FEEDBACK-BASED INFORMATION ROADMAP (FIRM): GRAPH-BASED ESTIMATION AND CONTROL OF ROBOTIC SYSTEMS UNDER UNCERTAINTY

A Dissertation

by

ALIAKBAR AGHAMOHAMMADI

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Nancy M. Amato
Co-Chair of Committee,	Suman Chakravorty
Committee Members,	Ricardo Gutierrez-Osuna
	John Junkins
	P. R. Kumar
	Dylan Shell
Head of Department,	Nancy M. Amato

May 2014

Major Subject: Computer Engineering

ABSTRACT

This dissertation addresses the problem of stochastic optimal control with imperfect measurements. The main application of interest is robot motion planning under uncertainty. In the presence of process uncertainty and imperfect measurements, the system's state is unknown and a state estimation module is required to provide the information-state (belief), which is the probability distribution function (pdf) over all possible states. Accordingly, successful robot operation in such a setting requires reasoning about the evolution of information-state and its quality in future time steps. In its most general form, this is modeled as a Partially-Observable Markov Decision Process (POMDP) problem. Unfortunately, however, the exact solution of this problem over continuous spaces in the presence of constraints is computationally intractable. Correspondingly, state-of-the-art methods that provide approximate solutions are limited to problems with short horizons and small domains. The main challenge for these problems is the exponential growth of the search tree in the information space, as well as the dependency of the entire search tree on the initial belief.

Inspired by sampling-based (roadmap-based) methods, this dissertation proposes a method to construct a "graph" in information space, called Feedback-based Information RoadMap (FIRM). Each FIRM node is a probability distribution and each FIRM edge is a local controller. The concept of belief stabilizers is introduced as a way to steer the current belief toward FIRM nodes and induce belief reachability. The solution provided by the FIRM framework is a feedback law over the information space, which is obtained by switching among locally distributed feedback controllers.

Exploiting such a graph in planning, the intractable POMDP problem over con-

tinuous spaces is reduced to a tractable MDP (Markov Decision Process) problem over the graph (FIRM) nodes. FIRM is the first graph generated in the information space that preserves the principle of optimality, i.e., the costs associated with different edges of FIRM are independent of each other. Unlike the forward search methods on tree-structures, the plans produced by FIRM are independent of the initial belief (i.e., plans are query-independent). As a result, they are robust and reliable. They are robust in the sense that if the system's belief deviates from the planned belief, then replanning is feasible in real-time, as the computed solution is a feedback over the entire belief graph. Computed plans are reliable in the sense that the probability of violating constraints (e.g., hitting obstacles) can be seamlessly incorporated into the planning law. Moreover, FIRM is a scalable framework, as the computational complexity of its construction is linear in the size of underlying graph as opposed to state-of-the-art methods whose complexity is exponential in the size of underlying graph.

In addition to the abstract framework, we present concrete FIRM instantiations for three main classes of robotic systems: holonomic, nonholonomic, and non-point-stabilizable. The abstract framework opens new avenues for extending FIRM to a broader class of systems that are not considered in this dissertation. This includes systems with discrete dynamics or in general systems that are not well-linearizable, systems with non-Gaussian distributions, and systems with unobservable modes. In addition to the abstract framework and concrete instantiations of it, we propose a formal technique for replanning with FIRM based on a rollout-policy algorithm to handle changes in the environment as well as discrepancies between actual and computational models. We demonstrate the performance of the proposed motion planning method on different robotic systems, both in simulation and on physical systems. In the problems we consider, the system is subject to motion and sensing

noise. Our results demonstrate a significant advance over existing approaches for motion planning in information space. We believe the proposed framework takes an important step toward making information space planners applicable to real world robotic applications.

DEDICATION

To my lovely wife, Negar

To my lovely parents, Faride and Gholamreza

ACKNOWLEDGEMENTS

First, I would like to thank my co-advisors, Prof. Suman Chakravorty and Prof. Nancy Amato. Suman has been an excellent teacher, mentor, and trustworthy friend. I am indebted to his constant support and great mentorship. Dr. Amato has taught me many skills throughout these years, and I am thankful for her as an advisor.

I would also like to thank my committee members, professors Dylan Shell, P.R. Kumar, John Junkins, and Ricardo Gutierrez-Osuna for their time, guidance, and precious advice during the PhD process. In addition to my committee members, my research has significantly benefited from discussions with professors John Valasek, Wolfgang Bangerth, Jean-Francois Chamberland, Dezhen Song, and Roger Smith. I am thankful to them for their constructive and insightful feedback on my work. I would also like to specially thank Dr. Ben Zoghi for his constant support, kindness, and advice to me and my wife during these years.

I am grateful to Anshu Narang who gave me a lot of precious advice during my PhD. I would like to thank Mrinal Kumar for sharing his expertise in uncertainty quantification (in particular on methods to solve the Fokker-Plank-Kolmogorov Equation), and Abhishek Halder for countless discussion on many different research topics.

I was lucky to have so many great friends through these years. Among them, I would like to specially thank and express my gratitude to Andy Giese and Jory Denny, who helped me in many different ways, in particular when I was getting stuck in local minima! They have been true friends to me in every sense of the word. I would also like to thank my great old friend Amir-hossein Tamjidi for his constant support. His integrity, calm demeanor, and soft speech have always been an inspiration for me. Many thanks to Saurav Agarwal, Aditya Mahadevan,

and Daniel Tomkins for the great moments we had together and for their help in implementing the FIRM method on physical robots. I would like to thank Alireza Alemi, Sandip Kumar, Reza Ghasemi, Roozbeh Denshvar, Lantao Liu who have contributed to make these years very enjoyable; these friendships are a precious legacy of my experience at Texas A&M.

I would like to thank all other friends and colleagues at Texas A&M University who made these years memorable, including: Neha Satak, Amirali Fayyaz, Babak mousakhani, Amin Hassanzadeh, Mani Zandifar, Elham Khabiri, Adam Fidel, Cindy Yeh, Huei-Fang Yang, Moein Sudavi, Mohammad-hossein Rafiei, Ahmad Bani-Yunes, Roshmik Saha, Faryaneh Poursardar, Dan Yu, Karan Bagadiya, Ammar Abbas, Ajinkya Jain, Tarek Elgohary, Shuvra Nath, Chinwe Ekenna, Mukulika Ghosh, Sam Jacobs, Antal Buss, Shishir Sharma, Olivier Rojo, Nicolas Castet, Harshvardhan, Shawna Thomas, Troy McMahon, Juan Burgos, Samuel Rodriguez, Nathan Thomas, Arnaud Marfaing, Zhijia Xu, Thibaut Preval, Timmie Smith, Yuriy Solodkyy, Ioannis Papadopoulos, Vincent Marsy, and all my friends in the Dynamics and Control group as well as the Parasol Laboratory.

I would also like to thank members of the Department of Aerospace Engineering, in particular Karen Knabe for her constant help, and members of the Department of Computer Science and Engineering, including Kay Jones and David Ramirez.

I am grateful to my mother Faride and to my father Gholamreza for their dedication and unconditional love as well as my lovely sister Mozhdeh; it has been hard to stay apart from each other.

Finally, and most importantly, I would like to thank my best friend and my wife, Negar, for her love, support, and all the sacrifices she has made in these rewarding but stressful years. She was always there for me and thanks to her I never felt alone. I owe her everything I have attained.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Motion planning under motion and sensing uncertainty is an instance of the stochastic optimal control problem with imperfect measurements, or in general the problem of sequential decision making under uncertainty. Many real-world problems can be modeled this way, ranging from robot motion planning under uncertainty [44], to medical needle steering for minimally invasive surgery [97], to assistive technologies for persons with dementia [41], and even to setting rules for conservation of threatened species [24].

In particular, sequential decision making under uncertainty plays an important role in robotic systems and it is a crucial capability for performing many tasks. The main sources of uncertainty in robotic systems arise from uncertainty in determining the robot's motion as well as uncertainty in sensory readings. Under these uncertainties, a state estimation module can provide a probability distribution over the possible states of the system, and therefore decision making is performed in the space of these distributions, the so called information space or belief space. Planning in the belief space in its most general form is formulated as a Partially Observable Markov Decision Process (POMDP) problem [12, 44, 86].

Due to the complexity of solving POMDP problems [59, 74], only a small class of problems can be solved exactly using a POMDP formulation. In particular, planning (i.e., solving POMDPs) over continuous state, control, and observation spaces is a challenge. In the presence of state and control constraints, the problem is even more difficult. Different approximation schemes have been proposed in the literature to provide approximate solutions to this problem. State-of-the-art methods mostly rely on forward search in the belief space [13, 54, 77, 79, 81, 82, 91, 94]. However, at

every step, each possible action-observation pair leads to a new belief and therefore the search tree grows exponentially and soon becomes intractable. Moreover, this search tree is only valid for a given initial belief, and in the case of large deviations in the belief state, or when starting from a new belief, the entire search tree must be reconstructed. These two limitations have limited the scalability of the forward search methods.

In this research, we propose a graph-based framework for handling the "constrained POMDP" problem, while preserving the closed-loop (feedback) structure of the solution. The method generates a representative graph in the belief space, whose nodes are certain probability distributions and whose edges are feedback controllers to take the system's belief form one node to another. The framework is a principled way of reducing the general "constrained POMDP" problem into a computationally tractable MDP problem, while preserving important properties of the solution of the original POMDP, such as robustness (feedback solution) and reliability (ability to incorporate accurate failure probabilities). Thus, the obtained policy can be executed online.

Next, we provide a simple example of such a graph pictorially and compare it with traditional forward search methods.

Example 1. *Figure 1.1 illustrates the main idea of generating the graph in belief space and coping with exponential growth of the search tree. Figure 1.1(a) depicts a simple graph in the state space. Figure 1.1(b) shows the evolution of a Gaussian belief b (with mean $\widehat{x}^+$ and covariance P) on the this graph from the left-most node to the right-most node. As is seen in this figure, the search tree in the belief space grows exponentially. For example, although there exists a single edge $e^{(10,11)}$ between nodes $\mathbf{v}^{10}$ and $\mathbf{v}^{11}$ in the state space graph (Fig. 1.1(a)), the belief evolution along $e^{(10,11)}$*

is not unique (Fig. 1.1(b-c)) since it depends on (i) the initial belief, (ii) obtained observations (observation history), and (iii) the taken path (action history) that has led to $\mathbf{v}^{10}$*. Figure 1.1(c) shows the corresponding random search tree in belief space. In FIRM, however, associated with each PRM node we have a predefined unique belief that is reachable independent of the initial belief. Such a reachability is ensured by using appropriate local feedback controllers (Fig. 1.1(d-e)).*

FIRM nodes are probability distributions that are reachable independent of the system's initial belief. As a result, the edges of the FIRM graph are independent of each other. This construct breaks the curse of history in POMDPs, allowing us to construct a graph in the belief space with independent edge costs. Therefore, in contrast to the main body of the literature in motion planning under uncertainty, FIRM can be re-used for future queries and does not need to reconstruct the graph every time a new query is submitted.

From an algorithmic perspective, this edge independence is an example of the optimal substructure property. A problem has an optimal substructure only if the optimal solution can be obtained from a combination of optimal solutions to its subproblems [31]. To solve a problem using Dynamic Programming (DP) or its successive approximation schemes such as Dijkstra's algorithm, the optimal substructure assumption has to hold [87], i.e., the cost of any sub-path has to be independent of what precedes it and what succeeds it. As mentioned, the direct transformation of sampling-based methods to belief space does not satisfy this assumption, while FIRM preserves it. Furthermore, edge independence allows the challenging task of computing collision probabilities to be done offline, separately for each edge, without performing costly computations repeatedly and without any simplifying assumptions.

To demonstrate the practical utility of the FIRM framework, we propose different

concrete instantiations of the abstract framework and apply them to a variety of robotic systems such as holonomic robots [2, 8], nonholonomic robots [6], systems with kinodynamical constraints (e.g., non-point-stabilizable systems), and robotic manipulators [7,8], in simulation. We apply the method to a physical mobile robotic system as well and investigate the robustness and performance of the method in a real-world setting [1]. In these experiments, we show how the method can incorporate the available information in the environment as well as constraints into the planning law. Online replanning using traditional forward search methods is a challenging task due to the expensive computations involved in reproducing the forward search tree. However, in the experiments with physical robots, we show how the method is capable of performing online replanning in the case of large deviations or unexpected changes in the environment such as unknown and changing obstacles.

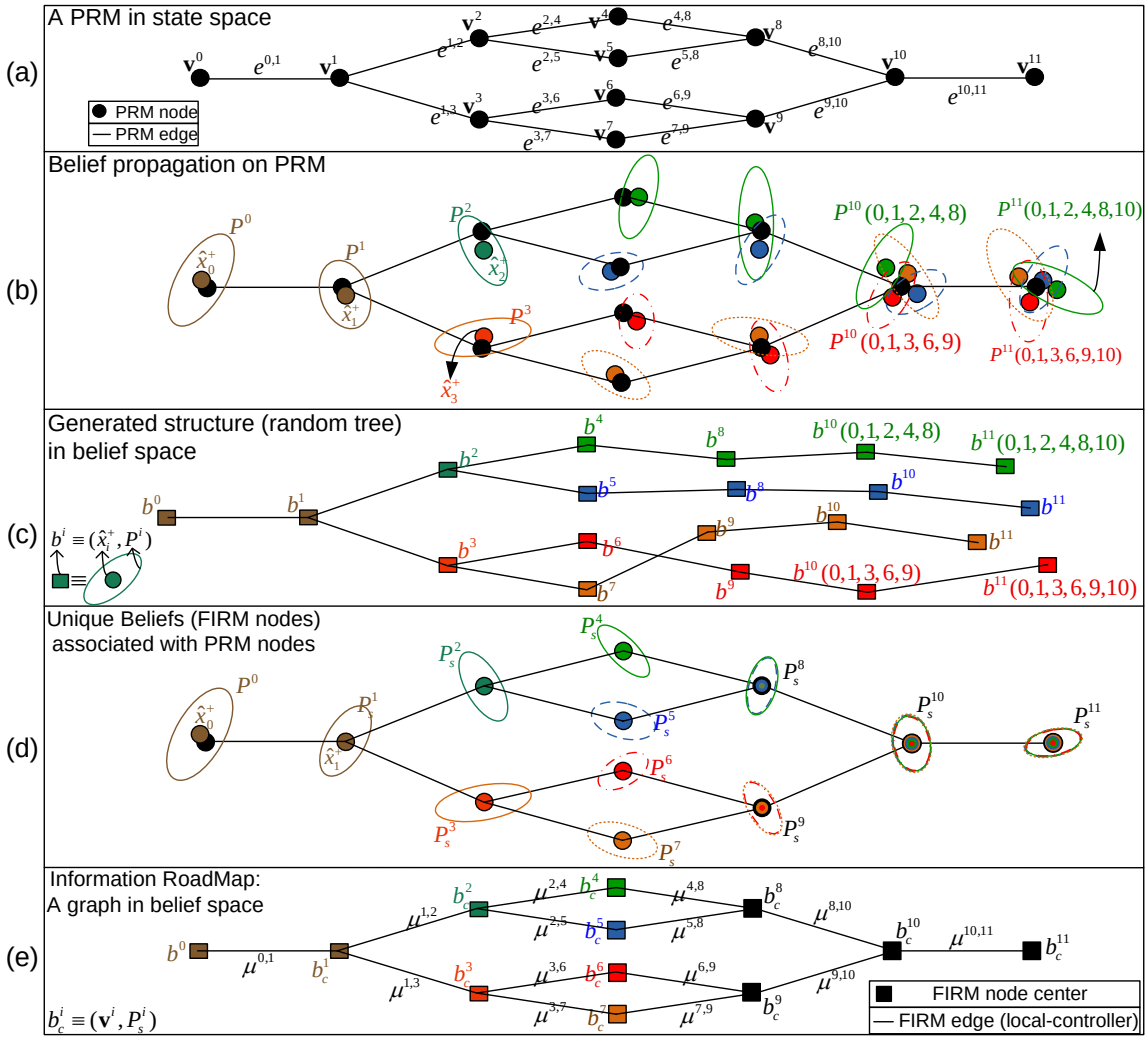


Figure 1.1: (a) A simple PRM in state-space. (b) Assuming Gaussian belief space, belief snapshots along different paths starting from $\mathbf{v}^0$ ending at $\mathbf{v}^{11}$ are shown. As shown here, the obtained belief depends on the path traveled by the robot. For example $P^{11}(0,1,3,6,9,10)$ denotes the estimation covariance at node $\mathbf{v}^{11}$, when the robot has traversed a path through nodes $(0,1,3,6,9,10)$ prior to node 11. (c) Corresponding belief paths in belief space. The belief at each node depends on the initial belief, taken actions (edges), and obtained observations (random). Therefore, the generated structure in the belief space is not a graph but a random tree. (d) Unique beliefs assigned to each PRM node. Using stabilizers, regardless of the action and observation history, the belief at each node reaches these predefined beliefs. (e) The FIRM corresponding to the given PRM. b_c^i's are graph nodes in belief space and μ^{ij}'s are local planners (graph edges).

1.1 Research Contributions

This dissertation provides a multi-query roadmap-based (graph-based) solution for the belief space planning problem as the first counterpart to the celebrated Probabilistic Roadmap Methods (PRMs) for the deterministic state space planning problem. The main application of interest is motion planning for robotic systems. We refer to the proposed graph-based framework for planning under uncertainty as the Feedback-based Information RoadMap (FIRM). This dissertation makes theoretical and practical contributions. The main theoretical highlights are noted below:

- *Graph in belief space*: We describe how the constrained POMDP problem can be reduced to a tractable MDP problem on the FIRM graph. FIRM is the first framework that generates a graph in belief space, or more precisely, a multi-query graph in belief space. This graph breaks the curse of history in POMDPs [76] by making the belief evolution along each edge independent of the belief evolution on the rest of the graph.

- *Belief reachability*: This research proposes different techniques based on feedback controllers to address the task of sampling belief nodes. Under the Gaussian assumption, we characterize the set of beliefs that are reachable under adopted feedback controllers, independent of the initial belief. We show that sampling these beliefs (Fig. 1.1), the controller can drive the belief into the ϵ-neighborhood of these sampled beliefs in finite time and hence ensure reachability of the FIRM nodes. Accordingly, different instantiations of the FIRM framework are proposed for three main classes of robotic systems: holonomic, nonholonomic, and non-point stabilizable systems.

- *Performance and completeness guarantees*: An important property of the FIRM

framework is that we can analytically characterize the success probability of the computed plan using FIRM offline. Therefore, one can increase the number of nodes to increase the success probability offline. Subsequently, we introduce the concept of "probabilistic completeness under uncertainty (PCUU)" for belief space planners, provide tools that can be used in analyzing belief space planners, and show that FIRM is a probabilistically complete algorithm under uncertainty.

- *Scalability (linear construction cost)*: As mentioned, belief space planners usually have an exponential planning complexity either in the number of nodes (if they are sampling-based methods) or in the size of grid (if they rely on discretizing the state space). However, the complexity of the FIRM construction is linear in the size of the underlying graph. As a result it can handle planning problems with much larger domains.

Equally important, this research offers a set of practical contributions, which we believe provides an important step toward utilizing POMDPs as a practical tool for robot motion planning under uncertainty. The main practical highlights and properties of FIRM are summarized below:

- *Efficient planning and Reliability (incorporating constraints in planning)*: The construction of FIRM is offline and thus online planning (and replanning) is very efficient. In addition, owing to its offline construction, in FIRM, accurate collision probabilities can be computed and incorporated in the planning law offline, which leads to more reliable plans compared to traditional forward search methods, where simplified collision measures are usually adopted due to the exponential growth of the search tree.

- *Robustness and online replanning*: In FIRM, the computed solution is a *feedback* policy. As a result, in case of large deviations, either no replanning is needed or just a computationally efficient local online replanning is needed, and the feedback over the belief space can compensate for the deviations. Therefore, the method is robust to large deviations. Also, a formal framework based on a rollout policy [15] is proposed to realize efficient local replanning with FIRM, which can also cope with changes in the environment.

This research has been reported in a number of publications. The abstract FIRM framework and a concrete FIRM method for holonomic systems are reported in [2,8]. Probabilistic completeness results are reported in [4, 8]. A concrete FIRM method for nonholonomic systems is reported in [6] and results on unifying the perfect and imperfect state information cases are reported in [7, 9]. Also, results on dynamic online replanning in belief space are reported in [1].

1.2 Dissertation Outline

In Chapter 2, we present the necessary background information on belief space planning. In Chapter 3, we review the literature related to our research and place our work in this context. The abstract FIRM framework is introduced in Chapter 4, where we detail the reduction of the POMDP problem to the MDP problem on the FIRM graph. We also introduce the concept of probabilistic completeness under uncertainty in this chapter. In Chapters 5, 6, and 7 we propose concrete FIRM frameworks to respectively handle three main classes of robotic systems: holonomic, nonholonomic, and non-point-stabilizable systems. We demonstrate the performance of these methods on a set of robotic systems in simulation. We also provide complexity analyses and comparisons with state-of-the-art methods. In Chapter 8, we apply the FIRM framework to a physical robotic system, where we propose a formal

replanning scheme to cope with large deviations, changes in the environment map, and discrepancies between computational and physical models. Chapter 9 concludes the dissertation and discusses future research directions.

2. PRELIMINARIES: FORMULATING THE PROBLEM OF MOTION PLANNING UNDER UNCERTAINTY

The problem of motion planning under motion and sensing uncertainty is an instance of the sequential decision making problem in belief space. In the first part of this chapter, we present necessary background information on belief space planning (i.e., the stochastic control problem with imperfect measurements), including (i) a definition of belief, (ii) a brief overview of recursive state estimation, (iii) a definition of the problem of planning under uncertainty, (iv) the corresponding formulation of dynamic programming, and (v) a definition of the constrained POMDP problem. In the second part of this chapter, we review a few unconstrained POMDP problems for which the analytical solution exist.

2.1 Problem of Motion Planning Under Uncertainty

The main sources of uncertainty in motion planning are the lack of exact knowledge of the robot's motion model, of the robot's sensing model, and of the environment model, which are referred to respectively as motion uncertainty, sensing uncertainty, and map uncertainty. In this research, we focus on motion and sensing uncertainty, but some of the concepts are extensible to problems with map uncertainty. The problem of motion planning under motion uncertainty is an instance of the stochastic optimal control problem with perfect state information, which is also known as the Markov Decision Process (MDP) problem. The problem of motion planning under both motion and sensing uncertainty is an instance of the stochastic optimal control problem with imperfect state information, which is known as the Partially-Observable Markov Decision Process (POMDP) problem. Figure 2.1 shows the different elements in a POMDP problem.

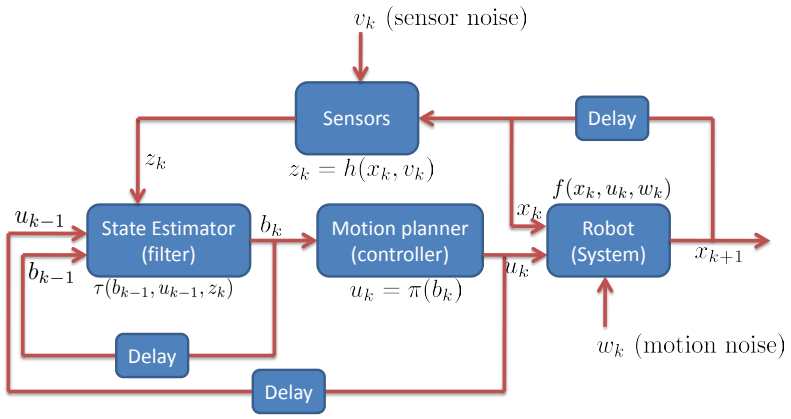


Figure 2.1: Block diagram corresponding to the problem of planning under motion and sensing uncertainty; i.e., the POMDP problem.

In the deterministic setting, we seek a path (or the optimal path) as the solution of motion planning problem. However, in the stochastic setting, we seek a feedback law (or the optimal feedback law) π as the solution of the motion planning problem. In the case of an MDP, π is a mapping from the state space to the control space, while in the case of a POMDP, π is a mapping from the belief space to the control space (see Fig. 2.1). In the rest of dissertation, we focus on POMDPs.

2.1.1 Stochastic Optimal Control with Imperfect Measurements

As previously mentioned, the POMDP formulation is the most general formulation for the planning problem under process (motion) uncertainty and imperfect state information (sensing uncertainty). POMDPs were introduced in [12, 44, 86]. In the following, we discuss elements of the POMDP problem, and then present a POMDP formulation that is known as the belief MDP problem [15, 50, 90].

The system state describes the system at every time step. Let us denote the state of the system at the k-th time step by x_k. We denote the space of all possible states by $\mathbb{X}$. It is worth noting that the state space in our problem can be continuous

(uncountable).

We denote the action (or control) at time step k, by u_k. The set $\mathbb{U}$ is the control space (possibly continuous) of the problem, containing all possible control inputs, $u \in \mathbb{U}$. Also, $u_{0:k} := \{u_0, u_1, \cdots, u_k\}$ denotes the control history up to step k.

The motion model f describes how the state of the system evolves over time based on the applied action and the motion noise. Thus, a generic motion model can be defined as:

$$x_{k+1} = f(x_k, u_k, w_k), \quad w_k \sim p(w_k|x_k, u_k) \tag{2.1}$$

where w_k is the process (motion) noise at time step k, distributed according to a conditional probability distribution denoted by $p(w_k|x_k, u_k)$. Another conventional way of representing the motion model is based on a transition pdf (probability distribution function), where $p(x'|x, u) : \mathbb{X} \times \mathbb{U} \times \mathbb{X} \to \mathbb{R}_{\geq 0}$ is the state transition pdf that encodes the probability density of the transition from state x to state x' under the control u.

In the presence of *imperfect* state measurements, we do not have access to the system state x. Instead, we obtain noisy measurements of the state through sensors. Let us denote the measurement (or observation) at time step k by z_k. $\mathbb{Z}$ is the observation space of the problem, containing all possible observations, $z \in \mathbb{Z}$. Also, $z_{0:k} := \{z_0, z_1, \cdots, z_k\}$ denotes the observation history up to step k.

The observation model h describes how the sensors sense the state by providing a mapping from the state space to the observation space as:

$$z_k = h(x_k, v_k), \quad v_k \sim p(v_k|x_k) \tag{2.2}$$

where v_k is the observation noise at time step k, which is distributed according to a conditional probability distribution denoted by $p(v_k|x_k)$. Another conventional way of representing the observation model is based on the likelihood pdf, where $p(z|x) : \mathbb{X} \times \mathbb{Z} \to \mathbb{R}_{\geq 0}$ is the observation pdf conditioned on the system's state.

In the absence of exact state, the available data for decision making at time step k is the observation history $z_{0:k}$ and control history $u_{0:k-1}$, which is denoted by $\mathcal{H}_k = \{z_{0:k}, u_{0:k-1}\}$. Thus, the policy at time step k, is a history dependent policy, denoted by $\pi^{hist} : \mathbb{Z}^{k+1} \times \mathbb{U}^k \to \mathbb{U}$ that maps the data history $\mathcal{H}_k$ to the action u_k, i.e., $u_k = \pi^{hist}(\mathcal{H}_k)$. Let us denote the space of all history-dependent policies by Π^{hist}.

The function $c : \mathbb{X} \times \mathbb{U} \to \mathbb{R}_{\geq 0}$ models the one-step cost of the problem based on the application. In other words, $c(x, u)$ denotes the cost of taking action u at state x. In the motion planning problem, the incurred cost is zero if the system reaches a stopping region, i.e., reaches the goal region or hits an obstacle. Otherwise, it is a non-zero value to prevent the system from falling into infinite loops or stopping at some point before reaching a stopping region.

To choose a policy in Π^{hist}, we define the cost-to-go function $J^{\pi^{hist}}(x_0) : \mathbb{X} \to \mathbb{R}_{\geq 0}$ that models the cost-to-go from every state x_0 under a given policy π^{hist}. Then we pick the policy π^{hist^*} that minimizes the defined cost-to-go for every initial state x_0. Assuming a stage-additive cost structure, starting from x_0, the cost-to-go would be:

$$
\begin{aligned}
J^{\pi^{hist}}(x_0) &= \sum_{k=0}^{\infty} \mathbb{E}\left[c(x_k, \pi^{hist}(\mathcal{H}_k))\right] \\
s.t. \quad & x_{k+1} = f(x_k, \pi^{hist}(\mathcal{H}_k), w_k), \quad w_k \sim p(w_k|x_k, u_k) \\
& z_k = h(x_k, v_k), \quad v_k \sim p(v_k|x_k) \\
& \mathcal{H}_k = \{z_{0:k}, u_{0:k-1}\}, \quad u_k = \pi^{hist}(\mathcal{H}_k)
\end{aligned}
$$

Note that in general π^{hist} can be a time-varying policy, but to simplify the notation we do not show its dependency on time.

Now, we can define the optimal cost-to-go from every state x_0 as the $J^*(x_0) : \mathbb{X} \rightarrow \mathbb{R}_{\geq 0}$. Accordingly, we can formally define the problem of stochastic control under process and measurement noise (i.e., the POMDP problem) as the problem of finding the optimal policy that leads to the optimal cost-to-go, i.e.,

$$\begin{aligned} J^*(x_0) &= \min_{\pi^{hist} \in \Pi^{hist}} J^{\pi^{hist}}(x_0) \\ \pi^{hist*} &= \arg \min_{\pi^{hist} \in \Pi^{hist}} J^{\pi^{hist}}(x_0) \end{aligned} \tag{2.3}$$

The history-dependent policy is a complex function as it depends on the data history which grows over time. So, as the first step toward simplifying the POMDP problem, we use state estimation techniques to reduce the data history into a more compressed representation.

2.1.2 Recursive State Estimation

As discussed above, policy $\pi^{hist}(\cdot)$ is a function that returns an action (control) u for a given data history $\mathcal{H}$. However, the data in a given history $\mathcal{H}$ can be compressed and represented as a probability distribution function (pdf) over all possible states. It can be shown that such a probability distribution retains all the information needed for decision making, and hence it is called the information-state or belief of the system [15, 50].

The information-state or belief is defined as the probability distribution over all possible states conditioned on the available data $\mathcal{H}$, i.e., conditioned on all taken actions and obtained measurements. $b_k : \mathbb{X} \times \mathbb{Z}^k \times \mathbb{U}^{k-1} \rightarrow \mathbb{R}_{\geq 0}$ denotes the belief at

the k-th step, which is formally defined as $b_k := p(x_k|\mathcal{H}_k) = p(x_k|z_{0:k}; u_{0:k-1})$. Notice that to simplify the expressions, we show the belief by b_k, but, rigorously speaking, since it is a function of x_k and $\mathcal{H}_k$, it should be written as $b_k(x_k, \mathcal{H}_k)$. The space of all possible beliefs is called belief space and denoted by $\mathbb{B}$.

In batch estimation, at each step the belief b_k is computed as a function of history, denoted by $b_k = \overline{\tau}(\mathcal{H}_k)$. In contrast, using recursive state estimation techniques, belief can be computed recursively at every time step. The dynamics introduced by this recursion are shown by function $\tau : \mathbb{B} \times \mathbb{U} \times \mathbb{Z} \to \mathbb{B}$, that computes the next belief based on the last action and current observation $b_{k+1} = \tau(b_k, u_k, z_{k+1})$. These dynamics can be derived using Bayes rule and the law of total probability [15], [90] as follows:

$$
\begin{aligned}
b_{k+1} &= p(x_{k+1}|\mathcal{H}_{k+1}) = p(x_{k+1}|\mathcal{H}_k \cup \{z_{k+1}, u_k\}) \\
&= \frac{p(z_{k+1}|x_{k+1}, \mathcal{H}_k, u_k)p(x_{k+1}|\mathcal{H}_k, u_k)}{p(z_{k+1}|\mathcal{H}_k, u_k)} \\
&= \frac{p(z_{k+1}|x_{k+1})p(x_{k+1}|\mathcal{H}_k, u_k)}{p(z_{k+1}|\mathcal{H}_k, u_k)} \\
&= \frac{p(z_{k+1}|x_{k+1}) \int_{\mathbb{X}} p(x_{k+1}|x_k, u_k)p(x_k|\mathcal{H}_k, u_k)dx_k}{p(z_{k+1}|\mathcal{H}_k, u_k)} \\
&= \frac{p(z_{k+1}|x_{k+1}) \int_{\mathbb{X}} p(x_{k+1}|x_k, u_k)p(x_k|\mathcal{H}_k)dx_k}{p(z_{k+1}|\mathcal{H}_k, u_k)} \\
&= \frac{p(z_{k+1}|x_{k+1}) \int_{\mathbb{X}} p(x_{k+1}|x_k, u_k)b_k dx_k}{p(z_{k+1}|\mathcal{H}_k, u_k)} \\
&=: \tau(b_k, u_k, z_{k+1})
\end{aligned}
\tag{2.4}
$$

Usually, the function τ is composed of two functions, a prediction function τ_{pred}

and an update function τ_{up}; i.e. $\tau = \tau_{up} \circ \tau_{pred}$:

$$b_{k+1} = \tau(b_k, u_k, z_{k+1}) = \tau_{up}(\tau_{pred}(b_k, u_k), z_{k+1})$$

where, τ_{pred} predicts the next belief based on the taken action:

$$b^-_{k+1} := \tau_{pred}(b_k, u_k) := \int_{\mathbb{X}} p(x_{k+1}|x_k, u_k) b_k dx_k \tag{2.5}$$

and τ_{up} updates the predicted belief based on the obtained measurements z_{k+1}:

$$b_{k+1} = \tau_{up}(b^-_{k+1}, z_{k+1}) := \frac{p(z_{k+1}|x_{k+1}) b^-_{k+1}}{p(z_{k+1}|\mathcal{H}_k, u_k)} \tag{2.6}$$

b^-_k is called the prior distribution (or predicted belief) and b_k is called the posterior distribution (or updated) belief, which is also shown by b^+_k. In this dissertation, by belief we mean the posterior belief and we usually do not write the superscript "+".

Another conventional way of representing a belief dynamics model is based on the transition pdf, where $p(b'|b, u) : \mathbb{B} \times \mathbb{U} \times \mathbb{B} \to \mathbb{R}_{\geq 0}$ is the belief transition pdf that encodes the probability density of a transition from belief b to belief b' under the control u, which is an equivalent way of representing $b_{k+1} = \tau(b_k, u_k, z_{k+1})$.

Now that the data history $\mathcal{H}_k$ is compressed to the distribution b_k, we can perform decision making based on the belief rather than the data history. It is worth noting that the solution of the POMDP problem is a history-dependent policy π^{hist}. However, instead, one can separately design a "filter" and a "separated policy", where filter returns the belief based on the available data history, and the separated policy

π returns the action based on the belief [50].

$$\pi^{hist} = \pi \circ \overline{\tau} \Rightarrow \pi^{hist}(\mathcal{H}_k) = \pi(\overline{\tau}(\mathcal{H}_k)) \tag{2.7}$$

Therefore, the separated policy is a mapping from the belief space to the control space: $\pi : \mathbb{B} \to \mathbb{U}$ that returns the action based on the belief $u_k = \pi(b_k)$. In this dissertation by "policy" we mean the "separated policy".

2.1.3 Belief MDP Formulation

Now, we can reformulate the POMDP problem in (2.3) in the belief space by defining the corresponding costs in the belief space.

We denote the one-step cost in belief space with the same function name c. Indeed, we overload the function $c : \mathbb{X} \times \mathbb{U} \to \mathbb{R}_{\geq 0}$ with the function $c : \mathbb{B} \times \mathbb{U} \to \mathbb{R}_{\geq 0}$, defined as follows:

$$c(b, u) = \mathbb{E}[c(x, u)|\mathcal{H}] = \int_{\mathbb{X}} c(x, u)p(x|\mathcal{H})dx \tag{2.8}$$

which models the one step-cost of the problem in the belief space. In other words, $c(b, u)$ denotes the cost of taking action u at the belief b.

In partially-observable environments, the goal region is defined in the belief space, since the state of the system is unknown. Thus, if we denote the goal region by B_{goal}, the incurred cost is zero if the system reaches B_{goal}, i.e., $c(b, u) = 0$ if $b \in B_{goal}$. We discuss the reason for this assignment and discuss other conditions on the one-step cost shortly.

To choose a policy in Π, we define the cost-to-go from every belief and then pick the policy π^* that minimizes the defined cost-to-go. Assuming a stage-additive cost structure, starting from b_0 and using the policy π, the cost-to-go (or value) function

$J^{\pi}(\cdot) : \mathbb{B} \to \mathbb{R}$ is formally defined as:

$$J^{\pi}(b_0) = \sum_{k=0}^{\infty} \mathbb{E}\left[c(b_k, \pi(b_k))\right]$$
$$s.t. \quad b_{k+1} = \tau(b_k, \pi(b_k), z_k), \quad z_k \sim p(z_k|x_k) \tag{2.9}$$

Again, note that in general the policy π is a time-varying policy, but to simplify the notation we do not show its dependency on time.

