Robust Online Belief Space Planning in Changing Environments: Application to Physical Mobile Robots

Ali-akbar Agha-mohammadi, Saurav Agarwal, Aditya Mahadevan,
Suman Chakravorty, Daniel Tomkins, Jory Denny, Nancy M. Amato

***Abstract*— Motion planning in belief space (under motion and sensing uncertainty) is a challenging problem due to the computational intractability of its exact solution. The Feedback-based Information RoadMap (FIRM) framework made an important theoretical step toward enabling roadmap-based planning in belief space and provided a computationally tractable version of belief space planning. However, there are still challenges in applying belief space planners to physical systems, such as the discrepancy between computational models and real physical models. In this paper, we propose a dynamic replanning scheme in belief space to address such challenges. Moreover, we present techniques to cope with changes in the environment (e.g., changes in the obstacle map), as well as unforeseen large deviations in the robot's location (e.g., the kidnapped robot problem). We then utilize these techniques to implement the first online replanning scheme in belief space on a physical mobile robot that is robust to changes in the environment and large disturbances. This method demonstrates that belief space planning is a practical tool for robot motion planning.**

I. Introduction

Sequential decision making under uncertainty is a key prerequisite for many robotics applications. Consider an autonomous, low-cost mobile robot that is subject to motion noise and lacks exact measurements due to sensor noise. Controlling this robot and planning motions for it is an instance of the Partially-Observable Markov Decision Process (POMDP) [13], [23] problem, which is a formal framework for sequential decision making under uncertainty. However, the POMDP problem is also notorious for its computational intractability. Methods such as [11], [15], [18], [24], [25] reduce the computation burden of POMDPs and aim to solve more challenging and realistic problems. Recently, the Feedback-based Information RoadMap (FIRM) framework [3] takes an important theoretical step toward realistic scenarios by significantly reducing the computational complexity of planning under uncertainty.

Additionally, handling changes in the environment (e.g., obstacles), changes in the goal location, and large deviations in the robot's location calls for online planning in uncertain, partially observable environments. However, when dealing with real-world physical systems, POMDP-based methods, including FIRM, encounter another important challenge: *discrepancy between real models with the models used for computation*. Such discrepancies can lead to deviations from the desired plan. Moreover, changes in the environment and large disturbances are other important challenges that needs to be handled. One strategy to address this problem is an ability to dynamically replan in belief space. In this paper, we propose a principled rollout-based extension of FIRM planning to facilitate its application to real-time stochastic (re)planning problems, and deal with changes in the environment and large disturbances in the robot's state.

(a)

(b)

Fig. 1. (a) A picture of robot (iRobot Create) in the operating environment. Landmarks can be seen on the walls. (b) Floorplan of the environment, in which experiments are conducted.

In the main body of POMDP literature, in particular sampling-based methods, the computed solution depends on the initial belief [4], [7], [9], [14], [21], [28] (sometimes referred to as single-query solvers). Therefore, in replanning (planning from a new initial belief) almost all the computations need to be done again, which prohibits their usage in cases where real-time dynamic replanning schemes, such as Receding Horizon Control (RHC), are needed. However, multi-query methods such as FIRM [2], [3] provide a construction mechanism independent of the initial belief of the system. As a result, they are suitable methods to be used for dynamic replanning purposes.

Trajectory optimization-based methods can also be used for replanning in an RHC scheme. The RHC framework was

Agha-mohammadi is with the Laboratory for Information and Decision Systems, MIT, Cambridge, MA, 02139. Agarwal and Chakravorty are with the Dept. of Aerospace Engineering, and Mahadevan, Tomkins, Denny, and Amato are with the Dept. of Computer Science and Engineering, Texas A&M University, TX, 77843, USA. Emails: aliagha@mit.edu {sauravag,mahadeven,schakrav,kittsil,jorydenny amato}@tamu.edu. This research supported in part by AFOSR Grant FA9550-08-1-0038 and by NSF awards CNS-0551685, CCF-0833199, CCF-0830753, IIS-0916053, IIS-0917266, EFRI-1240483, RI-1217991, by NIH NCI R25 CA090301-11, by Chevron, IBM, Intel, Oracle/Sun and by Award KUS-C1-016-04, made by King Abdullah University of Science and Technology (KAUST). J. Denny supported in part by an NSF Graduate Research Fellowship.

originally designed for deterministic systems. The most common approach is to approximate the stochastic system with a deterministic one by replacing the uncertain quantities with their mean (or maximum likelihood) values [5]. Methods such as [8], [10], [12], [20], [27] fall into this category and can be used in the RHC setting; they replace future random observations with their deterministic maximum likelihood value. However, in this form of RHC, the optimization is carried out only within a limited horizon. Also, removing the system's stochasticity may lead to unreliable plans.

The main contributions of this paper are threefold.

- We propose a principled method for real-time replanning in belief space by extending the idea of the rollout policy [5] to belief space using FIRM. This method considers all possible future observations.
- We propose techniques such as a "lazy feedback evaluation" algorithm to react to changes in the environment as well as large disturbances.
- We implement the proposed belief space planning scheme on a physical robotic system as an application of the FIRM framework. We demonstrate the robustness of the method to changes in the environment, failures in the sensory system, and large deviations.

These results lay the groundwork for further application of the theoretical POMDP framework to practical applications, thus moving toward long-term autonomy in robotic systems.

