rollout algorithm in belief space.

```
input : Initial belief b_current ∈ 𝔹, B^goal ⊂ 𝔹
while b_current ∉ B^goal do
    π_{0:T} = Solve optimization in Eq.(8) starting from
      b_0 = b_current;
    Apply the action u_0 = π(b_0) to the system;
    Observe the actual z;
    Compute the belief b_current ← τ(b_current, u_0, z);
```

A known shortcoming of the stated RHC formulation is its limited horizon, which might lead the system to local minima by choosing actions that guide the robot toward "favorable" states (with low cost) in the near future followed by a set of "unfavorable" states (with a high cost) in the long run. To improve the basic RHC, different variants have been proposed including the "rollout policy" [46]. Here, we discuss how they can be extended to and realized in belief space.

Rollout policy in belief space: Another class of methods that aims to reduce the complexity of the stochastic planning problem in (6) is the class of rollout policies [46], which are more powerful than the described version of RHC in the following sense: First, they do not approximate the system with a deterministic one. Second, they avoid local minima using a suboptimal policy that approximates the true cost-to-go beyond the horizon. This policy is referred to as the "base policy" and denoted by $\widetilde{J}$. Formally, at each step of the rollout policy scheme, the following closed-loop optimization is solved:

$$\pi_{0:T}(\cdot) = \arg\min_{\Pi_{0:T}} \mathbb{E}\left[\sum_{k=0}^{T} c(b_k, \pi_k(b_k)) + \widetilde{J}(b_{T+1})\right]$$
$$\text{s.t. } b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k)$$
$$x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)). \quad (8)$$

Then, only the first control law π_0 is used to generate the control signal u_0 and the remaining policies are discarded. Similar to the RHC, after applying the first control, a new sequence of policies is computed from the new point. The rollout algorithm is described in Algorithm 2.

Although the rollout policy in belief space efficiently reduces the computational cost compared to the original POMDP problem, it is still formidable to solve since the optimization is carried out over the policy space. Moreover, there should be a base policy that provides a reasonable cost-to-go $\widetilde{J}$. We now proceed to propose a rollout policy in belief space that exploits the FIRM-based cost-to-go.

A. Enabling SLAP via FIRM-Based Rollout in Belief Space

In this section, we discuss how a rollout policy in belief space (and hence SLAP) can be realized using the FIRM framework. As explained earlier, in FIRM, the system transitions between two nodes[2] B^i and B^j (centered at sampled beliefs b^i and b^j) using a *local* controller μ^{ij}. Global-level decision-making only occurs when the system is in a FIRM node (e.g., region B^i) and for the rest of the time, local controls are executed (e.g., μ^{ij}). In FIRM-based rollout, we raise this limitation by forcing the system to globally replan at every time step to enable SLAP. Specifically, denoting the system belief at time step t by b_t, we rely on the following procedure. At each time step t:

1) We connect b_t to all its neighboring FIRM nodes (in radius R) using suitable local controllers μ^{tj}. These local controllers are designed in the same way as FIRM edges.
2) We evaluate the transition costs $C(b_t, \mu^{tj})$ and the probability of landing in nodes B^γ under the influence of the controller μ^{tj}, i.e., $\mathbb{P}(B^\gamma | b_t, \mu^{tj})$.
3) We evaluate the best edge outgoing from b_t by solving

$$j^* = \arg\min_j \left\{ C(b_t, \mu^{tj}) + \sum_{\gamma=0}^{N} \mathbb{P}(B^\gamma | b_t, \mu^{tj}) J^g(B^\gamma) \right\} \tag{9}$$

where $J^g(B^\gamma)$ is the nominal cost-to-go under the FIRM policy from node B^γ and $J^g(B^0)$ is the failure cost-to-go as discussed in Section III.

4) We choose μ^{tj^*} as the local controller at b_t if the expected success probability exceeds the current one. In other words, we only switch from the current local controller (i.e., μ^{ij}) to μ^{tj^*} if the following condition holds:

$$\mathbb{E}[\text{success} | b_t, \mu^{tj^*}] > \mathbb{E}[\text{success} | b_t, \mu^{tj}] \tag{10}$$

where expected success probability is

$$\mathbb{E}[\text{success} | b_t, \mu^{t\alpha}] = \sum_{\gamma=1}^{N} \mathbb{P}(B^\gamma | b_t, \mu^{t\alpha}) P^{\text{success}}(B^\gamma) \tag{11}$$

and $P^{\text{success}}(B^\gamma) = \Gamma_\gamma^T (I - Q)^{-1} R_{\text{goal}}$ is the probability of success for reaching the goal from FIRM node B^γ under the nominal FIRM policy [see (4)].

Algorithm 3 describes planning with the proposed rollout process. We split the computation to offline and online phases. In the offline phase, we carry out the expensive computation of graph edge costs and transition probabilities. Then, we handle pose deviations and the changes in the start/goal location by repeated online replanning while reusing offline computations.

In the following, we discuss how Algorithm 3 provides a tractable variant of (8). Following the concepts and terminology in [46], here, the FIRM policy plays the role of the base policy; FIRM's cost-to-go values are used to approximate the cost-to-go beyond the rollout horizon. Given a dense FIRM graph, where nodes partition the belief space (i.e., $\cup_i B^i = \mathbb{B}$), the belief at the end of rollout [b_{T+1} in (8)] will fall into a FIRM node with a known cost-to-go. With a sparse FIRM graph, where nodes do not cover the entire belief space, we design local policies that drive the belief into a FIRM node at the end of horizon. However, since the belief evolution is random, reaching a FIRM node at deterministic time horizon T may not be guaranteed. Therefore, we propose a new variant of rollout method with a random horizon $\mathscr{T}$ as follows:

$$\begin{aligned} \pi_{0:\infty}(\cdot) = \arg\min_{\widetilde{\Pi}} \mathbb{E}\left[\sum_{k=0}^{\mathscr{T}} c(b_k, \pi_k(b_k)) + \widetilde{J}(b_{\mathscr{T}+1}) \right] \\ \text{s.t.} \quad b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k | x_k) \\ x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k | x_k, \pi_k(b_k)) \\ b_{\mathscr{T}+1} \in \cup_j B^j \end{aligned} \tag{12}$$

where for $b_{\mathscr{T}+1} \in B^j$, we have

$$\widetilde{J}(b_{\mathscr{T}+1}) = J^g(B^j). \tag{13}$$

$\widetilde{\Pi}$ is a restricted set of policies under which the belief will reach a FIRM node in finite time. In other words, if $\pi \in \widetilde{\Pi}$ and $\pi = \{\pi_1, \pi_2, \ldots\}$, then $\mathbb{P}(b_{\mathscr{T}+1} \in \cup_j B^j | \pi) = 1$ for finite $\mathscr{T}$. Thus, the last constraint in (12) is redundant and automatically satisfied for suitably constructed $\widetilde{\Pi}$. Also, the FIRM-based cost-to-go $J^g(\cdot)$ plays the role of the cost-to-go beyond the horizon [see Eq. (13)].

Note that based on Algorithm 3, we can provide guarantees on the performance of the proposed method. Before formally stating the results, recall that at each instance of rollout computation, the current belief b_t is added as a virtual node B_t^{virtual} to the FIRM graph to generate the augmented FIRM graph G_t^a. A virtual node being defined as a temporary node with no incoming edges. Virtual nodes are removed from the graph as soon as the system departs their vicinity.

Algorithm 3: Rollout algorithm with FIRM as base policy.

input : Initial belief b_t and goal belief region B^{goal}
Construct a FIRM graph and store nodes $\mathbb{V} = \{B^i\}$, edges $\mathbb{M} = \{\mu^{ij}\}$, Cost-to-go $J^g(\cdot)$, and Success probabilities $P^{success}(\cdot)$; `// offine phase`
while $b_t \notin B^{goal}$ `// online phase`
do
  Find neighboring nodes $\mathbb{V}_R = \{B^i\}_{i=1}^r$ to b_t;
  Set $B_t = \{b_t\}$, $J(B_t) = \infty$, and $S = 0$;
  forall $B^j \in \mathbb{V}_R$ **do**
    μ^{tj} = Generate controller from b_t to B^j ;
    $C(b_t, \mu^{tj}), \mathbb{P}(B^\gamma | b_t, \mu^{tj})$ = Simulate μ^{tj} to compute transition probability and expected cost;
    Compute the expected success $\mathbb{E}[success | b_t, \mu^{tj}]$;
    if $\mathbb{E}[success | b_t, \mu^{tj}] \geq S$ **then**
      Compute the candidate cost-to-go as $J^{cand} = C(b_t, \mu^{tj}) + \sum_{\gamma=0}^{N} \mathbb{P}(B^\gamma | b_t, \mu^{tj}) J^g(B^\gamma)$;
      if $J^{cand} < J(B_t)$ **then**
        $J(B_t) = J^{cand}$ and $S = \mathbb{E}[success | b_t, \mu^{tj}]$;
        $\mu^{tj^*} = \mu^{tj}$;
  Apply the action $u_t = \mu^{tj^*}(b_t)$ to the system;
  Get the actual measurement z_{t+1};
  Compute the next belief $b_t \leftarrow \tau(b_t, u_t, z_{t+1})$;
  if *user submits a new goal state* $\mathbf{v}^{goal}$ **then**
    $B^{goal} \leftarrow$ Sample the corresponding FIRM node;
    Add B^{goal} to the FIRM graph; $\mathbb{V} \leftarrow \mathbb{V} \cup \{B^{goal}\}$;
    Connect B^{goal} to its r nearest neighbors using edges $\{\mu^{(i,goal)}\}$. Also, $\mathbb{M} \leftarrow \mathbb{M} \cup \{\mu^{(i,goal)}\}$;
    $[J^g(\cdot), P^{success}(\cdot)]$ = DynamicProgramming($\mathbb{V}, \mathbb{M}$);

[2] In the cartoon in Fig. 4, it looks like B^j is the sole destination for μ^{ij}. However, in dense graphs the belief under μ^{ij} might be absorbed by a different funnel before reaching B^j. The summation over γ in the following equations takes this consideration into account.

Proposition 1: For a given static map, the performance and success probability of the FIRM-Rollout policy is lower bounded by the nominal FIRM policy at any belief state during execution of the planner.

Proof: See [64] for the proof.

Remark: If the desired factor was merely the success probability, one can ignore the cost-to-go comparison condition in Algorithm 3 and only maximize the success probability.

In addition to improving the performance while not compromising on the safety, the rollout procedure is particularly helpful in handling the changes in the environment map. We discuss this aspect in Section IV-C.

B. Complexity Analysis

In this section, we analyze the computational complexity of the offline and online phase of the proposed algorithm.

Offline phase: We assume the environment is a hypercube $[0, w]^d$. For constructing the offline policy on a k-regular graph with N nodes, we have to simulate kN edges offline. Let us denote the number of particles describing belief by n_b^{off}. Assuming a fixed velocity 1 m/s on edges, and assuming simulation steps occur at every Δt seconds, the number of simulation calls (including collision checks) is $n_{\text{coll}} = \sum_{s=1}^{kN} n_b^{\text{off}} \Delta t^{-1} l_s$, where l_s is the length of the sth edge.