When a policy is executed, there is a stopping condition that stops the execution. In the motion planning problem under uncertainty (also known as the stochastic shortest path problem), the system stops if it reaches the goal region in the belief space B_{goal}. For the system to not fall into an infinite cycle or to not stop before the termination condition is satisfied, the cost of taking any action before the stopping condition is satisfied has to be positive. It also has to be zero when the stopping condition is met:

$$\begin{cases} c(b, u) = 0 & \text{if } \ b \in B_{goal} \\ c(b, u) \geq \epsilon > 0 & \text{if } \ b \notin B_{goal} \end{cases} \tag{2.10}$$

which results in a zero cost-to-go from the goal region, i.e., $J^{\pi}(b) = 0$, for all $b \in B_{goal}$.

Belief MDP problem is just a reformulation of the original POMDP problem as an MDP problem defined over the belief space [15, 50, 90]. In the belief MDP problem we seek for the best separated policy that minimizes the cost-to-go function from every belief in the belief space. Formally, if we denote the *optimal* cost-to-go function by $J(\cdot)$, the belief MDP problem is the problem of finding the *optimal* policy

$\pi^*(\cdot) : \mathbb{B} \to \mathbb{U}$, which attains the minimum cost-to-go as follows:

$$J(\cdot) = \min_{\Pi} J^{\pi}(\cdot)$$
$$\pi^* = \arg\min_{\Pi} J^{\pi}(\cdot) \tag{2.11}$$

2.1.4 Dynamic Programming

Dynamic Programming (DP) is a formulation for solving the belief MDP problem in (2.11). It simplifies the problem by converting the minimization over the function space (policy space) Π into a minimization problem over the control space $\mathbb{U}$. Dynamic programming is based on the simple idea of the "principle of optimality", mainly introduced by Bellman [14].

A problem is said to satisfy the Principle of Optimality if the sub-solutions of an optimal solution of the problem are themselves optimal solutions for their sub-problem. In the present context this can be paraphrased as following: if the policy $\pi^*(\cdot)$ is the optimal policy starting from an initial state, it must be optimal from any other intermediate state that can be reached from the initial state, under π^*. From an algorithmic point of view, a problem that has this property is said to have "optimal substructure property" [31].

The intuition behind the *Principle of Optimality* is simple. If we start from x (or b in belief space) and reach the state x' (or belief b') on the way to goal, we have the "sub-problem" of reaching the goal from x' (or b'). Due to the stage-additive structure of the cost-to-go, the policy has to be optimal from x' (or b') to the goal for it to be optimal from x (or b) to the goal. Dynamic programming is based on this idea.

The Dynamic Programming (DP) equation (also known as Bellman's equation) is a necessary condition for optimality associated with MDP problems. Relying on the principle of optimality, DP breaks the optimization problem over the policy spaces into optimization problems over the control space to choose the *immediate control* (next action) separately, setting aside all future decisions. For the belief MDP in (2.11), the DP formulation would then be:

$$J(b) = \min_u \{c(b,u) + \int_{\mathbb{B}} p(b'|b,u)J(b')db'\}, \quad \forall b \in \mathbb{B} \tag{2.12a}$$

$$\pi^*(b) = \arg\min_u \{c(b,u) + \int_{\mathbb{B}} p(b'|b,u)J(b')db'\}, \quad \forall b \in \mathbb{B} \tag{2.12b}$$

2.1.5 Constrained POMDP problem

In many applications, including motion planning, there are constraints that have to be imposed on the system state.

In the motion planning problem, obstacles are considered to be constraints in system's state space. We can partition the state space $\mathbb{X}$ into the free space $\mathbb{X}_{free}$ and the obstacle space $\mathbb{X}_{obst}$, such that $\mathbb{X}_{free} \cup \mathbb{X}_{obst} = \mathbb{X}$ and $\mathbb{X}_{free} \cap \mathbb{X}_{obst} = \emptyset$. We also define the set F as the failure set such that if the system state hits the set F, then the control policy is considered to fail. In many applications, the obstacle set is the same as the failure set, i.e., $F = \mathbb{X}_{obst}$. However, there may be applications in which the obstacle set is smaller than the failure set. That is, there exist other causes of failure such as exceeding some maximum operation time, violating actuator saturation limits, etc.

In general, denoting the constraints on the state space by the failure set F, we

can formulate the *constrained belief MDP* problem as follows:

$$\begin{aligned} \pi^*(\cdot) = \arg\min_{\pi \in \Pi} & \sum_{k=0}^{\infty} \mathbb{E}\left[c(b_k, \pi(b_k))\right] \\ s.t. \quad & b_{k+1} = \tau(b_k, \pi(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\ & x_k \notin F, \quad \forall k \end{aligned} \tag{2.13}$$

In the stochastic case, since satisfying $x_k \notin F$ can be difficult (or infeasible), it is relaxed to the "chance constraint" $\Pr(x_k \in F) \leq \delta$, for some small positive real number δ. Thus, we have:

$$\begin{aligned} \pi^*(\cdot) = \arg\min_{\pi \in \Pi} & \sum_{k=0}^{\infty} \mathbb{E}\left[c(b_k, \pi(b_k))\right] \\ s.t. \quad & b_{k+1} = \tau(b_k, \pi(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\ & \Pr(x_k \in F) \leq \delta, \quad \forall k \end{aligned} \tag{2.14}$$

In this dissertation, we propose a method to reduce the computationally intractable constrained POMDP in (2.14) to a tractable MDP in belief space. The method is called Feedback-based Information RoadMap (FIRM), which will be detailed in Chapter 4.

2.2 Linear Quadratic Gaussian (LQG) Controllers

In this section, we restrict our attention to the POMDP problems with quadratic costs for the class of linear systems with Gaussian noises. For this class of systems, the unconstrained POMDP problem can be solved analytically and leads to the celebrated LQG controllers. We will use the results of this section in the following chapters, when we *locally* linearize nonlinear systems.

2.2.1 Time-varying LQG Controller

A time-varying Linear Quadratic Gaussian (LQG) controller is often used to track a pre-planned trajectory (also called nominal, desired, or open-loop trajectory) in the presence of process and observation noise. In principal it is designed (and optimal) for linear systems with Gaussian noise, but it is also can be utilized for stabilizing nonlinear systems locally around the planned trajectory. An LQG controller is composed of a Kalman Filter (KF) as an estimator and a Linear Quadratic Regulator (LQR) as a controller. At every time step k, the KF provides the *a posteriori* distribution over the system state (belief) b_k, and LQR generates control u_k based on b_k.

In this section, we first discuss the system linearization and planned nominal trajectory, and then discuss the KF, LQR and LQG corresponding with this nominal trajectory. Consider the nonlinear partially-observable state-space equations of the system as follows:

$$x_{k+1} = f(x_k, u_k, w_k), \quad w_k \sim \mathcal{N}(0, Q_k) \tag{2.15a}$$

$$z_k = h(x_k, v_k), \quad v_k \sim \mathcal{N}(0, R_k) \tag{2.15b}$$

A planned nominal trajectory for this system is a sequence of planned states $(x_k^p)_{k\geq 0}$ and planned controls $(u_k^p)_{k\geq 0}$, such that it is consistent with the noiseless dynamics model, i.e., we have:

$$x_{k+1}^p = f(x_k^p, u_k^p, 0) \tag{2.16}$$

The planned trajectory can be a finite sequence of some length N. The role of a closed-loop stochastic controller, during the trajectory tracking execution, is to

compensate for the robot's deviations from the planned trajectory due to noise and keeping the robot close to the planned trajectory in the sense of minimizing following quadratic cost:

$$J = \mathbb{E}\left[\sum_{k\geq 0}(x_k - x_k^p)^T W_x (x_k - x_k^p) + (u_k - u_k^p)^T W_u (u_k - u_k^p)\right] \tag{2.17}$$

where W_x and W_u are the positive definite weight matrices for state and control cost, respectively.

Since the state space is not fully observable and it is only partially observable, we do not have access to the perfect value of the state x_k. Thus, we provide the estimate x_k^+ of the state x_k based on the available observations $z_{0:k}$ from the beginning up to the current time step. Then, based on separation principle [15], it can be shown that in linear systems with Gaussian noise, the above minimization in terms of the error $x_k - x_k^p$ is equivalent to performing two separate minimizations based on the estimation error $x_k - \widehat{x}_k^+$ and the controller error $\widehat{x}_k^+ - x_k^p$, whose summation is the same as the original main error $x_k - x_k^p = (x_k - \widehat{x}_k^+) + (\widehat{x}_k^+ - x_k^p)$, where $\widehat{x}_k^+ = \mathbb{E}_x[x_k^+] = \mathbb{E}_x[x_k|z_{0:k}]$. As a major consequence, the design of the stochastic controller with a partially-observable state space (LQG), reduces to designing a controller with fully-observable state (LQR) and designing an estimator (KF), separately. In the following, we first discuss the linearization of a nonlinear model and then we discuss how a KF and an LQR can be designed for this linearized system and finally combine them to construct a time-varying LQG controller.

Given a nominal trajectory $(x_k^p, u_k^p)_{k\geq 0}$, we linearize the dynamics and observation model in Eq. (2.15), as follows:

$$x_{k+1} = f(x_k^p, u_k^p, 0) + A_k(x_k - x_k^p) + B_k(u_k - u_k^p) + G_k w_k, \quad w_k \sim \mathcal{N}(0, Q_k) \tag{2.18a}$$

$$z_k = h(x_k^p, 0) + H_k(x_k - x_k^p) + M_k v_k, \quad v_k \sim \mathcal{N}(0, R_k) \tag{2.18b}$$

where

$$\begin{aligned} A_k &= \frac{\partial f}{\partial x}(x_k^p, u_k^p, 0), \quad B_k = \frac{\partial f}{\partial u}(x_k^p, u_k^p, 0), \quad G_k = \frac{\partial f}{\partial w}(x_k^p, u_k^p, 0), \\ H_k &= \frac{\partial h}{\partial x}(x_k^p, 0), \quad M_k = \frac{\partial h}{\partial v}(x_k^p, 0) \end{aligned} \tag{2.19}$$

Now, let us define the following errors:

- LQG error (main error): $e_k = x_k - x_k^p$
- KF error (estimation error): $\widetilde{e}_k = x_k - \widehat{x}_k^+$
- LQR error (estimation of LQG error): $\widehat{e}_k^+ = \widehat{x}_k^+ - x_k^p$

Note that these errors are linearly dependent: $e_k = \widehat{e}_k^+ + \widetilde{e}_k$. Also, defining $\delta u_k = u_k - u_k^p$ and $\delta z_k = z_k - z_k^p := z_k - h(x_k^p, 0)$, we can rewrite the above linearized models as follows:

$$e_{k+1} = A_k e_k + B_k \delta u_k + G_k w_k \tag{2.20a}$$

$$\delta z_k = H_k e_k + M_k v_k \tag{2.20b}$$

In Kalman filtering, we aim to provide an estimate of the system's state based on the available partial information we have obtained until time k, i.e., $z_{0:k}$. The state estimate is a random vector denoted by x_k^+, whose distribution is the conditional distribution of the state on the obtained observations so far, which is called belief

and is denoted by b_k:

$$b_k = p(x_k^+) = p(x_k|z_{0:k}) \tag{2.21}$$

$$\widehat{x}_k^+ = \mathbb{E}[x_k|z_{0:k}] \tag{2.22}$$

$$P_k = \mathbb{C}[x_k|z_{0:k}] \tag{2.23}$$

where $\mathbb{E}[\cdot|\cdot]$ and $\mathbb{C}[\cdot|\cdot]$ are the conditional expectation and conditional covariance operators, respectively. In the Gaussian case, we have $b_k = \mathcal{N}(\widehat{x}_k^+, P_k)$, i.e., the belief can only be characterized by its mean and covariance, i.e., $b_k \equiv (\widehat{x}_k^+, P_k)$.

Kalman filtering consists of two steps at every time stage: a prediction step and an update step. In the prediction step, the mean and covariance of prior x_k^- is computed. For the system in Eq. (2.20) prediction step is:

$$\widehat{e_{k+1}^-} = A_k \widehat{e_k^+} + B_k \delta u_k \tag{2.24}$$

$$P_{k+1}^- = A_k P_k^+ A_k^T + G_k Q_k G_k^T \tag{2.25}$$

In the update step, the mean and covariance of posterior x_k^+ is computed. For the system in Eq. (2.20), the update step is:

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + M_k R_k M_k^T)^{-1} \tag{2.26}$$

$$\widehat{e_{k+1}^+} = \widehat{e_{k+1}^-} + K_{k+1}(\delta z_{k+1} - H_{k+1}\widehat{e_{k+1}^-}) \tag{2.27}$$

$$P_{k+1}^+ = (I - K_{k+1}H_{k+1})P_{k+1}^- \tag{2.28}$$

Note that

$$\widehat{x}_k^+ = \mathbb{E}[x_k|z_{0:k}] = x_k^p + \widehat{e}_k^+ = x_k^p + \mathbb{E}[e_k|z_{0:k}] \tag{2.29}$$

$$P_k = \mathbb{C}[x_k|z_{0:k}] = P_k^+ = \mathbb{C}[e_k|z_{0:k}] \tag{2.30}$$

Once we obtain the belief from the filter, a controller can generate an optimal control signal accordingly. In other words, we have a time-varying mapping μ_k from the belief space into the control space that generates an optimal control based on the given belief $u_k = \mu_k(b_k)$ at every time step k. An LQR controller is of this kind and it is optimal in the sense of minimizing following cost:

$$\begin{aligned} J_{LQR} &= \mathbb{E}\left[\sum_{k\geq 0}(\widehat{x}_k^+ - x_k^p)^T W_x(\widehat{x}_k^+ - x_k^p) + (u_k - u_k^p)^T W_u(u_k - u_k^p)\right] \\ &= \mathbb{E}\left[\sum_{k\geq 0}(\widehat{e}_k^+)^T W_x(\widehat{e}_k^+) + (\delta u_k)^T W_u(\delta u_k)\right] \end{aligned} \tag{2.31}$$

The linear control law that minimizes this cost function for a linear system is of the form:

$$\delta u_k = -L_k \widehat{e}_k^+ \tag{2.32}$$

where the time-varying feedback gains L_k can be computed recursively as follows:

$$L_k = (B_k^T S_{k+1} B_k + W_u)^{-1} B_k^T S_{k+1} A_k \tag{2.33}$$

$$S_k = W_x + A_k^T S_{k+1} A_k - A_k^T S_{k+1} B_k L_k \tag{2.34}$$

If the nominal path is of length N, then the $S_N = W_x$ is the initial condition of the above recursion, which is solved backwards in time. Note that the whole control is $u_k = u_k^p + \delta u_k$.

Plugging the obtained LQR control law into the Kalman filtering equations, we

can get the following error dynamics, for the defined errors:

$$\begin{pmatrix} e_{k+1} \\ \widetilde{e}_{k+1} \end{pmatrix} = \begin{pmatrix} A_k - B_k L_k & B_k L_k \\ 0 & A_k - K_{k+1} H_{k+1} A_k \end{pmatrix} \begin{pmatrix} e_k \\ \widetilde{e}_k \end{pmatrix} + \begin{pmatrix} G_k & 0 \\ G_k - K_{k+1} H_{k+1} G_k & -K_{k+1} M_{k+1} \end{pmatrix} \begin{pmatrix} w_k \\ v_{k+1} \end{pmatrix} \tag{2.35}$$

or equivalently,

$$\begin{pmatrix} e_{k+1} \\ \widehat{e}^{+}_{k+1} \end{pmatrix} = \begin{pmatrix} A_k & -B_k L_k \\ K_{k+1} H_{k+1} A_k & A_k - B_k L_k - K_{k+1} H_{k+1} A_k \end{pmatrix} \begin{pmatrix} e_k \\ \widehat{e}^{+}_{k} \end{pmatrix} + \begin{pmatrix} G_k & 0 \\ K_{k+1} H_{k+1} G_k & K_{k+1} M_{k+1} \end{pmatrix} \begin{pmatrix} w_k \\ v_{k+1} \end{pmatrix} \tag{2.36}$$

Defining $\zeta_k := (e_k, \widehat{e}^{+}_{k})^T$ and $q_k := (w_k, v_{k+1})^T$, we can rewrite Eq. (2.36) in a more compact form as

$$\zeta_{k+1} = \overline{F}_k \zeta_k + \overline{G}_k q_k, \quad q_k \sim \mathcal{N}(0, \overline{Q}_k), \quad \overline{Q}_k = \begin{pmatrix} Q_k & 0 \\ 0 & R_{k+1} \end{pmatrix} \tag{2.37}$$

with appropriate definitions for $\overline{F}_k$ and $\overline{G}_k$.

The above equation along with the following equation on estimation covariance propagation

$$P_{k+1} = (I - K_{k+1} H_{k+1})(A_k P_k A_k^T + G_k Q_k G_k^T) \tag{2.38}$$

characterize the evolution of the state x_k and belief $b_k \equiv (\widehat{x}^{+}_{k}, P_k)$ under the time-

varying LQG controller.

2.2.2 Stationary LQG Controller

A stationary Linear Quadratic Gaussian (SLQG) controller is often used to regulate (or stabilize) the system state to a pre-planned point (also called set-point, nominal, or desired point) in the presence of process and observation noise. In principal it is designed (and optimal) for linear systems with Gaussian noise, but it is also can be utilized for stabilizing nonlinear systems locally around the planned point. An SLQG controller is composed of a Stationary Kalman Filter (SKF) as an estimator and a Stationary Linear Quadratic Regulator (SLQR) as a controller. At every time step k, SKF provides the *a posteriori* distribution over the system state (belief) b_k, and SLQR generates control u_k based on b_k.

In this section, we first discuss the system linearization around the planned point, and then discuss the SKF, SLQR and SLQG corresponding to this nominal point. Consider the nonlinear partially-observable state-space equations of the system as follows:

$$x_{k+1} = f(x_k, u_k, w_k), \quad w_k \sim \mathcal{N}(0, Q_k) \tag{2.39a}$$

$$z_k = h(x_k, v_k), \quad v_k \sim \mathcal{N}(0, R_k) \tag{2.39b}$$

and consider a planned state point x^p, to whose vicinity the controller has to drive the system. If the system state reaches the x^p, it is assumed that the system remains there with zero control, $u^p = 0$, i.e.,

$$x^p = f(x^p, 0, 0) \tag{2.40}$$

The role of a closed-loop stochastic controller, during the state regulation, is com-

pensating robot deviations from the desired point due to the noise effects and driving the robot close to the desired point in the sense of minimizing following quadratic cost:

$$J = \mathbb{E}\left[\sum_{k\geq 0}(x_k - x^p)^T W_x (x_k - x^p) + (u_k)^T W_u (u_k)\right] \tag{2.41}$$

where W_x and W_u are the positive definite weight matrices for state and control cost, respectively.

Again, similar to the time-varying case, since we only have imperfect information of the state x_k, we have to make the estimate x_k^+ of the state based on the available observations $z_{0:k}$ and the controller generates the control signal based on the estimated value of the state, i.e., belief. Based on the separation principle [15], in linear systems with Gaussian noise, the above minimization is equivalent to performing two separate minimizations that lead to the separate design of the SKF and SLQR. In the following, we first discuss the linearization of a nonlinear model and then we discuss how the SKF and the SLQR can be designed for this linearized system and finally combine them to construct an SLQG controller.

Given a desired point x^p, we linearize the dynamics and observation model in Eq. (2.39), as follows:

$$x_{k+1} = f(x^p, 0, 0) + A_s(x_k - x^p) + B_s(u_k - 0) + G_s w_k, \quad w_k \sim \mathcal{N}(0, Q_s) \tag{2.42a}$$

$$z_k = h(x^p, 0) + H_s(x_k - x^p) + M_s v_k, \quad v_k \sim \mathcal{N}(0, R_s) \tag{2.42b}$$

where

$$A_s = \frac{\partial f}{\partial x}(x^p, 0, 0), \quad B_s = \frac{\partial f}{\partial u}(x^p, 0, 0), \quad G_s = \frac{\partial f}{\partial w}(x^p, 0, 0),$$

$$H_s = \frac{\partial h}{\partial x}(x^p, 0), \quad M_s = \frac{\partial h}{\partial v}(x^p, 0) \tag{2.43}$$

Now, let us define the following errors:

- SLQG error (main error): $e_k = x_k - x^p$.
- SKF error (estimation error): $\widetilde{e}_k = x_k - \widehat{x}_k^+$, where $\widehat{x}_k^+ = \mathbb{E}[x_k^+] = \mathbb{E}[x_k | z_{0:k}]$.
- SLQR error (estimation of SLQG error): $\widehat{e}_k^+ = \widehat{x}_k^+ - x^p$.

Note that these errors are linearly dependent: $e_k = \widehat{e}_k^+ + \widetilde{e}_k$. Defining $\delta u_k := u_k$ and $\delta z_k := z_k - z^p = z_k - h(x^p, 0)$, we can rewrite above linearized models as follows:

$$e_{k+1} = A_s e_k + B_s \delta u_k + G_s w_k \tag{2.44a}$$

$$\delta z_k = H_s e_k + M_s v_k \tag{2.44b}$$

In SKF, we aim to provide an estimate of the system's state based on the available partial information we have obtained until time k, i.e., $z_{0:k}$. The state estimate is a random vector denoted by x_k^+, whose distribution is the conditional distribution of the state on the obtained observations so far, which is called belief and is denoted by $b_k = p(x_k^+) = p(x_k | z_{0:k})$. In the Gaussian case, the belief can only be characterized by its mean and covariance, i.e., $b_k \equiv (\widehat{x}_k^+, P_k)$. Thus, in the Gaussian case, we can write:

$$b_k = p(x_k^+) = p(x_k | z_{0:k}) = \mathcal{N}(\widehat{x}_k^+, P_k) \Leftrightarrow b_k \equiv (\widehat{x}_k^+, P_k) \tag{2.45}$$

$$\widehat{x}_k^+ = \mathbb{E}[x_k | z_{0:k}], \qquad P_k = \mathbb{C}[x_k | z_{0:k}] \tag{2.46}$$

where $\mathbb{E}[\cdot|\cdot]$ and $\mathbb{C}[\cdot|\cdot]$ are the conditional expectation and conditional covariance operators, respectively.

SKF consists of two steps at every time stage: a prediction step and an update step. In the prediction step, the mean and covariance of prior x_k^- is computed. For the system in Eq. (2.44) the prediction step is:

$$\widehat{e_{k+1}^-} = A_s \widehat{e_k^+} + B_s \delta u_k \tag{2.47}$$

$$P_{k+1}^- = A_s P_k^+ A_s^T + G_s Q_s G_s^T \tag{2.48}$$

In the update step, the mean and covariance of posterior x_k^+ is computed. For the error system in Eq. (2.44), the update step is:

$$K_k = P_k^- H_s^T (H_s P_k^- H_s^T + M_s R_s M_s^T)^{-1} \tag{2.49}$$

$$\widehat{e_{k+1}^+} = \widehat{e_{k+1}^-} + K_{k+1}(\delta z_{k+1} - H_s \widehat{e_{k+1}^-}) \tag{2.50}$$

$$P_{k+1}^+ = (I - K_{k+1} H_s) P_{k+1}^- \tag{2.51}$$

Note that

$$\widehat{x_k^+} = x^p + \widehat{e_k^+}, \qquad P_k = P_k^+ \tag{2.52}$$

In SKF, if (A_s, H_s) is an observable pair and $(A_s, \check{Q}_s)$ is a controllable pair, where $G_s Q_s G_s^T = \check{Q}_s \check{Q}_s^T$, then the prior and posterior covariances P_k^- and P_k and the filter gain K_k all converge to their stationary values, denoted by P_s^-, P_s, and K_s, respectively [15]. The P_s^- can be computed by solving following DARE. Having P_s^-, stationary gain K_s and estimation covariance P_s is computed as follows:

$$P_s^- = G_s Q_s G_s^T + A_s (P_s^- - P_s^- H_s^T (H_s P_s^- H_s^T + M_s R_s M_s^T)^{-1} H_s P_s^-) A_s^T, \tag{2.53}$$

$$K_s = P_s^- H_s^T (H_s P_s^- H_s^T + M_s R_s M_s^T)^{-1}, \tag{2.54}$$

$$P_s = (I - K_s H_s) P_s^- \tag{2.55}$$

In Stationary LQR (SLQR) we have a stationary mapping μ_s from the belief space into the control space that generates an optimal control based on the given belief $u_k = \mu_s(b_k)$ at every time step k. SLQR controller is optimal in the sense of minimizing following cost:

$$\begin{aligned} J_{SLQR} &= \mathbb{E}\left[\sum_{k\geq 0} (\widehat{x}_k^+ - x^p)^T W_x (\widehat{x}_k^+ - x^p) + (u_k)^T W_u (u_k)\right] \\ &= \mathbb{E}\left[\sum_{k\geq 0} (\widehat{e}_k^+)^T W_x (\widehat{e}_k^+) + (\delta u_k)^T W_u (\delta u_k)\right] \end{aligned} \tag{2.56}$$

If the (A_s, B_s) is a controllable pair and $(A_s, \check{W}_x)$ is an observable pair, where $\check{W}_x{}^T \check{W}_x = W_x$, then, the stationary linear control law that minimizes the cost function J_{SLQR} for a linear system is of the form:

$$\delta u_k = -L_s \widehat{e}_k^+ \tag{2.57}$$

where the stationary feedback gain L_s can be computed as follows:

$$L_s = (B_s^T S_s B_s + W_u)^{-1} B_s^T S_s A_s \tag{2.58}$$

$$S_s = W_x + A_s^T S_s A_s - A_s^T S_s B_s L_s \tag{2.59}$$

where the second equation is indeed a Discrete Algebraic Riccati Equation (DARE) that can be efficiently solved for S_s. Plugging S_s into Eq. (2.58), we get the feedback gain L_s.

Plugging the obtained control law of SLQR into the SKF equations, we can get

the following stationary dynamics for the defined errors:

$$\begin{pmatrix} e_{k+1} \\ \widetilde{e}_{k+1} \end{pmatrix} = \begin{pmatrix} A_s - B_s L_s & B_s L_s \\ 0 & A_s - K_s H_s A_s \end{pmatrix} \begin{pmatrix} e_k \\ \widetilde{e}_k \end{pmatrix} + \begin{pmatrix} G_s & 0 \\ G_s - K_s H_s G_s & -K_s M_s \end{pmatrix} \begin{pmatrix} w_k \\ v_{k+1} \end{pmatrix} \tag{2.60}$$

or equivalently,

$$\begin{pmatrix} e_{k+1} \\ \widehat{e}^{+}_{k+1} \end{pmatrix} = \begin{pmatrix} A_s & -B_s L_s \\ K_s H_s A_s & A_s - B_s L_s - K_s H_s A_s \end{pmatrix} \begin{pmatrix} e_k \\ \widehat{e}^{+}_k \end{pmatrix} + \begin{pmatrix} G_s & 0 \\ K_s H_s G_s & K_s M_s \end{pmatrix} \begin{pmatrix} w_k \\ v_{k+1} \end{pmatrix} \tag{2.61}$$

Defining $\zeta_k := (e_k, \widehat{e}^{+}_k)^T$ and $q_k := (w_k, v_{k+1})^T$, we can rewrite Eq. (2.61) in a more compact form as

$$\zeta_{k+1} = \overline{F}_s \zeta_k + \overline{G}_s q_k, \quad q_k \sim \mathcal{N}(0, \overline{Q}_s), \quad \overline{Q}_s = \begin{pmatrix} Q_s & 0 \\ 0 & R_s \end{pmatrix} \tag{2.62}$$

with appropriate definitions for $\overline{F}_s$ and $\overline{G}_s$.

It can be shown that if $\overline{F}_s$ is a stable matrix, i.e. $\lim_{\kappa \to \infty} (\overline{F}_s)^\kappa = 0$, the ζ_k converges in distribution to $\zeta_s \sim \mathcal{N}(0, \mathcal{P}_s)$. The stationary covariance $\mathcal{P}_s$ is the solution of the following Lyapunov equation:

$$\mathcal{P}_s = \overline{F}_s \mathcal{P}_s \overline{F}_s^T + \overline{G}_s \overline{Q}_s \overline{G}_s^T \tag{2.63}$$

Note that $\mathcal{P}_s$ can be decomposed to four blocks

$$\mathcal{P}_s = \begin{pmatrix} \mathcal{P}_{s,11} & \mathcal{P}_{s,12} \\ \mathcal{P}_{s,21} & \mathcal{P}_{s,22} \end{pmatrix} \tag{2.64}$$

such that $\mathcal{P}_{s,11} = \lim_{k\to\infty} \mathbb{C}[e_k]$ and $\mathcal{P}_{s,22} = \lim_{k\to\infty} \mathbb{C}[\widehat{e}_k^+]$. Therefore, since $\widehat{x}_k^+ = x^p + \widehat{e}_k^+$, the estimation mean is also converging to a stationary random variable, denoted by $\widehat{x}_s^+$:

$$\widehat{x}_s^+ := \lim_{k\to\infty} \widehat{x}_k^+ \sim \mathcal{N}(x^p, \mathcal{P}_{s,22}) \tag{2.65}$$

Due to the linear relation $e_k = \widehat{e}_k^+ + \widetilde{e}_k$, we can also conclude $\lim_{k\to\infty} \mathbb{C}[\widetilde{e}_k] = \mathcal{P}_{s,11} + \mathcal{P}_{s,22} - 2\mathcal{P}_{s,12}$. It can be proven that in stationary LQG, the stability of matrix $\overline{F}_s$ is a direct consequence of the controllability of the pair (A_s, B_s) and the observability of pair (A_s, H_s) [15], [16].

Thus, collecting all the conditions, if (A_s, B_s) and $(A_s, \check{Q}_s)$ are controllable pairs, where $G_s Q_s G_s^T = \check{Q}_s \check{Q}_s^T$, and if (A_s, H_s) and $(A_s, \check{W}_x)$ are observable pairs, where $W_x = \check{W}_x^T \check{W}_x$, then under the stationary LQG the belief b_k converges in distribution to a stationary belief:

$$b_s := \lim_{k\to\infty} b_k = \mathcal{N}(\widehat{x}_s^+, P_s^+) \tag{2.66}$$

where P_s^+ is a deterministic quantity and we can characterize the distribution over the stationary belief as:

$$b_s \equiv (\widehat{x}_s^+, P_s^+) \sim \mathcal{N}\left(\begin{pmatrix} x^p \\ P_s^+ \end{pmatrix}, \begin{pmatrix} \mathcal{P}_{s,22} & 0 \\ 0 & 0 \end{pmatrix} \right) \tag{2.67}$$

2.2.3 Periodic LQG Controller

A periodic Linear Quadratic Gaussian (PLQG) controller is a time-varying LQG controller that is designed to track a pre-planned periodic trajectory (also called nominal, desired, or open-loop trajectory) in the presence of process and observation noise.

In this section, we first discuss the system linearization and planned nominal trajectory, and then discuss the KF, LQR and LQG corresponding with this nominal trajectory. Consider the nonlinear partially-observable state-space equations of the system as follows:

$$x_{k+1} = f(x_k, u_k, w_k), \quad w_k \sim \mathcal{N}(0, Q_k) \tag{2.68a}$$

$$z_k = h(x_k, v_k), \quad v_k \sim \mathcal{N}(0, R_k) \tag{2.68b}$$

A T-periodic planned nominal trajectory for the robot is a sequence of planned states $(x_k^p)_{k\geq 0}$ and planned controls $(u_k^p)_{k\geq 0}$, such that it is consistent with the noiseless dynamics model, i.e., we have:

$$x_{k+1}^p = f(x_k^p, u_k^p, 0), \quad x_{k+T}^p = x_k^p, \quad u_{k+T}^p = u_k^p \tag{2.69}$$

The role of a closed-loop stochastic controller, during the trajectory tracking execution, is to compensate for robot deviations from the planned trajectory due to noise effects and keeping the robot close to the planned trajectory in the sense of minimizing the following quadratic cost:

$$J = \mathbb{E}\left[\sum_{k\geq 0}(x_k - x_k^p)^T W_x (x_k - x_k^p) + (u_k - u_k^p)^T W_u (u_k - u_k^p)\right] \tag{2.70}$$

where W_x and W_u are the positive definite weight matrices for the state and control cost, respectively.

Since the state space is not fully observable and it is only partially observable, at every step of LQG execution, a Kalman filter estimates the system state and an LQR controller generates the optimal control based on this estimation. We first linearize the system along the nominal path and then describe the KF and LQR designed along this path.

Given a periodic nominal trajectory $(x_k^p, u_k^p)_{k\geq 0}$, we linearize the dynamics and observation model in (2.68), as follows:

$$x_{k+1} = f(x_k^p, u_k^p, 0) + A_k(x_k - x_k^p) + B_k(u_k - u_k^p) + G_k w_k, \quad w_k \sim \mathcal{N}(0, Q_k) \tag{2.71a}$$

$$z_k = h(x_k^p, 0) + H_k(x_k - x_k^p) + M_k v_k, \quad v_k \sim \mathcal{N}(0, R_k) \tag{2.71b}$$

where

$$\begin{cases} A_k &= \frac{\partial f}{\partial x}(x_k^p, u_k^p, 0), \quad B_k = \frac{\partial f}{\partial u}(x_k^p, u_k^p, 0), \quad G_k = \frac{\partial f}{\partial w}(x_k^p, u_k^p, 0), \\ H_k &= \frac{\partial h}{\partial x}(x_k^p, 0), \quad M_k = \frac{\partial h}{\partial v}(x_k^p, 0) \end{cases} \tag{2.72}$$

It is worth noting that the linearized system is a periodic one, i.e.,

$$A_{k+T} = A_k, \quad B_{k+T} = B_k, \quad G_{k+T} = G_k, \quad H_{k+T} = H_k, \quad M_{k+T} = M_k, \quad Q_{k+T} = Q_k, \quad R_{k+T} = R_k. \tag{2.73}$$

Now, let us define the following errors:

- LQG error (main error): $e_k = x_k - x_k^p$
- KF error (estimation error): $\widetilde{e}_k = x_k - \widehat{x}_k^+$

- LQR error (mean of estimation of LQG error): $\widehat{e}_k^+ = \widehat{x}_k^+ - x_k^p$

Note that these errors are linearly dependent: $e_k = \widehat{e}_k^+ + \widetilde{e}_k$. Also, defining $\delta u_k = u_k - u_k^p$ and $\delta z_k = z_k - z_k^p := z_k - h(x_k^p, 0)$, we can rewrite the above linearized models as follows:

$$e_{k+1} = A_k e_k + B_k \delta u_k + G_k w_k, \quad w_k \sim \mathcal{N}(0, Q_k) \tag{2.74a}$$

$$\delta z_k = H_k e_k + M_k v_k, \quad v_k \sim \mathcal{N}(0, R_k) \tag{2.74b}$$

which is a periodic linear system due to the (2.73).