II. Problem Statement and Target Application

We aim to design a belief space planner that can handle uncertainties associated with a typical low-cost robot. Moreover, the planner needs to be able to replan in real-time so that it can cope with changes in the environment as well as deviations resulting from model discrepancies, large disturbances, and sensor failures.

To formally define the problem, we start by defining the concept of belief and policy. Consider a system whose state, control, and motion noise are denoted by x_k, u_k, and w_k, respectively, at the k-th time step. Let us denote the state evolution model by $x_{k+1} = f(x_k, u_k, w_k)$. In a partially observable environment, the exact value of system state x_k is not known. However, we can get the measurement (or observation) vector z_k at the k-th time step through sensors. Let us denote the measurement model by $z_k = h(x_k, v_k)$, where v_k denotes sensing noise. Therefore, the only available data for decision making at the k-th time step are the observations we have received and the controls we have applied up to that time step, i.e., $\mathcal{H}_k = \{z_{0:k}, u_{0:k-1}\} = \{z_0, z_1, \cdots, z_k, u_0, \cdots, u_{k-1}\}$. A filtering module can encode this data into a probability distribution over all possible system states $b_k = p(x_k|\mathcal{H}_k)$, which is referred to as the *belief* or *information-state*. Therefore, the action u_k can be taken based on the belief b_k using a policy (planner) π_k, i.e., $u_k = \pi_k(b_k)$. In Bayesian filtering, belief can be computed recursively based on the last action and current observation, $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ [5], [26].

To find the policy π_k, we need to define the objective of planning. Although the objective function can be general, the cost function we will use in our experiments includes the localization uncertainty, control effort, and elapsed time.

$$c(b_k, u_k) = \zeta_p \text{tr}(P_k) + \zeta_u \|u_k\| + \zeta_T, \tag{1}$$

where $\text{tr}(P_k)$ is the trace of estimation covariance. 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 scalars that combine these costs to achieve a desirable behaviour. In the presence of a constraint set F (e.g., obstacles), we assume that the task fails if the robot violates these constraints (e.g., collides with obstacles). Therefore, in case of failure, the running-sum of costs (cost-to-go), i.e., $J(F) = \sum_{t'}^{\infty} c(b, u)$ is set to a suitably high cost-to-go.

Planning under uncertainty is defined as finding a sequence of policies $\pi_{0:\infty}(\cdot) = \{\pi_1(\cdot), \pi_2(\cdot), \pi_3(\cdot), \cdots\}$. Therefore, the original problem of stochastic control with imperfect state information is defined as follows:

Problem 1. ***(POMDP)*** *The problem of stochastic control with imperfect state information, or the Partially-Observable Markov Decision Process (POMDP) problem, is defined as the following optimization over the policy space:*

$$\pi_{0:\infty}(\cdot) = \arg\min_{\Pi_{0:\infty}} \sum_{k=0}^{\infty} \mathbb{E}\left[c(b_k, \pi_k(b_k))\right] \tag{2}$$
$$s.t. \quad b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k)$$
$$x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k))$$

where, 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, the solution is a stationary policy π_s, i.e., $\pi_1 = \pi_2 = \cdots = \pi_s$. However, Problem 1 is written in a more general setting to emphasize the connection with rollout policy, discussed further below. Solving the POMDP problem is computationally intractable over continuous state, action, and observation spaces. However, the main problem that this paper aims to solve is the following:

Problem 2. ***(Re-Solving POMDPs in real-time)*** *In case of a change in the failure set F (e.g. obstacles) or large deviation in the system's belief, re-solve Problem 1 in real-time.*

This paper aims at solving Problem 2 by exploiting FIRM, which re-use the computations performed for solving the POMDP problem a priori and hence can deal with such changes online.

A. Sample Application Scenario

We exercise the proposed planner in an office-like environment, where we use a low-cost iRobot Create platform (Figure 1(a)), on which a Dell Latitude laptop with an on-board camera is mounted. The robot obtains noisy measurements (relative range and bearing) from unique landmarks that are installed in the environment. The desired behaviour for the planner is to guide the robot to a goal through those regions of the environment where the robot can better localize itself and hence better avoid collisions. Most importantly, the

planner needs to be able to replan online so that it can handle changes in the environment and deviations resulting from model discrepancies, large disturbances, and sensor failures. We briefly discuss the environment, robot model, and sensory system. More detailed descriptions can be found in [1].

Environment: The specific environment for conducting experiments is the fourth floor of the Bright building at Texas A&M University. A floorplan is shown in Fig. 1(b). The hallway (yellow) and the experiment region (blue) are highlighted. The blue region contains a large cluttered office (room 407) with several doors.

System model (robot and sensors): We use an iRobot Create (Fig. 1(a)), whose state $x_k = (\mathrm{x}_k, \mathrm{y}_k, \theta_k)^T$ encodes its 2D position and heading angle at the k-th time step. The state evolution model $x_{k+1} = f(x_k, u_k, w_k)$ is the unicycle model, where the control command u_k consists of the linear and angular velocities $u_k = (V_k, \omega_k)^T$. Motion noise $w_k \sim \mathcal{N}(0, \mathbf{Q}_k)$ gets added to the control signal (see [1] for details). For sensing purposes, we use the laptop's on-board camera to detect landmarks (with unique black and white patterns) that are placed at known locations on walls (Fig. 1(a)). Denoting the j-th landmark position as ${}^jL$, the obtained measurement is the relative range and bearing to the landmark:

$$ {}^j z_k = [\|{}^j\mathbf{d}_k\|, \text{atan2}({}^jd_{2_k}, {}^jd_{1_k}) - \theta]^T + {}^j v,\ {}^j v \sim \mathcal{N}(\mathbf{0}, {}^j\mathbf{R}), $$

where ${}^j\mathbf{d}_k = [{}^jd_{1_k}, {}^jd_{2_k}]^T := [\mathrm{x}_k, \mathrm{y}_k]^T - L_j$. Experimentally, we have found that the intensity of measurement noise ${}^j v$ increases with the distance from the j-th landmark and the incidence angle. The incidence angle refers to the angle between the line connecting the camera to a landmark and a surface normal to the wall on which the landmark is mounted. Denoting the incident angle by $\phi \in [-\pi/2, \pi/2]$, we model the sensing noise associated with the j-th landmark as a zero mean Gaussian whose covariance is

$$ {}^j\mathbf{R}_k = \mathbf{diag}\left((\eta_{r_d}\|{}^j\mathbf{d}_k\| + \eta_{r_\phi}|\phi_k| + \sigma_b^r)^2, (\eta_{\theta_d}\|{}^j\mathbf{d}_k\| + \eta_{\theta_\phi}|\phi_k| + \sigma_b^\theta)^2\right) \quad (3) $$

In our implementation, we use $\eta_{r_d} = 0.1$, $\eta_{r_\phi} = 0.01$, $\sigma_b^r = 0.05$m, $\eta_{\theta_d} = 0.001$, $\eta_{\theta_\phi} = 0.01$, and $\sigma_b^\theta = 2.0$deg. The full vector of measurements z is the concatenation of measurements from visible landmarks.

III. Overview of FIRM

In this section, we briefly review the Feedback-based Information RoadMap (FIRM) framework [2], [3]. However, the concrete realization of FIRM constructed for conducting the experiments is detailed in [1]. An Information RoadMap (IRM) is a "multi-query" graph in belief space constructed independent of the initial belief space. Therefore, the intractable belief MDP problem can be reduced to a tractable MDP problem on this graph. Each node in an IRM is a small region $B = \{b : \|b - \grave{b}\| \leq \epsilon\}$ around a sampled belief $\grave{b}$. We denote the i-th node by B^i and the set of nodes by $\mathbb{V} = \{B^i\}$. Each edge in an IRM is a local controller. In FIRM, each edge (local controller) is a feedback controller whose goal is to drive the belief into the target node of the edge. We denote the edge (controller) between nodes i and j by μ^{ij} and the set of edges by $\mathbb{M} = \{\mu^{ij}\}$. A policy π^g on the graph is a mapping from graph nodes to edges; i.e., $\pi^g : \mathbb{V} \to \mathbb{M}$. Denote the set of all possible policies as Π^g.

Having such a graph in belief space, we can form the POMDP on the FIRM graph (so-called FIRM MDP):

$$ \pi^g = \arg\min_{\Pi^g} \mathbb{E} \sum_{n=0}^{\infty} C^g(B_n, \pi^g(B_n)) \quad (4) $$

where, B_n is the n-th visited node, and μ_n is the edge taken at B_n. $C^g(B, \mu) := \sum_{k=0}^{\mathcal{T}} c(b_k, \mu(b_k))$ is the generalized cost of taking local controller μ at node B centered at b_0.

We incorporate the failure set in planning by adding a hypothetical FIRM node $B^0 = F$ to the list of FIRM nodes. As the FIRM MDP in Eq.(4) is defined over the finite set of nodes, we can solve it by computing the graph cost-to-go through solving the following dynamic programming:

$$ J^g(B^i) = \min_{\mu}\{C^g(B^i, \mu) + \sum_{\gamma=0}^{N} \mathbb{P}^g(B^\gamma | B^i, \mu) J^g(B^\gamma)\} \quad (5) $$

where $\mathbb{P}^g(B^\gamma | B^i, \mu)$ is the probability of reaching B^γ from B^i under μ. The failure and goal cost-to-go's (i.e., $J^g(B^0)$ and $J^g(B^{goal})$) are set to a suitably high positive value and zero, respectively. Accordingly, the replanning algorithms, when start or goal changes, are presented in Algorithms 1 and 2. For a more detailed description of FIRM, see [1].

Algorithm 1: (Re)plan_from

```
input : Start belief b_0, cost-to-go J^g(·), nodes V = {B^i}
output : Next Local Controller μ*
Find r neighboring nodes N = {B^i}_{i=1}^r to b_0;
Set J*(B) = ∞;
for B ∈ N do
    Construct local planner μ from b_0 to B;
    Compute transition cost C(b_0, μ) and probability P(B|b_0, μ);
    if C(b_0, μ) + Σ_{γ=0}^{N} P(B^γ|b_0, μ) J^g(B^γ) < J*(B) then
        J*(B) = C(b_0, μ) + Σ_{γ=0}^{N} P(B^γ|b_0, μ) J^g(B^γ);
        μ* = μ;
return μ*;
```

Algorithm 2: (Re)plan_to

```
input : Goal node B^goal, FIRM Graph G = {V, M}
output : FIRM feedback π^g
Add B^goal to the graph; update V and M accordingly;
Compute the cost-to-go J^g and feedback π^g over the FIRM nodes by solving the MDP in Eq. (5);
return π^g;
```

IV. Dynamic Replanning in Belief Space

In this section, we first discuss the extension of the Receding Horizon Control (RHC) and Rollout Policy (ROP) [5] to belief space. Then we propose an ROP based on FIRM

that can cope with changes in the environment as well as large deviations.