Assuming a uniform distribution of the sampled points (in the sense of infinity norm) in the configuration space, the density of points is $\rho = Nw^{-d}$. Accordingly, the dispersion [65], [66] of the sampled points is $\delta = wN^{-d^{-1}}$. Assuming all edges have equal length (in the l^∞-norm sense), the edge length of the underlying PRM (over which FIRM has been built) is $l_s = \delta = w\sqrt[d]{N}^{-1}$

$$n_{\text{coll}} = (n_b^{\text{off}} \Delta t^{-1}) wk N^{1-d^{-1}}. \tag{14}$$

Online phase: In the online phase, we connect each node to all nodes in the neighborhood of radius R (in infinity norm). Thus, the size of neighboring area for connection is R^d, which encompasses $R^d * \rho$ neighboring points. For $R = r\delta$, it will encompass r^d points. Thus, we have r^d new edges in the online phase. It can be shown that the length of $(i+1)^d - i^d$ of these edges is in the range $i\delta < edgeLength < (i+1)\delta$.

For all edge lengths between $i\delta < l_s = edgeLength < (i+1)\delta$, let us approximate l_s by $i^+\delta$, where $i \le i^+ \le i+1$. Then, the sum of the length of all new edges is

$$L_s = \sum_{s=1}^{r^d} l_s = \sum_{i=1}^{r} \sum_{s=(i-1)^d+1}^{i^d} l_s = \delta \sum_{i=1}^{r} ((i)^d - (i-1)^d - 1) i^+.$$

Let us denote the number of particles describing the belief by n_b. The number of simulation calls (including collision checks) is

$$\begin{aligned} n_{\text{coll}} &= n_b \Delta t^{-1} L_s \\ &= n_b \Delta t^{-1} \sqrt[d]{N^{-1}} w \sum_{i=1}^{R\sqrt[d]{N} w^{-1}} ((i)^d - (i-1)^d - 1) i^+. \end{aligned}$$

Upper/lower bounds on the number of collision checks can be obtained by setting i^+ to its upper and lower bounds, i.e., $i+1$ and i. To gain further insight on the complexity, let us assume i^+ is a constant (i.e., all edge lengths are the same) and set it to its maximum value $i^+ = R\sqrt[d]{N} w^{-1}$. Then, the upper bound on collision checks n_{coll}^+ is

$$\begin{aligned} n_{\text{coll}}^+ &= (n_b \Delta t^{-1} w N^{-d^{-1}})(R\sqrt[d]{N} w^{-1}) \\ &\quad \times [(R\sqrt[d]{N} w^{-1})^d - R\sqrt[d]{N} w^{-1}] \\ &= n_b \Delta t^{-1} w^{-d} R^{d+1} N - n_b \Delta t^{-1} w^{-1} R^2 \sqrt[d]{N}. \end{aligned} \tag{15}$$

Given this upper-bound on the computation time and given uniform grid sampling strategy, the online computation time grows sublinearly with the number of underlying FIRM nodes N in the worst case. Also, for a given dimension, the online computation time is polynomial in the connection radius R. By removing the dimension from the equation and extending the results to random sampling, we can write the first term of the above equation as

$$n_{\text{coll}}^+ = (n_b \Delta t^{-1}) RV\rho$$

where ρ is the density of samples in the environment, V is the volume of the connection neighborhood, and R is the radius of the connection neighborhood.

C. Enabling SLAP in Changing Environments

In this section, we discuss the ability of the proposed planner to handle the changes in the obstacle map. We focus on a challenging case, where changes in the obstacle map are persistent and can possibly eliminate a homotopy class of solutions. Doors are an important example of this class. If the robot observes a door is closed (which was expected to be open), it might have to globally change the plan to get to the goal from a different passage. This poses a challenge to the state-of-the-art methods in the belief space planning literature.

To handle such changes in the obstacle map and replan accordingly, we propose a method for lazy evaluation of the generated feedback tree, referred to as "lazy feedback evaluation" algorithm, inspired by the lazy evaluation methods for PRM frameworks [67]. The basic idea is that at every node the robot reevaluates *only* the next edge (or the next few edges up to a fixed horizon) that the robot will most likely take. By re-evaluation, we mean it needs to recompute the collision probabilities along these edges. If there is a significant change (measured by α in Algorithm 4) in the collision probabilities, the DP problem is resolved and new cost-to-go values are computed. Otherwise, the cost-to-go values remain unchanged and the robot keeps following its rollout-policy. Such lazy evaluation (computing the collision probabilities for a single edge or a small number of edges) can be performed online. The method is detailed in Algorithm 4.

Remark: Another challenge with these persistent changes is that they stay in the memory. Imagine a case where the robot is in a room with two doors and the goal point is outside the room. Suppose that after checking both doors, the robot realizes they are closed. To solve such cases, the door state is reset to "open" after a specific amount of time to persuade the robot to recheck the state of doors. We further discuss the concept of the "forgetting time" in the experiments section.

It is important to note that it is the particular structure of the proposed planner that makes such replanning feasible online. The graph structure of the underlying FIRM allows us to *locally* change the collision probabilities in the environment without affecting the collision probability of the rest of the graph [i.e., properties of different edges on the graph are independent of each other; see Figs. 4 and 1(b)]. Such a property is not present in the state-of-the-art sampling-based belief space plan-

Algorithm 4: Lazy Feedback Evaluation (Lazy Replanning).

input : FIRM graph
output : Updated cost-to-go, $J^g(\cdot)$ and success probabilities $P^{success}(\cdot)$
Perceive the obstacles map;
if *there is a change in map* **then**
  $\mathscr{F} \leftarrow$ Retrieve the sequence of nominal edges returned by feedback up to horizon l; Set $ctr = 0$;
  forall *edges* $\mu \in \mathscr{F}$ **do**
    Re-compute collision probabilities $\mathbb{P}_{new}(B, \mu)$ from starting node B of edge μ;
    if $|\mathbb{P}_{new}(B, \mu) - \mathbb{P}(B, \mu)| > \alpha$ **then**
      $\mathbb{P}(B, \mu) \leftarrow \mathbb{P}_{new}(B, \mu)$;
      $ctr \leftarrow ctr + 1$;
  if $ctr > 0$ **then**
    Update edge set $\mathbb{M}$ based on new transition probabilities;
    $[J^g(\cdot), P^{success}(\cdot)]$ = DynamicProgramming($\mathbb{V}, \mathbb{M}$);
return $J^g(\cdot)$ and $P^{success}(\cdot)$;

ners (e.g., [16], [17]), where the collision probabilities and costs on *all* edges are dependent on each other and hence need to be recomputed.

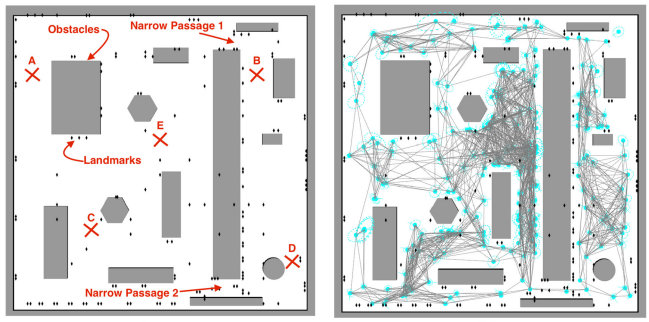
V. Simulation Results

In this section, we demonstrate the performance of the method in simulation. The robot is tasked to go from a start location to multiple goal locations sequentially in an obstacle-laden environment with narrow passages and asymmetrically placed landmarks.

Motion model: The state of the robot at time k is denoted by $x_k = (\mathrm{x}_k, \mathrm{y}_k, \theta_k)^T$ (2-D position and the heading angle). We denote the control as $u_k = (v_{x,k}, v_{y,k}, \omega_k)^T$ and the process noise by $w_k = (n_{v_x}, n_{v_y}, n_\omega)^T \sim \mathcal{N}(0, \mathbf{Q}_k)$. Let f denote the kinematics of the robot such that $x_{k+1} = f(x_k, u_k, w_k) = x_k + u_k \delta t + w_k \sqrt{\delta t}$.

Observation model: We use a range-bearing landmark-based observation model. Each landmark has a unique fully observable ID. Let ${}^i\mathbf{L}$ be the location of the ith landmark. The displacement vector ${}^i\mathbf{d}$ from the robot to ${}^i\mathbf{L}$ is given by ${}^i\mathbf{d} = [{}^i d_x, {}^i d_y]^T := {}^i\mathbf{L} - \mathbf{p}$, where $\mathbf{p} = [\mathrm{x}, \mathrm{y}]^T$ is the position of the robot. Therefore, the observation ${}^i z$ of the ith landmark can be described as ${}^i z = {}^i h(x, {}^i v) = [\|{}^i\mathbf{d}\|, \mathrm{atan2}({}^i d_y, {}^i d_x) - \theta]^T + {}^i v$. The observation noise is assumed to be zero-mean Gaussian such that ${}^i v \sim \mathcal{N}(\mathbf{0}, {}^i\mathbf{R})$, where ${}^i\mathbf{R} = \mathrm{diag}((\eta_r \|{}^i\mathbf{d}\| + \sigma_b^r)^2, (\eta_\theta \|{}^i\mathbf{d}\| + \sigma_b^\theta)^2)$. The measurement quality decreases as the robot gets farther from the landmarks and the parameters η_r and η_θ determine this dependency. σ_b^r and σ_b^θ are the bias standard deviations.

Environment and scenario: The environment in Fig. 5(a) represents a 2-D office space measuring 21 m × 21 m. The robot is a disk with diameter 1 m. There are two narrow passages $P1$ and $P2$ with high collision probability. The narrow passages are 1.25 m wide; thus, offering a very small clearance for the robot to pass through. The robot is placed at starting location *A* and tasked to visit four different locations (*B, C, D*, and *E*) in four sequential segments: 1) $A \rightarrow B$; 2) $B \rightarrow C$; 3) $C \rightarrow D$; and 4) $D \rightarrow E$. We compare the performance of the standard FIRM with the proposed rollout-based method.



(a) (b)

Fig. 5. (a) Simulation environment: landmarks (black diamonds) and obstacles (gray polygons). The locations of interest that the robot is tasked to visit are marked by red crosses. The two narrow passages P1 and P2 represent the regions of high collision probability (risky) due to the small clearance. (b) Underlying FIRM roadmap: edges (grey lines), nodes (cyan disks), covariance of the FIRM nodes (dashed ellipses).

A. Planning With Standard FIRM

First, we construct the FIRM roadmap offline [see Fig. 5(b)].