A Periodic Kalman Filter (PKF) is a time-varying Kalman filter, whose underlying linear system is periodic, as in (2.74). In Kalman filtering, we aim to provide an estimate of the system's state based on the available partial information we have obtained until time k, i.e., $z_{0:k}$. The state estimate is a random vector denoted by x_k^+, whose distribution is the conditional distribution of the state on the obtained observations so far, which is called belief and is denoted by b_k:

$$b_k = p(x_k^+) = p(x_k | z_{0:k}) \tag{2.75}$$

$$\widehat{x}_k^+ = \mathbb{E}[x_k | z_{0:k}] \tag{2.76}$$

$$P_k = \mathbb{C}[x_k | z_{0:k}] \tag{2.77}$$

where $\mathbb{E}[\cdot|\cdot]$ and $\mathbb{C}[\cdot|\cdot]$ are the conditional expectation and conditional covariance operators, respectively. In the Gaussian case, we have $b_k = \mathcal{N}(\widehat{x}_k^+, P_k)$, i.e., the belief can only be characterized by its mean and covariance, i.e., $b_k \equiv (\widehat{x}_k^+, P_k)$. Similar to the conventional Kalman filtering, PKF consists of two steps at every time stage: a prediction step and an update step. In the prediction step, the mean and covariance of prior x_k^- is computed. For the system in (2.74) the prediction step

is:

$$\widehat{e_{k+1}^{-}} = A_k \widehat{e_k^{+}} + B_k \delta u_k \tag{2.78}$$

$$P_{k+1}^{-} = A_k P_k^{+} A_k^T + G_k Q_k G_k^T \tag{2.79}$$

In the update step, the mean and covariance of posterior x_k^+ is computed. For the system in (2.74), the update step is:

$$K_k = P_k^{-} H_k^T (H_k P_k^{-} H_k^T + M_k R_k M_k^T)^{-1} \tag{2.80}$$

$$\widehat{e_{k+1}^{+}} = \widehat{e_{k+1}^{-}} + K_{k+1}(\delta z_{k+1} - H_{k+1}\widehat{e_{k+1}^{-}}) \tag{2.81}$$

$$P_{k+1}^{+} = (I - K_{k+1} H_{k+1}) P_{k+1}^{-} \tag{2.82}$$

Note that

$$\widehat{x_k^{+}} = \mathbb{E}[x_k | z_{0:k}] = x_k^p + \mathbb{E}[e_k | z_{0:k}] = x_k^p + \widehat{e_k^{+}} \tag{2.83}$$

$$P_k = \mathbb{C}[x_k | z_{0:k}] = \mathbb{C}[e_k | z_{0:k}] = P_k^{+} \tag{2.84}$$

Lemma 1. *In Periodic Kalman filtering (PKF), if for all k, the pair (A_k, H_k) is detectable and the pair $(A_k, \check{Q}_k)$ is stabilizable, where $G_k Q_k G_k^T = \check{Q}_k \check{Q}_k^T$, then the prior and posterior covariances P_k^- and P_k and the filter gain K_k, all converge to their T-periodic stationary values, denoted by $\check{P}_t^-$, $\check{P}_t$, and $\check{K}_t$, respectively [18]. Matrix $\check{P}_t^-$ is the unique Symmetric T-Periodic Positive Semi-definite (SPPS) solution [18] of the following Discrete Periodic Riccati Equation (DPRE):*

$$\check{P}_{k+1}^{-} = G_k Q_k G_k^T + A_k(\check{P}_k^{-} - \check{P}_k^{-} H_k^T (H_k \check{P}_k^{-} H_k^T + M_k R_k M_k^T)^{-1} H_k \check{P}_k^{-}) A_k^T \tag{2.85}$$

Having $\check{P}_k^-$, the periodic gain $\check{K}_k$ and estimation covariance $\check{P}_k$ is computed as follows:

$$\check{K}_k = \check{P}_k^- H_k^T (H_k \check{P}_k^- H_k^T + M_k R_k M_k^T)^{-1}, \tag{2.86}$$

$$\check{P}_k = (I - \check{K}_k H_k)\check{P}_k^- \tag{2.87}$$

where

$$\check{P}_{k+T}^- = \check{P}_k^-, \quad \check{K}_{k+T} = \check{K}_k, \quad \check{P}_{k+T} = \check{P}_k \tag{2.88}$$

Proof. See [18]. □

If the pair (A_k, H_k) is detectable and the pair $(A_k, \check{Q}_k)$ is stabilizable, then the pair (A_k, H_k) is observable and the pair $(A_k, \check{Q}_k)$ is controllable, and hence Lemma 9 follows.

A PLQR controller is the separated controller part of the PLQG controller. Once the Periodic Kalman filter produces the estimation (belief), the PLQR controller generates an optimal control signal accordingly. In other words, we have a time-varying mapping μ_k from the belief space into the control space that generates an optimal control based on the given belief $u_k = \mu_k(b_k)$ at every time step k. In PLQG, the mapping μ_k is the control law of the PLQR controller, which is optimal in the sense of minimizing following cost:

$$\begin{aligned} J_{PLQR} &= \mathbb{E}\left[\sum_{k\geq 0} (\widehat{x}_k^+ - x_k^p)^T W_x (\widehat{x}_k^+ - x_k^p) + (u_k - u_k^p)^T W_u (u_k - u_k^p)\right] \\ &= \mathbb{E}\left[\sum_{k\geq 0} (\widehat{e}_k^+)^T W_x (\widehat{e}_k^+) + (\delta u_k)^T W_u (\delta u_k)\right] \end{aligned} \tag{2.89}$$

The linear control law that minimizes this cost function for a linear system is:

$$\delta u_k = -L_k \widehat{e}_k^+, \quad L_{k+T} = L_k \tag{2.90}$$

Lemma 2. *In a Periodic LQR (PLQR), if for all k, the pair (A_k, B_k) is stabilizable and the pair $(A_k, \check{W}_x)$ is detectable, where $W_x = \check{W}_x^T \check{W}_x$, then the time-varying feedback gains L_k is a T-periodic gain, i.e., $L_{k+T} = L_k$ and is computed as follows:*

$$L_k = (B_k^T S_{k+1} B_k + W_u)^{-1} B_k^T S_{k+1} A_k \tag{2.91}$$

where S_k is the SPPS solution of the following DPRE:

$$S_k = W_x + A_k^T S_{k+1} A_k - A_k^T S_{k+1} B_k (B_k^T S_{k+1} B_k + W_u)^{-1} B_k^T S_{k+1} A_k \tag{2.92}$$

Note that the whole control is $u_k = u_k^p + \delta u_k$.

Plugging the obtained control law of PLQR into the PKF equations, we can get the following error dynamics, for the defined errors:

$$\begin{aligned} \begin{pmatrix} e_{k+1} \\ \widetilde{e}_{k+1} \end{pmatrix} &= \begin{pmatrix} A_k - B_k L_k & B_k L_k \\ 0 & A_k - \check{K}_{k+1} H_{k+1} A_k \end{pmatrix} \begin{pmatrix} e_k \\ \widetilde{e}_k \end{pmatrix} \\ &+ \begin{pmatrix} G_k & 0 \\ G_k - \check{K}_{k+1} H_{k+1} G_k & -\check{K}_{k+1} M_{k+1} \end{pmatrix} \begin{pmatrix} w_k \\ v_{k+1} \end{pmatrix} \end{aligned} \tag{2.93}$$

or equivalently,

$$\begin{pmatrix} e_{k+1} \\ \widehat{e}_{k+1}^+ \end{pmatrix} = \begin{pmatrix} A_k & -B_k L_k \\ \check{K}_{k+1} H_{k+1} A_k & A_k - B_k L_k - \check{K}_{k+1} H_{k+1} A_k \end{pmatrix} \begin{pmatrix} e_k \\ \widehat{e}_k^+ \end{pmatrix}$$

$$+ \begin{pmatrix} G_k & 0 \\ \check{K}_{k+1} H_{k+1} G_k & \check{K}_{k+1} M_{k+1} \end{pmatrix} \begin{pmatrix} w_k \\ v_{k+1} \end{pmatrix} \tag{2.94}$$

Defining $\zeta_k := (e_k, \widehat{e}_k^+)^T$ and $q_k := (w_k, v_{k+1})^T$, we can rewrite (2.94) in a more compact form as

$$\zeta_{k+1} = \overline{F}_k \zeta_k - \overline{G}_k q_k, \quad q_k \sim \mathcal{N}(0, \overline{Q}_k), \quad \overline{Q}_k = \begin{pmatrix} Q_k & 0 \\ 0 & R_{k+1} \end{pmatrix} \tag{2.95}$$

with appropriate definitions for $\overline{F}_k$ and $\overline{G}_k$. Thus, ζ_k is a random variable with a Gaussian distribution, i.e.,

$$\zeta_k = \mathcal{N}(0, \mathcal{P}_k) \tag{2.96}$$

or

$$\begin{pmatrix} x_k \\ \widehat{x}_k^+ \end{pmatrix} \sim \mathcal{N}(\begin{pmatrix} x_k^p \\ x_k^p \end{pmatrix}, \mathcal{P}_k) \tag{2.97}$$

where $\mathcal{P}_k$ is the solution of following Discrete Periodic Lyapunov Equation (DPLE):

$$\mathcal{P}_{k+1} = \overline{F}_k \mathcal{P}_k \overline{F}_k^T - \overline{G}_k \overline{Q}_k \overline{G}_k^T \tag{2.98}$$

which can be decomposed into four blocks

$$\mathcal{P}_k = \begin{pmatrix} \mathcal{P}_{k,11} & \mathcal{P}_{k,12} \\ \mathcal{P}_{k,21} & \mathcal{P}_{k,22} \end{pmatrix} \tag{2.99}$$

Lemma 3. *Under the preceding assumptions in Lemma 1 and Lemma 2, the solution*

of DPLE in (2.98) *converges to a unique SPPS solution* $\check{\mathcal{P}}_k$ *independent of the initial covariance* $\mathcal{P}_0$*, i.e.,* $\check{\mathcal{P}}_{k+T} = \check{\mathcal{P}}_k$.

Proof. See [18]. □

Therefore, the process in (2.95) converges to a cyclostationary process [17], i.e., the distribution over ζ_k is periodic. Hence, since $\widehat{x}_k^+ \sim \mathcal{N}(x_k^p, \mathcal{P}_{k,22})$, the distribution over the estimation mean also converges to a periodic distribution, i.e., $\widehat{x}_k^+ \sim \mathcal{N}(x_k^p, \check{\mathcal{P}}_{k,22}) = \mathcal{N}(x_{k+T}^p, \check{\mathcal{P}}_{k+T,22})$. Hence, this analysis leads to the following lemma:

Lemma 4. *Under a Periodic LQG, the belief falls into a Gaussian cyclostationary process, i.e., the distribution over belief* $b_k = (\widehat{x}_k^+, P_k)$ *converges to the following periodic Gaussian distribution:*

$$b_k \equiv (\widehat{x}_k^+, P_k) \sim \mathcal{N}\left(\begin{pmatrix} x_k^p \\ \check{P}_k \end{pmatrix}, \begin{pmatrix} \check{\mathcal{P}}_{k,22} & 0 \\ 0 & 0 \end{pmatrix}\right) \tag{2.100}$$

The degeneracy of the Gaussian distribution over belief in (2.100) is due to the fact that $\check{P}_k$ is a deterministic process. It is worth noting that the belief mean converges to the T-periodic belief $\mathbb{E}[b_{k+T}] = \mathbb{E}[b_k] = (x_k^p, \check{P}_k)$. Hence, Lemma 8 follows, as it is the same as Lemma 4, where we have:

$$b_k^c = \begin{pmatrix} x_k^p \\ \check{P}_k \end{pmatrix}, \quad \mathtt{C}_k = \begin{pmatrix} \check{\mathcal{P}}_{k,22} & 0 \\ 0 & 0 \end{pmatrix} \tag{2.101}$$

3. LITERATURE REVIEW

In this chapter, we review the literature most relevant to our research. We start by reviewing different methods for planning in belief space and explaining their relation to the original POMDP framework. Additionally, we discuss literature concerning the probabilistic completeness of roadmap-based algorithms in the deterministic setting. We also review related literature on nonholonomic motion planning. Lastly, we look into literature related to the physical implementation of belief space planners as well as the methods that are robust to model discrepancies and changing environments.

3.1 Belief Space Planning Algorithms

Uncertainty in robotic systems usually stems from three sources: (i) motion uncertainty, which is also called process or dynamic uncertainty, and results from the noise that affects system dynamics, (ii) sensing uncertainty, which is also referred to as imperfect state information, and can be caused by the noise that affects the measurements made by sensors, and (iii) uncertainty in the environment map, such as uncertain obstacle locations or uncertain location of features (information sources) in the environment.

While there are methods that deal with map uncertainty [38, 62, 65], we do not utilize them in our framework as we assume there is no uncertainty in the environment map. Methods such as [10, 26, 27, 60] use sampling-based motion planning ideas to deal with motion uncertainty. However, these techniques do not consider sensing uncertainty in the planning problem.

Another class of methods that are most related to FIRM consider both motion and sensing uncertainties in planning. The planning problems under motion and sensing uncertainty in their most general form are modeled as a POMDP problem.

The ultimate goal in planning under uncertainty (solving POMDPs) is finding the best policy that generates the optimal actions as a function of belief. However, due to the intractability of POMDP solution, practical results for these methods usually are limited to problems with a small set of states [44]. Point-based POMDP solvers such as [13, 52, 72, 80] have increased the size of problems that can be solved by POMDPs. However, they do not handle continuous state, control, and observation spaces. For the Gaussian belief case, there are some techniques such as [95, 96] that handle continuous spaces locally around a given trajectory in belief space. The method in [78] generalizes local approaches to non-Gaussian beliefs.

In continuous state, control, and observation space, most methods do not follow the POMDP framework due to its complexity. Instead, they return a nominal path as the solution to the planning problem, which is fixed regardless of the process and sensor noise in the execution phase. Censi et al. [23] propose two algorithms for planning based on forward graph search and backward constraint propagation on a grid-based representation of the space. Platt et al. [79] plan in continuous space by relying on maximum likelihood observations and finding the best nominal path through nonlinear optimization methods. In the LQG-MP method [93], the best path is found among a finite number of RRT paths by simulating the performance of LQG on them all. The method in [22] is another example of a tree-based approach, in which the underlying nominal trajectory is optimized using RRT*. Vitus and Tomlin [99] propose an approach to optimize the underlying trajectory by formulating the problem as a chance constrained optimal control problem. In [94], the authors extend the LQG-MP to roadmaps. Moreover, [81] and [43] utilize roadmap-based methods based on the PRM approach, where the best path is found through breadth-first search on the Belief Roadmap (BRM). However, on all these roadmap-based methods the optimal substructure assumption is broken, i.e., the edge costs depend

on each other.

Since these methods return a nominal path instead of a feedback policy, in the case of large deviations, or starting from a new point, replanning has to be performed. However, unless the planning domain is small [79], in the presence of large disturbances, replanning in these methods is computationally very expensive. The reason is that the constructed planning tree depends on the initial belief and thus all computations needed to construct the tree (i.e., to predict future costs) have to be reproduced for a query from a new starting initial belief. BRM ameliorates this expensive computation using covariance factorization techniques. However, it still does not satisfy the optimal substructure assumption and thus for a new query from a new initial point, the search algorithm has to be run again. In the presence of obstacles, recomputing the collision probabilities is also required, which makes replanning even more expensive. In other words, these methods are single-query in the sense that the edge costs are computed for each query.

Thus, online replanning can be done only if the planning domain is small (e.g., [79]) or if the planning horizon is short, such as Receding Horizon Control-based (RHC-based) approaches (e.g., [25]). The method proposed in [91] is an RHC-based method, where the nominal path is updated dynamically over an N-step horizon. The PUMA method proposed in [40] is also an RHC-based framework, where instead of a single action, a sequence of actions (macro-action) is selected at every decision stage. However, these methods entail repeatedly solving open loop optimal control problems at every time step, which is computationally very expensive as the previous computations, in general, cannot be reused for the queries from the new initial point.

In FIRM, however, the best feedback policy, i.e., a mapping from belief space to actions, is computed offline, which is the main goal of planning under uncertainty (POMDPs). Moreover, in FIRM, the optimal substructure assumption holds and

as a result, the costs of the edges on the roadmap are independent of each other. Therefore, the roadmap is independent of the initial point, and in replanning from a new initial point, the computations need not be reproduced. In other words, FIRM is a multi-query roadmap in the belief space in the sense that all edge costs are independent of query. If the goal belief is fixed, the feedback over the graph can be computed (see Eq. (4.17)) offline, in which case the algorithm is robust to changes in the start point of query, and if the goal is also varying, graph feedback can be computed (see Eq. (4.17)) online, which results in a multi-query roadmap that is robust to changes in the start and goal points of the query.

In the methods that account for sensing uncertainty, the state has to be estimated based on measurements. To handle unknown future measurements in the planning stage, methods in [23, 43, 79, 81, 91] consider the maximum likelihood (ML) observation sequence to predict the estimation performance. In contrast, [93, 94] and the FIRM method take all possible future observations into account in planning. As a result, these methods lead to more reliable plans since they take into account possible deviations of the belief caused by future unknown observations.

In the presence of obstacles, due to the dependency of collision events in different time, only methods such as Monte Carlo simulation that can take such dependence into account can be used to compute the collision probabilities reliably (see Fig. 3.1). However, these methods are computationally expensive. As a result, approximate safety measures have been designed to efficiently account for obstacles in planning [23, 91, 93, 94]. However, a problem with some of these collision probability measures is that they are built on the assumption that the collision probabilities at different stages along the path are independent of each other, which is not true in general, and may lead to very conservative or optimistic plans. Fig. 3.1 shows the dependence of the collision probability in time step k_1 and k_2, i.e., $\mathbb{P}(x_{k_1} \in F)$ and $\mathbb{P}(x_{k_2} \in F)$, where

x_k is the robot state at time step k and F is the obstacle set (shown by rectangles). The ellipses are 3σ ellipses of Gaussian distributions obtained by Kalman filtering. Also, the samples in a Monte-Carlo simulation are shown by small circles. The dark ones have collided with obstacles and do not get propagated, while the light ones are the safe samples. Although the overall collision probability in Fig. 3.1(a) is much more than the collision probability Fig. 3.1(b), simplified safety measures based on ellipse-obstacle intersection area lead to the same safety measure in Fig. 3.1(a) and 3.1(b), and are unable to capture this dependency. As a result, different methods (e.g. [75]) provide more accurate and faster ways of computing collision probabilities. In FIRM, however, collision probabilities can be computed and seamlessly incorporated into the planning stage without making simplifying assumptions.

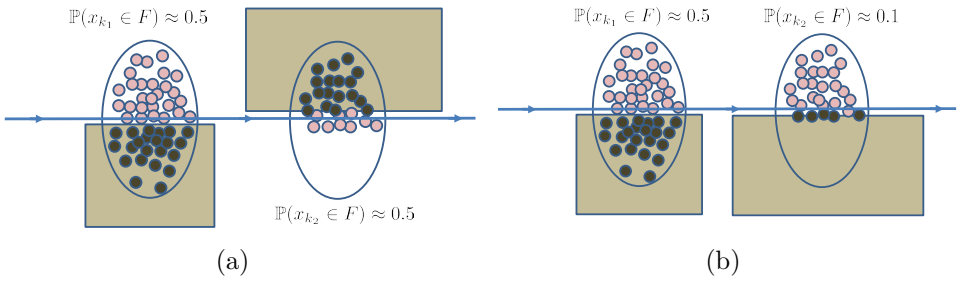


Figure 3.1: This figure shows the dependence of the collision probability in different time steps. Such a dependence can be captured by Monte Carlo simulations but not using the covariance ellipse-obstacle intersection area.

3.2 Probabilistic Completeness

Due to the success of sampling-based methods in many practical planning problems, many researchers have investigated the theoretical basis for this success. However, almost all of these investigations have been done for algorithms that are de-

signed for deterministic planning. The literature in this direction falls into two categories: path isolation-based methods and space covering-based methods.

In the path isolation-based approach, one path is chosen, and it is tiled with sets such as ϵ-balls in [46] or sets with arbitrary shapes but strictly positive measure in [55]. Then, the success probability is analyzed by investigating the probability of sampling in each of the sets that tile the given path in the obstacle-free space. Methods in [46], [55], [89], and [19] are among those that perform path isolation-based analysis of planning algorithms.

In the space covering-based analysis approach, an adequate number of sampled points to find a successful path is expressed in terms of a parameter ϵ, which is a property of the environment. A space is ϵ-good if every point in the state space can at least be connected to an ϵ fraction of the space, using local planners. Methods in [42] and [47] are among these methods.

These methods were developed for the situation where the desired result from the planning algorithm is a path. However, in the presence of uncertainty, the concept of "successful path" is no longer meaningful because on a given path different policies may result in different success probabilities, some interpreted as successful and some not. Thus, since the planning algorithm returns a policy instead of a path, the success has to be defined for a policy. This research extends these concepts to probabilistic spaces, i.e., to sampling-based methods concerning planning under uncertainty. In Chapter 4 Section 4.5, we define the concept of a successful policy and accordingly the concept of probabilistic completeness, and formulate them rigorously.

3.3 Belief Space Planning for Nonholonomic and/or Non-point-stabilizable Systems

Nonholonomic motion planning deals with planning open-loop or feedback (closed-loop) plans for moving an object that is subject to nonholonomic constraints. The unicycle model is an important example of a nonholonomic system, and is commonly used to approximately model a variety of systems ranging from differential drive and synchro drive single-body robots [57] to steerable needles in surgery [100]. One of the challenges in nonholonomic motion planning is stabilizing the system to a point. Thus, if we consider two basic motion tasks: *point-to-point motion*, which deals with driving a moving object from an initial state to a goal state, and *trajectory following*, which deals with following a trajectory in state space, then, in contrast to the holonomic case, point-to-point motion in nonholonomic systems is a more difficult task than trajectory tracking [73]. As mentioned, the main challenge is the state stabilization to the target node. Another class of systems, for which state stabilization (and this point-to-point motion) is a challenge is the class of non-point-stabilizable systems. This class includes fixed-wing aircraft as their velocity cannot fall under some threshold to maintain the lift requirement. Thus, stabilizing such systems to a fixed state is a challenge.

Similar to motion planning in state space, in belief space the basic motion tasks can be defined as: *point-to-point motion*, which deals with driving the belief of the moving object from a given belief to a target belief, and *trajectory following*, which deals with following a trajectory in belief space. Both these tasks are more challenging in belief space than in state space. The point-to-point motion in belief space is required to construct a query-independent roadmap such as FIRM in belief space.

In fully-observable environments, Generalized PRM [27] performs point-to-point motion under motion uncertainty. In partially observable environments, under motion and sensing uncertainty, the Feedback-based Information RoadMap (FIRM) utilizes feedback controllers for the purpose of belief stabilization, and hence embeds the point-to-point motion behavior in belief space.

In Chapter 6, we present an instantiation of the abstract FIRM framework for nonholonomic systems using Dynamic Feedback Linearization-based (DFL-based) controllers. Adopting a DFL-based controller along with a stationary Kalman filter, we embed point-to-point motion in belief space for nonholonomic systems and instantiate a FIRM. In Chapter 7, we propose a concrete instantiation of the abstract FIRM framework that can handle non-point-stabilizable systems using Periodic controllers as belief stabilizers.

3.4 Real-time Replanning in Belief Space

Real-time replanning in the belief space is a vital capability in many applications for two main reasons. First, the belief dynamics are usually more random than the state dynamics because the belief is directly affected by the system observation. Therefore, a spurious data association (detecting a wrong feature ID), which is a very common failure in many sensing systems (such as vision or laser range finders), can cause large changes in the belief. Hence, online replanning is necessary to recover from such failures. Second, in practical applications, discrepancies between real models and the models used for computation are a significant source of error and cause the belief to occasionally have behaviors different than expected nominal behavior. Again, by being able to replan, the robot can recover from such belief deviations. Besides these two points, online replanning can help the robot to cope better with changes in the environment as well as recover from large deviations that

may occur in its location.

However, the majority of the state-of-the-art methods for planning in belief space are not equipped with online replanning capabilities. Sampling-based methods for solving belief space planning such as Belief RoadMap (BRM) [81], LQG-MP [94], [22], or [51] are single query methods (the solution is valid for a given initial belief). Therefore, in case of replanning from a new belief all (or most) of the computation needs to be redone, which prevents their usage when frequent real-time replanning is required. Similarly, point-based methods such as [76], [52], [13], [28] are rooted in a single initial belief.

On the other hand, trajectory optimization-based methods can be used for replanning in a Receding Horizon Control (RHC) scheme. In an RHC scheme (as will be detailed in Chapter 8), the optimal trajectory is computed within a limited horizon. Then, only the first step of the trajectory is executed and the rest of it is discarded. From the new point, then, the optimal trajectory is recomputed and 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 noise) with their nominal values (e.g., zero, for zero-mean Gaussian noise), and then treat the stochastic system as a deterministic one and use it in an RHC framework. Methods such as [35], [79], [25], [40], and [91] fall into this category and are among the trajectory optimization-based methods that can be used in an RHC framework, with an additional assumption that future observations are deterministic. However, in such an approach the optimization is carried out only within a limited horizon, and therefore the system may locally choose good actions but after a while may find itself in a high cost region. Moreover, removing the stochasticity of the system state (or belief) may lead to unreliable plans. In

this research, we propose a method based on FIRM embedded in a rollout policy framework that addresses these issues. In other words, in the proposed belief space planning method, we consider all possible future (random) observations. Moreover, we pick the FIRM plan as the base policy of the rollout policy method and thus incorporate the cost-to-go beyond the optimization horizon into the planning. We detail this approach in Chapter 8.

4. FEEDBACK-BASED INFORMATION ROADMAP (FIRM)

The main goal and contribution of this dissertation is to develop a "graph-based framework for motion planning under uncertainty" to address the intractability of the "constrained POMDP" problem (see Eq. (2.14)). In this chapter, we discuss feedback controllers and belief reachability under them. Then, we detail how we can generate a graph in belief space using feedback controllers. We call this graph "Feedback-based Information RoadMap (FIRM)" and describe how the constrained POMDP problem can be reduced to a tractable problem on this graph. By solving the dynamic programming problem on this graph, we compute the feedback tree for this problem. Subsequently, we provide performance guarantees on the solution by computing the success probability of the obtained solution. Finally, we define the concept of "probabilistic completeness under uncertainty", introduce tools to analyze belief space planners in general, and prove the probabilistic completeness of the FIRM framework in particular.

The goal of this chapter is to construct an abstract FIRM framework, assuming that there exists a mechanism to guarantee belief reachability. Hence, if for a class of systems, one can design a controller that satisfies the belief reachability requirement, the controller can be plugged into the abstract FIRM framework to generate a concrete instantiation of it, i.e., a representative graph in the belief space for the considered class of systems. We discuss concrete instantiations of this abstract framework in Chapters 5, 6, and 7.

We start this chapter by defining elements and assumptions needed in FIRM

construction. Accordingly, we transform the original intractable POMDP problem (see Eq. (2.14)) into a Semi-Markov Decision Process (SMDP) problem in belief space using a representative graph of local feedback controllers. Then, we construct an arbitrarily good approximation to the solution of this belief SMDP. Doing so, we obtain a tractable MDP, the so called "FIRM MDP". We discuss this derivation first for the obstacle-free case, and then we add the obstacles to the planning framework. Finally, we characterize the quality of the solution obtained by FIRM via its success probability and provide a generic algorithm for planning with FIRM.

4.1 Feedback Controllers and Reachability

As discussed in Chapter 2, in partially observable environments, the available data for decision-making at time step k can be compressed into the information-state or belief b_k. As discussed, using dynamic estimation schemes, belief can be propagated as $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ (See Eq. (2.4)), which can be presented as a one-step transition pdf $p(b_{k+1}|b_k, u_k)$ or a one-step transition probability $\mathbb{P}(B|b_k, u_k) = \int_B p(b_{k+1}|b_k, u_k)$, where $B \subset \mathbb{B}$.

In partially observable environments, at each stage, the decision making process is performed based on the belief at that stage. Thus, a (separated) feedback controller in partially observable spaces is a mapping from the belief space to the control space, i.e., $\mu(\cdot) : \mathbb{B} \to \mathbb{U}$. Consider a given set of such controllers denoted by $\mathbb{M}$. Later, we discuss the structure of the set $\mathbb{M}$ in detail.

Generating controls based on a given controller μ, the belief evolves according to a Markov chain whose one-step transition density function (if one exists) is denoted by $p_1(b'|b, \mu) : \mathbb{B} \times \mathbb{M} \times \mathbb{B} \to \mathbb{R}_{\geq 0}$ and defined as $p_1(b'|b, \mu) := p(b'|b, \mu(b))$. Thus, the feedback controller μ essentially induces a Markov chain with the transition probability $p_1(b'|b, \mu)$ over the belief space $\mathbb{B}$. In general the n-step transition pdf

under the controller μ is recursively defined as:

$$p_n(b''|b,\mu) := \int_{\mathbb{B}} p(b''|b',\mu(b'))p_{n-1}(b'|b,\mu)db' \tag{4.1}$$

In a similar way, $\mathbb{P}_1(B|b,\mu) : \mathbb{B} \times \mathbb{M} \times \mathfrak{B}_{\mathbb{B}} \to [0,1]$ denotes the transition probability from b to B under the local controller μ after one step, i.e., $\mathbb{P}_1(B|b,\mu) := \mathbb{P}(B|b,\mu(b))$. The set $\mathfrak{B}_{\mathbb{B}}$ is the sigma-algebra on the belief space $\mathbb{B}$. Similar, $\mathbb{P}_n(B|b,\mu) : \mathbb{B} \times \mathbb{M} \times \mathfrak{B}_{\mathbb{B}} \to [0,1]$ denotes the transition probability from b to B under the local controller μ after n steps.

$\mathcal{T}(A|b,\mu) : \mathbb{B} \times \mathbb{M} \times \mathfrak{B}_{\mathbb{B}} \to [0,\infty]$ denotes the hitting time on the set A, under the controller μ starting from belief b. Formally it is defined as:

$$\mathcal{T}(A|b,\mu) := \min\{k \geq 0, b_k \in A|b_0 = b,\mu\} \tag{4.2}$$

Following the notation in [61], ${}_A\mathbb{P}_n(\cdot|b,\mu) : \mathbb{B} \times \mathbb{M} \times \mathfrak{B}_{\mathbb{B}} \times \mathfrak{B}_{\mathbb{B}} \to [0,1]$ denotes the probability of transition from b to B in n steps under the controller μ "avoiding" the set A. ${}_A\mathbb{P}_n(\cdot|b,\mu)$ is called the n-step taboo probability and is formally defined as:

$$\begin{aligned} {}_A\mathbb{P}_n(B|b,\mu) &:= \Pr(b_n \in B, b_k \notin A,\ \forall k = 0,\cdots,n-1|b_0 = b,\ \mu) \\ &= \Pr(b_n \in B, \mathcal{T}(A) \geq n|b,\mu) \end{aligned} \tag{4.3}$$

We call region $B \subset \mathbb{B}$ a stopping region of the controller μ if we force the controller to stop executing as the belief reaches the region B, i.e., for all $b \in B$, we impose

$\mathbb{P}_1(B|b,\mu) = 1$.

Therefore, ${}_B\mathbb{P}_n(\cdot|b,\mu)$ for the controller μ with stopping region B is the transition probability from b to B under μ in exactly n steps (not less or more), i.e., ${}_B\mathbb{P}_n(B|b,\mu) := \Pr(b_n \in B, b_k \notin B,\ \forall k = 0,\cdots,n-1|b_0 = b,\ \mu)$. Therefore, we can write:

$$\mathbb{P}_n(B|b,\mu) = \sum_{k=0}^{n} {}_B\mathbb{P}_k(B|b,\mu) \tag{4.4}$$

Also, we can write the n-step transition probability as the probability of landing in the stopping region B in *at most* n *steps*:

$$\mathbb{P}_n(B|b,\mu) = \Pr(\mathcal{T}(B|b,\mu) \leq n) \tag{4.5}$$

Consider the controller μ that starts executing from belief b and stops executing when the belief enters region B. Thus, we can define $p(b'|b,\mu)$ as the pdf (if it exists) over the belief space, when the controller μ, invoked at b, stops executing, i.e.:

$$p(b'|b,\mu) := \lim_{n\to\infty} p_n(b'|b,\mu) \tag{4.6}$$

Similarly, $\mathbb{P}(B|b,\mu)$ represents the transition probability from b to B induced by μ, when the controller stops executing. Thus, the probability of landing in stopping region B in finite time is $\mathbb{P}(B|b,\mu)$, which is computed as:

$$\begin{aligned}\mathbb{P}(B|b,\mu) &:= \lim_{n\to\infty} \mathbb{P}_n(B|b,\mu) \\ &= \Pr(\text{belief ever enters } B) \\ &= \Pr(\mathcal{T}(B|b,\mu) < \infty)\end{aligned}$$

$$= \sum_{n=0}^{\infty} {}_B\mathbb{P}_n(B|b,\mu) \tag{4.7}$$

The stopping region B is called reachable under a controller μ starting from b if $\mathbb{P}(B|b,\mu) = 1$. The stopping region B is called accessible under a controller μ starting from b, if $\mathbb{P}(B|b,\mu) > 0$.

The stopping region B is called αT-reachable under a controller μ starting from b if $\mathbb{P}_T(B|b,\mu) = \Pr(\mathcal{T}(B|b,\mu) \leq T) > \alpha$, i.e., the controller can drive the system into B in fewer than T steps with a probability greater than α.

The reachability basin $\breve{B}$ associated with the pair (μ, B) is the set of all beliefs from which B is reachable under μ in the absence of constraints. Hence, the reachability and αT-reachability basins, respectively, are defined as follows:

$$\breve{B} = \{b \in \mathbb{B} \ : \ \mathbb{P}(B|b,\mu) = 1\}, \tag{4.8}$$

$$\breve{B}(\alpha, T) = \{b \in \mathbb{B} \ : \ \mathbb{P}_T(B|b,\mu) > \alpha\}, \tag{4.9}$$

Clearly, $B \subset \breve{B}$, and in practical cases, B is much smaller than $\breve{B}$.

4.2 FIRM Graph

In this section, we assume that there are no constraints (i.e., $F = \emptyset$), and we reduce planning over the entire belief space to planning over a representative graph in the belief space. Doing so, we can reduce the MDP problem in (2.11) over the continuous space into a tractable MDP problem over the graph.

The first step in the construction of the proposed framework is to sample a set of stabilizers $\{\mu^j\}$, where each stabilizer $\mu(\cdot)$ is a mapping from the belief space to the control space. Typically, every stabilizer is characterized by a d_v-vector of parameters

$\mathbf{v} \in \mathbb{R}^{d_v}$, i.e., we can denote the j-th stabilizer more rigorously as $\mu^j(\cdot; \mathbf{v}^j) : \mathbb{B} \to \mathbb{U}$. As a result, we can sample the parameters $\mathcal{V} = \{\mathbf{v}^j\}$ and then construct a stabilizer corresponding to each parameter. One can view the set $\mathcal{V}$ as a set of underlying PRM nodes in the parameter space.

FIRM nodes $\{B_j\}$ are disjoint sets in belief space, where the j-th node has to be chosen such that it is reachable under the j-stabilizer, i.e., $\mathbb{P}(B^j|b, \mu^j) = 1$, with a sufficiently large $\breve{B}$. We discuss the size of $\breve{B}$ further below. Note that, for practical purposes, the reachability condition can be replaced by αT-reachability if needed.

Consider a set of N samples $\{(\mu^i, B^i)\}_{i=1}^N$, where the reachability basin of the i-th sample is denoted by $\breve{B}^i$. Now, consider $\{B^i\}_{i=1}^N$ as the nodes of a graph. The node B^i is connected to the node B^j if, starting from any $b \in B^i$, we can reach B^j using μ^j. In other words B^i is connected to the node B^j if $B^i \subset \breve{B}^j$. Again, the reachability condition can be replaced by the αT-reachability condition.

For simple systems (linear with Gaussian noise) and some controllers (such as SLQG), the connection condition can be checked analytically. However, in general, checking this connection condition analytically may be very difficult. In such cases, the Markov chain induced by the controller can be simulated numerically (e.g., using particle-based methods). Accordingly, we can approximate the reachability (or αT-reachability) probability and check if the condition is true or not. Since this process is done offline, the computational burden can be tolerated. However, as we will see further below, in many cases, designing suitable *edge controllers* in practice increases the reachability probability such that practically one can assume the reachability is satisfied and so there is no need to propagate the probability distribution.

By definition, the graph node B associated with the controller μ acts as the stopping region of the controller. However, if the process under the stabilizer hits another graph node before its corresponding graph node, we can stop the controller

and pick the best controller from this intermediate node. Therefore, we can extend the stopping region for all controllers to the union of all nodes $\Psi := \cup_{l=1}^{N} B^l$. As a result, we will not necessarily have $\mathbb{P}(B^i|b, \mu^i) = 1$ since the process may hit some other node before B^i. However, we will have $\mathbb{P}(\Psi|b, \mu^i) = 1$ for all i in the absence of constraints.

To ease the connection step, and to handle cases where we have distant nodes (in sparse graphs), we can precede each stabilizer by a time-varying controller (referred to as the edge-controller). To illustrate this idea, consider two nodes B^i and B^j, where $B^i \not\subseteq \breve{B}^j$, i.e., B^i cannot be connected to B^j through μ^j. In this case, we can connect the underlying state nodes $\mathbf{v}^i$ and $\mathbf{v}^j$ in the state space by a finite trajectory e^{ij} (say with length l) and then design a time varying controller $\overline{\mu}_k^{ij}$, for $k = 0, 1, \cdots, l$ to track this finite trajectory. Therefore, if node B^i is in the reachability basin of $(\overline{\mu}_k^{ij}, \breve{B}^j)$, then obviously B^i would be in the reachability basin of (μ^{ij}, B^j), where $\mu^{ij} = \{\overline{\mu}_{0:l}^{ij}, \mu^j\}$. We call μ^{ij} the (i, j)-th local controller, as it connects node B^i to node B^j.

Formally, we define the constructed graph with the set of nodes $\mathbb{V} = \{B^i\}_{i=1}^{N}$ and the set of edges (or local controllers) $\mathbb{M} = \{\mu^{ij}\}$. The set of controllers available at B^i is denoted by $\mathbb{M}(i)$, i.e., the set of edges starting from B^i. Similar to PRM, in which the path (final solution) is constructed as a concatenation of edges on the roadmap, in FIRM, the policy is constructed by the concatenation of the local policies. However, it is worth noting that by this construction we still perform planning in a continuous space and do not discretize the control space.