RHC in belief space: Receding horizon control (often referred to as rolling-horizon or model-predictive control) was originally designed for deterministic systems (to cope with model discrepancy). For stochastic systems, where the closed-loop (feedback) control law is needed, the best formulation of the RHC scheme is a subject of current research [8], [16], [22]. In the most common form of RHC [5], the stochastic system is approximated with a deterministic system by replacing the uncertain quantities with their typical values (e.g., maximum likelihood value). In belief space planning, the quantities that inject randomness into belief dynamics are unknown future observations. Thus, one can replace random observations z_k with their deterministic maximum likelihood value z_k^{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. To describe these stages, let us assume we are at step n and the belief is b_n. 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 by solving the following optimization problem:

$$\begin{aligned} u_{0:T} = \arg\min_{\mathbb{U}_{0:T}} & \sum_{k=0}^{T} c(b_k^d, u_k) \\ s.t. \quad & b_{k+1}^d = \tau(b_k^d, u_k, z_{k+1}^{ml}), \quad b_0^d = b_n \\ & z_{k+1}^{ml} = \arg\max_z p(z|x_{k+1}^d) \\ & x_{k+1}^d = f(x_k^d, u_k, 0), \end{aligned} \tag{6}$$

In the second stage, it 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}, belief b_{n+1} will be different than b_1^d. Subsequently, RHC performs these two computations from the new belief b_{n+1}. In other words, RHC computes an open loop sequence $u_{0:T}$ from this new belief. This process continues until the belief reaches a desired belief location. Algorithm 3 recaps this procedure.

Algorithm 3: RHC for Partially-observable stochastic systems

input : Initial belief $b_{current} \in \mathbb{X}$, $B_{goal} \subset \mathbb{B}$
while $b_{current} \notin B_{goal}$ **do**
  $u_{0:T}$ = Solve the optimization in Eq.(6) starting from $b_0^d = b_{current}$;
  Apply the action u_0 to the system;
  Observe the actual z;
  Compute the belief $b_{current} \leftarrow \tau(b_{current}, u_0, z)$;

State-of-the-art methods such as [27] and [19] utilize this form of RHC in belief space. This framework is also called Partially-Closed Loop RHC (PCLRHC) [27] since it partially exploits some information about the future observations (i.e., z^{ml}) and does not fully ignore them.

Issues with RHC: There are some issues regarding the presented form of the RHC framework: First, due to the limited horizon and ignoring the cost-to-go beyond the horizon, the method may get stuck into pitfalls 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. Second, the presented form of RHC ignores the stochasticity of the system within the horizon, which may lead to inaccurate approximations of the cost and unreliable control actions. To overcome these issues, researchers have proposed variants of RHC and different frameworks based on the idea of repeated planning [5]. Here, we discuss such a framework called "rollout policy" [5] and aim to realize it in belief space using the FIRM framework.

Rollout policy in belief space: A class of methods that aims to reduce the complexity of the stochastic planning problem in Eq.2 is the class of Rollout Policies (ROP) [5], which are more powerful than the described version of RHC in the following sense: First, they search for a sequence of policies (instead of open-loop controls) within the horizon, and do not approximate the system with a deterministic one. Second, they use a suboptimal policy, called the "base policy," to compute a cost-to-go function $\widetilde{J}$ that approximates the true cost-to-go beyond the horizon. In other words, at each step of the rollout policy scheme, the following closed-loop optimization is solved:

$$\begin{aligned} \pi_{0:T}(\cdot) = \arg\min_{\Pi_{0:T}} & \mathbb{E}\left[\sum_{k=0}^{T} c(b_k, \pi_k(b_k)) + \widetilde{J}(b_{T+1})\right] \\ s.t. \quad & b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\ & x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)) \end{aligned} \tag{7}$$

Then, only the first control law π_0 is used to generate the control signal u_0 and the rest of the policies are discarded. Similar to RHC, after applying the first control, a new sequence of policies is computed from the new point. The rollout algorithm is shown in Algorithm 4.

Algorithm 4: Rollout algorithm in Belief Space:

input : Initial belief $b_{current} \in \mathbb{B}$, $B_{goal} \subset \mathbb{B}$
while $b_{current} \notin B_{goal}$ **do**
  $\pi_{0:T}$ = Solve optimization in Eq.(7) starting from $b_0 = b_{current}$;
  Apply the action $u_0 = \pi(b_0)$ to the system;
  Observe the actual z;
  Compute the belief $b_{current} \leftarrow \tau(b_{current}, u_0, z)$;

Although the rollout policy in the 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 propose a rollout policy in the belief space based on the FIRM-based cost-to-go.

FIRM-based Rollout Policy: In the FIRM-based rollout policy, we adopt the FIRM policy as the base policy of the rollout algorithm. Accordingly, the cost-to-go of the FIRM policy will be used as the cost-to-go beyond the horizon. Now, if we have a dense FIRM graph such that FIRM nodes partition the belief space (i.e., $\cup_i B^i = \mathbb{B}$), then at the end of the horizon, the belief b_{T+1} belongs to a FIRM node B from which the FIRM cost-to-go is available. However, in practice, when the FIRM nodes cannot cover the entire belief space, we need to make sure that a truncated policy can drive the belief into a FIRM node at the end of horizon. Nevertheless, since the belief evolution is random, we may not be able to guarantee that the belief reaches a FIRM node at the end of a deterministic horizon T. Therefore, instead of truncating the policy over a fixed time, we truncate the policy once the belief reaches a pre-specified stopping region (which happens in a random time denoted by $\mathcal{T}$) as follows:

$$\begin{aligned}
&\pi_{0:\infty}(\cdot) = \arg\min_{\Pi_{0:\infty}} \mathbb{E}\left[\sum_{k=0}^{\mathcal{T}} c(b_k, \pi_k(b_k)) + \widetilde{J}(b_{\mathcal{T}+1})\right] \\
&s.t. \quad b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\
&\qquad x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)) \\
&\qquad b_{\mathcal{T}+1} \in \cup_j B^j, \qquad (8)
\end{aligned}$$