FIRM nodes: The roadmap is constructed by uniformly sampling configurations in the free space. Then, corresponding to each configuration node, we create a FIRM node (belief) by following the procedure in Section III-A. In short, we linearize the system dynamics and sensor model around the sampled configuration point. We create a Kalman Filter corresponding to this local linear system and find its reachable belief by solving the corresponding Riccati equation. At each node, there exists a stabilizing controller which locally drives all beliefs to the belief node.

FIRM edges: The edges of the FIRM roadmap are generated by first finding valid (collision free) straight line connections between neighboring nodes and then generating edge controllers which drive the belief from the starting belief of the edge to the vicinity of the target belief of the edge. For each edge in the graph, we run Monte Carlo simulations to compute the expected execution cost and transition probability. The constructed FIRM roadmap is stored for use in the online rollout phase.

Online phase: In the online phase, the planner receives a query (i.e., starting and goal configuration). These configurations are added to the existing roadmap by computing the appropriate stationary belief, stabilizer, and edge controllers. Since this construction preserves the optimal substructure property (i.e., edges are independent of each other; see Fig. 4 and 1(b)), we can solve DP on the graph for the given goal location to construct the feedback tree.

FIRM feedback tree: The solution of the DP problem (i.e., π^g), is visualized with a *feedback tree*. Recall that π^g is a mapping (look-up table) that returns the next best edge for any given graph node. The feedback tree is rooted at the goal node. For each node, the feedback tree contains only one outgoing edge ($\mu = \pi^g(B^i)$) that pulls the robot toward the goal.

Most-likely path (MLP): The most likely path is defined as a path followed by the FIRM feedback if there was no noise (i.e., it is a tree branch that connects start to goal.) Note that the actual solution (generated by FIRM) can be arbitrarily different from the MLP due to noise.

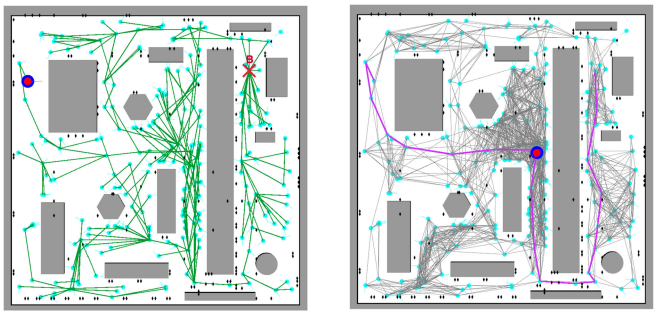


Fig. 6. Segment 1 of policy execution with FIRM, starting at *A* and going to *B*. The locations *A* and *B* are marked in Fig. 5(a). (a) Feedback tree (green) for goal location B, robot (blue disk), and initial belief (red). (b) Most likely path (purple) under FIRM from *A* to *B*.

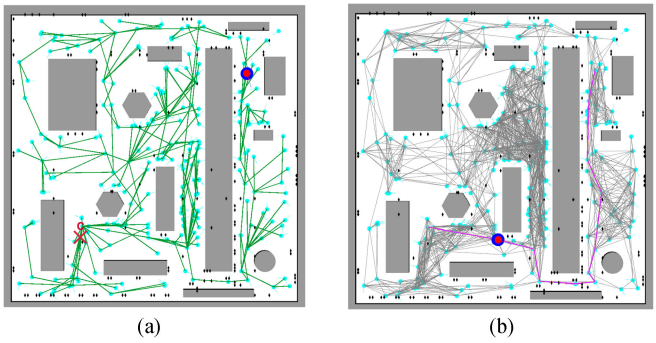


Fig. 7. Segment 2 ($B \rightarrow C$) of policy execution with FIRM. (a) FIRM feedback for goal location *C*. (b) MLP (purple) under FIRM from *B* to *C*.

In segment 1 ($A \rightarrow B$), the planner computes the feedback tree [see Fig. 6(a)] rooted in B. Fig. 6(b) shows the MLP. In this case, the noise is not high enough to change the homotopy class, and the robot is close to the MLP. To reach the goal, the robot follows edge controllers returned by the feedback policy and stabilizes to all FIRM nodes along the path. Once the robot reaches B, we submit a new goal C. A new feedback tree is computed online rooted in C [see Fig. 7(a)], with MLP shown in Fig. 7(b). We follow a similar procedure to accomplish segments $C \rightarrow D$ and $D \rightarrow E$.

B. Planning With the Proposed Method

For segment 1 ($A \rightarrow B$), as before, we begin with the underlying FIRM roadmap constructed offline and compute the feedback tree [see Fig. 6(a)]. However, when rollout planner follows the feedback tree, a different behavior emerges. At each time step (or more generally every T_{rollout} seconds), the planner connects the current robot belief to neighboring FIRM nodes in radius R (i.e., the planner locally generates edge controllers with their associated cost and transition probability). Then, the planner checks which connection provides the lowest sum of the edge-cost and cost-to-go from its landing node as in Eq. (9). The connection with the lowest sum is chosen as the next edge to follow. Fig. 8(a) shows the planner checking connections (red edges) locally to neighboring FIRM nodes.

An important behavior emerges in segment 1. As the robot proceeds, the rollout is able to find a shorter path through the relatively open area by skipping unnecessary stabilizations [see Figs. 8(b) and (c)]. As the robot traverses the narrow passage $P2$, the rollout realizes "stabilizing" to the FIRM node is the best option as it concludes it is better to reduce the uncertainty to a safe level before proceeding through the narrow passage [see Fig. 8(d)]. Eventually, the robot reaches location B through the path as marked in green in Fig. 8(f). Rollout gives the robot a distinct advantage over the nominal FIRM plan as it guides the robot through a shorter and faster route. Furthermore, it should be noted that although the last part of the two paths (after exiting the narrow passage) look similar, they differ significantly in the velocity profiles. Along the purple path, the robot stabilizes to each and every FIRM node. But, along the green path (rollout), the robot maintains a higher average velocity by skipping unnecessary stabilizations. The robot performs full or partial stabilization only when the gained information (reduced uncertainty and risk) is necessary to complete the mission.

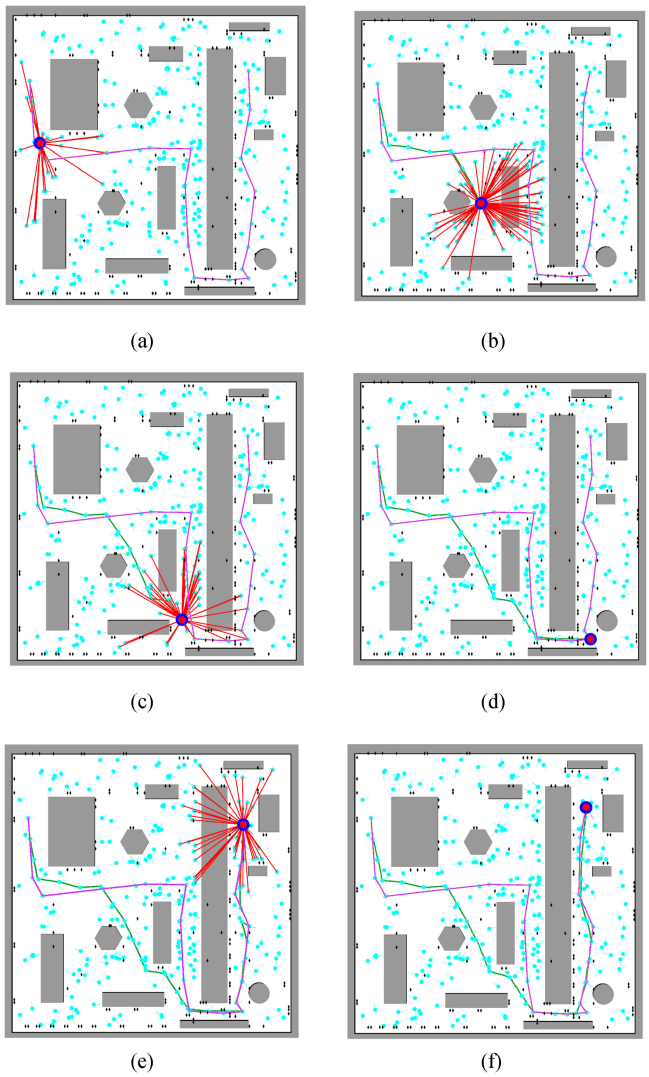


Fig. 8. Segment 1 with rollout: Starting at *A* and going to *B*. (a) Robot checks new connections with neighbors (red). MLP under the nominal FIRM feedback is in purple. (b) Rollout guides the robot away from the MLP to a shorter and faster path (green). The new path is in a different homotopy class. (c) Robot approaches narrow passage *P2* through a more direct path as compared to the MLP. (d) Robot stabilizes at a few FIRM nodes while passing through the narrow passage. (e) Robot approaches goal location *B*. (f) Rollout path is shorter and faster than the FIRM's MLP.

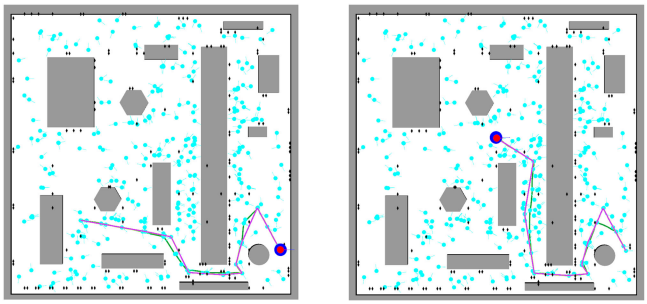


Fig. 9. Asymmetric costs and random execution noises. (a) Segment 3 ($C \rightarrow D$). (b) Segment 4 ($D \rightarrow E$).

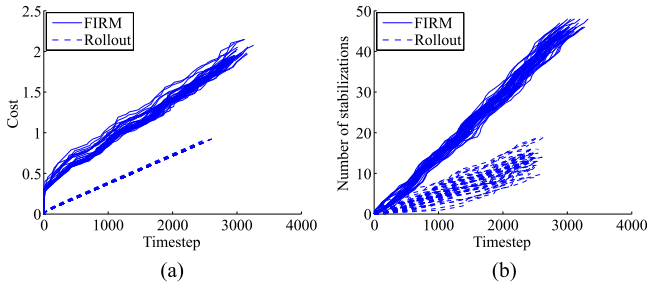


Fig. 10. Performance comparison between the original FIRM and proposed planner on 50 runs. (a) Execution cost for FIRM rises faster than the cost of the rollout-based policy. (b) Number of belief nodes that the robot stabilizes to, during plan execution, is consistently lower for the rollout.

Similar behaviors are observed when completing segments 2 ($B \rightarrow C$), 3 ($C \rightarrow D$), and 4 ($D \rightarrow E$). Fig. 9 shows final paths for segments 3 and 4. We observe different levels of stabilization and different path shapes when passing through passage $P2$ in segments 3 and 4. This is due to asymmetric distribution of information and risk in the environment.

C. Analysis of Simulation Results

In this section, we discuss the statistical analysis for the presented simulation results by running the planner multiple times. The results show that the proposed method significantly increases the performance of the standard FIRM implementation while preserving its robustness and scalability.