By the term "macro-action", we mean a sequence of control signals (actions) [39, 40]. In other words, a macro-action is a sequence of open-loop policies. It is important to note that a local controller is *not* a macro-action, but rather a sequence of policies (macro-policy), each of which is a mapping from belief space to the con-

tinuous control space. Using macro-actions results in an open-loop policy which cannot compensate for the belief state deviation from the planned path. However, under local-controllers (macro-policies), the feedback nature of the controllers enable compensation for controllers. Thus, the belief can be steered toward a stopping region.

4.2.1 Belief Semi-Markov Decision Processes (Belief SMDP)

In this section, we reduce the planning (MDP) problem over the entire belief space into a planning (SMDP) problem over a subset of belief space, which is actually the union of FIRM graph nodes, i.e., $\Psi = \cup_j B^j$.

First, we generalize the concept of one-step cost $c(b, u) : \mathbb{B} \times \mathbb{U} \rightarrow \mathbb{R}_{\geq 0}$ to the one-step SMDP cost $C^s(b, \mu) : \mathbb{B} \times \mathbb{M} \rightarrow \mathbb{R}_{\geq 0}$, which represents the cost of invoking the local controller $\mu(\cdot)$ at the belief state b, i.e.,

$$C^s(b, \mu) := \sum_{t=0}^{\mathcal{T}} c(b_t, \mu(b_t)|b_0 = b), \tag{4.10}$$

where $\mathcal{T} := \mathcal{T}(\Psi|b, \mu)$.

According to the above definitions, the original POMDP, formulated using DP in Eq. (2.12), can be reduced to a Semi-Markov Decision Process (SMDP) [88] in the belief space, referred to as a *belief SMDP* or *Semi-POMDP (SPOMDP)*:

$$J^s(b) = \min_{\mu \in \mathbb{M}(i)} C^s(b, \mu) + \int_{\Psi} p(b'|b, \mu) J^s(b') db', \ \forall b \in B^i, \ \forall i. \tag{4.11}$$

The integration over the entire belief space in Eq. (2.12) is reduced to integration over the sampled nodes, i.e., Ψ, in Eq. (4.11) as μ stops executing.

4.3 FIRM MDP

The DP in (4.11), though computationally more tractable than the original POMDP, is defined on the continuous neighborhoods B^i and thus, still formidable to solve. However, for sufficiently small B^i's, the cost-to-go of all beliefs in B^i, are approximately equal. A similar statement holds for the incremental cost. Thus, we can define the transition cost and probabilities $C^g : \mathbb{V} \times \mathbb{M} \to \mathbb{R}$ and $\mathbb{P}^g : \mathbb{V} \times \mathbb{V} \times \mathbb{M} \to [0, 1]$ on the FIRM graph, i.e., over the finite space $\mathbb{V}$, such that $\mathbb{P}^g(B^j|B^i, \mu^{ij})$ is the transition probability from B^i to B^j under the local planner μ^{ij}. Similarly, $C^g(B^i, \mu^{ij})$ denotes the cost of invoking local planner μ^{ij} at the FIRM node B^i. These roadmap level quantities are defined using the following "piecewise constant approximation", which is an arbitrarily good approximation for smooth enough functions and sufficiently small B^i's:

$$\forall b \in B^i, \forall i, j \quad \begin{cases} C^g(B^i, \mu^{ij}) := C(b^i_c, \mu^{ij}) \approx C(b, \mu^{ij}), \\ \mathbb{P}^g(\cdot|B^i, \mu^{ij}) := \mathbb{P}(\cdot|b^i_c, \mu^{ij}) \approx \mathbb{P}(\cdot|b, \mu^{ij}), \end{cases} \tag{4.12}$$

where b^i_c is a point in B^i, for example, its center, if B^i is ball. This approximation essentially states that any belief in the region B^i is represented by b^i_c for the purpose of decision making.

Graph policy $\pi^g : \mathbb{V} \to \mathbb{M}$ is a function that returns a local planner for any given node of the FIRM graph. Let us denote the space of all graph policies by Π^g.

To choose a policy in Π^g, we define the graph cost-to-go from every graph node (FIRM node), and then pick the policy π^{g^*} that minimizes the defined cost-to-go. Starting from B_0 and using the policy π^g, the cost-to-go (or value) function $J^g(\cdot; \pi^g)$:

$\mathbb{V} \times \Pi^g \to \mathbb{R}$ is formally defined as:

$$J^g(B_0; \pi^g) = \sum_{k=0}^{\infty} \mathbb{E}\left[C^g(B_k, \pi^g(B_k))\right]$$
$$s.t. \quad \mathbb{P}(B_{k+1}|B_k, \pi^g) \tag{4.13}$$

Optimal cost-to-go $J^g : \mathbb{V} \to \mathbb{R}_{\geq 0}$ is defined as follows:

$$J^g(B^i) = \min_{\pi^g \in \Pi^g} J^g(B^i; \pi^g) \tag{4.14}$$

Given the approximation in Eq. (4.12), the DP equation in Eq. (4.11) becomes:

$$\begin{aligned}
J^g(B^i) = J^s(b_c^i) &= \min_{\mu \in \mathbb{M}(i)} C^s(b_c^i, \mu) + \int_{\Psi} p(b'|b_c^i, \mu) J^s(b') db' \\
&= \min_{\mu \in \mathbb{M}(i)} C^s(b_c^i, \mu) + \sum_j \int_{B^j} p(b'|b_c^i, \mu) J^s(b') db' \\
&\approx \min_{\mu \in \mathbb{M}(i)} C^g(B^i, \mu) + \sum_j \int_{B^j} p(b'|b_c^i, \mu) J^g(B^j) db' \\
&= \min_{\mu \in \mathbb{M}(i)} C^g(B^i, \mu) + \sum_j J^g(B^j) \mathbb{P}(B^j|b_c^i, \mu) \\
&= \min_{\mu \in \mathbb{M}(i)} C^g(B^i, \mu) + \sum_j J^g(B^j) \mathbb{P}^g(B^j|B^i, \mu), \ \forall i
\end{aligned} \tag{4.15}$$

Accordingly, we can get the graph feedback $\pi^g : \mathbb{V} \to \mathbb{M}$ through the following DP:

$$J^g(B^i) = \min_{\mu \in \mathbb{M}(i)} C^g(B^i, \mu) + \sum_j \mathbb{P}^g(B^j|B^i, \mu) J^g(B^j), \ \forall i \tag{4.16a}$$

$$\pi^g(B^i) = \arg\min_{\mu \in \mathbb{M}(i)} C^g(B^i, \mu) + \sum_j \mathbb{P}^g(B^j|B^i, \mu) J^g(B^j), \ \forall i \tag{4.16b}$$

Thus, the original POMDP over the entire belief space, becomes a finite N_v-state

MDP in Eq. (4.16) defined on the finite set of FIRM nodes $\mathbb{V} = \{B^i\}_{i=1}^{N_v}$. We call the MDP in Eq. (4.16), the FIRM MDP in the absence of obstacles. It is worth noting that $J^g(\cdot) : \mathbb{V} \to \mathbb{R}$ is the cost-to-go function over the FIRM nodes, which assigns a cost-to-go for every FIRM node B^i and the mapping $\pi^g(\cdot) : \mathbb{V} \to \mathbb{M}$ is a mapping over the FIRM graph, from FIRM nodes into the set of local controllers that returns the optimal local controller that has to be taken at any FIRM node. Given $C^g(B, \mu)$ for all (B, μ) pairs, the DP in Eq. (4.16) can be solved *offline* using standard techniques such as the value/policy iteration to yield a feedback policy π^g over FIRM nodes $\{B^i\}$.

4.3.1 Incorporating Obstacles (Constraints) into FIRM MDP

In the presence of obstacles (i.e., state or control constraints), we cannot always ensure that the local controller $\mu^{ij}(\cdot)$ can drive any $b \in B^i$ into B^j with probability one. Instead, we have to specify the failure probabilities that the robot collides with an obstacle (hits the failure set $F \subset \mathbb{X} \times \mathbb{U}$).

We can generalize the stationary transition probabilities from $\mathbb{P}(\cdot|b, \mu) : \mathfrak{B}_{\mathbb{B}} \times \mathbb{M} \times \mathbb{B} \to [0, 1]$ into $\mathbb{P}(\cdot|b, \mu) : \{\mathfrak{B}_{\mathbb{B}}, F\} \times \mathbb{M} \times \mathbb{B} \to [0, 1]$ such that $\mathbb{P}(F|b, \mu^{ij})$ denotes the probability of hitting failure set F before hitting stopping region Ψ under μ starting from b.

Similarly, we generalize edge transition probabilities from $\mathbb{P}^g(\cdot|B, \mu) : \mathbb{V} \times \mathbb{V} \times \mathbb{M} \to [0, 1]$ into $\mathbb{P}^g(\cdot|B, \mu) : \{\mathbb{V}, F\} \times \mathbb{V} \times \mathbb{M} \to [0, 1]$ such that $\mathbb{P}^g(F|B^i, \mu) := \mathbb{P}(F|b_c^i, \mu)$. Again, given the function $\mathbb{P}(\cdot|b, \mu)$ is smooth and given that sets B^j are suitably small, we can make the approximation $\mathbb{P}^g(\cdot|B^i, \mu) := \mathbb{P}(\cdot|b_c^i, \mu) \approx \mathbb{P}(\cdot|b, \mu)$ for all $b \in B^i$ and for all i.

Finally, we generalize the cost-to-go function from $J^g : \mathbb{V} \to \mathbb{R}_{\geq 0}$ into $J^g : \{\mathbb{V}, F\} \to \mathbb{R}_{\geq 0}$, such that $J^g(F)$ is a user-defined suitably high cost for hitting

obstacles. Note that the cost-to-go from the goal node is zero, i.e., $J^g(B^{goal}) = 0$. Therefore, we can modify Eq. (4.16) to incorporate constraints, by repeating the procedure in the previous subsection to get the FIRM MDP in the presence of obstacles:

$$J^g(B^i) = \min_{\mu \in \mathbb{M}(i)} C^g(B^i, \mu) + J^g(F)\mathbb{P}^g(F|B^i, \mu) + \sum_j J^g(B^j)\mathbb{P}^g(B^j|B^i, \mu), \tag{4.17a}$$

$$\pi^g(B^i) = \arg\min_{\mu \in \mathbb{M}(i)} C^g(B^i, \mu) + J^g(F)\mathbb{P}^g(F|B^i, \mu) + \sum_j J^g(B^j)\mathbb{P}^g(B^j|B^i, \mu). \tag{4.17b}$$

All that is required to solve the above DP equation is the values of the costs $C^g(B^i, \mu)$ and the transition probability functions $\mathbb{P}^g(\cdot|B^i, \mu)$. Thus, the main difference from the obstacle free case is the addition of a "failure" state to the FIRM MDP along with associated probabilities of failure from various nodes B^i.

When a policy is executed, there is a stopping condition that stops the execution. In the motion planning problem under uncertainty (also known as the stochastic shortest path problem), the system stops if it reaches the goal region in the belief space B^{goal} or it fails (e.g., hits an obstacle), i.e., F happens.

For the system to not fall into an infinite cycle or to not stop before the termination condition is satisfied, the cost of taking any action before the stopping condition is satisfied has to be positive. It also has to be zero when the stopping condition is

met:

$$\begin{cases} c(b,u) = 0 & \text{if } \ (b \in B^{goal}) \ \text{ or } \ (F \text{ happens}) \\ c(b,u) \geq \epsilon > 0 & \text{if } \ \text{otherwise} \end{cases} \tag{4.18}$$

which results in a zero cost-to-go from the goal region, i.e., $J^g(B^{goal}) = 0$, for all $b \in B^{goal}$. To steer the system towards the successful stopping region, i.e., B^{goal}, and steer the system away from the failure stopping regions, we set the cost-to-go of the goal region to zero and we set the cost-to-go of the failure regions to a high value to keep the system away from them:

$$\begin{cases} J^g(B^{goal}) = 0 \\ J^g(F) = J_F \end{cases} \tag{4.19}$$

where J_F is a user-defined high cost-to-go for failing.

4.3.2 Overall Policy π

The overall feedback $\pi : \mathbb{B} \to \mathbb{U}$ is generated by combining the global policy π^g on the graph and local policies $\{\mu^{ij}\}$. Suppose at the k-th time step the active local controller is shown by μ_k^*. It remains unchanged $\mu_{k+1}^* = \mu_k^*$, and keeps generating control signals based on the belief b_k at each time step, until the belief reaches the corresponding stopping region, Ψ. Once the belief enters the stopping region $\Psi = \cup_j B^j$, it is in a graph node, say $B_k^* \in \mathbb{V}$. Accordingly, the global policy π^g chooses the next local controller, i.e., $\mu_{k+1}^* = \pi^g(B_k^*)$. Thus, this hybrid policy is

stated as follows:

$$u_k = \pi(b_k) = \begin{cases} \mu_k^*(b_k),\ \mu_k^* = \pi^g(B_{k-1}^*),\ \text{if } b_k \in B_{k-1}^* \\ \mu_k^*(b_k),\ \mu_k^* = \mu_{k-1}^*, \qquad \text{if } b_k \notin \Psi \end{cases} \tag{4.20}$$

Given the initial belief is b_0, if b_0 is in one of the graph nodes, then we just choose the best local controller using π^g. However, if b_0 does not belong to any of the graph nodes, we first make a singleton set $B^0 = \{b_0\}$ and connect it to the graph nodes based on the connect methods discussed earlier in this chapter. Denoting the outgoing edges (local controllers) from B^0 as $\mathbb{M}(0)$, we compute the transition cost $C^g(B^0, \mu)$, the transition probabilities $\mathbb{P}^g(B^j|B^0, \mu)$ for all j, and the failure probability $\mathbb{P}(F|B^0, \mu)$ for invoking local controllers $\mu \in \mathbb{M}(0)$ at B^0. Then, we choose the best initial controller μ_0^* as:

$$\mu_0^* = \begin{cases} \arg\min\limits_{\mu \in \mathbb{M}(0)} \{C^g(B^0, \mu) + \mathbb{P}^g(F|B^0, \mu) J^g(F) \\ \qquad\qquad + \sum\limits_j \mathbb{P}^g(B^j|B^0, \mu) J^g(B^j)\}, \qquad \text{if } \nexists r,\ \text{s.t. } b_0 \in B^r \\ \pi^g(B^r), \qquad\qquad \text{if } \exists r, \text{s.t. } b_0 \in B^r \end{cases} \tag{4.21}$$

It is worth noting that computing μ_0^* is the only part of the computation that depends on the initial belief b_0 and that has to be performed online, i.e., if a large deviation occurs, μ_0^* is the only part that needs to be reproduced for the new initial point. That is after μ_0^* drives the system to a graph node, then the optimal policy is known as it is associated with the graph node. Computing μ_0^* is feasible online as $\mathbb{M}(0)$ contains a limited number of edges.

4.3.3 Success Probability

We also quantify the quality of the solution π in the presence of obstacles. To this end, we require the probability of success of the policy π^g at the higher level Markov chain on FIRM nodes given by Eq. (4.17b). Without loss of generality let us assume that the first node B^1 is the goal node B^{goal}. The DP in Eq. (4.17) has $N+1$ states $\{F, B^{goal}, B^2, \cdots, B^N\}$ that can be decomposed into three disjoint classes: the failure class $\{F\}$, the goal class $\{B^{goal}\}$, and the transient class $\{B^2, B^3, \cdots, B^{N+1}\}$. The goal and failure classes are absorbing recurrent classes of this Markov chain. As a result, the transition probability matrix of this higher level $N+1$ state Markov chain can be decomposed as follows [66]:

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_f & 0 & 0 \\ 0 & \mathcal{P}_{goal} & 0 \\ \mathcal{R}_f & \mathcal{R}_{goal} & \mathcal{Q} \end{bmatrix}. \tag{4.22}$$

where, $\mathcal{P}_{goal} = \mathbb{P}^g(B^1|B^1, \cdot) = 1$ and $\mathcal{P}_f = \mathbb{P}^g(F|F, \cdot) = 1$, since goal and failure classes are the absorbing recurrent classes, i.e., the system stops once it reaches the goal or it fails. $\mathcal{Q}$ is a matrix that represents the transition probabilities between transient nodes in the transient class, whose (i,j)-th element is $\mathcal{Q}[i,j] = \mathbb{P}^g(B^{i+1}|B^{j+1}, \pi^g(B^{j+1}))$. Vectors $\mathcal{R}_{goal}$ and $\mathcal{R}_f$ are $(N-1)\times 1$ vectors that represent the probability of transient nodes $\mathbb{V} \setminus B^{goal}$ getting absorbed into the goal and failure node, respectively, i.e., $\mathcal{R}_{goal}[j] = \mathbb{P}^g(B^1|B^{j+1}, \pi^g(B^{j+1}))$ and $\mathcal{R}_f[j] = \mathbb{P}^g(F|B^{j+1}, \pi^g(B^{j+1}))$. Then, it can be shown that the success probability from any desired node $B^i \in \mathbb{V} \setminus B^{goal}$ is given as follows [66]:

$$\mathbb{P}(\text{success}|B^i, \pi^g) := \mathbb{P}(B^{goal}|B^i, \pi^g)$$

$$= \Gamma_{i-1}^{T}(I - \mathcal{Q})^{-1}\mathcal{R}_{goal}, \quad \forall i \geq 2, \tag{4.23}$$

where Γ_i is a column vector with all elements equal to zero except the i-th element which is set to one. Note that the vector $\mathcal{P}^s = (I - \mathcal{Q})^{-1}\mathcal{R}_{goal}$ includes the success probability from every graph node.

In the next section, we will discuss the success probability in more detail in the context of probabilistic completeness. However, according to the computed $\mathbb{P}(\text{success}|B^i, \pi^g)$, one can compute the success probability from any given initial belief b_0 as

$$\mathbb{P}(\text{success}|b_0, \pi) = \sum_j \mathbb{P}(B^j|b_0, \mu_0^*)\mathbb{P}(\text{success}|B^j, \pi^g), \tag{4.24}$$

where μ_0^* is given by Eq. (4.21). Then, this success probability is compared with a minimum acceptable success probability, denoted by p_{min}. If the condition $\mathbb{P}(\text{success}|b_0, \pi) > p_{min}$ is not satisfied, then the number of nodes in the graph has to be increased until the condition is satisfied. If, from the initial point b_0, a successful policy in the class of admissible policies exists, then this procedure will eventually find a successful policy by increasing the number of nodes, due to the probabilistic completeness of the method, which is discussed in Section 4.5 of the current chapter.

4.4 Generic FIRM Algorithms

The generic algorithms for the offline construction of FIRM and online planning with FIRM are presented in Algorithms 1 and 2, respectively. Concrete instantiations of these algorithms for SLQG-FIRM are given in Chapter 5.

As mentioned earlier, most approaches for planning in belief space in continuous state, action, and observation spaces result in query-dependent plans. However, one

Algorithm 1: Generic Construction of the FIRM graph (Offline)

1 Sample a set of stabilizer parameters $\mathcal{V} = \{\mathbf{v}^i\}$ and construct stabilizers $\mathbb{M} = \{\mu^i\}$ accordingly;
2 Sample set of belief nodes $\mathbb{V} = \{B^i\}$ such that they satisfy the reachability condition;
3 Connect the belief nodes using local controllers μ^{ij};
4 For each B^i and $\mu \in \mathbb{M}(i)$, compute the transition cost $C^g(B^i, \mu)$, and transition probabilities $\mathbb{P}^g(B^j|B^i, \mu)$ and $\mathbb{P}^g(F|B^i, \mu)$ associated with invoking μ at B^i;
5 Solve the graph DP in Eq. (4.17) to compute feedback π^g over graph nodes, and compute the π accordingly;

Algorithm 2: Generic planning (or replanning) on FIRM (Online)

1 Given an initial belief b_0, invoke the controller $\mu_0(\cdot)$ in Eq. (4.21), to take the robot into some FIRM node B;
2 **while** $B \neq B^{goal}$ **do**
3 Given the system is in FIRM node B, invoke the global feedback policy π^g to choose the local feedback policy $\mu(\cdot) = \pi^g(B)$;
4 Let the local controller $\mu(\cdot)$ execute until the robot is absorbed into a FIRM node B' or until it hits the failure set;
5 **if** *Collision happens* **then return** Collision;
6 Update current node $B \leftarrow B'$;

of the contributions of FIRM is that its construction does not depend on the query. In Algorithms 1 and 2, it is assumed that the goal is fixed for all queries; in this case in the planning phase we are only robust to changes in the starting point of the query. However, to make the algorithms also robust to changes in the goal belief, one can just move the last line of Algorithm 1 to the first line of Algorithm 2. Note that the computationally expensive part of Algorithm 1 is the computation of edge costs, which is independent of the start and goal location of the submitted query.

4.5 Probabilistic Completeness Under Uncertainty

In this section, we extend the concept of probabilistic completeness of planning algorithms for deterministic systems to the concept of probabilistic completeness of planning algorithms under uncertainty based on [4]. Accordingly, in the next subsection, we discuss the probabilistic completeness of FIRM. We start by reviewing the definition of success and probabilistic completeness in the deterministic case, and then we extend these definitions to the stochastic case.

In the deterministic case, such as conventional PRM, the outcome of the planning algorithm is a path. Thus, success is defined for paths: for a given initial and goal point, a successful path is a path connecting the start point to the goal point, which entirely lies in the obstacle-free space.

In the absence of uncertainty, a sampling-based motion planning algorithm is probabilistically complete if by increasing the number of samples, the probability of finding a successful path, if one exists, asymptotically approaches one.

In the presence of uncertainty, success cannot be defined for a path but is instead defined for a policy. Indeed, on a given path, different policies may result in different success probabilities. Moreover, under uncertainty, one can only assign a probability for reaching goal. Thus, to define success for a policy we consider a threshold $p_{min} \in [0, 1]$ and decide about success or failure accordingly.

In the presence of uncertainty, the solution of the planning algorithm is a function, called a closed-loop policy or feedback. Therefore, success is defined for policies: for a given initial belief b_0 and goal region B^{goal}, a successful policy is a policy under which the probability of reaching the goal from the given initial point is greater than some predefined threshold p_{min}. In other words, π is successful for a given b_0 if $\mathbb{P}(\text{success}|b_0, \pi) := \mathbb{P}(B^{goal}|b_0, \pi) > p_{min}$.

In sampling-based methods, a policy is parametrized by a set of samples. These samples can be in the state or belief space, depending on the algorithm. Let us denote these samples in a generic space by $\{\gamma_1, \gamma_2, \cdots, \gamma_N\}$. Thus, we can highlight the dependency of the sampling-based policy on the samples by the notation $\pi(\cdot; \{\gamma_1, \gamma_2, \cdots, \gamma_N\})$. The number of samples is denoted by N.

To define strong probabilistic completeness under uncertainty (SPCUU) let us suppose there exists a successful policy $\check{\pi}$. Then a sampling-based motion planning algorithm is SPCUU if increasing the number of samples without bound causes the probability of finding a successful policy to approach one. In other words, if there exists a successful policy $\check{\pi}$, then we have the following property for the sampling-based policy π:

$$\lim_{N\to\infty} \mathbb{P}(B^{goal}|b_0, \pi) > p_{min}, \tag{4.25}$$

where N is the number of samples in the sampling-based method.

Achieving an algorithm that is SPCUU requires searching in the entire space of policies, which is a computationally intractable task. Usually, in solving POMDPs the space of admissible policies is restricted to a sufficiently rich subset of policy space, denoted by Π, within which the method searches for the best policy. Restricting the successful policy to the set Π, we define a weaker notion of probabilistic completeness under uncertainty:

Suppose there exists a successful policy $\check{\pi} \in \Pi$. Then, a sampling-based motion planning algorithm is probabilistically complete under uncertainty (PCUU), if increasing the number of samples without bound, the probability of finding a successful policy approaches one. In other words, if there exists a successful policy $\check{\pi} \in \Pi$, then for the sampling-based policy π, we have $\lim_{N\to\infty} \mathbb{P}(B^{goal}|b_0, \pi) > p_{min}$.

As discussed earlier, in FIRM, inspired by the sampling-based PRM framework, this reduction from the entire function space to the restricted set of policies Π is performed by sampling feedback local planners and concatenating them. Therefore, the structure of local planners defines the set Π. Each local planner μ^{ij} is parametrized by its corresponding parameter $\mathbf{v}^j$. However, as mentioned, we can consider the set $\mathcal{V} = \{\mathbf{v}^i\}$ as the set of nodes which form an underlying PRM. Thus, any policy $\pi \in \Pi$ is parametrized by the set of underlying PRM nodes $\mathcal{V} = \{\mathbf{v}^i\}_{i=1}^{N_v}$. We highlight this dependency explicitly through the notation $\pi(\cdot; \mathcal{V})$. Therefore, the PCUU condition for FIRM can be written more explicitly as:

$$\lim_{N_v \to \infty} \mathbb{P}(B^{goal} | b_0, \pi(\cdot; \mathcal{V})) > p_{min}. \tag{4.26}$$

For a concrete instantiation of FIRM, we can explicitly characterize the set Π. For example, in SLQG-FIRM, Π is the set of all possible policies that can be generated by concatenating LQG controllers.

4.5.1 Probabilistic Completeness of FIRM

Obviously, FIRM-based methods are not SPCUU algorithms. However, in this section, we show that under mild practical conditions, FIRM-based methods are PCUU algorithms. We first provide an analysis of the local planners in belief space, and then state the assumptions more rigorously.

Throughout this section, the norm $\|\cdot\|$ denotes the supremum norm, when it is applied to functions. The norm $\|\cdot\|_{op}$ is applied on operators and it stands for the operator norm [49]. It is worth noting that in this section, by the word "continuous" we mean "Lipschitz continuous." Finally, we assume that $\mathbb{X}_{free}$ is a compact set.

$\mathcal{X} = (x, b) \in \mathbb{X}_h$ is referred to as hyper-state (or h-state), which is a state-belief pair. The space of all h-states is called hyper-state space (h-state space) $\mathbb{X}_h = \mathbb{X} \times \mathbb{B}$.

The $p^{\mu}(\mathcal{X}'|\mathcal{X})$ denotes the one-step transition pdf induced by the local controller μ, over the h-state space. Also, let $\mathbb{P}_n(S|\mathcal{X},\mu)$ denote the transition probability from h-state $\mathcal{X}$ into the set $S \subset \mathbb{X}_h$ in at most n steps.

The role of the (i,j)-th local planner or local controller is to drive the belief from the region B^i to its stopping region B^j in the belief space. For notational simplicity, we ignore the case that the controller can stop in any FIRM node, and we restrict its stopping region to B^j. In the presence of obstacles, we extend the concept of stopping region to include obstacles also. The stopping regions $\{B^j\}$ in the belief space and the stopping region F in the state space, both can be extended to the h-state space, respectively denoted by $\{\mathcal{B}^j\}$ and $\mathcal{F}$, where $\mathcal{B}^j \subset \mathbb{X}_h$ and $\mathcal{F} \subset \mathbb{X}_h$ are defined as:

$$\mathcal{B}^j := \{(X,b)|X \in \mathbb{X}_{free}, b \in B^j\}, \tag{4.27}$$

$$\mathcal{F} := \{(X,b)|X \in F, b \in \mathbb{B}\}, \tag{4.28}$$

$$\mathcal{S}^j := \mathcal{B}^j \cup \mathcal{F}, \quad \overline{\mathcal{S}}^j := \mathbb{X}_h \setminus \mathcal{S}^j \tag{4.29}$$

where $\mathcal{S}^j$ and $\overline{\mathcal{S}}^j$, respectively, denote the entire stopping region and transient region under the local controller μ^{ij}.

If, under dynamics induced by the local planner, the system reaches the target node $\mathcal{B}^j$, the local planner is considered to be successful, and if the system hits an obstacle, the local planner is considered to fail. The success probability of a local planner, i.e., the absorption probability into FIRM nodes, is computed through solving the following integral equation that results from the law of total probability:

$$\mathbb{P}(\mathcal{B}^j|\mathcal{X},\mu^{ij}) = \int_{\mathbb{X}_h} p^{\mu^{ij}}(\mathcal{X}'|\mathcal{X})\mathbb{P}(\mathcal{B}^j|\mathcal{X}',\mu^{ij})d\mathcal{X}'$$

$$= \int_{\mathcal{B}^j} p^{\mu^{ij}}(\mathcal{X}'|\mathcal{X})d\mathcal{X}' + \int_{\overline{\mathcal{S}^j}} p^{\mu^{ij}}(\mathcal{X}'|\mathcal{X})\mathbb{P}(\mathcal{B}^j|\mathcal{X}', \mu^{ij})d\mathcal{X}'. \tag{4.30}$$

where the second equality in Eq. (4.30) follows from substituting the following conditions, inherited from FIRM construction, into the first integral:

$$\mathbb{P}(\mathcal{B}^j|\mathcal{X}, \mu^{ij}) = \begin{cases} 1, \text{ if } \mathcal{X} \in \mathcal{B}^j \\ 0, \text{ if } \mathcal{X} \in \mathcal{F} \end{cases}. \tag{4.31}$$

Henceforth, we drop indices i and j to simplify expressions. Thus, we can write:

$$\begin{aligned} \mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) &= \int_{\mathcal{B}} p^{\mu}(\mathcal{X}'|\mathcal{X})d\mathcal{X}' + \int_{\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\mathbb{P}(\mathcal{B}|\mathcal{X}', \mu)d\mathcal{X}' \\ &= R(\mathcal{X}) + \mathbf{T}_{\mathcal{S}}\left[\mathbb{P}(\mathcal{B}|\cdot, \mu)\right](\mathcal{X}), \end{aligned} \tag{4.32}$$

where the operator $\mathbf{T}_{\mathcal{S}}$ and the function $R(\mathcal{X})$ are defined as:

$$\mathbf{T}_{\mathcal{S}}\left[f(\cdot)\right](\mathcal{X}) := \int_{\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})f(\mathcal{X}')d\mathcal{X}', \quad R(\mathcal{X}) := \int_{\mathcal{B}} p^{\mu}(\mathcal{X}'|\mathcal{X})d\mathcal{X}'. \tag{4.33}$$

The solution of the integral equation in Eq. (4.32) is expressed in the following as a Liouville-Neumann series [49], similar to the solution of the inhomogeneous Fredholm equation of second type [49].

$$\mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) = \sum_{n=1}^{\infty} \mathbf{T}_{\mathcal{S}}^{n}\left[R(\cdot)\right](\mathcal{X}). \tag{4.34}$$

We show that the series in Eq. (4.34) is a convergent series by resorting to the following assumption, which is a weaker version of the aforementioned FIRM condition on the design of nodes and local controllers.

Assumption 1. *We assume that there exists some time step N, at which the controller stops with a positive probability. Mathematically, there exists an $N < \infty$ and $\beta > 0$ such that $\mathbb{P}_N(\mathcal{S}^j|\mathcal{X}, \mu^{ij}) \geq \beta > 0$, for all $\mathcal{X}$.*

This assumption is almost always true, as it rephrases the role of a controller in driving the system toward the target region. For example, if we have Gaussian noise (as is the case in SLQG-FIRM), the assumption is true for $N = 1$ regardless of the utilized controller.

Lemma 1. *Given Assumption 1, we have:*

$$\begin{cases} \|\mathbf{T}_{\mathcal{S}}^n\|_{op} \leq 1, & n < N \\ \|\mathbf{T}_{\mathcal{S}}^n\|_{op} \leq 1 - \beta < 1, & n \geq N \\ \sum_{n=0}^{\infty} \|\mathbf{T}_{\mathcal{S}}^n\|_{op} \leq c < \infty. \end{cases} \tag{4.35}$$

Before proving Lemma 1, we state and prove the following lemma:

Lemma 2. *Consider the bounded function $0 \leq f(\mathcal{X}) \leq 1$, and kernel $k(\mathcal{X}', \mathcal{X}) \geq 0$. Then, for any set $\mathcal{A}$, we have:*

$$\| \int_{\mathcal{A}} k(\mathcal{X}', \mathcal{X}) f(\mathcal{X}') d\mathcal{X}' \| \leq \| \int_{\mathcal{A}} k(\mathcal{X}', \mathcal{X}) d\mathcal{X}' \|. \tag{4.36}$$

Proof. Given the properties of $f(\cdot)$ and $k(\cdot, \cdot)$, we have $k(\mathcal{X}', \mathcal{X}) f(\mathcal{X}') \leq k(\mathcal{X}', \mathcal{X})$, for all $\mathcal{X}$ and $\mathcal{X}'$. Taking the integral from both sides with respect to $\mathcal{X}'$ and then taking the supremum norm with respect to $\mathcal{X}$, the result follows. □

Now we prove Lemma 1.

Proof. If we denote the domain of operator $\mathbf{T}_{\mathcal{S}}$ by $\mathcal{D}$, we know that for all $f \in \mathcal{D}$, we have $0 \leq f(\mathcal{X}) \leq 1$, because $f(\mathcal{X})$ is the probability of reaching given set $\mathcal{S}$ under

some given controller invoked at point $\mathcal{X}$. Thus, it cannot be negative or greater than one and based on Lemma 2, we have:

$$\begin{aligned}\|\mathbf{T}_{\mathcal{S}}[f]\| &= \|\int_{\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X}) f(\mathcal{X}') d\mathcal{X}'\| \leq \|\int_{\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}'\| \\ &= \|\mathbb{P}_1(\overline{\mathcal{S}}|\mathcal{X},\mu)\| \leq 1. \end{aligned} \tag{4.37}$$

Therefore, based on the definition of operator norm, we have:

$$\|\mathbf{T}_{\mathcal{S}}\|_{op} = \sup_{f(\cdot)} \{\|\mathbf{T}_{\mathcal{S}}[f]\| \ : \ \forall f \in \mathcal{D}, \|f\| \leq 1\} \leq 1. \tag{4.38}$$

According to Assumption 1, there exists a finite number N, such that:

$$\inf_{\mathcal{X}} \mathbb{P}_n(\mathcal{S}|\mathcal{X},\mu) = \beta > 0 \ \ \forall n > N, \tag{4.39}$$

where "inf" and "sup" denote the infimum and supremum, respectively. Thus, we have

$$\begin{aligned}\|\mathbb{P}_n(\overline{\mathcal{S}}|\mathcal{X},\mu)\| &= \sup_{\mathcal{X}}(1 - \mathbb{P}_n(\mathcal{S}|\mathcal{X},\mu)) = 1 - \inf_{\mathcal{X}} \mathbb{P}_n(\mathcal{S}|\mathcal{X},\mu) \\ &= 1 - \beta < 1 \quad \forall n > N. \end{aligned} \tag{4.40}$$

Let us denote the n-th iterated kernel of $\mathbf{T}_{\mathcal{S}}$ as $p_n(\mathcal{X}'|\mathcal{X},\mu)$. Since this iterated kernel is a pdf, we have $p_n(\mathcal{X}'|\mathcal{X},\mu) \geq 0, \ \forall \mathcal{X}, \forall \mathcal{X}', \forall n$. We can write:

$$\begin{aligned}&\|\mathbf{T}_{\mathcal{S}}^{N}[f]\| = \|\int_{\overline{\mathcal{S}}} p_N(\mathcal{X}'|\mathcal{X},\mu) f(\mathcal{X}') d\mathcal{X}'\| \\ &\leq \|\int_{\overline{\mathcal{S}}} p_N(\mathcal{X}'|\mathcal{X},\mu) d\mathcal{X}'\| = \|\mathbb{P}_N(\overline{\mathcal{S}}|\mathcal{X},\mu)\| \leq \alpha < 1, \end{aligned} \tag{4.41}$$

where $\alpha = 1 - \beta$, and similar to Eq. (4.38), we get $\|\mathbf{T}_{\mathcal{S}}^{N}\|_{op} \leq \alpha < 1$. From the operator norm properties, we have:

$$\|\mathbf{T}_{\mathcal{S}}^{N+1}\|_{op} \leq \|\mathbf{T}_{\mathcal{S}}^{N}\|_{op}\|\mathbf{T}_{\mathcal{S}}\|_{op} \leq \alpha < 1$$

and similarly for all $n \geq N$, we have:

$$\|\mathbf{T}_{\mathcal{S}}^{n}\|_{op} \leq \alpha < 1 \;\; \forall n \geq N.$$

Now, consider the series: $\sum_{i=1}^{\infty} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op}$. We can split the sum to smaller pieces as follows:

$$\sum_{n=1}^{\infty} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op} = \sum_{n=1}^{N} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op} + \sum_{i=1}^{\infty} \sum_{n=iN+1}^{(i+1)N} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op}.$$

But because $\|\mathbf{T}_{\mathcal{S}}^{n+1}\|_{op} \leq \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op}$ for all $n \geq N$, we have

$$\sum_{n=iN+1}^{(i+1)N} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op} \leq N\|\mathbf{T}_{\mathcal{S}}^{iN}\|_{op}.$$

Also, we know

$$\|\mathbf{T}_{\mathcal{S}}^{iN}\|_{op} \leq \|\mathbf{T}_{\mathcal{S}}^{N}\|_{op}^{i} \leq \alpha^{i}$$

and thus, we have:

$$\begin{aligned}\sum_{n=1}^{\infty} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op} &= \underbrace{\sum_{n=1}^{N} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op}}_{\leq N} + \sum_{i=1}^{\infty} \sum_{n=iN+1}^{(i+1)N} \|\mathbf{T}_{\mathcal{S}}^{n}\|_{op} \\ &\leq N + \sum_{i=1}^{\infty} N\alpha^{i} = N + \frac{N}{1-\alpha} = c < \infty.\end{aligned}$$

□

Corollary 1. *The series* $\sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R]$ *is a convergent series, and therefore we can define the resolvent operator* $(I - \mathbf{T}_{\mathcal{S}})^{-1}[R] = \sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R]$*, where* $\|(I - \mathbf{T}_{\mathcal{S}})^{-1}\|_{op} \leq c < \infty$.