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

$$\widetilde{J}(b_{\mathcal{T}+1}) = J^g(B^j) \qquad (9)$$

The last condition in Eq.8 can be written more rigorously as $\mathbb{P}(b_{\mathcal{T}+1} \in \cup_j B^j | \pi) = 1$ for a finite $\mathcal{T}$. Also, as noted in Eq.(9), it is worth noting that the FIRM-based cost-to-go $J^g(\cdot)$ plays the role of the cost-to-go beyond the horizon $\widetilde{J}(\cdot)$.

Therefore, in solving the FIRM-based rollout policy problem, we aim to find a sequence of policies that ends up in a FIRM node and minimizes the cost in Eq.8. To find this optimal policy, we parametrize the policy space and perform minimization over the parameter space.

In our implementation, we adopt a variant of the Open-Loop Feedback Control (OLFC) scheme [5] along with a Kalman Filter as the belief controller. In this variant of OLFC, for a given $\mathbf{v}$, we compute an open-loop control sequence starting from the current estimation mean and ending at $\mathbf{v}$. Then, we apply a truncated sequence of the first l controls ($l = 5$ in our experiments). This process repeats every l steps until we reach the graph node. More details can be found in [1]. Therefore, the policy can be characterized by the next node; i.e., $\pi(\cdot; \mathbf{v})$. Thus, to solve the optimization in Eq.7 we search for the FIRM node $\grave{b}^j = (\mathbf{v}^j, P^j)$ whose mean, i.e., $\mathbf{v}^j$, leads to the best local policy $\pi(\cdot; \mathbf{v}^j)$. Accordingly, we implement the rollout technique in Algorithm 4.

V. Replanning in Changing Environments and Presence of Large Deviations

In this section, we discuss how we handle changes in the obstacle map and large deviations in the robot's belief. In general, handling these cases in belief space is a big challenge as they require online updating of the planning structure in belief space. It is important to note that it is the graph structure of FIRM that makes such an update and replanning feasible in real-time. The graph structure of FIRM allows us to *locally* change collision probabilities without affecting the rest of the graph (i.e., properties of different edges on the graph are independent of each other). It is important to note that such a property is not present in other state-of-the-art belief space planners, including SARSOP [15], BRM (Belief Roadmap Method) [21], or LQG-MP [28]. In those methods, collision probabilities and costs on *all* edges (number of possible edges is exponential in the size of underlying PRM) need to be re-computed.

A. Lazy Feedback Evaluation in Changing Environments

To adapt the proposed framework to handle changing environments, we rely on lazy evaluation methods. Inspired by the lazy evaluation methods for PRM frameworks [6], we propose a variant of the lazy evaluation methods for evaluating the generated feedback law. The basic idea is that at every node the robot re-evaluates *only* the next edge that it needs to take or a limited set of edges in the vicinity of the robot. By re-evaluation, we mean it re-computes collision probabilities along those edges. If there is a significant change in the local collision probabilities, then the dynamic programming problem is re-solved and a new feedback tree is computed. Otherwise, the feedback tree remains unchanged and the robot keeps following it. This lazy evaluation scheme can be performed in real-time. The method is outlined in Algorithm 5.

Algorithm 5: Lazy Feedback Re-Evaluation

```
input : Feedback π^g, current belief b_current
output : Updated feedback π^g
Update the obstacles map;
if there is a change in map then
    W ← Retrieve the sequence of nominal edges
    returned by feedback up to horizon l;
    forall the edges μ ∈ W do
        Re-compute the collision probabilities
        P_new(B, μ) from the start node B of edge;
    if exists μ ∈ W such that
    |P_new(B, μ) − P(B, μ)| > α then
        P(B, μ) ← P_new(B, μ);
        π^g ← Replan(b_current);
return π^g;
```

B. Handling Large Disturbances (kidnapped robot problem)

In robotics, the kidnapped robot problem commonly refers to a situation where an autonomous robot in operation is carried to an arbitrary location. This problem introduces different challenges such as (i) how to detect kidnapping, (ii) how to localize the robot, and (iii) how to control the robot to recover from this situation and accomplish its goal. The third part of this problem calls for online replanning in belief space.

Detecting a kidnapped situation: To detect the kidnapped situation, we constantly monitor the innovation signal $\widetilde{z}_k = z_k - z_k^-$ (the difference between actual and predicted observations). Recall that in our setting the observation at time step k from the j-th landmark is the relative range and bearing of the robot to the j-th landmark, i.e., ${}^j z_k = ({}^j r_k, {}^j \theta_k)$. The predicted version of this measurement is shown by ${}^j z_k^- = ({}^j r_k^-, {}^j \theta_k^-)$. We monitor the following measures of the innovation signal:

$$\widetilde{r}_k = \max_j(|{}^j r_k - {}^j r_k^-|), \quad \widetilde{\theta}_k = \max_j(d^\theta({}^j \theta_k, {}^j \theta_k^-)), \quad (10)$$

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 $\overline{\theta}_k$, we monitor the conditions $\overline{r}_k < r_{max}$ and $\overline{\theta}_k < \theta_{max}$. If both of them are satisfied, we follow the FIRM feedback (i.e., we are in the *Feedback Following Mode* (FFM)). However, violation of either of these conditions means that the robot is constantly observing high innovations, and thus it is not in the location that it was supposed to be (i.e., it is kidnapped). In Section VI, we show the innovation signal for a sample run on a physical robot. In our implementation, we consider $r_{max} = 1$ (meters) and $\theta_{max} = 50$ (degrees).