Cost of execution: We recorded the amount of localization uncertainty (trace of covariance) along the robot's path. Fig. 10(a) shows the cumulative version of this cost on 50 runs for the same task under the rollout-based planner and standard FIRM. We note that the cost for the rollout based policy rises slower than the cost for FIRM, and as the planning horizon increases, rollout offers increasing returns in performance.

Selective stabilization: Node stabilization makes FIRM robust and scalable while maintaining the optimal substructure property on the graph (via edge independence; see Fig. 4). Although the stabilization allows the FIRM to provide certain guarantees, it adds stabilization time and cost at each node to the time and cost of the mission. The rollout-based planner brings a higher level of intelligence to the process of node stabilization. Rollout performs stabilization when required and bypasses it when possible. Bypassing the stabilization allows the robot to complete the task faster and with less cost. Fig. 10(b) shows the number of nodes the robot has stabilized to on 50 different runs. In this example, the robot stabilizes to $\sim$45 nodes under FIRM compared to $\sim$10 nodes under the rollout-based planner ($\sim$75% reduction), while the difference is growing as the task becomes longer.

Time of task completion: Another quantitative performance measure is the time it takes for a planner to complete the task while guaranteeing a high likelihood of success. From Fig. 10(a) and (b), the time taken to complete the task with rollout is around 2500 time-steps (250 s) compared to 3000 time-steps (300 s) for FIRM. There is $\sim$15% reduction in the time to complete the task compared to the standard FIRM algorithm. The improvement in execution time makes the rollout-based planner a better candidate than FIRM for time-sensitive applications.

Varying node density: To further analyze the results, we study the performance of the method as a function of offline graph density. Fig. 11 shows how the cost, number of stabilizations, and time to complete the task change as the density of underlying graph increases.

D. Comparison With State-of-the-Art

We compare our proposed method with iterative local optimization-based (ILO-based) methods. Extended LQR [68] for deterministic systems, iterative linear quadratic Gaussian control [69], and its belief space variant [70] (referred to as BSP-iLQG, here) are among the pioneering ILO-based methods. Due to their optimization-based nature, ILO methods perform well when the problem has a single local optimum (i.e., the global optimum). When there are multiple local minima, the performance of ILO methods is sensitive to the initial solution.

In belief space variants of ILO methods, the problem is typically solved in two phases. First, ignoring all uncertainties, a deterministic motion planning problem is solved (e.g., using RRT in [70]) to find an initial trajectory from start to goal. Second, the generated trajectory is utilized as the initial solution for a local optimization process in belief space. In our simulations, we use a holonomic 2-D robot and point beacon observation model similar to the one used in [70, Sec. 6.2.2]. whose signal strength decreases quadratically with robot's distance from the beacon. We compare ILO-based approaches to the proposed method in two aspects: 1) The sensitivity of the quality of the solution to the initial guess and 2) replanning time.

In [64], we show an environment, where there exists a single optimum (i.e., the local optimum is identical to the global optimum). In this environment, ILO-based methods perform well and can converge to the optimal solution. However, environments with more obstacles and multiple homotopy classes and/or environments with more complex information distribution induce multiple local minima, which degrades the performance of the ILO-based methods. Fig. 12 shows one such environment. The initial RRT as shown in Fig. 12(a) computes a path that takes the robot toward the goal diagonally. This limits the resulting local optimum solution to a homotopy class, quite different from the global optimum. On the other hand, the proposed rollout-based method does not require any initial solution, and will be able to find the optimal homotopy spanned by the underlying graph. This key difference is shown in Fig. 12(b), where the generated solution (green) optimally exploits the information distribution (beacons are in the upper left corner) in the environment. In addition, the rollout-based method can update and repair the homotopy class during the execution to compensate for potential deviations due to the noise.

In addition to the solution quality, the replanning time in ILO-based methods grows with the problem horizon. For example,

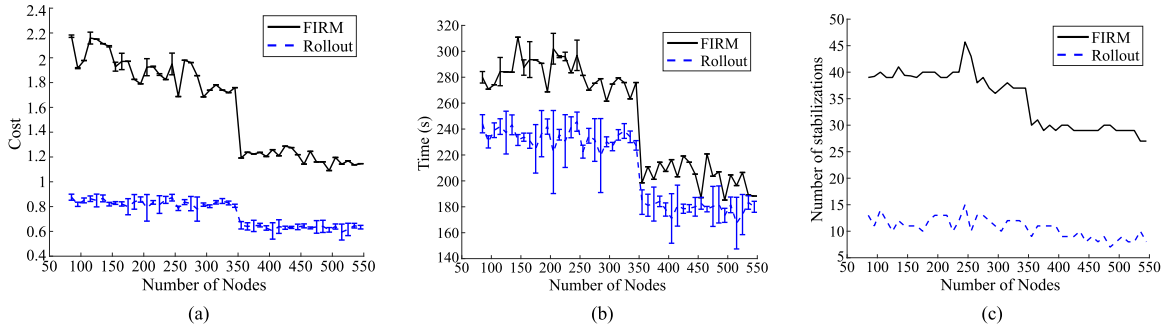


Fig. 11. Effect of increasing FIRM graph density on the rollout solution behavior. Each graph node is connected to all nodes within its radius $R = 5$. As the number of nodes in the graph crosses 350, a new connection through narrow passage 2 is found, which leads to sharp changes in the graphs. (a) Cost to complete the task reduces as the number of underlying graph nodes increases (due to availability of more options). Sharp dips in the graph correspond to cases where adding a new node captures a new low-cost homotopy class. (b) Time taken by the robot to complete the mission. As graph density increases, robot finds more nodes to connect to during the rollout phase and thus, can take more shortcuts, which reduces its total driving time. (c) Number of visited nodes (the number of stabilizations) in rollout is significantly smaller than FIRM.

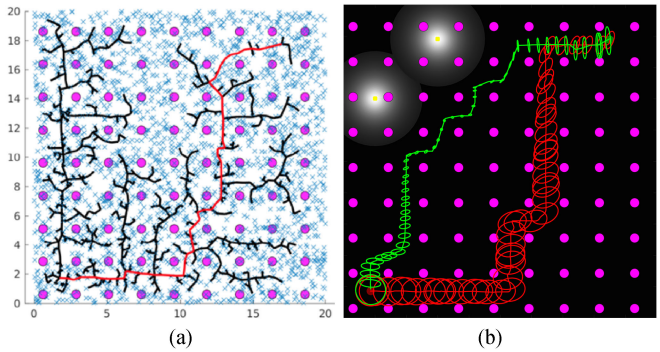


Fig. 12. Comparison of rollout versus local optimization-based methods. Obstacles (magenta) and information beacons (yellow light sources) are shown, whose signal strength declines quadratically with the robot-beacon distance. (a) Local optimization-based methods require an initial solution. Deterministic RRT (black) generates an initial solution (red). (b) Final solution computed by belief space ILO (red) is restricted to the homotopy of the RRT solution. On the other hand, the rollout-based policy guides the robot (green) toward the global optimum by exploiting the underlying global feedback structure. (a) Initial guess (in red) for ILO-based methods using deterministic RRT planner (black tree). (b) Locally optimized solution (red) and the path under the rollout policy (green).

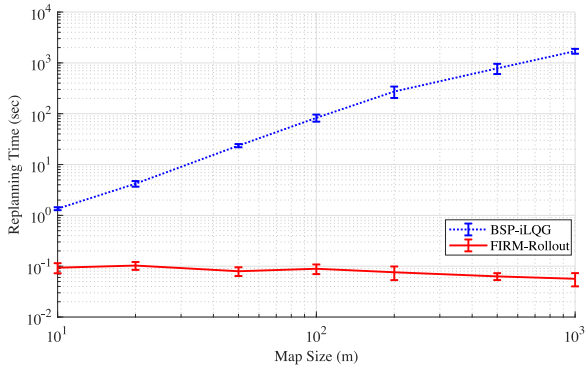


Fig. 13. Planning time: ILO in belief space (blue) versus the proposed method (red). As the planning horizon (distance between start and goal) increases, local optimization-based methods take more time to plan and converge, whereas the planning time in rollout methods is not a function of planning horizon.

in BSP-iLQG, the complexity of the optimization algorithm is in the order of $\mathcal{O}(Init) + \mathcal{O}(N_l N_i)$, where $\mathcal{O}(Init)$ refers to the complexity of computing an initial guess (e.g., solving a deterministic motion planning algorithm such as RRT), and $\mathcal{O}(N_l N_i)$ refers to the complexity of belief optimization. N_l is the trajectory length and N_i is the number of iterations for the optimization to converge. Thus, the computational complexity grows unboundedly with the planning horizon (path length).

Here, we experimentally compare the computational complexity (replanning time) of the proposed method with ILO-based methods. Although computing the initial solution $\mathcal{O}(Init)$ in ILO-based methods can take significant amount of time, here, to simplify the results, we only report the time ILO spends on belief optimization, i.e., $\mathcal{O}(N_l N_i)$, and do not include the initial solution generation time in the graphs for ILO-based methods.

We compare the results on the forest environment (see Fig. 12). Comparison is carried out in C++ on a PC with 3.40 GHz Quad-Core Intel i7-3770 CPU and 16 GB RAM running Ubuntu 14.04. We grow the environment and planning horizon at each step. As the forest grows, we maintain the same obstacle density as well as the information source density. The starting point is at the bottom-left and the goal is at the top-right for all environment sizes. For each environment size, we run the planners five times to record the statistical variations. Fig. 13 shows how the planning time of ILO-based methods increases as the problem size grows. On the other hand, the complexity of the proposed rollout-based method is constant regardless of the planning horizon. These results confirm the analyses in Section IV-B. In this forest environment, the average replanning time is 80.1 ms with a variance of 35.1 ms for rollout connections (edge length) of 2.25 m length (on average). Such rapid replanning capability allows for dynamic replanning in belief space to enable SLAP.

VI. Experimental Results for a Physical System

In this section, we demonstrate the proposed online POMDP-based solution for the SLAP on a physical robot. We discuss the details of implementation and report the important lessons learned. A video demonstrating the experiments is available in [71].

A. Target Application and System Setup

Consider a scenario, where the robot needs to operate and reach a goal in an office environment. Each time the robot reaches a goal, a new goal is submitted by a higher level applica-

Fig. 14. Floor-plan of the environment, in which experiments are conducted.

tion (e.g., manually by a user or multiple users). We investigate the performance and robustness of the method to: changing obstacles, such as doors, and moving people, changes in the goal location, deviations due to intermittent sensory failures, and kidnap situations (significant sudden deviation in the robot's location). In all these cases, an online replanning scheme can help the robot to recover from the off-nominal situation and accomplish its goal. In particular, we study the "kidnapped" situation, where a person might move the robot to an unknown location during the plan execution, and the robot needs to recover from this catastrophic deviation. The main focus of the experiments in this section is to demonstrate SLAP on physical robots by enabling online belief space (re)planning.