Proof. We know $\|R\| \leq 1$, and thus we can write:

$$\|\sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R]\| \leq \sum_{n=0}^{\infty} \|\mathbf{T}_{\mathcal{S}}^n\|_{op}\|R\| \leq \sum_{n=0}^{\infty} \|\mathbf{T}_{\mathcal{S}}^n\|_{op} \leq c < \infty.$$

Thus, series $\sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R]$ is a convergent series and we can define the operator $(I - \mathbf{T}_{\mathcal{S}})^{-1}[R] = \sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n[R]$. We have

$$\|(I - \mathbf{T}_{\mathcal{S}})^{-1}\|_{op} = \|\sum_{n=0}^{\infty} \mathbf{T}_{\mathcal{S}}^n\|_{op} \leq c < \infty. \tag{4.42}$$

□

According to Corollary 1, the success probability of the local controller μ can be written using the defined resolvent operator as:

$$\mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) = (I - \mathbf{T}_{\mathcal{S}})^{-1}[R(\cdot)](\mathcal{X}). \tag{4.43}$$

As the first result of this section (Proposition 1), we aim to show that this absorption probability varies continuously with respect to changes in the parameters of the local planner. However, we will first state two assumptions.

Assumption 2. *We assume the local planning law and induced transition probabilities are smooth, i.e.,*

- *Local control laws are continuous in their parameters, i.e., for the* (i, j)*-th local controller, mapping* $\mu^{ij}(\cdot; \mathbf{v}^j) : \mathbb{B} \to \mathbb{U}$ *is a continuous function in its parameter* $\mathbf{v}^j$.

- *The transition pdf on h-state, i.e., $p(\mathcal{X}'|\mathcal{X}, u)$ is a continuous function of the control u, i.e., there exists a $c_1 < \infty$, such that $\|p(\mathcal{X}'|\mathcal{X}, u) - p(\mathcal{X}'|\mathcal{X}, \check{u})\| \leq c_1\|u - \check{u}\|$.*

Finally, we state the following assumption, in which we emphasize the fact that, as $\mathbf{v} \to \check{\mathbf{v}}$, the transition probability induced by the local controller $\mu(\cdot; \mathbf{v})$ into the sets $\mathcal{B}$ and $\check{\mathcal{B}}$ has to converge also, which is a reasonable assumption for a smooth control law.

Assumption 3. *Consider the controllers $\mu(\cdot; \mathbf{v})$, and $\check{\mu}(\cdot; \check{\mathbf{v}})$, whose corresponding extended absorption regions are denoted by $\mathcal{B}$ and $\check{\mathcal{B}}$, respectively. We assume that there exist real numbers $r > 0$ and $c' < \infty$, such that for $\|\mathbf{v} - \check{\mathbf{v}}\| \leq r$, we have:*

$$\|\mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)\| \leq c'\|\mathbf{v} - \check{\mathbf{v}}\| \tag{4.44}$$

where $\ominus$ is the symmetric difference operator, i.e., $\mathcal{B} \ominus \check{\mathcal{B}} = (\mathcal{B} \setminus \check{\mathcal{B}}) \cup (\check{\mathcal{B}} \setminus \mathcal{B})$.

Now we state the following proposition on the continuity of the success probability of local planners:

Proposition 1. (Continuity of absorption probabilities): Given Assumptions 1, 2, and 3, the absorption probability $\mathbb{P}(B^j|b, \mu^{ij})$ is continuous in parameter $\mathbf{v}^j$ for all $i, j,$ and b.

We first state the following lemma on the continuity of the transition probability in the local controller parameter.

Lemma 3. *Given Assumption 2, there exists a $c_2 < \infty$ such that*

$$\|p(\mathcal{X}'|\mathcal{X}, \mu(b; \mathbf{v})) - p(\mathcal{X}'|\mathcal{X}, \check{\mu}(b; \check{\mathbf{v}}))\| \leq c_2\|\mathbf{v} - \check{\mathbf{v}}\|. \tag{4.45}$$

Proof. The result directly follows by combining two parts of Assumption 2. □

Now we are ready to prove Proposition 1.

Proof. To show $\mathbb{P}(\mathcal{B}|\mathcal{X},\mu)$ is continuous in $\mathbf{v}$, we perturb $\mathbf{v}$ to some $\check{\mathbf{v}}$, such that $\|\mathbf{v}-\check{\mathbf{v}}\|<r$. The local controller associated with node $\check{\mathbf{v}}$ is referred to as $\check{\mu}$, whose successful absorption region is denoted by $\check{\mathcal{B}}$ and stopping region is $\check{\mathcal{S}}$. Similarly the corresponding transient operator and recurrent function are referred to as $\check{\mathbf{T}}_{\check{\mathcal{S}}}$ and $\check{R}$. Finally, the success probability associated with the perturbed node $\check{\mathbf{v}}$ is $\mathbb{P}(\check{\mathcal{B}}|\mathcal{X},\check{\mu})$. To shorten the statements, we refer to $\mathbb{P}(\mathcal{B}|\mathcal{X},\mu)$ and $\mathbb{P}(\check{\mathcal{B}}|\mathcal{X},\check{\mu})$ respectively by $\mathfrak{P}(\mathcal{X})$ and $\check{\mathfrak{P}}(\mathcal{X})$. As a result of node perturbation, the success probability is perturbed as:

$$\begin{aligned}
&\mathbb{P}(\mathcal{B}|\mathcal{X},\mu)-\mathbb{P}(\check{\mathcal{B}}|\mathcal{X},\check{\mu}):=\mathfrak{P}-\check{\mathfrak{P}}=R+\mathbf{T}_{\mathcal{S}}[\mathfrak{P}]-\check{R}-\check{\mathbf{T}}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\\
&=R-\check{R}+\mathbf{T}_{\mathcal{S}}[\mathfrak{P}]-\mathbf{T}_{\mathcal{S}}[\check{\mathfrak{P}}]+\mathbf{T}_{\mathcal{S}}[\check{\mathfrak{P}}]-\mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]+\mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]-\check{\mathbf{T}}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\\
&=(R-\check{R})+\mathbf{T}_{\mathcal{S}}[\mathfrak{P}-\check{\mathfrak{P}}]+(\mathbf{T}_{\mathcal{S}}-\mathbf{T}_{\check{\mathcal{S}}})[\check{\mathfrak{P}}]+(\mathbf{T}_{\check{\mathcal{S}}}-\check{\mathbf{T}}_{\check{\mathcal{S}}})[\check{\mathfrak{P}}],
\end{aligned}$$

where

$$\mathbf{T}_{\check{\mathcal{S}}}\left[f(\cdot)\right](\mathcal{X}):=\int_{\overline{\check{\mathcal{S}}}}p^{\mu}(\mathcal{X}'|\mathcal{X})f(\mathcal{X}')d\mathcal{X}'. \tag{4.46}$$

Let us define the operators $\mathbf{T}_{\Delta\mathcal{S}}:=(\mathbf{T}_{\mathcal{S}}-\mathbf{T}_{\check{\mathcal{S}}})$ and $\Delta\mathbf{T}_{\check{\mathcal{S}}}:=(\mathbf{T}_{\check{\mathcal{S}}}-\check{\mathbf{T}}_{\check{\mathcal{S}}})$. Now, based on Corollary 1, we can write:

$$\mathfrak{P}-\check{\mathfrak{P}}=(I-\mathbf{T}_{\mathcal{S}})^{-1}\left[R-\check{R}+\mathbf{T}_{\Delta\mathcal{S}}[\check{\mathfrak{P}}]+\Delta\mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\right], \tag{4.47}$$

and thus the following inequality holds on the supremum norm of the perturbation

of the absorption probability:

$$
\begin{aligned}
&\|\mathfrak{P} - \check{\mathfrak{P}}\| \\
&\leq \|(I - \mathbf{T}_{\mathcal{S}})^{-1}\|_{op} \left(\|R - \check{R}\| + \|\mathbf{T}_{\Delta\mathcal{S}}[\check{\mathfrak{P}}]\| + \|\Delta\mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\| \right) \\
&\leq c \left(\|R - \check{R}\| + \|\mathbf{T}_{\Delta\mathcal{S}}[\check{\mathfrak{P}}]\| + \|\Delta\mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\| \right) \\
&= c \left(\|K_1(\mathcal{X})\| + \|K_2(\mathcal{X})\| + \|K_3(\mathcal{X})\| \right), \qquad (4.48)
\end{aligned}
$$

where $K_1(\mathcal{X}) := R(\mathcal{X}) - \check{R}(\mathcal{X})$, $K_2(\mathcal{X}) := \mathbf{T}_{\Delta\mathcal{S}}[\check{\mathfrak{P}}(\cdot)](\mathcal{X})$, and $K_3(\mathcal{X}) := \Delta\mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}(\cdot)](\mathcal{X})$. In the following we bound K_1, K_2, and K_3, and thus bound $\|\mathfrak{P} - \check{\mathfrak{P}}\|$, accordingly.

4.5.1.1 Bound for $K_1(\mathcal{X})$

The supremum norm of $K_1(\mathcal{X})$ is:

$$
\begin{aligned}
&\|K_1(\mathcal{X})\| = \|R(\mathcal{X}) - \check{R}(\mathcal{X})\| \\
&= \| \int_{\mathcal{B}} p^{\mu}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' - \int_{\check{\mathcal{B}}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \| \\
&= \| \int_{\mathcal{B}\cap\check{\mathcal{B}}} [p^{\mu}(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X})] d\mathcal{X}' \\
&+ \int_{\mathcal{B}-\check{\mathcal{B}}} p^{\mu}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' - \int_{\check{\mathcal{B}}-\mathcal{B}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \| \\
&\leq \int_{\mathcal{B}\cap\check{\mathcal{B}}} \|p^{\mu}(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X})\| d\mathcal{X}' \\
&+ \| \int_{\mathcal{B}-\check{\mathcal{B}}} p^{\mu}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' + \int_{\check{\mathcal{B}}-\mathcal{B}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) d\mathcal{X}' \| \\
&\overset{\text{from (4.45)}}{\leq} \int_{\mathcal{B}\cap\check{\mathcal{B}}} c_2 \|\mathbf{v} - \check{\mathbf{v}}\| d\mathcal{X}' + \|\mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)\| \\
&\qquad + \|\mathbb{P}_1(\check{\mathcal{B}} \ominus \mathcal{B}|\mathcal{X}, \check{\mu})\|
\end{aligned}
$$

$$\overset{\text{from (4.44)}}{\leq} c_2'\|\mathbf{v}-\check{\mathbf{v}}\| + 2c'\|\mathbf{v}-\check{\mathbf{v}}\| = \gamma_1\|\mathbf{v}-\check{\mathbf{v}}\|, \tag{4.49}$$

where $c_2' < \infty$ and $\gamma_1 = c_2' + 2c' < \infty$. In the penultimate inequality, we also used the fact that $\mathbb{P}_1(\check{\mathcal{B}} - \mathcal{B}|\mathcal{X}, \check{\mu}) \leq \mathbb{P}_1(\check{\mathcal{B}} \ominus \mathcal{B}|\mathcal{X}, \check{\mu})$ and $\mathbb{P}_1(\mathcal{B} - \check{\mathcal{B}}|\mathcal{X}, \mu) \leq \mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)$ because $\check{\mathcal{B}} - \mathcal{B} \subseteq \check{\mathcal{B}} \ominus \mathcal{B}$ and $\mathcal{B} - \check{\mathcal{B}} \subseteq \mathcal{B} \ominus \check{\mathcal{B}}$.

4.5.1.2 Bound for $K_2(\mathcal{X})$

We have:

$$\begin{aligned}
&\|K_2(\mathcal{X})\| = \|\mathbf{T}_{\Delta\mathcal{S}}[\check{\mathfrak{P}}]\| = \|\mathbf{T}_{\mathcal{S}}[\check{\mathfrak{P}}] - \mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\| \\
&= \|\int\limits_{\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\check{\mathfrak{P}}(\mathcal{X}')d\mathcal{X}' - \int\limits_{\overline{\check{\mathcal{S}}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\check{\mathfrak{P}}(\mathcal{X}')d\mathcal{X}'\| \\
&= \|\int\limits_{\overline{\mathcal{S}}-\overline{\check{\mathcal{S}}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\check{\mathfrak{P}}(\mathcal{X}')d\mathcal{X}' - \int\limits_{\overline{\check{\mathcal{S}}}-\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\check{\mathfrak{P}}(\mathcal{X}')d\mathcal{X}'\| \\
&\leq \|\int\limits_{\overline{\mathcal{S}}-\overline{\check{\mathcal{S}}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\check{\mathfrak{P}}(\mathcal{X}')d\mathcal{X}' + \int\limits_{\overline{\check{\mathcal{S}}}-\overline{\mathcal{S}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\check{\mathfrak{P}}(\mathcal{X}')d\mathcal{X}'\| \\
&= \|\int\limits_{\overline{\mathcal{S}}\ominus\overline{\check{\mathcal{S}}}} p^{\mu}(\mathcal{X}'|\mathcal{X})\check{\mathfrak{P}}(\mathcal{X}')d\mathcal{X}'\| \overset{\text{from (4.36)}}{\leq} \|\int\limits_{\overline{\mathcal{S}}\ominus\overline{\check{\mathcal{S}}}} p^{\mu}(\mathcal{X}'|\mathcal{X})d\mathcal{X}'\| \\
&= \|\mathbb{P}_1(\overline{\mathcal{S}} \ominus \overline{\check{\mathcal{S}}}|\mathcal{X}, \mu)\| \leq \|\mathbb{P}_1(\overline{\mathcal{B}} \ominus \overline{\check{\mathcal{B}}}|\mathcal{X}, \mu)\| \\
&= \|\mathbb{P}_1(\mathcal{B} \ominus \check{\mathcal{B}}|\mathcal{X}, \mu)\| \overset{\text{from (4.44)}}{\leq} \gamma_2\|\mathbf{v}-\check{\mathbf{v}}\|,
\end{aligned} \tag{4.50}$$

where $\gamma_2 = c' < \infty$. The penultimate inequality and equality follow from the relations $\overline{\mathcal{S}} \ominus \overline{\mathcal{S}'} \subseteq \overline{\mathcal{B}} \ominus \overline{\mathcal{B}'}$ and $\overline{\mathcal{B}} \ominus \overline{\mathcal{B}'} = \mathcal{B} \ominus \mathcal{B}'$, respectively.

4.5.1.3 Bound for $K_3(\mathcal{X})$

We have:

$$\begin{aligned}
&\|K_3(\mathcal{X})\| = \|\Delta \mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\| = \|\mathbf{T}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}] - \check{\mathbf{T}}_{\check{\mathcal{S}}}[\check{\mathfrak{P}}]\| \\
&= \| \int_{\bar{\check{\mathcal{S}}}} p^{\mu}(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' - \int_{\bar{\check{\mathcal{S}}}} p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \| \\
&= \| \int_{\bar{\check{\mathcal{S}}}} \left(p^{\mu}(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) \right) \check{\mathfrak{P}}(\mathcal{X}') d\mathcal{X}' \| \\
&\leq \int_{\bar{\check{\mathcal{S}}}} \| p^{\mu}(\mathcal{X}'|\mathcal{X}) - p^{\check{\mu}}(\mathcal{X}'|\mathcal{X}) \| \| \check{\mathfrak{P}}(\mathcal{X}') \| d\mathcal{X}' \\
&\overset{\text{from (4.45)}}{\leq} \int_{\bar{\check{\mathcal{S}}}} c_2 \|\mathbf{v} - \check{\mathbf{v}}\| d\mathcal{X}' = \gamma_3 \|\mathbf{v} - \check{\mathbf{v}}\|,
\end{aligned} \tag{4.51}$$

where $\gamma_3 < \infty$.

Therefore, based on Eq. (4.49), Eq. (4.50), Eq. (4.51), and Eq. (4.48), we can conclude that:

$$\|\mathbb{P}(\mathcal{B}|\mathcal{X}, \mu) - \mathbb{P}(\check{\mathcal{B}}|\mathcal{X}, \check{\mu})\| \leq \gamma \|\mathbf{v} - \check{\mathbf{v}}\|, \tag{4.52}$$

where $\gamma = c(\gamma_1 + \gamma_2 + \gamma_3) < \infty$, which completes the proof that the absorption probability under the controller μ is continuous in the PRM node $\mathbf{v}$. □

Now we present the main result regarding the probabilistic completeness of FIRM-based methods:

Theorem 1. *Given Assumptions 1, 2, and 3, any planning algorithm under uncertainty that is generated based on the FIRM framework (i.e., guarantees belief node reachability and induces a roadmap in the belief space with independent edge costs) is probabilistically complete under uncertainty (PCUU).*

Before starting with the proof of Theorem 1, we state the following proposition that concludes the continuity of the success probability of π (overall planner) given the continuity of the success probability of the individual local planners (μ^{ij}s).

Proposition 2. (Continuity of success probability of π): The success probability $\mathbb{P}(\text{success}|b_0, \pi)$ is continuous in $\mathcal{V}$, if the absorption probabilities $\mathbb{P}(B^j|b, \mu^{ij})$ are continuous in $\mathbf{v}^j$ for all i, j, and b.

Proof. Given that $\mathbb{P}(B^j|b, \mu^{ij})$ is continuous in $\mathbf{v}^j$, for all i, j, we want to show that $\mathbb{P}(\text{success}|\pi, b_0)$ is continuous in all $\mathbf{v}^j$. First, let us look at the structure of the success probability.

$$\mathbb{P}(\text{success}|b_0, \pi) = \mathbb{P}(B(\mu_0)|b_0, \mu_0)\mathbb{P}(\text{success}|B(\mu_0), \pi^g), \tag{4.53}$$

where μ_0 is computed using Eq. (4.21). The term $\mathbb{P}(B(\mu_0)|b_0, \mu_0)$ in the right hand side of Eq. (4.53) is continuous because the continuity of $\mathbb{P}(B^j|b, \mu^{ij})$ for all i, j is assumed in this proposition. Thus, we only need to show the continuity of the second term in Eq. (4.53). Without loss of generality we can consider $B^i = B(\mu_0)$. Then, we need to show that $\mathbb{P}(\text{success}|B^i, \pi^g)$ is continuous in $\mathbf{v}^i$ for all i.

As we saw in Section 4.3.3, the probability of success from the i-th FIRM node is as follows:

$$\mathbb{P}(\text{success}|B^i, \pi^g) = \Gamma_i^T (I - \mathcal{Q})^{-1} \mathcal{R}_g, \tag{4.54}$$

Moreover, we can consider $B^{goal} = B^N$ without loss of generality; then, the (i, j)-th element of matrix $\mathcal{Q}$ is $\mathcal{Q}[i, j] = \mathbb{P}(B^i|B^j, \pi^g(B^j))$, and the j-th element of vector $\mathcal{R}_g$ is $\mathcal{R}_g[j] = \mathbb{P}(B^N|B^j, \pi^g(B^j))$. Since we considered the B^j as the stopping region of

the local controller μ^{ij}, we have:

$$\mathbb{P}(B^j|B^i, \mu^{il}) = 0, \text{ if } l \neq j. \tag{4.55}$$

Therefore, all the non-zero elements in the matrices $\mathcal{R}_g$ and $\mathcal{Q}$ are of the form $\mathbb{P}(B^j|B^i, \mu^{ij})$. Thus, given the continuity of $\mathbb{P}(B^j|b, \mu^{ij})$, the transition probability $\mathbb{P}(B^j|B^i, \mu^{ij})$ is continuous and the matrices $\mathcal{R}_g$ and $\mathcal{Q}$ are continuous. Therefore, $\mathbb{P}(\text{success}|B^i, \pi^g)$ and thus $\mathbb{P}(\text{success}|b_0, \pi)$ are continuous in underlying PRM nodes.

□

Now we are ready to prove Theorem 1:

Proof. Based on the definition of probabilistic completeness under uncertainty, if there exists a successful policy $\check{\pi}$, FIRM has to find a successful policy π as the number of FIRM nodes increases unboundedly. Thus, we start by assuming that there exists a successful policy $\check{\pi} \in \Pi$ for a given initial belief b_0. Since each policy in Π is parametrized by a PRM graph, there exists a PRM with nodes $\check{\mathcal{V}} = \{\check{\mathbf{v}}^i\}_{i=1}^N$ that parametrizes the policy $\check{\pi}$. Since $\check{\pi}$ is a successful policy, we know $\mathbb{P}(\text{success}|b_0, \check{\pi}) > p_{min}$. Thus, we can define $\epsilon^* = \mathbb{P}(\text{success}|b_0, \check{\pi}) - p_{min} > 0$.

Given Assumptions 1, 2, and 3, and based on Propositions 1 and 2, we know that $\mathbb{P}(\text{success}|b_0, \pi)$ is continuous with respect to the parameters of the local planners, i.e., for any $\epsilon > 0$, there exists a $\delta > 0$, such that if $\|\mathcal{V} - \check{\mathcal{V}}\| < \delta$, then $|\mathbb{P}(\text{success}|b_0, \pi(\cdot; \mathcal{V})) - \mathbb{P}(\text{success}|b_0, \check{\pi}(\cdot; \check{\mathcal{V}}))| < \epsilon$. The notation $\|\mathcal{V} - \check{\mathcal{V}}\| < \delta$ means that $\|\mathbf{v}^i - \check{\mathbf{v}}^i\| < \delta$, for all i, or equivalently, $\mathbf{v}^i \in \check{\Omega}_i$, for all i, where $\check{\Omega}_i$ is a ball with radius δ, centered at $\check{\mathbf{v}}^i$.

Therefore, for the introduced ϵ^*, there exists a δ^* and corresponding regions $\{\check{\Omega}_i\}_{i=1}^N$, such that if we have a PRM whose nodes (or a subset of nodes – A subset

of nodes is sufficient, because the success probability is a non-decreasing function in terms of the number of nodes) satisfy the condition $\mathbf{v}_i^* \in \check{\Omega}_i$, for all $i = 1, \cdots, N$, then the planner π parametrized by this PRM has a success probability greater than p_{min}, i.e., $\mathbb{P}(\text{success}|b_0, \pi(\cdot; \mathcal{V})) > p_{min}$, and hence π is successful.

Since $\delta > 0$, the regions $\check{\Omega}_i$ have nonempty interiors. Consider a PRM with a sampling algorithm, under which there is nonzero probability of sampling in $\check{\Omega}_i$, such as uniform sampling. In other words, consider a sampling algorithm under which $\check{\Omega}_i$ are the sets with nonzero probability measures. Thus, starting with any PRM, if we increase the number of nodes, a PRM node will eventually be chosen at every $\check{\Omega}_i$, with probability one. Therefore the policy constructed based on these nodes will have a success probability greater than p_{min}, i.e., we eventually get a successful policy if one exists. Thus, FIRM is probabilistically complete under uncertainty (PCUU). $\square$

The basic idea of probabilistic completeness under uncertainty stems from an idea similar to the one in the path isolation-based analysis for planners in deterministic systems. Roughly speaking, in the path isolation argument for sampling-based planners in the absence of uncertainty, if there is a successful path and a non-zero neighborhood of this path, in which every path is successful, we can eventually find a path in this neighborhood, by increasing the number of samples, unboundedly. Similarly, in the presence of uncertainty, if there is a successful policy, it is parametrized by some parameters (set of PRM nodes, in FIRM). Thus, if there exists a non-zero measure neighborhood of these parameters, within which selected parameters lead to a successful policy, then we can eventually reach a successful policy by increasing the number of samples unboundedly and choosing samples in the target neighborhoods.

4.6 Rollout Policy for Dynamic Replanning in Belief Space

To handle frequent changes in the environment, changes in the goal location, large deviations in the robot's location, and in general to handle discrepancies between models used for simulation and the actual models, we resort to dynamic replanning in belief space. In this section, we discuss the extension of the RHC and Rollout policy [15] to the belief space to design a principled scheme for online replanning in the belief space that can cope with large deviations and changes in the environment map.

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

$$\begin{aligned} \pi_{0:\infty}(\cdot) &= \arg\min_{\Pi_{0:\infty}} \sum_{k=0}^{\infty} \mathbb{E}\left[c(b_k, \pi_k(b_k))\right] \\ s.t. \quad & b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\ & x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)) \end{aligned} \tag{4.56}$$

In the above problem, we seek for a sequence of policies $\pi_{0:\infty} = \{\pi_0(\cdot), \pi_1(\cdot), \pi_2(\cdot), \cdots\}$, where π_k maps any given 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 the one introduced earlier in this chapter. However, we keep the time-varying format for the reasons that will be clear further below.

As discussed earlier, solving the POMDP problem is computationally intractable over continuous state, action, and observation spaces. However, the more difficult problem is to solve the POMDP problem online as needed to cope with changes in the models or map (the case in our application). In this section, we discuss how

FIRM can be exploited to make it possible to re-solve POMDPs online.

Constructing a FIRM and Computing the optimal graph policy π^g, it is straightforward to handle the changes in the start and goal location (see Algorithms 3 and 4). Note that if the desired factor is the success probability only, one can initialize the success probability from b_0 to 0; i.e., $P^*(b_0) = 0$ before the `for` loop in Algorithm 3. Then, the one can replace the condition in line 8 of Algorithm 3 by $\mathbb{P}(B|b_0, \mu)P^{success}(B) > P^*(b_0)$ and add the condition update statement $P^*(b_0) = \mathbb{P}(B|b_0, \mu)P^{success}(B)$ into the `for` loop.

Algorithm 3: (Re)plan_from

1 **input** : Start belief b_0, Graph Cost-to-go $J^g(\cdot)$, FIRM nodes $\mathbb{V} = \{B^i\}$, Success probabilities $P^{success}(\cdot)$
2 **output** : Next Local Controller μ^*
3 Find r neighboring nodes $\mathfrak{N} = \{B^i\}_{i=1}^r$ to b_0;
4 Set $J^*(B) = \infty$;
5 **for** $B \in \mathfrak{N}$ **do**
6 Construct local planner μ from b_0 to B;
7 Compute the transition cost $C(b_0, \mu)$ and probability $\mathbb{P}(B|b_0, \mu)$;
8 **if** $C(b_0, \mu) + \mathbb{P}(B|b_0, \mu)J(B) + (1 - \mathbb{P}(B|b_0, \mu))J(F) < J^*(B)$ **then**
9 $J^*(B) = C(b_0, \mu) + \mathbb{P}(B|b_0, \mu)J(B) + (1 - \mathbb{P}(B|b_0, \mu))J(F)$;
10 $\mu^* = \mu$;
11 **return** μ^*;

Receding horizon control (often referred to as rolling horizon or model predictive control) was originally designed for deterministic systems [36] 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 [25, 56, 84, 98]. In the most common form of RHC [15] the stochastic system is approximated with a deterministic system by replacing the uncertain quantities with their typical values

Algorithm 4: (Re)plan_to

input : Goal state $\mathbf{v}^{goal}$, FIRM Graph $\mathcal{G} = \{\mathbb{V}, \mathbb{M}\}$
output : FIRM feedback π^g
$B^{goal} \leftarrow$ Sample the FIRM node associated with $\mathbf{v}^{goal}$;
Add B^{goal} to the FIRM graph; i.e., $\mathbb{V} \leftarrow \mathbb{V} \cup \{B^{goal}\}$;
Connect B^{goal} to its r nearest neighbors using edges $\{\mu^{(i,goal)}\}$. Also, $\mathbb{M} \leftarrow \mathbb{M} \cup \{\mu^{(i,goal)}\}$;
Compute the cost-to-go J^g and feedback π^g over the FIRM nodes by solving the graph DP in Eq. (4.17);
return π^g;

(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:

$$
\begin{aligned}
u_{0:T} = \arg\min_{\mathbb{U}_{0:T}} & \sum_{k=0}^{T} c(b_k^d, u_k) \\
s.t. \quad & b_{k+1}^d = \tau(b_k^d, u_k, z_{k+1}^{ml}) \\
& z_{k+1}^{ml} = \arg\max_z p(z|x_{k+1}^d) \\
& x_{k+1}^d = f(x_k^d, u_k, 0)
\end{aligned}
\tag{4.57}
$$

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}, the the belief b_{k+1} will be different that b^d_{k+1}. 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 5 recaps this procedure.