Information Gathering Mode (IGM): Once the robot detects it has been kidnapped, the estimation covariance is replaced with a large covariance to get an approximately uniform distribution over the state space. Then, we enter the Information Gathering Mode (IGM), where we take small and conservative steps (e.g., turning in place or taking random actions with small velocities) to obtain more measurements. Once the robot gets these measurements, the localization module corrects the estimation value and the innovation signal reduces. When conditions $\overline{r}_k < r_{max}$ and $\overline{\theta}_k < \theta_{max}$ are satisfied again, we exit the information gathering mode.

Post-IGM replanning: After recovering from being kidnapped, controlling the robot in belief space remains a significant challenge because the system can be far from where it was expected to be. However, using the proposed method and assuming the FIRM graph has enough nodes distributed well in the space, the robot needs to go only to a neighboring node from this new point. Therefore, there is no need for a costly replanning procedure. Indeed, the only required computation is to evaluate the cost of edges that connect the new start point to the neighboring FIRM nodes based on Algorithm 1.

VI. Experimental Results

In this section, we first discuss the results of PRM and FIRM-based motion planning and show how belief space planning can improve the performance. Then, we distinguish our method from the state-of-the-art by examining and discussing the robustness properties of the proposed method to changes in the obstacle map, and to large deviations in the robot's location and the goal location. The experiments are conducted on a low-cost iRobot Create equipped with a laptop and an integrated monocular web-camera (Fig. 1(a)).

A. Planning with PRM and FIRM

The goal of this section is to compare the performance of FIRM with deterministic planners such as Medial Axis PRM (MAPRM) [29]. The solution of the dynamic programming problem, i.e., π^g, is visualized with a *feedback tree* (FT). For each node, FT contains only one outgoing edge ($\mu = \pi^g(B^i)$). FT is rooted at the goal node.

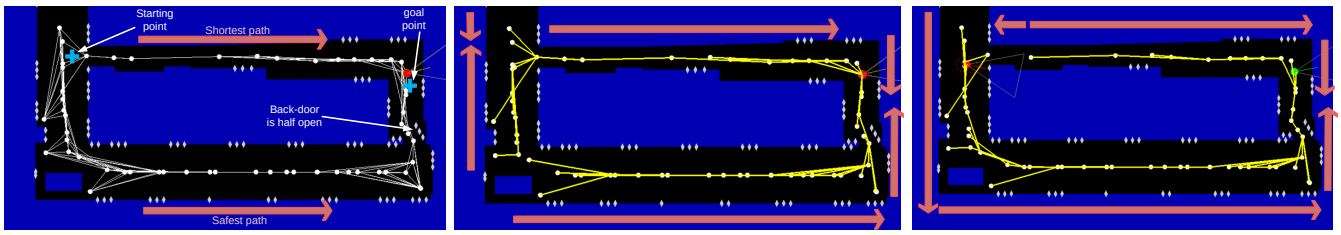
MAPRM-based planning: As one of the best variants of PRM when it comes to collision avoidance, we construct an MAPRM [29] in the environment (Fig. 2(a)). As is seen in Fig. 2(a), the path with maximum obstacle clearance (and the shortest path) is the one through the front door of room 407 (see Fig. 1(b)). Therefore, based on the obstacle clearance, MAPRM leads to the feedback tree shown in Fig. 2(b) that guides the robot through the front door. To execute the MAPRM plan we design LQG controllers to track the computed path. However, due to the lack of enough information along the solution path, the success rate of this plan is 27% (27 runs out of 100 Monte Carlo runs were successful) and the robot frequently collides with obstacles.

FIRM-based planning: In planning with FIRM, the information distribution in the environment is encoded in the planning via a framework which leads to a better judgement of the narrowness of passages in the belief space. Although in this environment the path through the front door is shorter, the success probability of traversing through the back door is more due to the presence of more information sources. 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 gives the feedback shown in Fig. 2(c), which results in an 88% success probability.

B. Robustness to Changes in the obstacle map

In this section, we investigate the robustness of the proposed algorithm to changes in obstacles for a physical system. In our experiments, we consider two types of obstacles. The first set of obstacles (most of the map) are static obstacles such as walls. The second class of obstacles include those that discretely change their state such as doors (state changes between "open" or "closed") in the environment. As discussed earlier, handling such changes is a challenge in state-of-the-art belief space planners since the planner cannot be updated locally and all computation for constructing the planner needs to be reproduced, which is not a feasible operation in real-time. The main focus of the following experiments is to demonstrate how our method can replan in real-time when faced with changes in the obstacle map.