1) Environment: Our experiments are conducted on the fourth floor of the Harvey Bum Bright building at the Texas A&M University campus. The floor plan is shown in Fig. 14. The floor spans almost 40 m of hallways whose average width is approximately 2 m, which is distinguished in yellow and blue in Fig. 14. The particular set of experiments reported in this paper is conducted in the region which is highlighted in blue in Fig. 14, part of which contains a large cluttered office area (407-area). This area has interesting properties that makes the planning more challenging:

a) 407-area is obstacle-laden (chairs/desks and workstations).
b) As is seen in Fig. 14, there are several doors in this area which may be open or closed. Two of these doors (front-door and back-door) are labeled in Fig. 14.
c) There are objects such as chairs and trash-cans in this environment which usually get displaced.
d) There are moving people, who are avoided using a reactive behavior, which may displace the robot from its planned path.

2) Robot model: The physical platform utilized in our experiments is an iRobot Create mobile robot (see Fig. 15). The robot can be modeled as a unicycle with the following kinematics:

$$x_{k+1} = f(x_k, u_k, w_k) = \begin{pmatrix} \mathrm{x}_k + (V_k \delta t + n_v \sqrt{\delta t}) \cos\theta_k \\ \mathrm{y}_k + (V_k \delta t + n_v \sqrt{\delta t}) \sin\theta_k \\ \theta_k + \omega_k \delta t + n_\omega \sqrt{\delta t} \end{pmatrix} \tag{16}$$

Fig. 15. Picture of the robot (an iRobot Create) operating in the office environment. Landmarks can be seen on the walls.

where $x_k = (\mathrm{x}_k, \mathrm{y}_k, \theta_k)^T$ describes the robot state at the k-th time step. $(\mathrm{x}_k, \mathrm{y}_k)^T$ is the 2-D position and θ_k is the heading angle of the robot. Control commands are the linear and angular velocities $u_k = (V_k, \omega_k)^T$. We use the Player robot interface [72] to send the control commands to the robot.

Motion noise: The motion noise vector is denoted by $w_k = (n_v, n_\omega)^T \sim \mathcal{N}(0, \mathbf{Q}_k)$, which is mostly resulted from uneven tiles on the floor, wheel slippage, and inaccuracy in the duration of the applied control signals. Experimentally, we observed that in addition to the fixed uncertainty associated with the control commands, there exists a portion of the noise that is proportional to the signal strength. Thus, we model the variance of the process noise at the kth time-step as $\mathbf{Q}_k = \mathrm{diag}((\eta V_k + \sigma_b^V)^2, (\eta\omega_k + \sigma_b^\omega)^2)$, where in our implementations, we have $\eta = 0.03$, $\sigma_b^V = 0.01$ m/s, and $\sigma_b^\omega = 0.001$ rad.

3) Sensing model: For sensing purposes, we use the onboard laptop webcam. We perform a vision-based landmark detection based on ArUco (a minimal library for Augmented Reality applications) [73]. Each landmark is a black and white pattern printed on a letter-size paper. The pattern on each landmark follows a slight modification of the Hamming code, and has a unique id, so that it can be detected robustly and uniquely. Landmarks are placed on the walls in the environment (see Fig. 15). The absolute position and orientation of each landmark in the environment are known. The ArUco library performs the detection process and presents the relative range and bearing to each visible landmark along with its id. Denoting the jth landmark position in the global 2-D coordinate frame as ${}^jL$, we can model the observation as a range-bearing sensing system:

$${}^jz_k = [{}^j\bar{d}_k, \mathrm{atan2}({}^jd_{2_k}, {}^jd_{1_k}) - \theta]^T + {}^jv,\ {}^jv \sim \mathcal{N}(\mathbf{0}, {}^j\mathbf{R})$$

where ${}^j\bar{d}_k = \|{}^j\mathbf{d}_k\|$ and ${}^j\mathbf{d}_k = [{}^jd_{1_k}, {}^jd_{2_k}]^T := [\mathrm{x}_k, \mathrm{y}_k]^T - L_j$.

Measurement noise: A random vector ${}^jv$ models the measurement noise associated with the measurement of the jth landmark. Experimentally, we observed that the intensity of the measurement noise increases by the distance from the landmark and by the incident angle. The incident angle refers to the angle between the line connecting the camera to landmark and the wall, on which the landmark is mounted. Denoting the incident angle by ${}^j\phi \in [-\pi/2, \pi/2]$ for the jth landmark, we model the sensing noise associated with the jth landmark as a zero mean Gaussian, whose covariance is

$${}^j\mathbf{R}_k = \mathrm{diag}((\eta_{r_d}{}^j\bar{d}_k + \eta_{r_\phi}|{}^j\phi_k| + \sigma_b^r)^2, (\eta_{\theta_d}{}^j\bar{d}_k + \eta_{\theta_\phi}|{}^j\phi_k| + \sigma_b^\theta)^2)$$

where in our implementations, we have $\eta_{r_d} = 0.1$, $\eta_{r_\phi} = 0.01$, $\sigma_b^r = 0.05$ m, $\eta_{\theta_d} = 0.001$, $\eta_{\theta_\phi} = 0.01$, and $\sigma_b^\theta = 2.0°$.

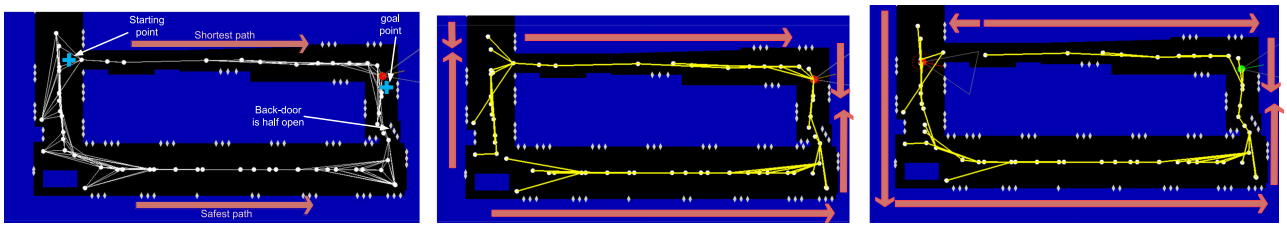


Fig. 16. (a) Environment: obstacles (blue), free space (black), and landmarks (white diamonds). An MAPRM graph (white) approximates the connectivity of the free space. (b) Feedback tree (yellow), generated by solving DP on MAPRM. From each node, there is only one outgoing edge, guiding the robot toward the goal. Arrows in pink coarsely represent the direction on which the feedback guides the robot. (c) Feedback tree (yellow), generated by solving DP on FIRM. Computed feedback tree guides the robot through the more informative regions, leading to more accurate localization and less collision probabilities. Arrows in pink coarsely represent the direction on which the feedback guides the robot.

Full vector of measurements: At each step, the robot observes the set of landmarks that fall into its field of view. Given that the robot can see r landmarks $\{L_{i_1}, \ldots, L_{i_r}\}$, the total measurement vector is $z = [{}^{i_1}z^T, \ldots, {}^{i_r}z^T]^T$. Due to the independence of measurements of different landmarks, the full observation model can be written as $z = h(x) + v$, where $v = [{}^{i_1}v^T, \ldots, {}^{i_r}v^T]^T \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and $\mathbf{R} = \text{diag}({}^{i_1}\mathbf{R}, \ldots, {}^{i_r}\mathbf{R})$.

B. SLAP Versus Decoupled Localization and Planning

In this section, we contrast the results of a traditional decoupled localization and planning with the proposed SLAP solution. Decoupled localization and planning here refers to a method, where the planner first generates a plan (ignoring the localizer) and then, in the execution phase, the localizer estimates the state to aid the controller to follow the planned trajectory. However, in the proposed SLAP solution, the planner takes the localizer into account in the planning phase and replans simultaneously as the localizer updates its estimation.

The test environment is shown in Fig. 16(a). Blue regions are obstacles and black regions are free space. Landmarks are shown by small white diamonds. The start and goal locations for the motion planning problem are marked in Fig. 16(a). The goal location is inside 407-area (see Fig. 14) and the starting location is close to the front door.

Decoupled planning and localization: To select a suitable planner, we tried a variety of traditional planners, such as PRM, RRT, and their variants. We observed that following the plan generated by most of these methods leads to collisions with obstacles and cannot reach the goal point due to the high motion noise (of our low-cost robot) and due to the sparsity of the information in certain parts of the test environment. The best results are achieved using the Medial-Axis PRM (MAPRM) method [74]. This planner is computationally more expensive than the other variants, but is more powerful in dealing with collisions by sampling points on the medial axis of the free space and constructing a PRM that has the most clearance from obstacles. An MAPRM graph (in white) for this environment is shown in Fig. 16(a).

In this environment, there are two main homotopy classes of paths between the start and goal nodes: Through the front door of 407-area and through the back door of the 407-area. From Fig. 16(a), it is obvious that the path through the front door is shorter. Moreover, the path through the front door has a larger obstacle clearance (larger minimum distance from obstacles along the path) compared to the path through the back door (since the back door is half-open). Therefore, based on conventional metrics in deterministic settings, such as shortest path or maximum clearance, MAPRM chooses the path through the front door over the path through the back door. The feedback tree that results from solving DP in this case is shown in Fig. 16(b). As expected, feedback guides the robot toward the goal through the front door. To execute MAPRM's plan, we use time-varying LQG controllers to keep the robot close to the generated path. However, due to the lack of enough information along the nominal path, the success rate of this plan is low, and the robot frequently collides with obstacles. The success probability along the nominal path is computed by multiple (100) runs and is equal to 27% (27 runs out of 100 runs were collision free).

FIRM-based SLAP: As can be seen in Fig. 16(a), the distribution of information is not uniform in the environment. The density of landmarks (information sources) along the path through the back door is higher than that of the path through the front door. FIRM-based SLAP can incorporate this information in the planning phase in a principled way. This leads to a better judgment of how narrow the passages are. For example, in this experiment, although the path through the front door is shorter than the path through the back door, considering the information sources, the success probability of going through the back door is much greater than going through the front door. Such knowledge about the environment is reflected in the FIRM cost-to-go and success probability. As a result, it generates a policy that suits the application, taking into account the uncertainty and available information in the environment. Solving DP on the FIRM graph generates the feedback tree shown in Fig. 16(c), which results in 88% success probability.

C. Online Replanning Aspect of SLAP

In this section, we focus on the "simultaneous" part in SLAP, which emphasizes the ability of the robot to replan after every localization update. In other words, in SLAP, the robot dynamically replans based on the new information coming from its sensors.

We study two important cases to illustrate the effect of online replanning. We first look into a challenging case, where the obstacle map changes and possibly eliminates a homotopy class of solutions. This means the planner has to switch to a different homotopy class in real time, which is a challenging situation for the state-of-the-art methods in the belief space planning literature. Second, we look into deviations from the path, where we focus on the kidnapped robot problem as the most severe and general case of belief deviation. Finally, we demonstrate

the performance of the proposed method on a complex scenario that includes changes in the obstacles, deviations in the robot pose, online changes in goal location, etc.