Algorithm 5: RHC 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.(4.57) starting from b^d_0 = 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);
```

State-of-the-art methods such as [91] and [77] utilize the RHC-in belief space. This framework is also called Partially-closed loop RHC (PCLRHC) [91] since it partially exploits some information about future observations (i.e., z^{ml}) and does not fully ignore them.

There are some issues regarding the presented RHC framework. First, due to the limited horizon and ignoring the cost-to-go beyond the horizon, the method may get stuck 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 approximation of the cost and unreliable control actions. To overcome these issues, researchers have proposed variants of RHC and different frameworks, such as the "rollout policy", based on the idea of repeated planning [15].

Another class of methods that aims to reduce the complexity of the stochastic planning problem in Eq. (4.17) is the class of rollout policies [15], 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:

$$\begin{aligned} \pi_{0:T}(\cdot) = \arg\min_{\Pi_{0:T}} \mathbb{E}\left[\sum_{k=0}^{T} c(b_k, \pi_k(b_k)) + \widetilde{J}(b_{T+1})\right] \\ s.t. \quad b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\ x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)) \end{aligned} \tag{4.58}$$

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 detailed as shown in Algorithm 6.

Algorithm 6: 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.(4.58) 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 $\widetilde{J}$. In the following, we propose a rollout policy in the belief space based on the FIRM-based cost-to-go.

In 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 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 the truncated policy can drive the belief into a FIRM node at the end of horizon. However, since the belief evolution is random, we may not be able to guarantee the belief reaches a FIRM node at the end of a deterministic horizon T. Therefore, instead of truncating the policy over time, we truncate the policy over the belief and leave the horizon length to be random (denoted by $\mathcal{T}$) as follows:

$$\begin{aligned}
\pi_{0:\infty}(\cdot) = \arg\min_{\widetilde{\Pi}} \mathbb{E}\left[\sum_{k=0}^{\mathcal{T}} c(b_k, \pi_k(b_k)) + \widetilde{J}(b_{\mathcal{T}+1})\right] \\
s.t. \quad b_{k+1} = \tau(b_k, \pi_k(b_k), z_k), \quad z_k \sim p(z_k|x_k) \\
x_{k+1} = f(x_k, \pi_k(b_k), w_k), \quad w_k \sim p(w_k|x_k, \pi_k(b_k)) \\
b_{\mathcal{T}+1} \in \cup_j B^j,
\end{aligned} \tag{4.59}$$

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

$$\widetilde{J}(b_{\mathcal{T}+1}) = J^g(B^j) \tag{4.60}$$

where $\widetilde{\Pi}$ is a restricted set of policies under which the belief will reach a FIRM node B^j in finite time (possibly random). More rigorously, if $\pi \in \widetilde{\Pi}$ and $\pi = \{\pi_1, \pi_2, \cdots\}$, then for finite $\mathcal{T}$, we have $\mathbb{P}(b_{\mathcal{T}+1} \in \cup_j B^j | \pi) = 1$, i.e., belief will enter into a FIRM node under π after finite time. In other words, the last constraint in (4.59) is redundant as it is already satisfied by the definition of $\widetilde{\Pi}$. However, it is explicitly written in (4.59) to emphasize this constraint. Also, 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 $\widetilde{J}(\cdot)$ (Equation (4.60)). In Chapter 8, we implement FIRM-based rollout policy to handle changes in the environment map and large deviations in the robot's location.

4.7 Discussion

In summary, in FIRM we aim to transform the original POMDP problem into a belief SMDP problem and solve it on a subset of belief space. Given the smoothness of the cost function and transition probabilities, the solution of the FIRM MDP is arbitrarily close to the solution of the belief SMDP over FIRM nodes. The important characteristic of FIRM is that it is solved offline and thus performing the online phase of planning (or replanning) is computationally feasible in online. To exploit the generic FIRM framework, one has to find (B, μ) pairs, where B is reachable (or αT-reachable) under μ, as FIRM nodes and edges. Also, transition costs and probabilities need to be computed. Finally, the corresponding FIRM MDP needs to be solved, which provides a global feedback policy on the graph that can be used in planning, as detailed in Algorithm 2. SLQG-FIRM, presented in the next chapter, is an instance of FIRM, in which the design of local controllers μ^{ij} and FIRM nodes

B^i is based on the properties of SLQG controllers.

5. FIRM INSTANTIATION FOR HOLONOMIC SYSTEMS

In this chapter, we develop a concrete FIRM for holonomic systems where belief reachability is accomplished by Stationary Linear Quadratic Gaussian (SLQG) controllers. We refer to this variant of FIRM as the SLQG-based FIRM. We discuss how an SLQG controller can satisfy belief node reachability. We characterize SLQG-based FIRM nodes and edges, and we provide concrete sampling and connecting methods.particular.

We start this chapter by reviewing LQG controllers. Then, we restrict our attention to the class of systems that SLQG-FIRM can handle and address how we can define nodes in belief space to satisfy reachability using SLQG controllers. Next, we explain the procedure of constructing local controllers (i.e., FIRM edges) and the SLQG-based FIRM graph. We compute transition probabilities and costs associated with each graph edge and compute the graph feedback. Finally, we describe algorithms for planning with this framework and demonstrate its performance on different systems and in different scenarios.

5.1 Preliminaries on SLQG

To construct the SLQG-based FIRM we assume the noise is Gaussian. It is worth noting that the abstract FIRM framework does not make any assumption on the form of the belief (e.g., it does not require the belief to be Gaussian). We start this section by defining the notation needed to deal with Gaussian beliefs.

We denote the random estimation vector by x^+, whose distribution is $b_k =$

$p(x_k^+) = p(x_k|z_{0:k}, u_{0:k-1})$, and denote the mean and covariance of x^+ by $\widehat{x}^+ = \mathbb{E}[x^+]$ and $P = \mathbb{E}[(x^+ - \widehat{x}^+)(x^+ - \widehat{x}^+)^T]$, respectively. Denoting the Gaussian belief space by $\mathbb{GB}$, every function $b(\cdot) \in \mathbb{GB}$, can be characterized by a mean-covariance pair $(\widehat{x}^+, P)$. Abusing notation, we also show this pair by $b \equiv (\widehat{x}^+, P) \in \mathbb{R}^n \times \mathbb{S}_+^n$, where the mean vector belongs to the n-dimensional Euclidean space $\mathbb{R}^n$ and the covariance matrix belongs to the space of all positive semi-definite $n \times n$ matrices $\mathbb{S}_+^n$.

An LQG controller is composed of a Kalman filter as the state estimator and an LQR controller (see Fig. 2.1). Thus, the belief dynamic $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ is known and comes from the Kalman filtering equations, and the controller $u_k = \mu(b_k)$ that acts on the belief comes from the LQR equations. Considering a quadratic cost for state error and control error, LQG is an optimal controller for linear systems with Gaussian noise [15]. However, it is also often used for stabilization of nonlinear systems around a given trajectory or around a given point.

Time-varying LQG is designed to track a given trajectory, in which at every time step, a different feedback policy is utilized. Stationary LQG is a time-invariant policy, in which LQG is designed around a given point, say $\mathbf{v}$, to steer the state of the system to $\mathbf{v}$ [15]. In Sections 2.2.1 and 2.2.2 we have discussed these controllers in detail.

Let us denote a configuration of a robotic system [58] by q. Kinematic models are specified in terms of the configuration variable q, while dynamical models are specified by the state $x = (q, \dot{q})$, where $\dot{q}$ denotes the corresponding velocities. In SLQG-FIRM, we sample the underlying PRM nodes (stabilizer parameters) from the configuration space. Thus, for dynamical systems, we impose the condition $\dot{q} = 0$ on the samples, i.e., we sample from the equilibrium space of the system, which is denoted by X in this dissertation.

Remark 1. *FIRM can be generalized to cases that do not need to sample in equilibrium space. For example, in systems such as fixed-wing aircraft, the system cannot reach the zero velocity $\dot{q} = 0$. In such cases, SLQG is not a suitable choice and one needs to design more appropriate controllers, such as periodic controllers as detailed in Chapter 7. In such a case, we sample periodic maneuvers as FIRM nodes. In other words, we go from periodic trajectory to periodic trajectory instead of going from point to point.*

5.2 Belief Stabilizers

In SLQG-FIRM nodes, we use Stationary LQG (SLQG) controllers as belief stabilizers, i.e., as a tool to reach (stabilize to) a predefined belief (FIRM node). To explain how SLQG works as a belief stabilizer, consider a fixed point $\mathbf{v} \in \mathsf{X}$ in the state space and consider the following linear (linearized) system about $\mathbf{v}$:

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k + \mathbf{G}w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{5.1a}$$

$$z_k = \mathbf{H}x_k + v_k, \qquad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \tag{5.1b}$$

The goal of the SLQG controller designed about $\mathbf{v}$ is to keep the state as close as possible to the desired point $\mathbf{v}$ and also keep the consumed energy at a reasonable level. More rigorously, SLQG minimizes the following quadratic cost:

$$J = \mathbb{E}\{\sum_{k\geq 0}(x_k - \mathbf{v})^T \mathbf{W}_x (x_k - \mathbf{v}) + u_k^T \mathbf{W}_u u_k\}, \tag{5.2}$$

where $\mathbf{W}_x$ and $\mathbf{W}_u$ are positive definite weight matrices that are defined by the user. As discussed in Chapter 2, under the SLQG controller minimizing the above cost,

the belief propagation and control generation is carried out as follows:

$$b_{k+1} \equiv \begin{bmatrix} \widehat{x}^+_{k+1} \\ P^+_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}\widehat{x}^+_k + \mathbf{B}u_k + \mathbf{K}_{k+1}(z_{k+1} - \mathbf{H}(\mathbf{A}\widehat{x}^+_k + \mathbf{B}u_k)) \\ (I - \mathbf{K}_{k+1}\mathbf{H})(\mathbf{A}P^+_k\mathbf{A}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T) \end{bmatrix} \equiv \tau(b_k, u_k, z_{k+1}), \tag{5.3}$$

where $\mathbf{K}_k$ is called the Kalman gain at the k-th time step and is computed as follows:

$$\mathbf{K}_{k+1} = (\mathbf{A}P^+_k\mathbf{A}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T)\mathbf{H}^T(\mathbf{H}(\mathbf{A}P^+_k\mathbf{A}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T)\mathbf{H}^T + \mathbf{M}\mathbf{R}\mathbf{M}^T)^{-1}. \tag{5.4}$$

The control signal is generated using a stationary feedback gain $\mathbf{L}_s$:

$$u_k = -\mathbf{L}_s(\widehat{x}^+_k - \mathbf{v}) =: \mu(b_k), \qquad \mathbf{L}_s = (\mathbf{B}_s^T S_s \mathbf{B}_s + \mathbf{W}_u)^{-1}\mathbf{B}_s^T S_s \mathbf{A}_s, \tag{5.5}$$

where, S_s is the solution of the following Discrete Algebraic Riccati Equation (DARE):

$$S_s = \mathbf{W}_x + \mathbf{A}_s^T S_s \mathbf{A}_s - \mathbf{A}_s^T S_s \mathbf{B}_s(\mathbf{B}_s^T S_s \mathbf{B}_s + \mathbf{W}_u)^{-1}\mathbf{B}_s^T S_s \mathbf{A}_s. \tag{5.6}$$

Consider an $n \times n$ matrix $\mathbf{A}$. A pair of matrices $(\mathbf{A}, \mathbf{B})$ is called a controllable pair if $\mathfrak{C} = [\mathbf{B}, \mathbf{AB}, \mathbf{A}^2\mathbf{B}, \cdots, \mathbf{A}^{n-1}\mathbf{B}]$ (referred to as controllability matrix) has rank n [15]. A pair of matrices $(\mathbf{A}, \mathbf{H})$ is called observable if the pair $(\mathbf{A}^T, \mathbf{H}^T)$ is controllable [15].

Let us also define the matrices $\check{\mathbf{Q}}$ and $\check{\mathbf{W}}_x$ such that $\mathbf{G}\mathbf{Q}\mathbf{G}^T = \check{\mathbf{Q}}\check{\mathbf{Q}}^T$, $\mathbf{W}_x = \check{\mathbf{W}}_x^T\check{\mathbf{W}}_x$. We next consider a class of linear systems and quadratic cost weights that satisfy the following property:

Property 1. *Pairs* $(\mathbf{A}, \mathbf{B})$ *and* $(\mathbf{A}, \check{\mathbf{Q}})$ *are controllable pairs, and pairs* $(\mathbf{A}, \mathbf{H})$ *and* $(\mathbf{A}, \check{\mathbf{W}})$ *are observable pairs.*

In the following, we present three lemmas, through which we can construct reachable SLQG-FIRM nodes for the systems that satisfy Property 1. However, approaches such as dynamic feedback linearization-based FIRM (see Chapter 6) or periodic LQG-based FIRM (see Chapter 7) extend this class of systems by excluding the controllability part in Property 1, and thus consider a broader class of systems.

Lemma 4. *Consider the SLQG controller designed to drive the state of the system in Eq.* (5.1) *to a point* $\mathbf{v} \in \mathsf{X}$*. Given that Property 1 is satisfied, in the absence of a stopping region, the belief* b_k *under an SLQG controller converges to a unique stationary belief* b_s*, in distribution (i.d.). In other words, the distribution over belief converges to a unique distribution. That is,*

$$b_k \xrightarrow{i.d.} b_s \sim \mathcal{N}(b_c, \mathtt{C}). \tag{5.7}$$

Note that b_k *is a random belief that converges to another random belief* b_s*. In the Gaussian setting, the distribution over the random belief* b_s *is* $\mathcal{N}(b_c, \mathtt{C})$*, where,* $b_c = \mathbb{E}[b_s] \equiv (\mathbf{v}, P_s)$*. The stationary estimation covariance matrix* P_s *is characterized in Lemma 5, and the covariance* $\mathtt{C}$ *is characterized in Section 2.2.2.*

Proof. See Section 2.2.2. □

Lemma 5. *Given Property 1, the following Algebraic Riccati equation (DARE) has a unique symmetric positive definite solution [15], denoted by* P_s^-*:*

$$P_s^- = \mathbf{G}\mathbf{Q}\mathbf{G}^T + \mathbf{A}(P_s^- - P_s^-\mathbf{H}^T(\mathbf{H}P_s^-\mathbf{H}^T + \mathbf{R})^{-1}\mathbf{H}P_s^-)\mathbf{A}^T. \tag{5.8}$$

Moreover, the stationary covariance matrix P_s *introduced in Lemma 4 is computed*

as:

$$P_s = P_s^- - P_s^- \mathbf{H}^T (\mathbf{H} P_s^- \mathbf{H}^T + \mathbf{R})^{-1} \mathbf{H} P_s^-. \tag{5.9}$$

Proof. See Section 2.2.2 or [15]. □

Now we state the main result, through which we can construct *reachable* FIRM nodes under SLQG-based belief stabilizers:

Lemma 6. *Consider the SLQG controller designed to drive the state of the system in Eq.* (5.1) *to a point* $\mathbf{v} \in \mathsf{X}$*. Suppose matrix* $\mathbf{H}$ *is full rank and Property 1 is satisfied. Then, any set* $B \subset \mathbb{B}$*, whose interior contains* $b_c \equiv (\mathbf{v}, P_s)$*, is reachable under the designed SLQG controller starting from any Gaussian distribution. Moreover, the estimation covariance* P_k *converges to the unique deterministic stationary covariance* P_s*.*

Proof. Let us consider the state space model of the linear system of interest as follows:

$$x_{k+1} = \mathbf{A} x_k + \mathbf{B} u_k + \mathbf{G} w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{5.10a}$$

$$z_k = \mathbf{H} x_k + v_k, \qquad\qquad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{5.10b}$$

Based on Lemma 4, if $(\mathbf{A}, \mathbf{B})$ and $(\mathbf{A}, \check{\mathbf{Q}})$ are controllable pairs, where $\mathbf{G}\mathbf{Q}\mathbf{G}^T = \check{\mathbf{Q}}\check{\mathbf{Q}}^T$, and if $(\mathbf{A}, \mathbf{H})$ and $(\mathbf{A}, \check{\mathbf{W}}_x)$ are observable pairs, where $\mathbf{W}_x = \check{\mathbf{W}}_x^T \check{\mathbf{W}}_x$, then the estimation covariance deterministically tends to a stationary covariance P_s. Therefore, for any $\epsilon > 0$, after a deterministic finite time, P_k enters the ϵ-neighborhood of the stationary covariance, denoted by P_s.

The estimation mean dynamics, however, is stochastic and is as follows for the

system in Eq. (5.10):

$$\begin{aligned}\widehat{x}_{k+1}^{+} &= \mathbf{v} + (\mathbf{A} - \mathbf{BL} - \mathbf{K}_{k+1}\mathbf{HA})(\widehat{x}_k^{+} - \mathbf{v}) \\ &\quad + \mathbf{K}_{k+1}\mathbf{HA}(x_k - \mathbf{v}) + \mathbf{K}_{k+1}\mathbf{HG}w_k + \mathbf{K}_{k+1}v_{k+1} \\ &= \mathbf{v} - (\mathbf{A} - \mathbf{BL})\mathbf{v} + (\mathbf{A} - \mathbf{BL} - \mathbf{K}_{k+1}\mathbf{HA})\widehat{x}_k^{+} \\ &\quad + \mathbf{K}_{k+1}\mathbf{HA}x_k + \mathbf{K}_{k+1}\mathbf{HG}w_k + \mathbf{K}_{k+1}v_{k+1} \end{aligned} \tag{5.11}$$

where the Kalman gain $\mathbf{K}_k$ is:

$$\mathbf{K}_k = P_k^{-}\mathbf{H}^T(\mathbf{H}P_k^{-}\mathbf{H}^T + \mathbf{R})^{-1} \tag{5.12}$$

Since $\mathbf{K}$ is full rank (due to the condition on the rank of $\mathbf{H}$), and since the v and w are Gaussian noises, the Eq. (5.11) induces an irreducible Markov process over the state space [61]. Thus, if we have a stopping region for the estimation mean with size $\epsilon > 0$, the estimation mean process will hit this stopping region in a finite time [61], with probability one.

Based on the estimation mean dynamics in Eq. (5.11) and the state dynamics in Section 2.2.2, in the absence of stopping region, if the estimation mean process and state process start from $\widehat{x}_0^{+}$ and x_0, respectively, such that $\mathbb{E}[\widehat{x}_0^{+}] = \mathbf{v}$ and $\mathbb{E}[x_0] = \mathbf{v}$ (which indeed is the case in FIRM due to the usage of edge-controllers), "the mean of estimation mean" remains on the $\mathbf{v}$, i.e., $\mathbb{E}[\widehat{x}_k^{+}] = \mathbf{v}$, for all k. As a result, if we center the stopping region for the estimation mean at $\mathbf{v}$, the probability of hitting the stopping region is maximized and the stopping time is minimized.

Combining the results for estimation covariance and estimation mean, if we define the region B as a set in the Gaussian belief space with a non-empty interior centered at $(\mathbf{v}, P_s)$, then the belief $b_k \equiv (\widehat{x}_k^{+}, P_k)$ enters region B in finite time with probability

one. Thus, the pair (B, μ) is a proper pair over whole $\mathbb{GB}$. $\square$

Therefore, based on Lemma 6, SLQG can accomplish the belief reachability for an appropriately chosen region B. In the next subsection we explicitly characterize the region B.

5.3 Designing SLQG-FIRM Nodes

As mentioned, to construct a FIRM we first construct an underlying PRM [48]. In the SLQG-FIRM, nodes of the underlying PRM, denoted by $\{\mathbf{v}^j\}_{j=1}^{N_v}$, are sampled from the obstacle-free space. Considering linear systems or nonlinear systems that are locally well approximated by linearization, we linearize the system about every PRM node. Let us denote the linear (linearized) system about $\mathbf{v}^j$ as follows:

$$x_{k+1} = \mathbf{A}^j x_k + \mathbf{B}^j u_k + \mathbf{G}^j w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^j) \tag{5.13a}$$

$$z_k = \mathbf{H}^j x_k + v_k, \qquad\qquad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^j). \tag{5.13b}$$

where w_k and v_k are motion and measurement noise, respectively, drawn from zero-mean Gaussian distributions with covariances $\mathbf{Q}^j$ and $\mathbf{R}^j$.

To design the j-th FIRM node B^j, we first design the SLQG controller μ_s^j (see Eq. (5.5)) corresponding to the system in Eq. (5.13). The controller μ_s^j is called the j-th node controller or the j-th belief stabilizer. Given Property 1, based on Lemma 4, the limiting random belief $b_s^j \equiv (\widehat{x}_s^{+^j}, P_s^j)$ exists. $\widehat{x}_s^{+^j}$ and P_s^j are the stationary estimation mean and covariance, respectively. Note that under SLQG, $\widehat{x}_s^{+^j}$ is a random variable and P_s^j is a deterministic matrix. Moreover, in Lemma 4, it is shown that $b_c^j = \mathbb{E}[b_s^j] \equiv (\mathbf{v}^j, P_s^j)$, where P_s^j is shown to be unique and computed in

Lemma 5. Thus, we can characterize the j-th node center:

$$b_c^j \equiv (\mathbf{v}^j, P_s^j). \tag{5.14}$$

As a result, considering B^j as a ball with an arbitrary radius $\epsilon > 0$ centered at b_c^j, the pair (B^j, μ_s^j) is a *proper pair*, based on Lemma 6; i.e., B^j is reachable under μ_s^j. Thus, one can define the j-th FIRM node as $B^j = \{b \ : \ \|b - b_c^j\|_b < \delta\}$, where $\|\cdot\|_b$ denotes a suitable norm in belief space and δ defines the FIRM node size. A typical example of such a FIRM node in Gaussian belief space can be defined by considering mean and covariance separately:

$$B^j = \{b \equiv (x, P) \ : \ \|x - \mathbf{v}^j\| < \delta_1, \|P - P_s^j\|_m < \delta_2\} \tag{5.15}$$

where δ_1 and δ_2 are suitably small thresholds that determine the size of FIRM node B^j. $\|\cdot\|$ is a suitable vector norm and $\|\cdot\|_m$ is a suitable matrix norm. We denote the set of all SLQG-FIRM nodes as $\mathbb{V} = \{B^i\}$.

5.4 Designing SLQG-FIRM Edges

A FIRM edge is actually a local planner (local feedback controller). In SLQG-based FIRM, the local controller representing the (i, j)-th edge is denoted by μ^{ij}. The role of μ^{ij} is to drive the belief from the node B^i to the node B^j. Based on Lemma 6, for a linear system, if we choose $\mu^{ij} = \mu_s^j$, as has been done in [2], the node B^j is reachable under μ^{ij}. However, to better cope with nonlinearities, we construct the local controller μ^{ij} by preceding the node-controller with a time-varying LQG controller $\overline{\mu}_k^{ij}$, which is called an *edge-controller* here. Time-varying LQG controllers have been described in detail in Section 2.2.1.

To design edge-controllers, first the underlying PRM edges, denoted by $\mathcal{E} =$

$\{\mathbf{e}^{ij}\}$, have to be constructed. For kinematics-based models there are many different methods in the PRM literature to construct such edges. For dynamical models, there are fewer choices. A few examples are [92] or [5].

An edge-controller $\overline{\mu}_k^{ij}$ in SLQG-FIRM is built by linearizing the system along the (i,j)-th PRM edge $\mathbf{e}^{ij}$ and designing a time-varying LQG controller to track it (see Section 2.2.1). The edge-controller has two major roles. First it tries to track the PRM edge and thus exploits the available information on the PRM edges, such as some clearance from the obstacles. Second, in the case that the neighboring PRM nodes are not close to each other, it takes the belief into the valid linearization region of the j-th belief stabilizer, where it hands over the system to the belief stabilizer, and the belief stabilizer in turn takes the system to the j-th FIRM node.

Thus, overall, the (i,j)-th local controller μ^{ij} is the concatenation of the (i,j)-th edge controller $\overline{\mu}_k^{ij}$ and j-th node-controller (belief stabilizer) μ_s^j. We denote the set of all SLQG-FIRM edges by $\mathbb{M} = \{\mu^{ij}\}$ and the set of all SLQG-FIRM edges originating from B^i by $\mathbb{M}(i)$.

Formally, we define SLQG-FIRM as a graph with the set of nodes $\mathbb{V} = \{B^i\}$ and the set of edges (or local controllers) $\mathbb{M} = \{\mu^{ij}\}$. The set of controllers originating from B^i is denoted by $\mathbb{M}(i) \subset \mathbb{M}$.

5.5 Transition Probabilities and Edge Costs

To find a feedback on a FIRM graph, we need to compute the cost associated with the graph edges. Moreover, we include the constraint set F in the planning with FIRM by computing the probability of violating the constraint $(x,u) \notin F$ along the graph edges. Let us denote the cost of taking controller μ^{ij} at node B^i by $C^g(B^i, \mu^{ij})$. Superscript g refers to the "global" (or "graph-level") quantities, as these quantities are used to find the global policy (or policy on the graph). Similarly,

let $\mathbb{P}^g(B^j|B^i,\mu^{ij})$ and $\mathbb{P}^g(F|B^i,\mu^{ij})$ denote the probability of the transition to B^j and F under μ^{ij}, respectively. These quantities are rigorously defined in Chapter 4 and their connection with the original POMDP is established. In this subsection, we give examples of how such costs and transition probabilities can be computed.

Computing transition probabilities $\mathbb{P}^g(\cdot|B^i,\mu^{ij})$ in general can be computationally expensive. Here, we utilize particle-based methods to approximate the distributions and thus compute the collision probabilities. Basically, we can approximate the failure and reachability probabilities based on the number of particles that violate the constraints (hit the set F) and based on the number of particles that can reach the target node (hit the set B^j). The method is described in more detail with the experiments in Section 5.8 of this chapter. The dependency of collision events in different time steps, which is ignored in most collision probability computation methods in the POMDP literature, can be taken into account rigorously in particle-based methods. Owing to the offline construction of FIRM, the high computational burden of particle-based approaches can be tolerated. However, any other method for computing transition probabilities can also be adopted, such as [75].

The FIRM edge costs in general and their derivation based on the one-step costs of the original POMDP problem are defined in Chapter 4. However, roughly speaking, we can define the cost $C^g(B^i,\mu^{ij})$ as the sum of all one-step costs along the edge until the system reaches the target node B^j or hits the failure set F. Depending on the application, one can define a variety of cost functions. Here, we form a cost function based on a linear combination of the estimation accuracy and edge traversal time. This cost function aims to find paths for which the estimator (and hence the controller) can perform well and also to find faster paths. An indicator of estimation error is the trace of estimation covariance. Thus, we define $\Phi^{ij} = \mathbb{E}[\sum_{k=1}^{\mathcal{T}} \mathrm{tr}(P_k^{ij})]$ along the edge. In stationary LQG, the covariance matrix evolves deterministically

and thus the expectation operator can be omitted. However, if the filter of choice in the edge-controller is the Extended Kalman Filter (EKF), the covariance matrix evolution is also stochastic, and this measure can take into account its stochasticity. Let us denote the mean stopping time under controller μ^{ij} as $\widehat{\mathcal{T}}^{ij}$. Then, the total edge cost is considered as a linear combination of estimation accuracy and expected stopping time, with suitable coefficients α_1 and α_2.

$$C^g(B^i, \mu^{ij}) = \alpha_1 \Phi^{ij} + \alpha_2 \widehat{\mathcal{T}}^{ij}. \tag{5.16}$$

5.6 Graph Feedback on SLQG-FIRM

Graph policy $\pi^g : \mathbb{V} \to \mathbb{M}$ is a function that returns an edge (local controller) for any given node of the graph. We denote the space of all graph policies by Π^g. To choose the best graph policy in Π^g we define the optimal graph cost-to-go J^g from every graph node.

The cost-to-go from a given node B^i is equal to the cost of the next taken controller, i.e., $C^g(B^i, \pi^g(B^i))$, plus the expected cost-to-go from the next node or from the failure set. In other words, the dynamic programming equations for this graph are:

$$\begin{aligned} J^g(B^i) = \min_{\mathbb{M}(i)} & C^g(B^i, \mu^{ij}) + J^g(F)\mathbb{P}^g(F|B^i, \mu^{ij}) \\ & + J^g(B^j)\mathbb{P}^g(B^j|B^i, \mu^{ij}), \end{aligned} \tag{5.17a}$$

$$\begin{aligned} \pi^g(B^i) = \arg\min_{\mathbb{M}(i)} & C^g(B^i, \mu^{ij}) + J^g(F)\mathbb{P}^g(F|B^i, \mu^{ij}) \\ & + J^g(B^j)\mathbb{P}^g(B^j|B^i, \mu^{ij}). \end{aligned} \tag{5.17b}$$

in which, $J(F)$ is a suitably high user-defined cost-to-go for hitting the obstacles.

The cost-to-go from goal node B^{goal} is defined to be zero, i.e., $J^g(B^{goal}) = 0$.

The DP in Eq. (5.17) is a tractable DP as it is defined on a finite number of graph nodes. Computing the transition costs and probabilities offline, this DP can be solved online using standard techniques, such as value/policy iteration methods, for any submitted query. As a result, FIRM is indeed a multi-query roadmap in belief space. Moreover, if the goal node is fixed and only the starting point of the query changes, then this DP can be solved offline and π^g can be stored as a look-up table.

Algorithm 7 details the construction of SLQG-FIRM with a given goal node.

Algorithm 7: Offline Construction of SLQG-FIRM

input : Free space map, X_{free}
output : FIRM graph $\mathcal{G}$
Sample PRM nodes $\mathcal{V} = \{\mathbf{v}^j\}_{j=1}^{N_v}$ and construct its edges $\mathcal{E} = \{\mathbf{e}^{ij}\}$;
forall the *PRM nodes* $\mathbf{v}^j \in \mathcal{V}$ **do**
    Design the node controller (stationary LQG) μ_s^j about the node $\mathbf{v}^i$ using Eq. (5.5);
    Compute associated b_c^j using Eq. (5.14);
    Construct FIRM node B^j using Eq. (5.15);
Construct $\mathbb{V} = \{B^i\}$;
forall the *PRM edges* $\mathbf{e}^{ij} \in \mathcal{E}$ **do**
    Design the edge controller (time-varying LQG) $\overline{\mu}_k^{ij}$ along the edge $\mathbf{e}^{ij}$ (detailed in Section 2.2.1);
    Construct the local controller μ^{ij} by concatenating edge controller $\overline{\mu}_k^{ij}$ and node controller μ_s^j;
    Set $b_0 = b_c^i$;
    Generate sample belief paths $b_{0:\mathcal{T}}$ and ground truth paths $x_{0:\mathcal{T}}$ induced by controller μ^{ij} invoked at B^i;
    Compute the transition probabilities $\mathbb{P}^g(F|B^i, \mu^{ij})$ and $\mathbb{P}^g(B^j|B^i, \mu^{ij})$ and transition cost $C^g(B^i, \mu^{ij})$;
Construct $\mathbb{M} = \{\mu^{ij}\}$;
Compute the cost-to-go J^g and feedback π^g over the FIRM nodes by solving the DP in Eq. (5.17);
$\mathcal{G} = (\mathbb{V}, \mathbb{M}, J^g, \pi^g)$;
return $\mathcal{G}$;

5.7 Planning with SLQG-FIRM (Query-phase)

Given that the FIRM graph is computed offline, the online phase of planning (and replanning) on the roadmap becomes very efficient, and thus feasible in real time. In this section, we assume that the goal node is fixed and we just input the start point as the query. However, as discussed in the previous subsection, one can easily submit queries with different goal locations by solving DP online. If the initial belief b_0 of the submitted query does not belong to any B^i, we create a singleton

set $B_0 = \{b_0\}$ as the initial FIRM node. To connect B_0 to the FIRM graph, we go back into the state space, where the underlying PRM is constructed. There, we add a new PRM node to the graph $\mathbf{v}_0$, which is the expected value of the robot state, i.e., $\mathbf{v}_0 = \mathbb{E}[x_0]$. Then, we connect $\mathbf{v}_0$ to the underlying PRM graph based on the connecting function of the adopted PRM. We denote the set of newly added edges originating from $\mathbf{v}_0$ by $\mathcal{E}(0)$. Then, corresponding to each edge in $\mathcal{E}(0)$, we design a local controller and call the set of them $\mathbb{M}(0)$. Finally, we choose the best initial controller among the local controllers in $\mathbb{M}(0)$ using:

$$\mu_0^*(\cdot) = \arg\min_{\mu \in \mathbb{M}(0)} \{C^g(B_0, \mu) + \mathbb{P}^g(B(\mu)|B_0, \mu) J^g(B(\mu)) + \mathbb{P}^g(F|B_0, \mu) J^g(F)\}, \quad (5.18)$$

where $B(\mu)$ is the target node of the controller μ. Under the controller μ_0^*, belief evolves and enters one of FIRM nodes, if no collision occurs. From this FIRM node, a combination of the global graph policy π^g and the local edge policies $\{\mu^{ij}\}$ can take the belief to the goal node, as explained below.

After computing a global graph feedback π^g and local edge feedbacks $\{\mu^{ij}\}$, we can construct a full feedback π. Actually, at every time instance, π is equal to one of the local feedbacks, which is chosen by the global feedback in the last visited node. In other words, given the current FIRM node, we use policy π^g defined on FIRM nodes to find μ^* and pick μ^* to move the robot into $B(\mu^*)$. This process is continued until the system reaches the goal region or hits the failure set. Algorithm 8 illustrates this procedure.

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 [29].

Consider a kidnapped robot problem in a known environment. Just after the

Algorithm 8: Online Phase Algorithm (Planning or Replanning with SLQG-FIRM)

input : Initial belief b_0, FIRM graph $\mathcal{G}$
if $\exists B^i \in \mathbb{V}$ *such that* $b_0 \in B^i$ **then**
    Compute $\mu^{ij} = \pi^g(B^i)$;
else
    Compute $\mathbf{v}_0 = \mathbb{E}[x_0]$ based on b_0, and connect $\mathbf{v}_0$ to the PRM. Let $\mathcal{E}(0)$ denote the set of outgoing edges from $\mathbf{v}_0$;
    Set $B_0 = \{b_0\}$; Design local controllers associated with edges in $\mathcal{E}(0)$. Call the set of these local controllers $\mathbb{M}(0)$;
    forall the $\mu \in \mathbb{M}(0)$ **do**
        Generate sample belief paths $b_{0:\mathcal{T}}$ and ground truth paths $x_{0:\mathcal{T}}$ induced by controller μ invoked at b_0;
        Compute the transition probabilities $\mathbb{P}^g(F|B_0, \mu)$ and $\mathbb{P}^g(B(\mu)|B_0, \mu)$ and transition costs $C^g(B_0, \mu)$;
    Set $i = 0$ and choose the best initial local controller μ^{ij} within the set $\mathbb{M}(0)$ using Eq. (5.18);
while $B^i \neq B^{goal}$ **do**
    Denote the target node of μ^{ij} by B^j;
    while $b_k \notin B^j$ *and "no collision"* **do**
        Apply the control $u_k = \mu^{ij}(b_k)$ to the system;
        Get the measurement z_{k+1} from sensors;
        if *Collision happens* **then return** Collision;
        Update belief as $b_{k+1} = \tau(b_k, \mu^{ij}(b_k), z_{k+1})$;
    Set $B^i = B^j$, then compute $\mu^{ij} = \pi^g(B^i)$;

robot is kidnapped, it would be risky to apply any control, because the robot may be close to an obstacle. Thus, in such a scenario, we first initialize the system belief with a Gaussian with large covariance and go into an "information gathering" mode, where we do not apply any control signal and only gather measurements, until the covariance shrinks to a reasonable covariance or it remains unchanged for a significant amount of time (i.e., when there is no additional information to reduce the uncertainty). Afterwards, we connect the resulting belief to the FIRM nodes and continue applying the FIRM policy to move the robot toward the goal region.