We consider the environment shown in Fig. 1(b). The start and goal locations are shown in Fig. 3(a). We construct a PRM in the environment ignoring the changing obstacles (e.g., assuming all doors are open). Leveraging PRM to construct a FIRM and solving the dynamic programming problem on it, we get the feedback tree shown in Fig. 3(a)



(a) (b) (c)

Fig. 2. (a) The environment including obstacles (blue), free space (black), and landmarks (white diamonds) on the walls are shown. An MAPRM graph approximating the connectivity of free space, starting point, and goal point are shown. (b) The feedback tree generated by solving DP on MAPRM is shown in yellow. From each node there is only one outgoing edge (in yellow), computed by DP, guiding the robot toward the goal. Arrows in pink coarsely represent the direction on which the feedback guides the robot. (c) The feedback tree generated by solving DP on FIRM; As is seen, the computed feedback guides robots through more informative regions that leads to more accurate localization and less collision probabilities.

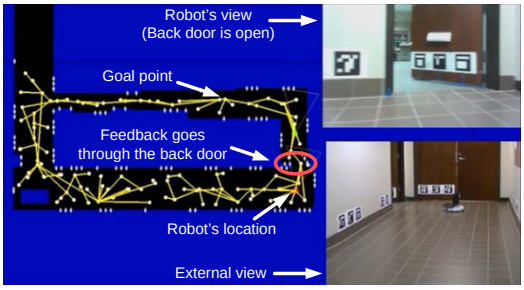
that guides the robot toward the goal through the back-door of room 407. However, the challenge is that the door may be closed when the robot reaches it, and there may be new obstacles in the environment. The robot needs to replan in real-time once it encounters such changes in the environment. For details on the obstacle detection mechanism see [1].

Figure 3(b) shows a snapshot of our run when the robot detects the door is in a different situation than expected. As a result, the robot updates the obstacle map as can be seen in Fig. 3(b), in which the door is closed. Accordingly, the robot replans in belief space based on Algorithm 5. Figure 3(b) shows the feedback tree resulting from replanning. As seen, the new feedback guides the robot through the front door, since it detects the back door is closed. The video of a long run (see Section VI-D) provides more detail on this procedure. Moreover, this video shows the robustness of the method to temporary failures in the perception system (e.g., missing landmarks due to blockages, blur, etc.), which is discussed more in [1].

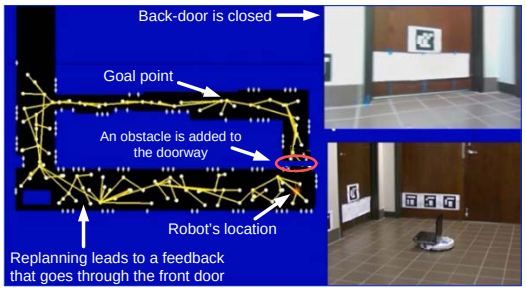
C. Robustness to large deviations

In this section, we investigate the robustness of the proposed framework in dealing with large deviations in the robot's position. As a more general form of this problem, we consider the *kidnapped robot problem* as discussed in the previous section. The need for online replanning in belief space makes this problem challenging.

Figure 4(a) shows a snapshot of a run that involves two kidnappings and illustrates the robustness of the planning algorithm to the kidnapping situation. The start and goal positions are shown in Fig. 4(a). The feedback tree (shown in yellow) guides the robot toward the goal through the front door. However, before reaching the goal point the robot gets kidnapped in the hallway (cf. Fig. 4(a)) and placed in an unknown location within room 407 (cf. Fig. 4(a)). The first jump in 4(b) shows this deviation. Once the robot recovers from being kidnapped (i.e., when both innovation signals in Fig. 4(b) fall below their corresponding thresholds), replanning from the new point is performed. Feedback guides the robot toward the goal point from within room 407. However, again, before robot reaches the goal point, it is kidnapped and placed in an unknown location (see Fig. 4(a)). The second jump in the innovation signals in Fig. 4(b) corresponds to this kidnapping. Again, replanning from the new point, the robot follows the feedback and reaches the goal point.



(a)

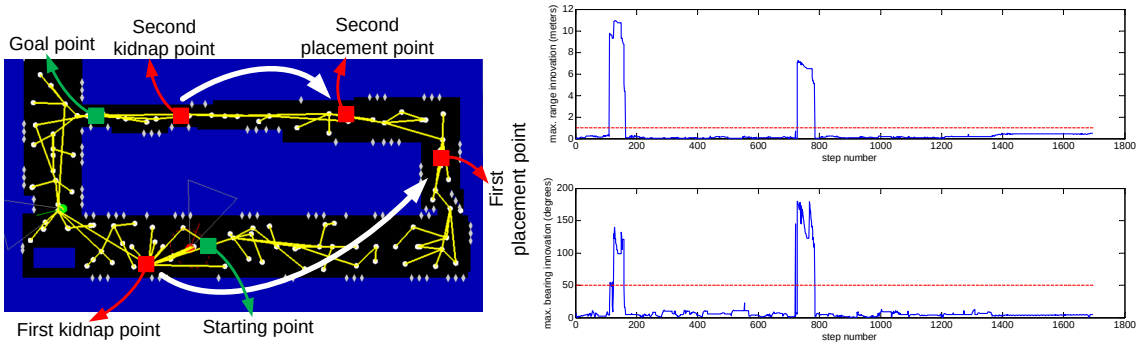


(b)

Fig. 3. (a) The back door is open at this snapshot. The feedback guides the robot toward goal through the back door. (b) The back door is closed at this snapshot. Robot detects the door is closed and updates the obstacle map (adds door). Accordingly robot replans and computes the new feedback. The new feedback guides the robot through the front door.