1) Changes in the Obstacle map

Here, we show how enabling simultaneous planning and localization, online, can handle the changes in the obstacle map.

In this paper, we assume no prior knowledge about the environment dynamics. As a result, we have a simple model for obstacle dynamics: All new obstacles will be added to the map with a large forgetting time of 10 min (i.e., almost-permanent). The only exception in this model is moving people: if a moving person is detected, a new obstacle will not be added to the map. Instead, we assume there exists a lower level reactive behavior (e.g., stopping or dodging) in a subsumption-like architecture [75] that suppresses the belief space planner in the vicinity of the moving person. Once the control is back to the SLAP layer, the robot might have deviated from its nominal plan, and thus, the SLAP layer has to replan to recover from such deviations.

Therefore, the method is very efficient in dealing with persistent/slow changes in the map (e.g., closed/open doors). An important aspect of the method is that it can deal with severe changes that might eliminate or create homotopy classes of solutions. Doors are an important example of this class. If the robot observes a closed door (which was expected to be open), it might have to *globally* change the plan to get to the goal from a different passage. This is a very challenging problem for today's belief space planners.

As the first experiment, we consider the environment shown in Fig. 14. The start and goal locations are shown in Fig. 17(a). We construct a PRM in the environment ignoring the changing obstacles (assuming all doors are open and there are no people in the environment). Then, we construct the corresponding FIRM and solve DP on it. As a result, we get the feedback tree shown in Fig. 17(a) that guides the robot toward the goal through the back door of 407-area. However, the challenge is that the door may be closed when the robot reaches it, and there may be people moving in the environment. Moreover, for various reasons (such as motion blur in the image or blocked landmarks by people), the robot might misdetect landmarks temporarily during the run.[3] To handle such a change in the obstacle map and replan accordingly, we use the "lazy feedback evaluation" method outlined in Algorithm 4.

Results on physical robots: Fig. 17(b) shows a snapshot of the system during the operation when the robot detects the change signal, i.e., detects that the door is in a different state than its recorded situation in the map. As a result, the robot updates the obstacle map as can be seen in Fig. 17(b) (door is closed). Accordingly, the robot replans; Fig. 17(b) shows the feedback tree resulting from the replanning. The new feedback guides the robot through the front door since it detects the back door is closed. The full video of this run provides much more details and is available in [71].

It is important to note that it is the particular structure of the proposed SLAP framework that makes such online replanning feasible. The graph structure of the underlying FIRM allows us to *locally* change the collision probabilities in the environment without affecting the collision probability of the rest of the graph (i.e., properties of different edges on the graph are independent of each other; see Fig. 4). This independence property is not present in the state-of-the-art belief planners, such as BRM (Belief Roadmap Method) [16] or LQG-MP [17]. In those methods, collision probabilities and costs on *all* edges need to be recomputed if a change in the obstacle map is detected. The general purpose planners, such as ABT [33], are also not applicable to this setting due to the size of the problem and the need to recompute collision probabilities. In ABT, if the robot detects a change in the obstacle map in the vicinity of the robot, it needs to alter the uncertainty evolution in an ABT tree branch near the tree root (i.e., near the robot pose). This, in turn, will require the whole subtree (including collision probabilities) under the affected branch to be updated, which is not a real-time operation for long-horizon planning.

[3]Designing perception mechanisms for obstacle detection is not a concern of this research; thus, we circumvent the need for this module by sticking small markers with specific IDs on moving objects (doors or people's shoes).

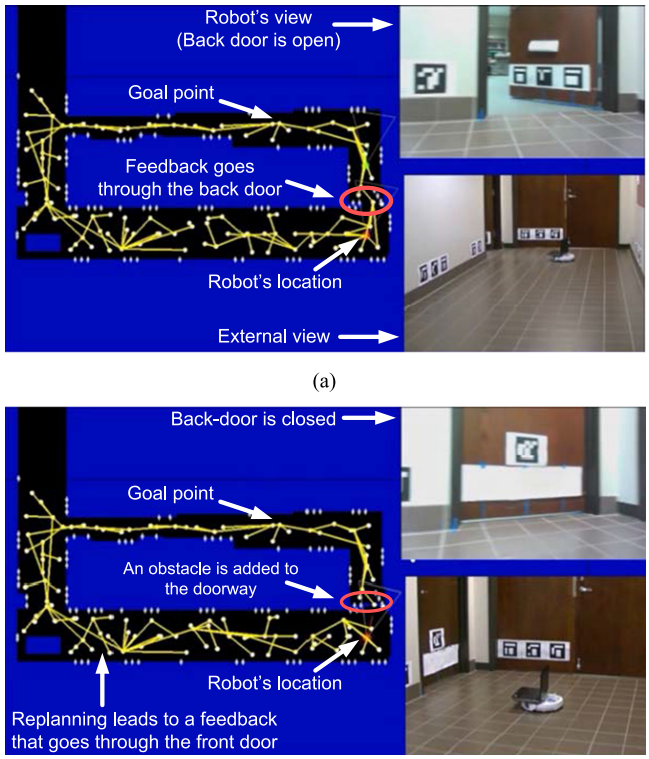


Fig. 17. (a) Back door is open at this snapshot. The feedback guides the robot toward the goal through the back door. (b) Back door is closed at this snapshot. Robot detects the door is closed and updates the obstacle map (adds the door to the map). Accordingly robot replans and computes the new feedback. The new feedback guides the robot toward the goal through the front door.

2) Deviations in the Robot's Pose

In this section, we demonstrate how online replanning enables SLAP in the presence of large deviations in the robot's position. As the most severe form of this problem, we consider the *kidnapped robot problem*. In the following, we discuss this problem and challenges it introduces.

Kidnapped robot problem: An autonomous robot is said to be in the kidnapped situation if it is carried to an unknown location while it is in operation. The problem of recovering from this situation is referred to as the kidnapped robot problem [76]. This problem is commonly used to test the robot's ability to recover from catastrophic localization failures. This problem introduces different challenges, such as how to detect kidnapping, how to relocalize the robot, and how to control the robot to accomplish its goal. Our main focus, here, is on the third part, i.e., how

to replan in belief space from the new belief resulted from the kidnapped situation. This is in particular challenging because large deviations in the robot's pose can globally change the plan and the homotopy class of the optimal solution. Therefore, the planner should be able to change the global plan online.

Detecting the kidnapped situation: To embed the kidnapped situation into the framework in a principled way, we add a Boolean observation z^{lost} to the observation space. Let us denote the innovation signal as $\widetilde{z}_k = z_k - z_k^-$ (the difference between the actual observations and predicted observation). Recall that in our implementation, the observation at time step k from the jth landmark is the relative range and bearing of the robot to the jth landmark, i.e., ${}^j z_k = ({}^j r_k, {}^j\theta_k)$. The predicted version of this measurement is denoted by ${}^j z_k^- = ({}^j r_k^-, {}^j\theta_k^-)$. We monitor the following measures of the innovation signal:

$$\widetilde{r}_k = \max_j \left(|{}^j r_k - {}^j r_k^-|\right), \quad \widetilde{\theta}_k = \max_j \left(d^\theta({}^j\theta_k, {}^j\theta_k^-)\right) \tag{17}$$

where $d^\theta(\theta, \theta')$ returns the absolute value of the smallest angle that maps θ onto θ'. Passing these signals through a low-pass filter, we filter out the outliers (temporary failures in the sensory reading). Denoting the filtered signals by $\overline{r}_k$ and $\overline{\theta}_k$, if both conditions $\overline{r}_k < r_{\max}$ and $\overline{\theta}_k < \theta_{\max}$ are satisfied, then $z^{\text{lost}} = 0$, otherwise $z^{\text{lost}} = 1$. When $z^{\text{lost}} = 0$, we follow the current rollout planner. $z^{\text{lost}} = 1$ means that the robot is constantly observing high innovations, and thus, it is not in the location in which it believes to be (i.e., it is kidnapped). Once it is detected that the robot is kidnapped, we replace the estimation covariance with a large covariance (to get an approximately uniform distribution over the state space).

Replanning from the kidnapped situation: The rollout-FIRM algorithm can inherently handle such replanning. In other words, the kidnapped situation, i.e., a deviated mean and very large covariance, will just be treated as a new initial belief and a new query. Accordingly, the FIRM rollout creates the best macro-action (i.e, graph edge or funnel) on the fly and execute it. Note that the belief deviation might change the optimal homotopy class and the plan should be updated globally, which makes it challenging for many POMDP planners. Using the proposed rollout planner, the robot just needs to go to a neighboring node from this deviated point. Since the underlying FIRM graph is spread in the belief space, the only required computation is to evaluate the cost of edges that connect the new starting point to the neighboring FIRM nodes.

To get safer plans when replanning from $z^{\text{lost}} = 1$ situation, we update the rollout planning mechanism slightly: in addition to the new initial belief, we add one more belief node to the graph, as described below. Consider a new kidnapped initial belief $b_0 \equiv (\widehat{x}_0^+, P_0)$. Let δ denote the distance between the mean of this new belief $\widehat{x}_0^+$ and the closest mean on the graph nodes. If $z^{\text{lost}} = 1$ and δ is not small, the mean belief is far from actual robot position and moving the robot δ meters based on a wrong belief might lead to collision. To ensure that the proposed rollout-based planner can take this case into account, we add a FIRM node b' to the graph at (or very close to) the configuration point $v = \widehat{x}_0^+$.

In such a case starting from a deviated belief b_0 with large covariance, the planner will take the robot to b' first, which is a belief with the same mean but smaller covariance (i.e., turning in-place or taking very small velocities). Planner will make this choice since moving to a farther node when the covariance is very large will lead to high collision probability; this risk is reflected in the transition probabilities of the rollout edges.

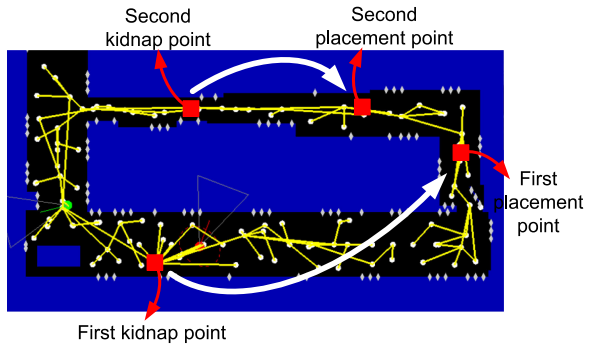


Fig. 18. This figure shows the setup for the experiment containing two kidnapping.