A more efficient approach of handling this problem is detailed in Chapter 8 using innovation signals.

5.8 Experimental Results

In this section, we first illustrate theoretical results from the previous sections on a planar robot in a small three-dimensional planning domain. Then, we present planning results for a larger three-dimensional state space. Finally, we report the results of the method on a dynamical model of an eight-arm manipulator (sixteen-DOF state space). This section is followed by a brief comparison with other state-of-the-art methods in this domain.

5.8.1 Planar 3D Omni-directional Robot

The main goal of this subsection is to illustrating steps in construction and planning with SLQG-FIRM. In this subsection, we focus on an omni-directional robot. Its state is composed of its 2D position in the plane and its heading angle. The goal in this section is to illustrate the steps of constructing SLQG-FIRM and planning with it.

A 3-wheel omni-directional mobile robot is used in experiments with the nonlinear kinematic model given in [45]. The state vector is composed of a 2D location and heading angle $x = [{}^1x, {}^2x, \theta]^T$ in a global world frame. $u = [{}^1u, {}^2u, {}^3u]^T$ is the vector of controls, where ${}^iu$ is the linear velocity of the i-th wheel. w is the motion noise, which is drawn from a zero-mean Gaussian distribution. The motion dynamics for this robot, in its original continuous form is [45]:

$$\dot{x} = \mathbf{f}_c(x, u, w) = T(x)u + w, \tag{5.19}$$

where

$$T(x)=\begin{pmatrix} -\frac{2}{3}\sin(\theta) & -\frac{2}{3}\sin(\frac{\pi}{3}-\theta) & \frac{2}{3}\sin(\frac{\pi}{3}+\theta) \\ \frac{2}{3}\cos(\theta) & -\frac{2}{3}\cos(\frac{\pi}{3}-\theta) & -\frac{2}{3}\cos(\frac{\pi}{3}+\theta) \\ \frac{1}{3r} & \frac{1}{3r} & \frac{1}{3r} \end{pmatrix}, \tag{5.20}$$

where r is the distance of the wheels from the robot's center of mass. The discrete motion dynamics is shown by:

$$x_k = \mathbf{f}(x_{k-1}, u_{k-1}, w_{k-1}). \tag{5.21}$$

$w_k \sim \mathcal{N}(0, \mathbf{Q})$ is the motion noise at the k-th time step, which is drawn from a zero-mean Gaussian distribution with covariance matrix $\mathbf{Q}$. It can be shown that if we linearize this system, the linearized motion model satisfies the controllability condition in Property 1.

In experiments, the robot is equipped with exteroceptive sensors that provide range and bearing measurements from existing landmarks (radio beacons) in the environment. The 2D location of the j-th landmark is denoted by L_j. Measuring L_j can be modeled as follows:

$${}^j z = {}^j h(x, {}^j v) = [\|{}^j\mathbf{d}\|, \text{atan2}({}^j d_2, {}^j d_1) - \theta]^T + {}^j v, \quad {}^j v \sim \mathcal{N}(\mathbf{0}, {}^j\mathbf{R}),$$

where ${}^j\mathbf{d} = [{}^j d_1, {}^j d_2]^T := [{}^1 x, {}^2 x]^T - L_j$. The vector ${}^j v$ is a state-dependent observation noise, with covariance

$${}^j\mathbf{R} = \text{diag}((\eta_r \|{}^j\mathbf{d}\| + \sigma_b^r)^2, (\eta_\theta \|{}^j\mathbf{d}\| + \sigma_b^\theta)^2). \tag{5.22}$$

In other words, the uncertainty (standard deviation) of the sensor reading increases as

the robot gets farther from the landmarks. $\eta_r = \eta_\theta = 0.3$ determines this dependence, and $\sigma_b^r = 0.01$ meter and $\sigma_b^\theta = 0.5$ degrees are the bias standard deviations. A similar model for range sensing is used in [81]. We assume the robot observes all N_L landmarks at all times and their observation noise is independent. Thus, the total measurement vector is denoted by $z = [{}^1z^T, {}^2z^T, \cdots, {}^{N_L}z^T]^T$, and, due to the independence of measurements of different landmarks, the observation model for all landmarks can be written as:

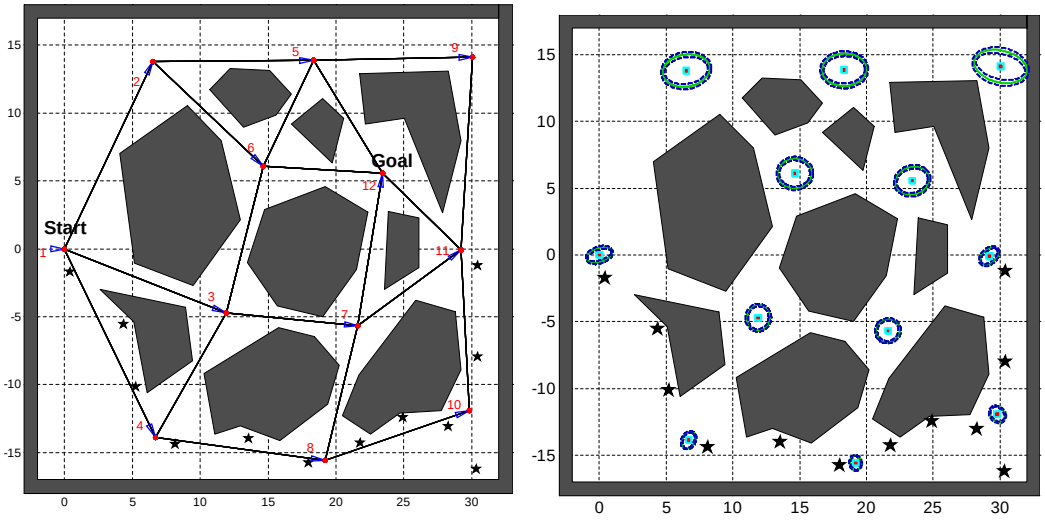
$$z = h(x) + v, \quad v \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad \mathbf{R} = \text{diag}({}^1\mathbf{R}, \cdots, {}^{N_L}\mathbf{R}). \tag{5.23}$$

It is straightforward to show that the linearized version of this observation model satisfies the observability condition in Property 1. Therefore, this entire system model (motion and sensing models) satisfies Property 1 and thus the SLQG-FIRM can be used for planning.

Figure 5.1(a) shows a sample environment, including obstacles, landmarks, and enumerated nodes in $({}^1x, {}^2x, \theta)$ space. Nodes are shown by blue triangles, which encode the position $({}^1x, {}^2x)$ and heading angle θ of the robot. Landmarks are shown by black stars. The corresponding FIRM nodes are computed and shown in Fig. 5.1(b). All elements in Fig. 5.1(b) are defined in $({}^1x, {}^2x, \theta)$ space but only the $({}^1x, {}^2x)$ portion of them is shown. Each $b_c^j \equiv (\mathbf{v}^j, P_s^j)$ is illustrated by a red dot representing $\mathbf{v}^j$ and a green ellipse, representing 3σ ellipse of covariance P_s^j. Each FIRM node B^j is a neighborhood around b_c^j. In the experiments, we define the node region using the component-wise version of Eq. (5.15), to handle the error scale difference in position and orientation variables:

$$B^j = \{b \equiv (x, P) | \ \ |x - \mathbf{v}^j| \dot{<} \epsilon, |P - P_s^j| \dot{<} \Delta\}, \tag{5.24}$$

where $|\cdot|$ and $\dot{<}$ stand for the absolute value and component-wise comparison operators, respectively. We also set $\epsilon = [0.07(meter), 0.07(meter), 1(degree)]^T$ and $\Delta = \epsilon\epsilon^T$ to quantify B^j's. The projection of B^j onto the space of estimation mean, i.e., $B^j_x = \{\widehat{x}^+ \ : \ |\widehat{x}^+ - \mathbf{v}^j| \dot{<} \epsilon\}$ is a neighborhood around $\mathbf{v}^j$, which is shown by a cyan rectangle centered at $\mathbf{v}^j$. The projection of B^j onto the space of estimation covariances, i.e., $B^j_P = \{P \ : \ |P - P^j_s| \dot{<} \Delta\}$ is a neighborhood around P^j_s. However, in a 2D plot B^j_P cannot be shown due to its high dimension. Thus, we partially illustrate it only by two dashed green ellipses that represent 3σ covariances of $P^j_s - \Delta_d$ and $P^j_s + \Delta_d$, where Δ_d is the matrix Δ, whose off-diagonal elements are set to zero. For illustration purposes, both of these neighborhoods, i.e., B^j_x and B^j_P, are five times magnified in Fig. 5.1(b).



(a) (b)

Figure 5.1: (a) Figure depicts the underlying PRM graph. Gray polygons are the obstacles and black stars represent the landmarks' locations. (b) FIRM nodes corresponding to PRM nodes.

After designing FIRM nodes and local controllers, the transition costs and probabilities have to be computed. Based on the given task and needed accuracy, different approaches can be taken. Here, we use a particle-based approximation of the distribution to compute these quantities, and we use $M = 100$ particles. In other words, for every (B, μ) pair, we perform 100 runs. At every run, a sample path of state x, a sample path of estimation mean $\widehat{x}^+$, and a sample path of estimation covariance P is generated. If the filter of choice in the edge-controller is the Linearized Kalman Filter (LKF) [32], [85], the covariance evolution is deterministic and there is no need to generate 100 different sample covariance paths. However, if the filter of choice in the edge-controller is the Extended Kalman Filter (EKF) [32], [85], then we have to generate the sample covariance paths too, to take into account the stochasticity of the covariance matrix. Figure 5.2(a) depicts sample paths of the true state x and estimation mean $\widehat{x}^+$ in green and dark red, respectively, for $M = 100$ particles. Note that when a true state path (green path) collides with an obstacle, the process stops and failure happens. However, in this figure, for illustration purposes, we continue the process and ignore the obstacles to better show the uncertainty tube and information availability at different parts of the space. As seen in Fig. 5.2(a), the behavior of the true state on the edges which have access to more accurate observations is remarkably close to the planned behavior. In contrast, on the edges that receive less informative observations, the controller cannot effectively compensate for deviations of the ground truth from the nominal path, which can lead to collision with obstacles.

To simplify the figure, Fig. 5.2(b) depicts sample estimation covariance evolution only for a single particle. In this figure, to keep the centers of ellipses (i.e., the estimation mean) on the planned points, we let the process and observation noise be zero. However, note that, in general, the estimation mean is affected by the noise (as is seen in Fig. 5.2(a)). Indeed, Fig. 5.2(b) can be seen as the maximum-likelihood

estimation uncertainty tube over the roadmap.

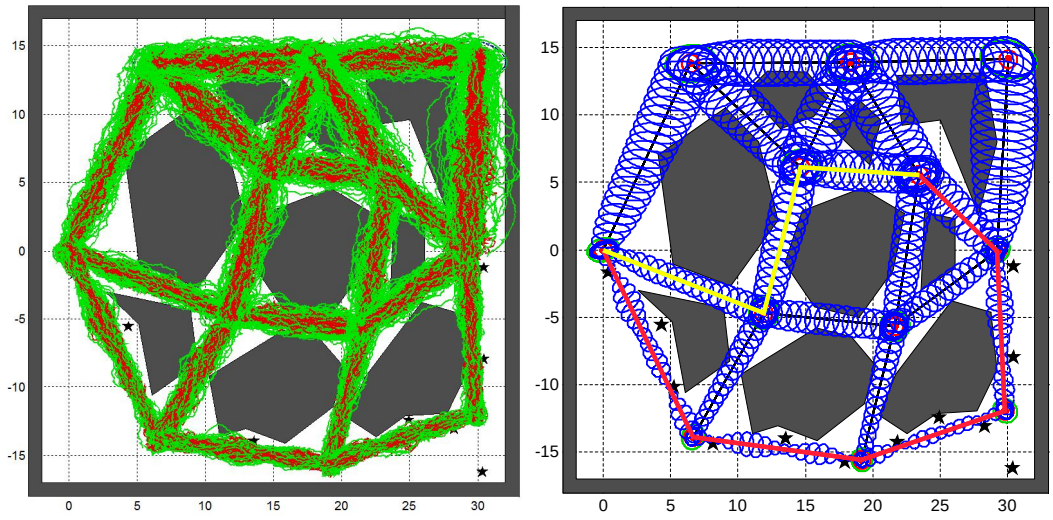


Figure 5.2: Sample paths induced by controllers invoked at different nodes. (a) For $M = 100$ particles, sample ground truth paths and sample estimation mean paths are shown in green and dark red, respectively. (b) The most likely path under the optimal policy and shortest path are shown in red and yellow respectively. The 3σ ML estimation uncertainty tube is drawn in blue.

Let us denote the q-th sample path for the true state by $x^{(q)}_{0:\mathcal{T}^q}$, for the estimation mean by $\widehat{x}^{+(q)}_{0:\mathcal{T}^q}$, and for the estimation covariance by $P^{(q)}_{0:\mathcal{T}^q}$, where $\mathcal{T}^q$ is the stopping time of the q-th particle in executing μ at B. Moreover, one can assign a weight to each particle q based on its probability of occurrence. There are different ways proposed to compute these weights in the Sequential Monte Carlo literature [34]. However, the main condition is that they have to sum to one, i.e., $\sum_{q=1}^{M} w^{(q)} = 1$. Here we simply consider $w^{(q)} = M^{-1}$. Note that if we run μ^{ij} at B^i, all these quantities also have to have a ij superscript. Having these sample paths, we can compute the transition costs and probabilities associated with invoking the μ^{ij} at

B^i. For the collision probability, we have:

$$\mathbb{P}^g(F|B^i,\mu^{ij}) = \mathbb{E}[\mathbb{I}_F|B^i,\mu^{ij}] \approx \sum_{q=1}^{M} w^{(q)}\mathbb{I}_F(x^{(q)}_{0:\mathcal{T}^q}) \tag{5.25}$$

$$\mathbb{P}^g(B^j|B^i,\mu^{ij}) = 1 - \mathbb{P}^g(F|B^i,\mu^{ij}) \tag{5.26}$$

where $\mathbb{I}_F$ is the failure indicator. $\mathbb{I}_F(x^{(q)}_{0:\mathcal{T}^q})$ is one if $x_k \in F$ for some $k \leq \mathcal{T}^{(q)}$. Otherwise it is zero. $\mathcal{T}^q$, or more rigorously $\mathcal{T}^{ij^{(q)}}$, is the stopping time of the q-th particle in executing μ^{ij} at B^i. To compute $\mathcal{T}^{ij^{(q)}}$, we only need to check the condition $b \in B^j$ at every time step and find the first time step that belief b enters the stopping region B^j. Thus, we can compute the mean stopping time as

$$\widehat{\mathcal{T}}^{ij} = \mathbb{E}[\mathcal{T}^{ij}] \approx \sum_{q=1}^{M} w^{(q)}\mathcal{T}^{ij^{(q)}}. \tag{5.27}$$

To compute the filtering cost defined in Section 5.5, again we use the particle-based representation of belief:

$$\Phi^{ij} = \mathbb{E}[\sum_{k=1}^{\mathcal{T}^{ij}} \mathrm{tr}(P_k)|B^i,\mu^{ij}] \approx \sum_{q=1}^{M}\sum_{k=1}^{\mathcal{T}^q} w^{(q)}\mathrm{tr}(P_k^{(q)}), \tag{5.28}$$

where $P_k^{(q)}$ is the estimation covariance at the k-th time step of the q-th particle. Finally, the cost of taking μ^{ij} at B^j is as follows:

$$C^g(B^i,\mu^{ij}) = \alpha_1\Phi^{ij} + \alpha_2\widehat{\mathcal{T}}^{ij}$$

where we used the coefficients $\alpha_1 = 0.95$ and $\alpha_2 = 0.05$. Table 5.1 shows these quantities for several (B^i,μ^{ij}) pairs in corresponding to Fig. 5.2.

Plugging the computed transition costs and probabilities into Eq. (4.17), we can

Table 5.1: Computed costs for several Node-controller pairs in FIRM using 100 particles

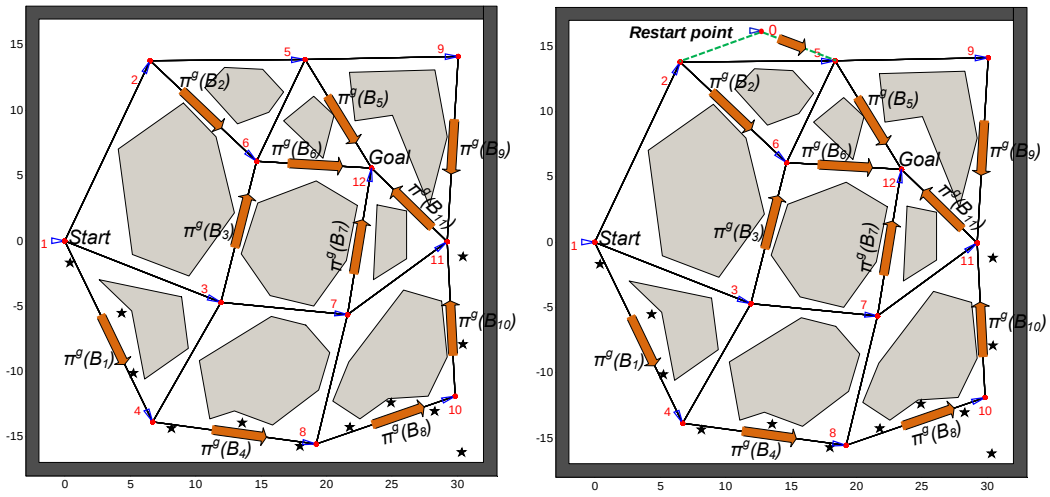
(B^i,μ^{ij}) pair	$B^1,\mu^{1,4}$	$B^4,\mu^{4,8}$	$B^8,\mu^{8,10}$	$B^{10},\mu^{10,11}$	$B^{11},\mu^{11,12}$	$B^1,\mu^{1,3}$	$B^3,\mu^{3,6}$	$B^6,\mu^{6,12}$
$\mathbb{P}^g(B^j\|B^i,\mu^{ij})$	%97	%95	%99	%77	%79	%87	%55	%79
Φ^{ij}	18.5967	11.2393	6.8229	15.1148	26.2942	23.6183	48.8189	43.6207
$\mathbb{E}[\mathcal{T}^{ij}]$	238.2	193.0	150.0	209.6	170.8	200.3	242.4	219.2
$\sigma[\mathcal{T}^{ij}]$	21.8	28.7	15.1	24.5	22.6	22.7	30.1	26.7

solve the DP and compute the graph policy π^g. This process is performed once offline if the goal location is fixed. Fig. 5.3(a) shows the policy π^g on the constructed FIRM in this example. Indeed, at every FIRM node B^i, the policy π^g decides which local controller has to be taken, which in turn aims to take the robot to the next FIRM node. Thus, the online part of the planning is quite efficient and reduces to executing the controller and generating the control signal.

An important consequence of this framework is that replanning can be performed using FIRM efficiently. Suppose due to some unmodeled large disturbance, the robot's belief deviates significantly from the planned path, i.e., for some appropriate norm $\|\cdot\|$ on belief space we have $\|b_k - \mathbb{E}[b_k^p]\| > \varrho$, where b_k^p is the planned belief at k-th time step, and ϱ is the threshold for deciding if replanning is needed or not. In such cases, replanning occurs and based on Algorithm 8. In Fig. 5.3(b), we illustrate a simple replanning process. In this figure, it is assumed that an unmodeled large disturbance affects the system, such that the estimation mean significantly deviates from the planned path. The deviated mean is shown on the figure as the "restart point". Thus, based on Algorithm 8, we connect this point to the PRM. In Fig. 5.3(b) the newly added PRM edges, i.e., $\mathcal{E}(0)$, are shown by dashed green lines. Then, for every edge in $\mathcal{E}(0)$, we design a local controller. Call the set of newly constructed local controllers $\mathbb{M}(0)$. For every $\mu \in \mathbb{M}(0)$ compute corresponding transition costs

and probabilities. Finally, according to Bellman's principle of optimality, we use the precomputed cost-to-go's $J^g(\cdot)$ to decide which controller will be taken at the "restart point" using Eq. (4.21). Taking this controller, the belief state returns to the FIRM nodes, and from there again we can use the precomputed π^g to control the robot toward the goal region.

We show the most likely path under π^g in red in Fig. 5.2(b). The shortest path is also illustrated in Fig. 5.2(b) in yellow. It can be seen that the "most likely path under the best policy" detours from the shortest path to a path along which the filtering uncertainty is smaller and it is easier for the controller to avoid collisions.

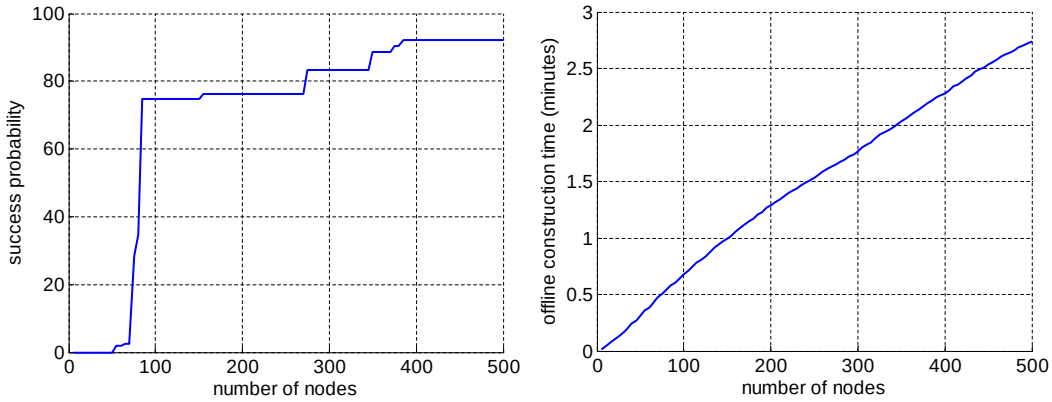


(a) (b)

Figure 5.3: Planning and replanning on FIRM. (a) Policy π^g resulted from solving DP in Eq. (4.17) is shown by red arrows. Indeed for every FIRM node, the policy π^g tells that which controller has to be taken. (b) In this figure it is assumed that an unmodeled large disturbance affects the system, such that the estimation mean significantly deviates from the planned path. The deviated mean is denoted by "restart point" on the figure.

5.8.2 *Larger Environment*

In this section, we consider the same omni-directional robot with the same observation model, and we perform planning in a larger environment (shown in Fig. 5.5), whose size is almost 10,000 square meters. Every grid square is a 10 by 10 area. The standard deviation of the process noise is assumed to be 1 meter for the positional degrees of freedom and 7 degrees for the angular degree of freedom. We start with a 5-node FIRM and at every step we randomly sample five more nodes until we reach 500 nodes. Thus, overall, we construct 100 FIRM graphs in this environment, for each of which we measure the construction time (cumulative) and compute the success probability. Plots in Fig. 5.5 show these quantities as a function of the number of nodes for a sample run on an Intel i5 dual-core 1.7 GHz machine with 4GB memory. 50 particles are used for collision checking, and every node in the underlying PRM is connected to its 3 nearest neighbors.



(a) (b)

Figure 5.4: This figure shows (for a sample run) the success probability of the generated plan versus the number of nodes, as well as the construction time (offline) for the plan.

Basically, FIRM construction is an anytime algorithm in the sense that one can increase the number of nodes and stop enlarging the graph when a termination condition is satisfied such as: *(i)* achieving a desirable success probability or a desirable cost-to-go, *(ii)* no change is observed in the success probability or in the cost-to-go for a significant time, or *(iii)* exceeding the maximum allowed time for offline computation.

Again, as is seen from Fig. 5.5, the highest likelihood path under the optimal policy detours from the shortest path towards the more informative regions in the environment. As a result, it reduces the collision probability and at the same time increases the estimation accuracy and controller efficiency. However, it is important to note that the returned solution is not a single path, but it is a feedback law over the entire space. For the video of executing this plan (with less number of nodes to unclutter the video) see `https://parasol.tamu.edu/groups/amatogroup/movies/ExtensionOne\_v5.mp4` and `https://parasol.tamu.edu/groups/amatogroup/movies/LargeEnvironmentDoubleSpeed.mp4` ([70] and [71]).

We also conducted a simulation to illustrate the robustness of the method to large deviations. In this simulation, the robot is pushed away from the roadmap several times by some large disturbances, and replanning is performed online based on Algorithm 8. The video of this simulation is available at `https://parasol.tamu.edu/groups/amatogroup/movies/ExtensionTwo\_v6.mp4` and `https://parasol.tamu.edu/groups/amatogroup/movies/Replanning4TimesFaster.mp4` ([68] and [69]).

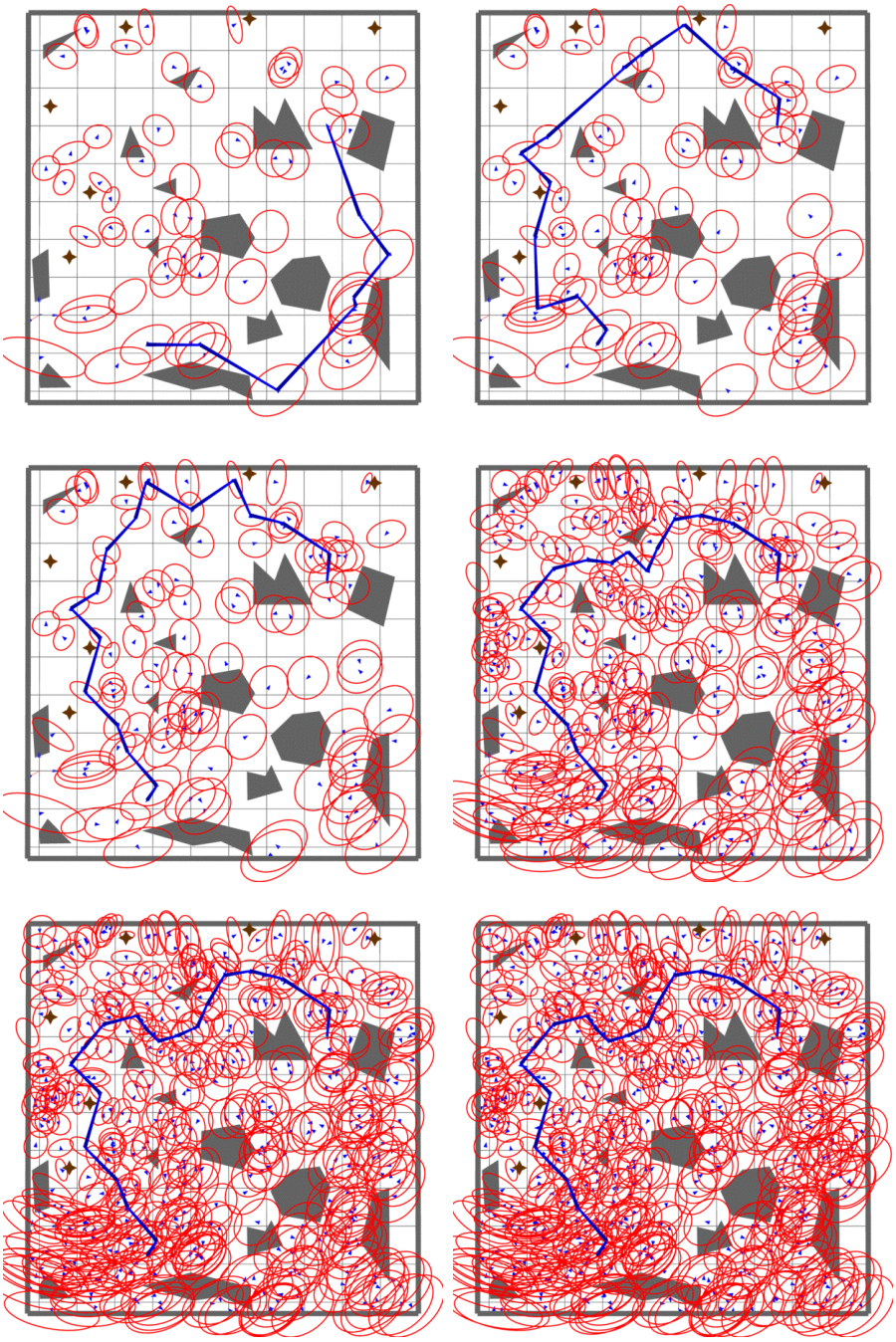


Figure 5.5: This figure shows different snapshots of the roadmap for 50, 75, 105, 275, 425, and 500 nodes, respectively. The most likely path under the optimal plan is also shown in blue. Stars indicate landmarks. Mean and covariance of the FIRM node centers are shown by small blue triangles and their associated red ellipses, respectively. Also, see [68–71] for videos of planning with FIRM in this environment.

5.8.3 8-arm Manipulator

On a given graph, the number of paths between two given points grows exponentially with the size of graph. Thus, in the direct propagation of uncertainty on a roadmap, the number of edge costs and transition probabilities that need to be computed is exponential in the number of underlying PRM nodes (see Section 5.9 for a detailed analysis). As a result, when we deal with high dimensional state spaces, where PRM needs to have many edges and nodes, it is not feasible to use the methods that perform direct uncertainty propagation. However, using FIRM, we only need to compute the costs and transition probabilities for as many edges as the underlying PRM has. Thus, we can easily increase the dimension to the level that PRM can handle, and the complexity of the algorithm is increased only by a constant factor (involving computation of costs and transition probabilities of a single edge). In the following experiment, we verify the effectiveness of FIRM in handling high-dimensional systems through a simple example of an 8-arm manipulator. To the best of our knowledge, this is the first belief space planner that can provide a plan over an entire roadmap for an eight-dimensional system, while incorporating expensive costs in planning such as computing collision probabilities. This experiment shows that FIRM can be used as a practical tool in real-world problems.

We consider an 8-arm manipulator with eight revolute joints in the plane. The state of the system is described by the angles of joints and their velocities $x = (\theta_1, \cdots, \theta_8, \dot{\theta}_1, \cdots, \dot{\theta}_8)^T$, and the available control is considered to be the angular acceleration (or torque) of joints $u = (\alpha_1, \alpha_2, \cdots, \alpha_8)$. The process noise $w = (w_1, w_2, \cdots, w_8)$ is modeled as a zero-mean Gaussian noise on angular accelerations. Therefore, the continuous motion model for every link is $\ddot{\theta}_i = \alpha_i + w_i$, whose discrete

version for the entire state can be written as:

$$x_{k+1} = Ax_k + Bu_k + Gw_k \tag{5.29}$$

where

$$A = \begin{pmatrix} I_8 & I_8\delta t \\ 0_8 & I_8 \end{pmatrix}, \quad B = \begin{pmatrix} 0_8 \\ I_8\delta t \end{pmatrix}, \quad G = \begin{pmatrix} 0_8 \\ I_8\sqrt{\delta t} \end{pmatrix}. \tag{5.30}$$

δt is the time interval between two consecutive time steps, and I_n and 0_n are the identity matrix and square zero matrix of dimension n, respectively.

We use the light-dark environment setting as the observation model, which is also used in [78, 79]. In the light-dark environment, the accuracy of sensory readings is encoded by a gray level, in which the regions that have access to more accurate sensory readings are lighter than the regions that do not have access to such informative sensory readings. In this experiment, we assume that we measure the state of the system, but this measurement is more accurate as we get closer to the left wall on which our sensor is mounted. (This model is adopted from [79].) Thus, we have $z = h(x) = [z^1, \cdots, z^8]^T$, where

$$z^i = \theta_i + v^i, \qquad v^i \sim \mathcal{N}(0, (\eta|\mathtt{x}^i - l| + \sigma_b)^2) \tag{5.31}$$

where $\mathtt{x}^i$ is the x coordinate of the i-th joint location, and l is the location of the vertical wall. η defines the dependency of the noise standard deviation on the distance from wall, and σ_b is the bias standard deviation. Figure 5.6 shows an example of such an environment, in which $l = -1.5$, $\eta = .1$, and $\sigma_b = 10^{-4}$. The full observation

model can be written as:

$$z_k = h(x_k) = Hx_k + Mv_k \tag{5.32}$$

where, $H = [I_8, 0_8]$ and $M = I_8$.

The described system is a controllable and observable system, and thus we adopt the SLQG controller as the stabilizing controller. Therefore, the parameters of the controller are points in the equilibrium space, as explained in previous sections. In other words, to generate sample nodes in the state space, we need to sample the configuration space $(\theta_1, \cdots, \theta_8)$ and append zero angular velocities to it. To connect these samples in the state space we design simple trajectories between nodes, along which we accelerate the joints (angles) by a constant acceleration until half way to the next node and thereafter we decelerate the joints until reaching the next node.

First, corresponding to sampled nodes in the state space, we compute corresponding FIRM nodes and then design local controllers according to Algorithm 7. In a similar procedure to the one in the previous experiment, we compute the transition costs and probabilities.

To solve the DP, we need to characterize the goal nodes. In Fig. 5.6, the goal region for the tip location of the manipulator is shown by a purple circle. We mark all PRM samples whose tip locations are within the goal region, as goal nodes. Setting the cost-to-go to zero for all goal nodes, we solve the DP and compute the optimal feedback on the graph according to Algorithm 7. Finally, we execute the plan based on Algorithm 8 and we illustrate the propagation of the covariance of the manipulator tip in Fig. 5.6 in red. As can be seen in Fig. 5.6, there are two passages among the obstacles to reach the goal region. Although the right passage is closer to the initial configuration of the manipulator, the manipulator detours to a longer path through

the left passage, because there is more accurate sensory information available in the left passage than the right one. As is seen in this example, the feedback plan minimizes the collision probability and picks the safest path, while it is robust to deviations. In other words, if for any reason the manipulator deviates significantly from the underlying PRM, the feedback plan connects the deviated belief to the best neighboring FIRM node, in real-time, and continues the pre-computed plan from this node.

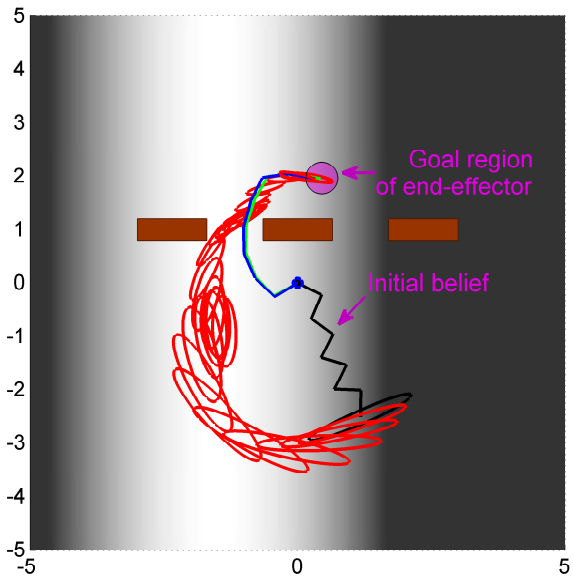


Figure 5.6: This figure shows a result of executing the FIRM plan for an 8-arm manipulator in a light-dark (sensing) environment. The manipulator is attached to the origin $(0, 0)$ and the purple region is the goal region for the manipulator tip. To simplify the figure, we only show the uncertainty of the manipulator tip (end-effector). The initial mean and covariance is shown by black, and the evolution of the tip covariance during the plan execution is shown in red. The final estimation mean and the true configuration of the manipulator are shown in blue and green, respectively. Obstacles are shown in brown. The manipulator follows a longer but safer path to the goal region through the left passage, compared to the shorter but risky (with high collision probability) path through the right passage.

5.9 Comparison

In this section, we perform a short comparison of SLQG-FIRM against the two most related methods in the literature: BRM [81] and LQG-MP on roadmaps [94]. Both methods are belief space planners that exploit roadmap-based ideas. We compare the methods in terms of the offline construction and online planning complexity, and also in terms of some other properties listed in Table 5.2. In the following, we go over the complexity analysis that leads to the entries in this table.

In a general graph, the number of paths between two given nodes is exponential in the number of nodes N. For example, if each node in a graph is connected to k-nearest neighbor nodes on the graph, for a search depth of d edges on the graph, the corresponding search tree contains k^d paths. Notice that each of these paths has d edges on it. Thus, if we directly (without using belief stabilizers) propagate the uncertainty on a roadmap for a depth of d, we have to evaluate the cost on dk^d edges. So, the asymptotic complexity of the overall problem is of the order $O(Nk^N)$. Now, if computing the cost and transition probabilities associated with each edge under uncertainty is a constant multiplier $O(c)$ of computing its cost in deterministic case, then the overall complexity of the methods based on direct belief propagation is also $O(Nk^N)$. On the other hand, in any variant of FIRM, due to the edge independence, only the cost of $O(Nk)$ edges needs to be constructed as in PRM, and thus the overall complexity of offline construction of FIRM is $O(Nk)$.

If the system deviates from the valid region of the plan, in direct propagation methods, edge costs need to be recomputed for all edges. So, in BRM and LQG-MP on roadmaps, the replanning complexity will be of the order $O(Nk^N)$. If the cost of each edge is defined in such a way that it only depends on the belief at the start and end of edge (i.e., does not depend on the belief along the edge), BRM can reduce

the computation complexity to $O(\frac{N}{l}k^N)$ through covariance factorization techniques, where l is assumed to be the length (number of steps) of each edge. In FIRM, in the case of replanning (submitting a query with new starting point), it is only required to connect the deviated belief to k neighboring FIRM nodes. Thus, we only need to compute the cost for the k new edges. It is worth noting that if the underlying PRM is dense enough such that the valid region of the local controllers covers the space, edge cost computation in the replanning phase reduces to zero because if the system deviates out of a valid region of a local planner, it will fall into the valid region of some other planner.

To reduce the complexity of the search algorithm in BRM and LQG-MP on roadmaps, it is assumed that the costs on different edges of the roadmap are independent. This heuristic can reduce the complexity of the algorithm, but still it may be significantly high compared to the PRM or FIRM. Moreover, this heuristic (edge independent assumption) is not true without having belief stabilizers, and thus search algorithms relying on such a heuristic may result in solutions arbitrarily different from the true solution of the search algorithm. Assuming that no such heuristic is used in the search algorithm, Table 5.2 summarizes the complexity of these algorithms.

Table 5.2: Belief Space Roadmap-based Method Comparison (without using heuristic in search algorithms)

Algorithm	**offline construction complexity (no heuristic)**	**replanning (online planning) complexity**	**future observations**	**System requirement**	**valid region of plan**	**Collision probabilities**
Generic PRM	$O(Nk)$	$O(k)$	———	assumes a controller exists to drive the system from node-to-node	on the graph only	———
BRM	$O(Nk^N)$	$O(\frac{N}{l}k^N)$ or $O(Nk^N)$	maximum likelihood observation	well-linearizable systems	vicinity of the nominal path	not considered
LQG-MP on Roadmaps	$O(Nk^N)$	$O(Nk^N)$	All observations	well-linearizable systems	vicinity of the nominal path	simplified measures are used
Generic FIRM	$O(Nk)$	$O(k)$	———	assumes a controller exists to drive the system from node-to-node	union of convergence regions of local controllers	———
SLQG-FIRM	$O(Nk)$	$O(k)$ or $O(1)$	All observations	well-linearizable, and linear controllable and observable systems	vicinity of whole PRM (entire space for a dense PRM)	computed

The huge reduction in the computational complexity of the planning algorithm (in particular, in the online phase), opens many possibilities in utilizing POMDP solvers in real-world applications. Moreover, due to its sampling-based nature, it ameliorates the curse of dimensionality just as PRM does in the deterministic case. In other words, if the dimension of the system increases, we need a greater number of nodes N in the underlying PRM to capture the free space connectivity, in which case we cannot use direct methods due to their complexity. However, FIRM can tolerate the increase in the dimension since its complexity is only a constant multiplier of the PRM complexity.

6. FIRM INSTANTIATION FOR NONHOLONOMIC SYSTEMS

In this chapter, we develop a concrete FIRM for nonholonomic systems. The belief reachability in this case is accomplished by combining the Kalman filter and Dynamic Feedback Linearization (DFL) based controller respectively as the estimator and separated controller. We refer to this variant of FIRM as DFL-based FIRM.particular.

We start this chapter by discussing some of the challenges in dealing with nonholonomic systems. Then, we present the procedure of constructing stabilizers and local controllers (i.e., FIRM edges) based on a combination of Kalman filter and DFL-based separated controllers. Finally, we describe the planning algorithms with this framework and demonstrate its performance.

6.1 Controllability in Nonholonomic Systems

An implicit assumption in roadmap-based methods such as PRM is that on every edge there exists a controller to drive the robot from the start node to the end node of the edge or to an ϵ-neighborhood of the end node, for some small $\epsilon > 0$. For a linearly controllable robot, a linear controller can locally track a PRM edge and drive the robot to its endpoint node. However, for a nonholonomic robot such as a unicycle, the linearized model at any state point (and zero velocity) is not controllable, and hence, a linear controller cannot stabilize the robot to the PRM nodes. Consider the

discrete unicycle model:

$$x_{k+1} = f(x_k, u_k, w_k) = \begin{pmatrix} \mathsf{x}_k + (V_k + n_v)\delta t \cos\theta_k \\ \mathsf{y}_k + (V_k + n_v)\delta t \sin\theta_k \\ \theta_k + (\omega_k + n_\omega)\delta t \end{pmatrix}, \tag{6.1}$$

where $x_k = (\mathsf{x}_k, \mathsf{y}_k, \theta_k)^T$ describes the robot state, in which $(\mathsf{x}_k, \mathsf{y}_k)^T$ is the 2D position of the robot and θ_k is the heading angle of the robot, at time step k. The vector $u_k = (V_k, \omega_k)^T$ is the control vector consisting of linear velocity V_k and angular velocity ω_k. The motion noise vector is denoted by $w_k = (n_v, n_\omega)^T \sim \mathcal{N}(0, \mathbf{Q}_k)$. Linearizing this system about the point (node) $\mathbf{v} = (\mathsf{x}^p, \mathsf{y}^p, \theta^p)$, one can conclude that the system is linearly controllable if and only if $V^p > 0$. Thus, in stabilizing the robot to a PRM node, where the nominal control is zero, $u^p = (V^p, \omega^p)^T = (0, 0)^T$, the system is not linearly controllable. Therefore, a linear controller cannot stabilize the unicycle to a PRM node. Moreover, based on the necessary condition in Brockett's theorem [21], even a smooth time-invariant nonlinear control law cannot drive the unicycle to a PRM node, and the stabilizing controller must be either discontinuous and/or time-varying.

On roadmaps in belief space, the situation is even more complicated, since the controller has to drive the robot to the ϵ-neighborhood of a belief node in belief space. Again, if the linearized system is controllable, using a linear stochastic controller such as the stationary LQG controller, one can drive the robot belief to the belief node as discuss in Chapter 5. However, if the linearized system about the desired point is not controllable, the belief stabilization, if possible, is much more difficult than state stabilization. Consider a unicycle robot, equipped with sensors measuring the range and bearings from a set of landmarks in the environment. Linearizing the motion and

sensing models of this system for stabilization purposes, we get a linearly observable system but not a linearly controllable system. In this section, we handle this situation by utilizing a DFL-based controller along with a Kalman filter to steer the system belief toward a pre-defined node in belief space.