D. A Longer and more complex experiment

We next demonstrate the ability of the system to perform long-term tasks in a complex scenario that consists of visiting several goals (each time therobot reaches a goal, a user submits a new goal). The replanning ability allows the robot to change the plan online in belief space as the goal location changes. Moreover, the robot frequently encounters changes in the obstacle map (open/closed doors and new obstacles in the environment) as well as missing information sources and kidnapped robot situations. Thus, the robot frequently needs to perform a replanning operation in belief space to deal with such frequent changes. A 25 minute video of this run is recorded and available in [17] (a shorter version has been submitted along with the paper) that shows the robot's performance in this complex scenario. In this video, the robot faces three changes in the goal location, three changes in the door's state (open/closed), several new obstacles in



(a) (b)

Fig. 4. (a) The set up for the experiment containing two kidnapping. (b) Innovation signals $\widehat{r}_k$ and $\widehat{\theta}_k$ during this run. When both of the signals are below specified thresholds r_{max} and θ_{max} (dashed red lines), robot follows the FIRM feedback. Otherwise, the system enters the information gathering mode.

the environment, three kidnapping situations, and numerous failures of the sensory systems due to missing landmarks, blur in image, and etc.

VII. Conclusion

In this paper, we present an application of the FIRM motion planning method to a physical robotic system. This paper proposes a robust method for belief space planning based on efficient online replanning. Such replanning is a key ability in handling discrepancies between real world models and computational models, changes in the environment and obstacles, large deviations, and changes in information sources. We implemented this belief space planner on a physical system and demonstrate the robustness to such discrepancies that occur in practice. We believe this work provides an important step toward making POMDP methods applicable to real world robotic systems.

References

[1] Aliakbar Agha-mohammadi, Saurav Agarwal, Aditya Mahadeval, Daniel Tomkins, Jory Denny, Suman Chakravorty, and Nancy M. Amato. Dynamic real-time replanning in belief space: An experimental study on physical mobile robots. *Technical Report: TR13-007, Parasol Lab., CSE Dept., Texas A&M University*, 2013.

[2] Aliakbar Agha-mohammadi, Suman Chakravorty, and Nancy Amato. FIRM: Feedback controller-based Information-state RoadMap -a framework for motion planning under uncertainty-. In *International Conference on Intelligent Robots and Systems (IROS)*, 2011.

[3] Aliakbar Agha-mohammadi, Suman Chakravorty, and Nancy Amato. FIRM: Sampling-based feedback motion planning under motion uncertainty and imperfect measurements. *International Journal of Robotics Research*, 33(2):268–304, Feb. 2014.

[4] Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A. Ngo. Monte carlo value iteration for continuous-state pomdps. In *WAFR*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 175–191. Springer, 2010.

[5] Dimitri Bertsekas. *Dynamic Programming and Optimal Control: 3rd Ed.* Athena Scientific, 2007.

[6] Robert Bohlin and Lydia E Kavraki. Path planning using lazy prm. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 521–528. IEEE, 2000.

[7] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *ICRA*, pages 723–730, 2011.

[8] S. Chakravorty and R. Scott Erwin. Information space receding horizon control. In *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, April 2011.

[9] Pratik Chaudhari, Sertac Karaman, David Hsu, and Emilio Frazzoli. Sampling-based algorithms for continuous-time pomdps. In *the American Control Conference (ACC)*, Washington DC, 2013.

[10] Tom Erez and William D Smart. A scalable method for solving high-dimensional continuous pomdps using local approximation. In *the International Conference on Uncertainty in Artificial Intelligence*, 2010.

[11] Devin Grady, Mark Moll, and Lydia E. Kavraki. Automated model approximation for robotic navigation with POMDPs. In *ICRA*, 2013.

[12] R. He, E. Brunskill, and N. Roy. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40:523–570, February 2011.

[13] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[14] H. Kurniawati, T. Bandyopadhyay, and N.M. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, pages 1–18, 2012.

[15] H. Kurniawati, D. Hsu, and W.S. Lee. SARSOP: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Proceedings of Robotics: Science and Systems*, 2008.

[16] Duan Li, Fucai Qian, and Peilin Fu. Variance minimization approach for a class of dual control problems. *IEEE Trans. Aut. Control*, 47(12):2010–2020, 2002.

[17] Video on Parasol Lab. webpage. http://youtu.be/m3t3udm0ftu.

[18] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

[19] R. Platt. Convex receding horizon control in non-gaussian belief space. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2012.

[20] Robert Platt, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems (RSS)*, June 2010.

[21] Sam Prentice and Nicholas Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research*, 28(11-12), October 2009.

[22] Shridhar K. Shah, Chetan D. Pahlajani, Nicholaus A. Lacock, and Herbert G. Tanner. Stochastic receding horizon control for robots with probabilistic state constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[23] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations Research*, 21(5):1071–1088, 1973.

[24] T. Smith and R. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *Proceedings of Uncertainty in Artificial Intelligence*, 2005.

[25] M. Spaan and N. Vlassis. Perseus: Randomized point-based vallue iteration for pomdps. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[26] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[27] Noel Du Toit and Joel W. Burdick. Robotic motion planning in dynamic, cluttered, uncertain environments. In *ICRA*, May 2010.

[28] Jur van den Berg, Pieter Abbeel, and Ken Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *International Journal of Robotics Research*, 30(7):895–913, 2011.

[29] Steven A Wilmarth, Nancy M Amato, and Peter F Stiller. MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1024–1031, 1999.