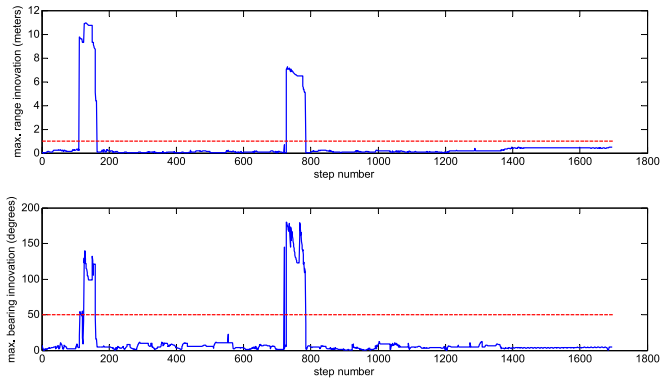


Fig. 19. This figure shows the innovation signals $\bar{r}_k$ and $\bar{\theta}_k$, along with the thresholds $r_{\max}$ and $\theta_{\max}$ (dashed red lines). Large jumps correspond to the kidnapping events.

Results on physical robots: Fig. 18 shows a snapshot of a run that contains two kidnappings and illustrates the robustness of the planning algorithm to the kidnapping situation. The feedback tree (shown in yellow) guides the robot toward the goal through the front door. However, before reaching the goal point, the robot is kidnapped in the hallway and placed in an unknown location within 407-area (see Fig. 18). In our implementations, we consider $r_{\max} = 1$ m and $\theta_{\max} = 50°$. The first jump in Fig. 19 shows this deviation. Once the robot recovers from being kidnapped (i.e., when both innovation signals in Fig. 19 fall below their corresponding thresholds), replanning from the new point is performed. This time, the feedback guides the robot toward the goal point from within 407-area. However, before the robot reaches the goal point, it is kidnapped again and placed in an unknown location (see Fig. 18). The second jump in the innovation signals in Fig. 19 corresponds to this kidnapping.

D. Longer and More Complex Experiments: Robustness to Changing Goals, Obstacles, and to Large Deviations

To emphasize the ability of the system to perform long-term SLAP, we conduct a complex experiment that consists of visiting several goals. The user(s) submit a new goal for the robot every time it reaches its current goal. While the robot needs to change the plan each time a new goal is submitted, it frequently encounters changes in the obstacle map (open/closed doors and moving people) as well as intermittent sensor fail-

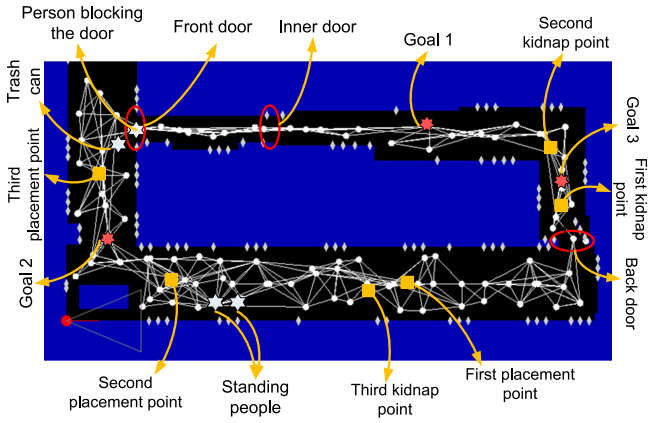


Fig. 20. Scenario for the long-term autonomous operation with a sequence of goals as well as various intermediate events and changes in the environment map.

ures and kidnapping situations. In [64], we provide an itemized and detailed description of the specific steps involved in this experiment based on Fig. 20 and accompanying video [71].

This long and complicated scenario demonstrates how simultaneous planning and localization can lead to robust behaviors in the presence of intermittent model discrepancies, changes in the environment, and large deviations in the robot's location. It is worth noting that online replanning in belief space is a challenge for the state-of-the-art belief planners as they mainly rely on search structures that depend on the system's initial belief. Hence, when the system's localization pdf encounters a significant deviation, replanning from the new localization belief requires the structure to be rebuilt, which is not a feasible operation online. However, constructing a query-independent graph (the underlying FIRM) allows us to embed it in a replanning scheme such as the proposed rollout policy technique and perform online replanning to enable SLAP.

VII. Method Limitations and Future Work

In this section, we recap the method assumptions and limitations mentioned in previous sections.

Restricted class of POMDPs: As discussed in the previous sections, it is worth noting that the proposed method is not a general-purpose POMDP solver. It provides a solution for a class of POMDP problems (including SLAP), where one can design closed-loop controllers with a funneling behavior in belief space. In the proposed instantiation of FIRM in this paper, designing funnels requires knowledge about the closed-form dynamics and sensor model. Also, the system needs to be locally linearizable at belief nodes, and the noise is assumed to be Gaussian. Further, designing a funnel/controller in belief space requires the uncertainty to be over the part of the state space that is controllable (e.g., the ego-vehicle). For example, the proposed SLAP solution is not applicable to two-player games, where there is no direct control on the opponent's motion or sensing.

Combining FIRM with general-purpose online solvers: Most of the general-purpose tree-based POMDP solvers can be combined with FIRM, where an online tree-based planner creates and searches the tree and use FIRM as the approximate policy (and cost-to-go) beyond the tree horizon. In particular, when the problem in hand does not satisfy the above-mentioned assumptions, one can approximate the original problem with a problem that does satisfy the above assumptions, create a FIRM graph, and use it as the base policy. Leveraging this base policy, one can use general-purpose online POMDP solvers in the vicinity of the current belief, such as Despot [32], ABT [33], POMCP [13], AEMS [31], that act on the original exact problem.

Dealing with dynamic environments: In this paper, we assume no prior knowledge about the environment dynamics. As a result, the simple model we use for new obstacles is: they either enter the map with a large forgetting time of 10 min (e.g., doors) or avoided reactively (e.g., moving people). A more sophisticated and efficient solution can be obtained by learning and modeling changes over time [44] or using some prior on the motion of moving objects. Incorporating such knowledge in the proposed planning framework is a subject of future work.

VIII. Conclusion

In this paper, we proposed a rollout-policy-based algorithm for online replanning in belief space to enable SLAP. The proposed algorithm is able to switch between different homotopy classes of trajectories in real time. It also bypasses the belief stabilization process of the state-of-the-art FIRM framework. A detailed analysis was presented, which shows that the method can recover the performance and success probability that was traded off in the stabilization phase of FIRM. Further, by reusing the costs and transition probabilities computed in the offline construction phase, the method is able to enable SLAP, via online replanning, in the presence of changes in the environment and large deviations in the robot's pose. Via extensive simulations, we demonstrated performance gains when using the rollout-based belief planner. As a key focus of the work, we also demonstrated the results of the proposed belief space planner on a physical robot in a real-world indoor scenario. Our experiments show that the proposed method is able to perform SLAP and guide the robot to its target locations while dynamically replanning in real time and reacting to the changes in the obstacles and deviations in the robot's state. Such replanning is an important ability for physical systems, where stochasticity in the system's dynamics and measurements can often result in failure. Hence, we believe that the proposed SLAP solution takes an important step toward bringing belief space planners to physical applications and advances long-term safe autonomy for mobile robots.

References

[1] R. D. Smallwood and E. J. Sondik, "The optimal control of partially observable Markov processes over a finite horizon," *Oper. Res.*, vol. 21, no. 5, pp. 1071–1088, 1973.

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, pp. 99–134, 1998.

[3] J. Pineau, G. Gordon, and S. Thrun, "Point-based value iteration: An anytime algorithm for POMDPs," in *Proc. Int. Joint Conf. Artif. Intell.*, Acapulco, Mexico, 2003, pp. 1025–1032.

[4] A. Agha-Mohammadi, S. Chakravorty, and N. Amato, "FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements," *Int. J. Robot. Res.*, vol. 33, pp. 268–304, Feb. 2014.

[5] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proc. Conf. Uncertainty Artif. Intell.*, 2005, pp. 542–547.

[6] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *J. Artif. Intell. Res.*, vol. 24, pp. 195–220, 2005.

[7] H. Kurniawati, D. Hsu, and W. Lee, "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces," in *Proc. Robot., Sci. Syst.*, 2008, pp. 65–72.

[8] H. Kurniawati, Y. Du, D. Hsu, and W. S. Lee, "Motion planning under uncertainty for robotic tasks with long time horizons," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 308–323, 2011.

[9] D. Grady, M. Moll, and L. E. Kavraki, "Automated model approximation for robotic navigation with POMDPs," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2013, pp. 78–84.

[10] Z. Littlefield, D. Klimenko, H. Kurniawati, and K. Bekris, "The importance of a suitable distance function in belief-space planning," in *Proc. Int. Symp. Robot. Res.*, 2015, pp. 683–700.

[11] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online POMDP planning for autonomous driving in a crowd," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2015, pp. 454–460.

[12] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel, "Scaling up Gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation," in *Algorithmic Foundations Robot. XI*. New York, NY, USA: Springer, 2015, pp. 515–533.

[13] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Proc. 23rd Int. Conf. Neural Inf. Process. Syst.*, 2010, pp. 2164–2172.

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[15] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Trans. Robot. Autom.*, vol. 17, no. 3, pp. 229–241, Jun. 2001.

[16] S. Prentice and N. Roy, "The belief roadmap: Efficient planning in belief space by factoring the covariance," *Int. J. Robot. Res.*, vol. 28, no. 11/12, pp. 1448–1465, Oct. 2009.

[17] J. van den Berg, P. Abbeel, and K. Goldberg, "LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 895–913, 2011.

[18] H. Carrillo, Y. Latif, M. Rodriguez-Arevalo, J. Neira, and J. Castellanos, "On the monotonicity of optimality criteria during exploration in active slam," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2015, pp. 1476–1483.

[19] H. Carrillo, I. Reid, and J. Castellanos, "On the comparison of uncertainty criteria for active slam," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2012, pp. 2080–2087.

[20] L. Carlone, J. Du, M. Kaouk Ng, B. Bona, and M. Indri, "Active slam and exploration with particle filters using Kullback-Leibler divergence," *J. Intell. Robot. Syst.*, vol. 75, no. 2, pp. 291–311, Aug. 2014. [Online]. Available: http://dx.doi.org/10.1007/s10846-013-9981-9

[21] A. Kim and R. Eustice, "Perception-driven navigation: Active visual SLAM for robotic area coverage," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 3196–3203.

[22] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments," *Int. J. Robot. Res.*, vol. 34, no. 7, pp. 849–882, 2015.

[23] L. Carlone and D. Lyons, "Uncertainty-constrained robot exploration: A mixed-integer linear programming approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2014, pp. 1140–1147.

[24] H. Kurniawati, T. Bandyopadhyay, and N. Patrikalakis, "Global motion planning under uncertain motion, sensing, and environment map," *Auton. Robots*, vol. 33, pp. 1–18, 2012.

[25] H. Bai, D. Hsu, W. S. Lee, and V. A. Ngo, "Monte Carlo value iteration for continuous-state POMDPs," in *Proc. Workshop Algorithmic Found. Robot.*, 2010, pp. 175–191.

[26] P. Chaudhari, S. Karaman, D. Hsu, and E. Frazzoli, "Sampling-based algorithms for continuous-time POMDPs," in *Proc. Amer. Control Conf.*, Washington, DC, USA, 2013, pp. 4604–4610.

[27] T. Smith and R. Simmons, "Heuristic search value iteration for POMDPs," in *Proc. 20th Conf. Uncertainty Artif. Intell.*, 2004, pp. 520–527.

[28] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee, "Planning under uncertainty for robotic tasks with mixed observability," *Int. J. Robot. Res.*, vol. 29, no. 8, pp. 1053–1068, 2010.

[29] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Auton. Agents Multi-Agent Syst.*, vol. 27, no. 1, pp. 1–51, 2013.

[30] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for POMDPs," *J. Artif. Intell. Res.*, vol. 32, pp. 663–704, 2008.

[31] S. Ross *et al.*, "AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs." in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 2592–2598.

[32] A. Somani, N. Ye, D. Hsu, and W. S. Lee, "DESPOT: Online POMDP planning with regularization," in *Proc. Adv.Neural Inf. Process. Syst.*, 2013, pp. 1772–1780.

[33] H. Kurniawati and V. Yadav, "An online POMDP solver for uncertainty planning in dynamic environment," in *Proc. Int. Symp. Robot. Res.*, 2013, pp. 611–629.

[34] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez, "Belief space planning assuming maximum likelihood observations," in *Proc. Robot., Sci. Syst.*, Jun. 2010.

[35] J. van den Berg, S. Patil, and R. Alterovitz, "Efficient approximate value iteration for continuous gaussian POMDPs," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 1832–1838.

[36] T. Erez and W. D. Smart, "A scalable method for solving high-dimensional continuous POMDPs using local approximation," in *Proc. 26th Int. Conf. Uncertainty Artif. Intell.*, 2010, pp. 160–167.

[37] S. Chakravorty and R. S. Erwin, "Information space receding horizon control," in *Proc. IEEE Symp. Adaptive Dyn. Program. Reinforcement Learn.*, Apr. 2011, pp. 302–309.

[38] R. He, E. Brunskill, and N. Roy, "Efficient planning under uncertainty with macro-actions," *J. Artif. Intell. Res.*, vol. 40, pp. 523–570, Feb. 2011.

[39] N. D. Toit and J. W. Burdick, "Robotic motion planning in dynamic, cluttered, uncertain environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 966–973.

[40] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2011, pp. 723–730.

[41] N. Kuhnen, A. Sieverling, and O. Brock, "Sensor-based, task-constrained motion generation under uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, China, 2014, pp. 4348–4355.

[42] L. P. Kaelbling and T. Lozano-Pérez, "Integrated task and motion planning in belief space," *Int. J. Robot. Res.*, vol. 32, no. 9/10, pp. 1194–1227, 2013.

[43] C. Bowen and R. Alterovitz, "Closed-loop global motion planning for reactive execution of learned tasks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 1754–1760.

[44] B. Marthi, "Robust navigation execution by planning in belief space," in *Proc. Robot. Sci. Syst.*, Jul. 2012.

[45] A. *et al.*, "Robust online belief space planning in changing environments: Application to physical mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, China, May 2014, pp. 149–156.

[46] D. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2007.

[47] S. Bhattacharya, M. Likhachev, and V. Kumar, "Identification and representation of homotopy classes of trajectories for search-based path planning in 3D," in *Proc. Robot., Sci. Syst.*, Los Angeles, CA, USA, Jun. 2011.

[48] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "FIRM: Feedback controller-based information-state roadmap: A framework for motion planning under uncertainty-," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Systems*, San Francisco, CA, USA, Sep. 2011, pp. 4284–4291.

[49] L. Kavraki, P. Švestka, J. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[50] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *Int. J. Robot. Res.*, vol. 18, no. 6, pp. 534–555, 1999.

[51] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "On the probabilistic completeness of the sampling-based feedback motion planners in belief space," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, USA, Sep. 2012, pp. 3983–3990.

[52] P. R. Kumar and P. P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1986.

[53] W. F. Arnold III and A. J. Laub, "Generalized eigenproblem algorithms and software for algebraic Riccati equations," *Proc. IEEE*, vol. 72, no. 12, pp. 1746–1754, 1984, pp. 1746–1754.

[54] G. Oriolo, A. De Luca, and M. Vandittelli, "WMR control via dynamic feedback linearization: Design, implementation, and experimental validation," *IEEE Trans. Control Syst. Technol.*, vol. 10, no. 6, pp. 835–851, Nov. 2002.

[55] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. Autom. Control*, vol. 38, no. 5, pp. 700–716, May 1993.

[56] C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in cartesian space," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1991, pp. 1136–1141.

[57] A. De Luca, G. Oriolo, and M. Vendittelli, "Control of wheeled mobile robots: An experimental overview," in *Ramsete*. New York, NY, USA: Springer, 2001, pp. 181–226.

[58] A. Agha-mohammadi, S. Chakravorty, and N. Amato, "Nonholonomic motion planning in belief space via dynamic feedback linearization-based FIRM," in *Proc. Int. Conf. Intell. Robots Syst.*, Vilamoura, Portugal, Oct. 2012, pp. 4433–4440.
[59] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practicea survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.
[60] D. Li, F. Qian, and P. Fu, "Variance minimization approach for a class of dual control problems," *IEEE Trans. Autom. Control*, vol. 47, no. 12, pp. 2010–2020, Dec. 2002.
[61] D. van Hessem and O. Bosgra, "A full solution to the constrained stochastic closed-loop MPC problem via state and innovations feedback and its receding horizon implementation," in *Proc. 42nd IEEE Conf. Decision Control*, Maui, HI, USA, Dec. 2003, pp. 929–934.
[62] S. K. Shah, C. D. Pahlajani, N. A. Lacock, and H. G. Tanner, "Stochastic receding horizon control for robots with probabilistic state constraints," in *Proc. IEEE Int. Conf. Robot. Autom.*, Minneapolis, MN, USA, Sep. 2012, pp. 2893–2898.
[63] R. Platt, "Convex receding horizon control in non-Gaussian belief space," in *Proc. Algorithmic Found. Robot.*, 2012, pp. 443–458.
[64] A. Agha-mohammadi, S. Agarwal, S.-K. Kim, S. Chakravorty, and N. M. Amato, "SLAP: Simultaneous localization and planning under uncertainty for physical mobile robots via dynamic replanning in belief space: Extended version," *CoRR*, arXiv:1510.07380, 2015. [Online]. Available: http://arxiv.org/abs/1510.07380
[65] S. M. LaValle, M. S. Branicky, and S. R. Lindemann, "On the relationship between classical grid search and probabilistic roadmaps," *Int. J. Robot. Res.*, vol. 23, no. 7/8, pp. 673–692, 2004.
[66] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *Int. J. Robot. Res.*, vol. 25, no. 7, pp. 627–643, 2006.
[67] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2000, vol. 1, pp. 521–528.
[68] J. van den Berg, "Extended LQR: Locally-optimal feedback control for systems with non-linear dynamics and non-quadratic cost," in *Proc. Robot. Res.*, 2016, pp. 39–56.
[69] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *Proc. Amer. Control Conf.*, Jun. 2005, vol. 1, pp. 300–306.
[70] J. van den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *Int. J. Robot. Res.*, vol. 31, no. 11, pp. 1263–1278, 2012.
[71] "Video of FIRM plan execution on a physical robot in a changing environment." 2013. [Online]. Available: https://www.youtube.com/watch?v=6cKzcfVDes8
[72] B. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc. IEEE 11th Int. Conf. Adv. Robot.*, 2003, vol. 1, pp. 317–323.
[73] R. Munoz-Salinas. 2018. [Online]. Available: http://sourceforge.net/projects/aruco/.
[74] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1999, vol. 2, pp. 1024–1031.
[75] R. Brooks, "A robust layered control system for a mobile robot," *IEEE J. Robot. Autom.*, vol. 2, no. 1, pp. 14–23, Mar. 1986.
[76] H. Choset *et al.Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA, USA: MIT Press, 2005.

Ali-akbar Agha-mohammadi received the B.S. and M.S. degrees in electrical and computer engineering (control systems) from Tabriz University and K.N. Toosi University of Technology, in 2005 and 2008, respectively. He received the Ph.D. degree in computer science and engineering from Texas A&M University, TX, USA, in 2013.

He is currently a Robotics Research Technologist with NASA-JPL, California Institute of Technology, Pasadena, CA, USA. He was a Research Engineer with Qualcomm Research and a Postdoctoral Researcher with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include robotic autonomy and mobility, stochastic systems, control systems, estimation, and filtering theory.

Dr. Agha became a NASA NIAC Fellow in 2018.

Saurav Agarwal received the B.S. degree in aerospace engineering from Indian Institute of Technology Bombay, Mumbai, India, and the M.S. degree in aerospace engineering from Cranfield University, Cranfield, U.K., and the Ph.D. degree in aerospace engineering from Texas A&M University, TX, USA, respectively.

He is currently the CEO and Co-Founder of Stocked Robotics. His research interests include SLAM, motion planning, estimation and control, and computer vision.

Dr. Agarwal was a recipient of Roberto Padovani Scholarship from Qualcomm in 2015.

Sung-Kyun Kim received the B.S. degree in mechanical and aerospace engineering from Seoul National University, Seoul, South Korea, and the M.S. degree in mechanical engineering from Korea Advanced Institute of Science and Technology, Daejeon, South Korea. He is currently a Ph.D. candidate at the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA.

He is currently a Research Scientist on leave of absence in the Center for Robotics Research, Korea Institute of Science and Technology. His current research interests include motion planning under uncertainty, search-based motion planning, and dexterous/mobile manipulation.

Mr. Kim was a recipient of Fulbright Foreign Student Scholarship.

Suman Chakravorty received the B.Tech. degree in mechanical engineering from Indian Institute of Technology, Madras, India, in 1997 and the M.S. and Ph.D. degrees in aerospace engineering from University of Michigan, Ann Arbor, USA, in 2000 and 2004, respectively.

He is currently an Associate Professor with the Aerospace Engineering Department, Texas A&M University, TX, USA. His research interests include autonomous robotic mapping and planning, stochastic dynamical systems including stochastic hybrid systems and space-based high-resolution imaging systems.

Nancy M. Amato received the undergraduate degrees in mathematical sciences and economics from Stanford, CA, USA, and the M.S. and Ph.D. degrees in computer science from the University of California, Berkeley, CA, USA and the University of Illinois at Urbana-Champaign, USA, respectively. She is Regents Professor and Unocal Professor of Computer Science and Engineering at Texas A&M University, TX, USA. In January 2019, she will join the Department of Computer Science at the University of Illinois at Urbana-Champaign, USA, as Department Head and as Abel Bliss Professor of Engineering. Her research focuses on robotics motion planning, computational biology and geometry, and parallel computing. She is a AAAI, AAAS, ACM, and IEEE Fellow.