6.2 DFL-based FIRM

To exploit the generic FIRM framework, one has to determine proper (B, μ) pairs of the FIRM nodes and local controllers. Also, there has to be a way of computing transition costs and probabilities. In this Chapter, we propose one such approach for nonholonomic systems, where we construct a FIRM in which belief stabilization is performed by compositing a Kalman Filter as the estimator and a Dynamic Feedback Linearization-based (DFL-based) controller as the belief controller. Accordingly, we design the reachable FIRM nodes B^j, and local planners μ^{ij}, required in (4.17). Then we discuss how the transition probabilities $\mathbb{P}^g(\cdot|B^i, \mu^{ij})$, and costs $C^g(B^i, \mu^{ij})$ in (4.17) are computed. Finally, we solve the corresponding FIRM MDP and provide the algorithms for offline construction of DFL-based FIRM and online planning with it. To deal with Gaussian beliefs, we rely on the notation defined in Chapter 5.

In Chapter 5, the optimal LQG controller is utilized for the construction of local controllers in FIRM. However, it is limited to models which are linearly stabilizable to a point in state space, which excludes the class of nonholonomic systems such as a unicycle model. Here, we combine the Kalman filter and a DFL-based controller to construct a belief stabilizable for the unicycle model with partial information. This construct is a suboptimal design for the controller in a partially-observable environment. However, it is efficient in practice, and shows a promising solution for constructing a Feedback-based Information RoadMap (FIRM) for nonholonomic systems such as the unicycle model.

6.2.1 Estimator Design

For the state estimation we adopt the Kalman Filter (KF), which is commonly utilized for localizing unicycle robots [90]. Thus, the belief dynamic $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ is a result of the Kalman filtering equations. In Section 2.2.2, we review the Kalman filter and its stationary behavior in detail. Here, we only discuss the limiting belief behavior under the stationary KF (SKF).

Consider a PRM node $\mathbf{v}$. Let us denote the linear (linearized) system about the node $\mathbf{v}$ by the tuple $\Upsilon = (\mathbf{A}, \mathbf{B}, \mathbf{G}, \mathbf{Q}, \mathbf{H}, \mathbf{M}, \mathbf{R})$ that represents the following state space model:

$$x_{k+1} = \mathbf{A}x_k + \mathbf{B}u_k + \mathbf{G}w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{6.2a}$$

$$z_k = \mathbf{H}x_k + \mathbf{M}v_k, \qquad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{6.2b}$$

where w_k and v_k are motion and measurement noises, respectively, drawn from zero-mean Gaussian distributions with covariances $\mathbf{Q}$ and $\mathbf{R}$.

Consider a stationary KF (SKF), which is designed to estimate the state of the system in (6.2). Let us also define the matrix $\check{\mathbf{Q}}$ such that $\mathbf{G}\mathbf{Q}\mathbf{G}^T = \check{\mathbf{Q}}\check{\mathbf{Q}}^T$. Now, consider the class of systems that satisfy the following property:

Property 2. *The pair* $(\mathbf{A}, \check{\mathbf{Q}})$ *is a controllable pair [15], and the pair* $(\mathbf{A}, \mathbf{H})$ *is an observable pair [15].*

Lemma 7. *Given Property 2, the estimation covariance under the SKF, designed for the system in* (6.2)*, converges to the matrix* P_s*, independent of its initial covariance:*

$$P_s = P_s^- - P_s^-\mathbf{H}^T(\mathbf{H}P_s^-\mathbf{H}^T + \mathbf{M}\mathbf{R}\mathbf{M}^T)^{-1}\mathbf{H}P_s^-, \tag{6.3}$$

where, P_s^- is the unique symmetric positive semi-definite solution of the following Discrete Algebraic Riccati Equation (DARE):

$$P_{k+1}^- = \mathbf{A}(P_k^- - P_k^-\mathbf{H}^T(\mathbf{H}P_k^-\mathbf{H}^T + \mathbf{M}\mathbf{R}\mathbf{M}^T)^{-1}\mathbf{H}P_k^-)\mathbf{A}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T. \tag{6.4}$$

Proof. See [15]. □

The estimation mean, however, evolves randomly, as it is a function of obtained observations. In SKF, the evolution of estimation mean is as follows:

$$\widehat{x}_{k+1}^+ = (I - \mathbf{K}\mathbf{H})\mathbf{A}\widehat{x}_k^+ + (I - \mathbf{K}\mathbf{H})\mathbf{B}u_k + \mathbf{K}z_{k+1} + (I - \mathbf{K}\mathbf{H})(I - \mathbf{A})\mathbf{v}, \tag{6.5}$$

where, $\mathbf{K} = P_s^-\mathbf{H}^T(\mathbf{H}P_s^-\mathbf{H}^T + \mathbf{M}\mathbf{R}\mathbf{M}^T)^{-1}$.

6.2.2 Belief Controller (Separated Controller) Design

The belief system is an underactuated system, i.e., the dimension of control space is less than the dimension of belief space. However, we can have full control of the estimation mean, while based on Lemma 7 the estimation covariance under the SKF, tends to P_s. As a result, if we design a feedback controller to control the estimation mean towards node $\mathbf{v}$ in state space, and assuming the system remains in the valid linearization region of the SKF (which is a reasonable assumption), then the belief will approach $b_c = (\mathbf{v}, P_s)$ in belief space. Considering a stopping region in belief space, whose interior contains b_c, the belief process under the feedback control will hit the stopping region in a finite time.

For a nonholonomic system such as the unicycle model, the system is not linearly

controllable. Thus, we resort to the original nonlinear model, and utilize a Dynamic Feedback Linearization-based (DFL-based) controller to control the estimation mean for the nonholonomic unicycle model. The nonlinear form of (6.5) is:

$$\widehat{x}_{k+1}^{+} = f(\widehat{x}_{k}^{+}, u_k, 0) + \mathbf{K}\widetilde{z}_{k+1}, \tag{6.6}$$

where, $\widetilde{z}_{k+1}$ is the observation error, defined as

$$\begin{aligned}\widetilde{z}_{k+1} &= h(f(x_k, u_k, w_k), v_{k+1}) - h(f(\widehat{x}_{k}^{+}, u_k, 0), 0) \\ &\approx \mathbf{H}\mathbf{A}\widehat{e}_{k}^{+} + \mathbf{H}\mathbf{G}w_k + \mathbf{M}v_{k+1},\end{aligned} \tag{6.7}$$

in which the function h is the observation model that maps the states to observations, i.e., $z_k = h(x_k, v_k)$, where v_k models the zero-mean Gaussian sensing noise. Random variable $\widehat{e}_{k}^{+}$ is the estimation error defined by $\widehat{e}_{k}^{+} = x_k - \widehat{x}_{k}^{+}$. The approximation in (6.7) results from linearizing functions h and f. The important point is that the equation in the right hand side of (6.7) does not depend on u_k. Also, note that $\widehat{e}_{k}^{+}$ is unknown as it depends on the (unknown) true state x_k.

Kalman filters are vastly used for state estimation in non-holonomic systems and show great success in practice; thus, it is reasonable to assume that the estimation error $\widehat{e}_{k}^{+}$ is small. Therefore, the whole term $\mathbf{K}\widetilde{z}_{k+1}$ in (6.6) can be treated as a small control-independent perturbation affecting the system. Therefore, we adopt a controller to control the unicycle model, which is effectively robust to the noise injected by $\mathbf{K}\widetilde{z}_{k+1}$. The controller of choice is a DFL-based controller proposed in [73], as it offers a robust behavior with respect to disturbances. Moreover, it provides an exponentially fast stabilization procedure, and has a natural, and smooth, transient performance. Experimental results verify the robustness of this controller to the

above-mentioned disturbance $\mathbf{K}\widetilde{z}_{k+1}$.

To construct the DFL-based controller for unicycle model, first we transform the system state such that the target node coincides with the origin, i.e., if we denote $\mathbf{v} = (\mathbf{v}_\mathtt{x}, \mathbf{v}_\mathtt{y}, \mathbf{v}_\theta)$ and $\widehat{x}_k^+ = (\widehat{\mathtt{x}}_k^+, \widehat{\mathtt{y}}_k^+, \widehat{\theta}_k^+)$, we can transfer the system state as:

$$\begin{pmatrix} \breve{\mathtt{x}}_k^+ \\ \breve{\mathtt{y}}_k^+ \end{pmatrix} = \begin{pmatrix} \cos \mathbf{v}_\theta & -\sin \mathbf{v}_\theta \\ \sin \mathbf{v}_\theta & \cos \mathbf{v}_\theta \end{pmatrix}^{-1} \begin{pmatrix} \widehat{\mathtt{x}}_k^+ - \mathbf{v}_\mathtt{x} \\ \widehat{\mathtt{y}}_k^+ - \mathbf{v}_\mathtt{y} \end{pmatrix}, \tag{6.8a}$$

$$\breve{\theta}_k^+ = \widehat{\theta}_k^+ - \mathbf{v}_\theta. \tag{6.8b}$$

Now the controller has to drive the $\breve{x}_k^+ = (\breve{\mathtt{x}}_k^+, \breve{\mathtt{y}}_k^+, \breve{\theta}_k^+)$ to the origin. Ignoring the disturbance term in estimation mean dynamics, and assuming $\widehat{x}_{k+1}^+ = f(\widehat{x}_k^+, u_k, 0)$, we compute the estimation mean derivative in the last time step, based on the previous control signal $u_{k-1} = (V_{k-1}, \omega_{k-1})$:

$$\begin{pmatrix} \dot{\breve{\mathtt{x}}}_{k-1}^+ \\ \dot{\breve{\mathtt{y}}}_{k-1}^+ \\ \dot{\breve{\theta}}_{k-1}^+ \end{pmatrix} = \begin{pmatrix} V_{k-1} \cos \breve{\theta}_{k-1}^+ \\ V_{k-1} \sin \breve{\theta}_{k-1}^+ \\ \omega_{k-1} \end{pmatrix} \tag{6.9}$$

Accordingly, we compute the intermediate controls:

$$u_1' = -k_{p1} \breve{\mathtt{x}}_{k-1}^+ - k_{d1} \dot{\breve{\mathtt{x}}}_{k-1}^+ \tag{6.10}$$

$$u_2' = -k_{p2} \breve{\mathtt{y}}_{k-1}^+ - k_{d2} \dot{\breve{\mathtt{y}}}_{k-1}^+ \tag{6.11}$$

where, as described in [73], the condition on the gains are $k_{pi}, k_{di} > 0$ for $i = 1, 2$ and also $k_{d1}^2 - 4k_{p1} = k_{d2}^2 - 4k_{p2} > 0$ and $k_{d2} - k_{d1} > 2(k_{d2}^2 - 4k_{p2})^{.5}$.

Finally, we compute the control signal at time step k:

$$V_k = V_{k-1} + (u_1' \cos \breve{\theta}_{k-1}^+ + u_2' \sin \breve{\theta}_{k-1}^+)\delta t \tag{6.12}$$

$$\omega_k = (u_2' \cos \breve{\theta}_{k-1}^+ - u_1' \sin \breve{\theta}_{k-1}^+)V_{k-1}^{-1} \tag{6.13}$$

Therefore, the controller, parametrized by the target point $\mathbf{v}$, receives current estimation mean $\widehat{x}_k^+$ and the previous control u_k to generate the next control u_{k+1}. We show this mapping by $u_{k+1} = \mu(\widehat{x}_k^+, u_k)$.

6.2.3 Designing FIRM Nodes $\{B^j\}$

As mentioned, to construct a FIRM, we first construct its underlying PRM, characterized by its nodes and edges $\{\{\mathbf{v}^j\}, \{\mathbf{e}^{ij}\}\}$. Linearizing the system about the PRM node $\mathbf{v}^j$ results in a linear system $\Upsilon^j = (\mathbf{A}^j, \mathbf{B}^j, \mathbf{G}^j, \mathbf{Q}^j, \mathbf{H}^j, \mathbf{M}^j, \mathbf{R}^j)$:

$$x_{k+1} = \mathbf{A}^j x_k + \mathbf{B}^j u_k + \mathbf{G}^j w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}^j) \tag{6.14a}$$

$$z_k = \mathbf{H}^j x_k + \mathbf{M}^j v_k, \qquad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^j) \tag{6.14b}$$

Then, we design a stationary Kalman filter τ^j and a DFL-based belief controller μ^j corresponding to the system Υ^j. The controller μ^j is called the j-th *node-controller*. Accordingly, we choose the belief nodes B^j such that B^j is an ϵ-ball in belief space, centered at $b_c^j \equiv (\mathbf{v}^j, P_s^j)$, where P_s^j is the stationary covariance of the SKF designed for the system Υ^j, computed using (6.3). Therefore, B^j can be written as:

$$B^j = \{b \equiv (x, P) : \|x - \mathbf{v}^j\| < \delta_1, \|P - P_s^j\|_m < \delta_2\}, \tag{6.15}$$

where $\|\cdot\|$ and $\|\cdot\|_m$ denote suitable vector and matrix norms, respectively. The size of the FIRM nodes are determined by δ_1 and δ_2, which are sufficiently small to satisfy

the approximation in (4.12). Based on Lemma 7, the condition $\|P_k - P_s^j\|_m < \delta_2$ is satisfied after a deterministic finite time $k > N$ and based on the adopted DFL design, which has a global attractive behavior in state space (and exponentially fast stabilization), the condition $\|x - \mathbf{v}^j\| < \delta_1$ is satisfied in a finite random time.

6.2.4 DFL-based Local Controllers μ^{ij}

The role of the local controller μ^{ij} is to drive the belief from the node B^i to node B^j. To construct the local controller μ^{ij}, we precede the node-controller μ^j with a so called edge-controller $\overline{\mu}_k^{ij}$.

The main role of the edge-controller $\overline{\mu}_k^{ij}$ is that it takes the belief at node B^i and drives it to the vicinity of the node B^j, where it hands the system over to the node-controller μ^j, which in turn takes the system into a FIRM node B^i. As opposed to the point-stabilization procedure, if we linearize the unicycle model along the PRM edge $\mathbf{e}^{ij}$, where the nominal linear velocity is greater than zero, the unicycle is linearly controllable. As a result, we use a time-varying LQG controller to track the edge $\mathbf{e}^{ij}$.

Thus, overall, the local controller μ^{ij} is the concatenation of the edge-controller $\overline{\mu}_k^{ij}$ and the node-controller μ^j. By this construct, the expected stopping time of the node-controller decreases as the initial belief of the node-controller is closer to the target node B^j, due to the usage of the edge-controllers.

6.2.5 Transition Probabilities and Costs

In general, it can be a computationally expensive task to compute the transition probabilities $\mathbb{P}(\cdot|B^i, \mu^{ij})$ and costs $C(B^i, \mu^{ij})$ associated with invoking local controller μ^{ij} at node B^i. However, owing to the offline construction of FIRM, this is not an issue. We utilize sequential Monte Carlo methods [34] to compute the collision and absorption probabilities. For the transition cost, we first consider estimation accuracy to find the paths, on which the estimator, and consequently, the controller can

perform better. A measure of estimation error is the trace of estimation covariance. Thus, we use $\Phi^{ij} = \mathbb{E}[\sum_{k=1}^{\mathcal{T}} \mathrm{tr}(P_k^{ij})]$, where P_k^{ij} is the estimation covariance at the k-th time step of the execution of local controller μ^{ij}. The outer expectation operator is useful in dealing with the Extended Kalman Filter (EKF), whose covariance is stochastic [32,85]. Moreover, since we are also interested in faster paths, we take into account the corresponding mean stopping time, i.e., $\widehat{\mathcal{T}}^{ij} = \mathbb{E}[\mathcal{T}^{ij}]$, and the total cost of invoking μ^{ij} at B^i is considered as a linear combination of estimation accuracy and expected stopping time, with suitable coefficients ξ_1 and ξ_2.

$$C(B^i, \mu^{ij}) = \xi_1 \Phi^{ij} + \xi_2 \widehat{\mathcal{T}}^{ij}. \tag{6.16}$$

6.2.6 Construction of DFL-based FIRM and Planning With it

Algorithm 9 details the construction of FIRM. A crucial feature of FIRM is that it can be constructed offline and stored, independent of future queries. In the construction, presented in Algorithm 9, it is independent of the starting point of query. However, it can also be independent of the goal location, by postponing the DP equation solver (Line 16 of Algorithm 9) to the online phase (beginning of Algorithm 10). Moreover, owing to the reduction from the original POMDP to an n-state MDP on belief nodes, the FIRM MDP can be solved using standard DP techniques such as value/policy iteration to yield the optimal policy π^g that picks the optimal local planner $\mu^* = \pi^g(B^i)$ at each FIRM node B^i among all controller $\mu \in \mathbb{M}(\alpha, i)$. Given that the FIRM graph is computed offline, the online phase of planning (and replanning) on the roadmap becomes very efficient and thus, feasible in real time. If the given initial belief b_0 does not belong to any B_i, we create a singleton set $B_0 = b_0$. To connect the B_0 to FIRM, we first, compute the expected value of the robot state, i.e., $\mathbb{E}[x_0]$ using its distribution b_0 and add the $\mathbb{E}[x_0]$ to the

PRM nodes, and connect it to the PRM graph. The set of newly added edges going from $\mathbb{E}[x_0]$ to the nodes on PRM are called $\mathcal{E}(0)$. We design the local controllers associated with each edge in $\mathcal{E}(0)$ and call the set of them as $\mathbb{M}(0)$. Then choosing a local controller in $\mathbb{M}(0)$, the belief enters one of the FIRM nodes, if no collision occurs. Thus, given the current node, we use policy π^g defined in (4.17) over FIRM nodes to find μ^*, and pick μ^* to move the robot into $B(\mu^*)$. Algorithm 10 illustrates this procedure.

Algorithm 9: Offline Construction of DFL-based FIRM

input : Free space map, $\mathbb{X}_{free}$
output : FIRM graph $\mathcal{G}$
Construct a PRM with nodes $\mathcal{V} = \{\mathbf{v}^j\}$, and edges $\mathcal{E} = \{\mathbf{e}^{ij}\}$, where $i, j = 1, \cdots, n$;
forall the *PRM nodes* $\mathbf{v}^j \in \mathcal{V}$ **do**
    Design the node-controller (DFL-based) μ^j to stabilize the system to $\mathbf{v}^j$;
    Compute the FIRM node center $b_c^j = (\mathbf{v}^j, P_s^j)$ using (6.3);
    Construct FIRM node B^j using (6.15) centered at b_c^j;
Collect all FIRM nodes $\mathbb{V} = \{B^j\}$;
forall the $(B^i, \mathbf{e}^{ij})$ *pairs* **do**
    Design the edge-controller $\overline{\mu}_k^{ij}$, as discussed in Section 6.2.4;
    Construct the local controller μ_k^{ij} by concatenating edge-controller $\overline{\mu}_k^{ij}$ and node-controller μ_k^j;
    Set the initial belief b_0 equal to the center of B^i, based on the approximation in (4.12);
    Generate sample belief paths $b_{0:\mathcal{T}}$ and state paths $x_{0:\mathcal{T}}$ induced by controller μ^{ij} invoked at B^i;
    Compute the transition probabilities $\mathbb{P}^g(F|B^i, \mu^{ij})$ and $\mathbb{P}^g(B^j|B^i, \mu^{ij})$ and transition costs $C^g(B^i, \mu^{ij})$;
Collect all local controllers $\mathbb{M} = \{\mu^{ij}\}$;
Compute cost-to-go J^g and feedback π^g over the FIRM by solving the DP in (4.17);
$\mathcal{G} = (\mathbb{V}, \mathbb{M}, J^g, \pi^g)$;
return $\mathcal{G}$;

Algorithm 10: Online Phase Algorithm (Planning with DFL-based FIRM)

input : Initial belief b_0, FIRM graph $\mathcal{G}$, Underlying PRM graph
if $\exists B^i \in \mathbb{V}$ *such that* $b_0 \in B^i$ **then**
    Choose the next local controller $\mu^{ij} = \pi^g(B^i)$;
else
    Compute $\mathbf{v}_0 = \mathbb{E}[x_0]$ based on b_0, and connect $\mathbf{v}_0$ to the PRM nodes. Call the set of newly added edges $\mathcal{E}(0) = \{\mathbf{e}^{0j}\}$;
    Design local planners associated with edges in $\mathcal{E}(0)$; Collect them in set $\mathbb{M}(0) = \{\mu^{0j}\}$;
    forall the $\mu \in \mathbb{M}(0)$ **do**
        Generate sample belief and state paths $b_{0:\mathcal{T}}$, $x_{0:\mathcal{T}}$ induced by taking μ at b_0;
        Compute the transition probabilities $\mathbb{P}(\cdot|b_0, \mu)$ and transition costs $C(b_0, \mu)$;
    Set $i = 0$; Choose the best initial local planner μ^{0j} within the set $\mathbb{M}(0)$ using (4.21);
while $B^i \neq B_{goal}$ **do**
    while $(\nexists B^j, s.t.,\ b_k \in B^j)$ *and "no collision"* **do**
        Apply the control $u_k = \mu_k^{ij}(b_k)$ to the system;
        Get the measurement z_{k+1} from sensors;
        if *Collision happens* **then return** Collision;
        Update belief as $b_{k+1} = \tau(b_k, \mu_k^{ij}(b_k), z_{k+1})$;
    Update the current FIRM node $B^i = B^j$;
    Choose the next local controller $\mu^{ij} = \pi^g(B^i)$;

6.3 Experimental Results

In this section, we illustrate the results of FIRM construction on a simple PRM. As a motion model, we consider the nonholonomic unicycle model whose kinematics are given in (6.1). As the observation model, in experiments, the robot is equipped with exteroceptive sensors that provide range and bearing measurements from existing radio beacons (landmarks) in the environment. The 2D location of the j-th landmark is denoted by L_j. Measuring L_j can be modeled as follows:

$$
\begin{aligned}
{}^j z &= [\|{}^j\mathbf{d}\|, \operatorname{atan2}({}^j d_2, {}^j d_1) - \theta]^T + {}^j v, \quad {}^j v \sim \mathcal{N}(\mathbf{0}, {}^j\mathbf{R}), \\
{}^j\mathbf{R} &= \operatorname{diag}((\eta_r \|{}^j\mathbf{d}\| + \sigma_b^r)^2, (\eta_\theta \|{}^j\mathbf{d}\| + \sigma_b^\theta)^2),
\end{aligned} \tag{6.17}
$$

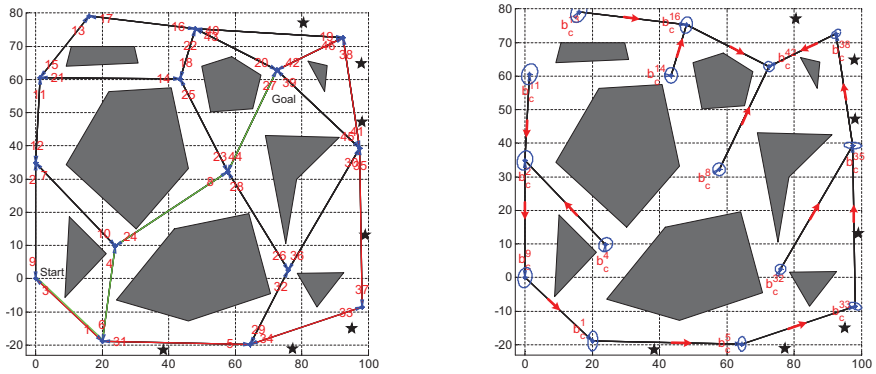
where ${}^j\mathbf{d} = [{}^jd_1, {}^jd_2]^T := [\mathtt{x}, \mathtt{y}]^T - L_j$. The uncertainty (standard deviation) of sensor readings increases as the robot gets farther from the landmarks. The parameters $\eta_r = \eta_\theta = 0.3$ determine this dependency, and $\sigma_b^r = 0.01$ meters and $\sigma_b^\theta = 0.5$ degrees are the bias standard deviations. A similar model for range sensing is used in [81]. The robot observes all N_L landmarks at all times and their observation noises are independent. Thus, the total measurement vector is denoted by $z = [{}^1z^T, {}^2z^T, \cdots, {}^{N_L}z^T]^T$ and due to the independence of measurements of different landmarks, the observation model for all landmarks can be written as $z = h(x) + v$, where $v \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and $\mathbf{R} = \text{diag}({}^1\mathbf{R}, \cdots, {}^{N_L}\mathbf{R})$.

Figure 6.1(a) shows a simple environment with nine radio beacons (black stars). A PRM is constructed in the 3D space of $(\mathtt{x}, \mathtt{y}, \theta)$ with 46 nodes and 102 edges. PRM nodes are shown by blue triangles with their numbers in red. To construct the FIRM nodes, we first solve the DARE corresponding to each PRM node and design its corresponding DFL-based node-controller. Then, we form the node centers $b_c^j = (\mathbf{v}^j, P_s^j)$, some of which are drawn in Fig.6.1(a), by the 3σ ellipse (in blue) of the covariance P_s^j. Finally, to handle the error scale difference in position and orientation variables, we construct the FIRM nodes based on the component-wise version of (6.15), as follows:

$$B^j = \{b \equiv (x, P) | \ \ |x - \mathbf{v}^j| \dot{<} \epsilon, |P - P_s^j| \dot{<} \Delta\}, \tag{6.18}$$

where $|\cdot|$ and $\dot{<}$ stand for the absolute value and component-wise comparison operators, respectively. We set $\epsilon = [0.8, 0.8, 5^\circ]^T$ and $\Delta = \epsilon\epsilon^T$ to quantify the B^j's.

After designing FIRM nodes and local controllers, the transition costs and probabilities are computed in the offline construction phase. Here, we use sequential weighted Monte Carlo based algorithms [34] to compute these quantities. In other



(a) (b)

Figure 6.1: A sample PRM, with numbered nodes. Seven landmarks are shown by black stars and obstacles are shown by gray polygons. (a) Node 9 is the start node and nodes 20, 27, 39, and 42 are goal nodes. Shortest path and the most-likely path under FIRM policy are shown in green and red, respectively. (b) The center of FIRM node, i.e., b_c is drawn for a selected number of PRM nodes. The feedback π^g is visualized for those FIRM nodes by red arrows.

words, for every (B^i, μ^{ij}) pair, we perform M runs and accordingly approximate the transition probabilities $\mathbb{P}^g(B^j|B^i, \mu^{ij})$, $\mathbb{P}^g(F|B^i, \mu^{ij})$, and costs $C^g(B^i, \mu^{ij})$. A similar approach is detailed in [3]. Table 6.1 shows these quantities along the best path resulting from the FIRM policy (see Fig.6.1(a)), where $M = 101$ and the coefficients in (6.16) are $\xi_1 = 0.98$ and $\xi_2 = 0.02$. Along edges where none of the 101 particles have collided with obstacles the collision probability is approximated by a value less than $1/101 = 0.0099$. The expected value and standard deviation of the time it takes for the controller to drive the belief into the target node is also reported in Table 6.1. Table 6.2 shows the same quantities for the edges along the shortest path. Comparing these two tables, it is seen that the path returned by the FIRM policy is safer, in terms of collision probability, and more informative, in the sense of localization uncertainty, when compared to the shortest path.

Using the computed transition costs and probabilities in (4.17), we can solve the

Table 6.1: Computed costs for several pairs of node-and-controller using 101 particles along the path returned by π^g.

(B_i,μ^{ij}) pair	$B_9,\mu^{9,1}$	$B_1,\mu^{1,5}$	$B_5,\mu^{5,33}$	$B_{33},\mu^{33,35}$	$B_{35},\mu^{35,38}$	$B_{38},\mu^{38,42}$
$\mathbb{P}^g(F\|B_i,\mu^{ij})$	15.3846%	7.6923%	<0.99%	<0.99%	<0.99%	15.3846%
Φ^{ij}	4.5967	1.9831	0.68936	1.6048	0.58705	0.53226
$\mathbb{E}[\mathcal{T}^{ij}]$	144.4545	217.3077	86.1538	161.2308	73	180.5455
$\sigma[\mathcal{T}^{ij}]$	66.7224	28.2396	9.109	5.7757	2.7433	40.6924

Table 6.2: Computed costs for several pairs of node-and-controller using 101 particles along the shortest path.

(B_i,μ^{ij}) pair	$B_9,\mu^{9,1}$	$B_1,\mu^{1,4}$	$B_4,\mu^{4,8}$	$B_8,\mu^{8,27}$
$\mathbb{P}^g(F\|B_i,\mu^{ij})$	15.3846%	38.4615%	46.1538%	38.4615%
Φ^{ij}	4.5967	2.0181	2.8001	2.1664
$\mathbb{E}[\mathcal{T}^{ij}]$	144.4545	168.375	127.2857	111.25
$\sigma[\mathcal{T}^{ij}]$	66.7224	50.3841	12.9192	38.1042

DP problem and compute the policy π^g on the graph. This process is performed only once offline, independent of the starting point of the query. Fig. 6.1(b) shows the policy π^g on the constructed FIRM in this example. Indeed, at every FIRM node B^i, the policy π^g decides which local controller should be invoked, which in turn aims to take the robot belief to the next FIRM node.

Thus, the online part of planning is quite efficient, i.e., it only requires executing the controller and generating the control signal, which is done online. An important consequence of feedback π^g is efficient replanning. In other words, since π^g is independent of the query, if due to some unmodeled large disturbance, the robot's belief deviates significantly from the planned path, it suffices to bring the robot back to the closest FIRM node and from there π^g drives the robot to the goal region as shown in

Fig. 6.1(b). In Fig. 6.1(a), we also show the shortest path (green) and the resulting path under policy π^g (red). It can be seen that the path returned by the best policy detours from the shortest path to a path along which the filtering uncertainty is smaller, and on which it is easier for the controller to avoid collisions.

7. FIRM INSTANTIATION FOR NON-POINT-STABILIZABLE SYSTEMS

This chapter is concerned with designing a concrete FIRM method for systems with kinodynamical constraints, in particular, systems, whose velocity cannot fall below a certain threshold (referred to as "non-stoppable" or "non-point-stabilizable" systems). Consider a control problem where the system state is composed of the position and velocity $(x, \dot{x})$ of an object. Stabilizing this system to a state $(x = a, \dot{x} = b)$ where $b \neq 0$ is not possible, because in order to stabilize x to a, $\dot{x}$ must go to zero. As an example of a non-stoppable system, consider a system whose state only consists of a position x, but it has constraints on its velocity $\dot{x} > b > 0$. All fixed-wing aircraft fall into this category as their velocity cannot fall under some threshold to maintain the lift requirement. Thus, stabilizing such systems to a fixed state is a challenge. This challenge is even more difficult when stabilization has to be achieved under uncertainty. In this chapter, we propose a framework that circumvents the need for point stabilization in graph-based (roadmap-based) methods by means of stabilization to suitably designed periodic maneuvers.

The main contribution of this chapter is proposing an instantiation of the abstract FIRM framework that can handle non-stoppable systems, and in general dynamical systems (which are not stabilizable to a point with zero-velocity), such as fixed-wing aircraft. To do so, we first introduce periodic-node PRM in state space, whose nodes lie on periodic trajectories, each one called an orbit. Then, we analyze the use of Periodic Linear Quadratic Gaussian (PLQG) controllers as belief stabilizers. Accordingly, we propose an approach to characterize and select reachable regions in belief space under PLQG controllers. Then, periodic-node PRM is leveraged to a corresponding graph (FIRM) in belief space, which is constructed offline, independent

of future queries, and can be used efficiently for replanning purposes. Collision probabilities are incorporated in the planner's construction.

7.1 Periodic-Node PRM

We circumvent the problem of stabilization to roadmap nodes by designing a variant of PRM, called Periodic-Node PRM (PNPRM). Although there are different ways to address this problem in state space, the critical property of PNPRM is that it can be extended to the belief space and forms a roadmap in belief space such that the belief nodes are reachable without a point-stabilization process. Let us denote the motion model with $x_{k+1} = f(x_k, u_k, w_k)$, where state, control, and process noise at the k-th time step are denoted by x_k, u_k, and w_k, respectively.

Similar to traditional PRM, PNPRM also consists of nodes and edges. However, in PNPRM, the nodes lie on small T-periodic trajectories (trajectories with period T) in the state space, called orbits, which satisfy the control constraints and non-holonomic constraints of the moving robot. To construct a PNPRM, we first sample a set of orbits in the state space, and then on each orbit, a number of state nodes are selected. Let us denote the j-th orbit trajectory by $O^j := (x_k^{p^j}, u_k^{p^j})_{k\geq 0}$, where $x_{k+1}^{p^j} = f(x_k^{p^j}, u_k^{p^j}, 0)$, $x_{k+T}^{p^j} = x_k^{p^j}$, and $u_{k+T}^{p^j} = u_k^{p^j}$. The set of PNPRM nodes that are chosen on O^j is denoted by $\mathtt{V}^j = \{\mathbf{v}_1^j, \mathbf{v}_2^j, \cdots, \mathbf{v}_m^j\}$ where $\mathbf{v}_\alpha^j = x_{k_\alpha}^{p^j}$ for some $k_\alpha \in \{1, \cdots, T\}$. Edges in PNPRM do not connect nodes to nodes, but they connect orbits to orbits in a way that respect all the control constraints and nonholonomic constraints. Thus, the (i, j)-th edge denoted by $\mathbf{e}^{ij}$ connects O^i to O^j.

As a result, a node $\mathbf{v}_\alpha^i$ is connected to the node $\mathbf{v}_\gamma^j$ through concatenation of three path segments: (i) the first segment is a part of O^i that connects $\mathbf{v}_\alpha^i$ to the starting point of $\mathbf{e}^{ij}$. This part is called a *pre-edge* and is denoted by $\mathbf{e}^{i_\alpha j}$, (ii) the second segment is the edge $\mathbf{e}^{ij}$ itself that connects O^i to O^j, and (iii) the third segment is

a part of O^j that connects the ending point of $\mathbf{e}^{ij}$ to the $\mathbf{v}^j_\gamma$. This part is called the *post-edge* and is denoted by $\mathbf{e}^{ij_\gamma}$.

One form of constructing orbits is based on circular periodic trajectories, where the edges are the lines that are tangent to the orbits. Figure 7.1(a) shows a simple PNPRM with three orbits O^i, O^r, and O^j. On each orbit four nodes are selected which are drawn (dots) with different colors. Edges $\mathbf{e}^{ij}$ and $\mathbf{e}^{rj}$ connect the corresponding orbits. The covariance ellipses associated with each PNPRM node are shown in Fig. 7.1(b) and discussed further below.

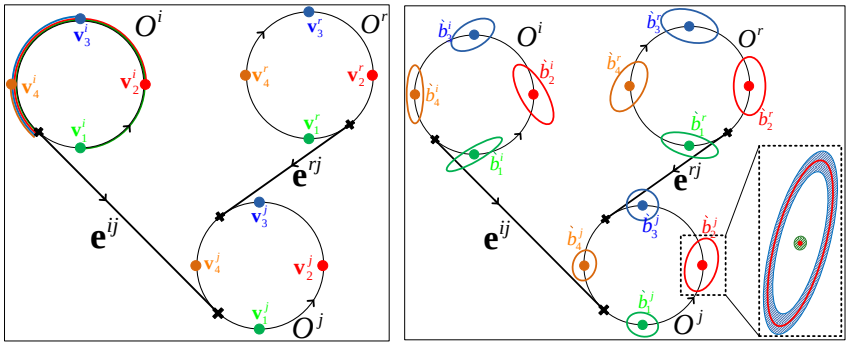


Figure 7.1: (a) A simple PNPRM with three orbits, twelve nodes, and two edges. (b) $\grave{b}^l_\alpha = (\mathbf{v}^l_\alpha, \check{P}^l_{k_\alpha})$ is the center of corresponding belief nodes, where $\check{P}^l_{k_\alpha}$'s are shown by their 3σ-ellipse. As an example of FIRM node, the magnified version of B^j_2, which is a small neighborhood centered at $\grave{b}^j_2$, is shown in the dotted box, where the blue shaded region depicts the covariance neighborhood and green shaded region depicts the mean neighborhood.

7.2 PLQG-based FIRM Construction

In this section, we construct a FIRM, in which local controllers are Periodic LQG (PLQG) controllers. Utilizing PLQG controllers, we design reachable FIRM nodes B^j_γ, and local planners $\mu^{\alpha,ij}$, required in (4.17). Then we discuss how the transition

probabilities $\mathbb{P}^g(\cdot|B^i_\alpha, \mu^{\alpha,ij})$, and costs $C^g(B^i_\alpha, \mu^{\alpha,ij})$ in (4.17) are computed. We start by defining notation needed for dealing with Gaussian beliefs.

7.2.1 Designing PLQG-based FIRM Nodes $\{B^j_\alpha\}$

An LQG controller is composed of a Kalman filter as the state estimator and an LQR controller. Thus, the belief dynamics $b_{k+1} = \tau(b_k, u_k, z_{k+1})$ are known, and come from the Kalman filtering equations, and the controller $u_k = \mu(b_k)$ that acts on the belief, comes from the LQR equations. LQG is an optimal controller for linear systems with Gaussian noise [15]. However, it is most often used for stabilizing nonlinear systems to a given trajectory or to a given point.

Periodic LQG (PLQG) is a time-varying LQG that is designed to track a given periodic trajectory [18, 30]. In Section 2.2.3 we have reviewed the periodic LQG controller in detail. Here, we only state the reachability result under the PLQG controller.

Consider a T-periodic PNPRM orbit $O = (x^p_k, u^p_k)_{k\geq 1}$ and the set of nodes $\{\mathbf{v}_\alpha\}$ on it. Let us denote the time-varying linear (linearized) system along the orbit O by the tuple $\Upsilon_k = (\mathbf{A}_k, \mathbf{B}_k, \mathbf{G}_k, \mathbf{Q}_k, \mathbf{H}_k, \mathbf{M}_k, \mathbf{R}_k)$ that represents the following state space model, where $\Upsilon_k = \Upsilon_{k+T}$:

$$x_{k+1} = \mathbf{A}_k x_k + \mathbf{B}_k u_k + \mathbf{G}_k w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k) \tag{7.1a}$$

$$z_k = \mathbf{H}_k x_k + \mathbf{M}_k v_k, \qquad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k), \tag{7.1b}$$

Consider the Periodic LQG (PLQG) controller that is designed for the system in (7.1) to track the orbit $(x^p_k, u^p_k)_{k\geq 1}$, through minimizing the following quadratic cost:

$$J = \mathbb{E}\left[\sum_{k\geq 0} \overline{x}^T_k \mathbf{W}_x \overline{x}_k + \overline{u}^T_k \mathbf{W}_u \overline{u}_k\right] \tag{7.2}$$

where $\overline{x}_k = x_k - x_k^p$ and $\overline{u}_k = u_k - u_k^p$. Matrices $\mathbf{W}_x$ and $\mathbf{W}_u$ are the positive definite weight matrices for state and control cost, respectively. Let us also define the matrices $\check{\mathbf{Q}}_k$ and $\check{\mathbf{W}}_x$ such that $\mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T = \check{\mathbf{Q}}_k\check{\mathbf{Q}}_k^T$, $\mathbf{W}_x = \check{\mathbf{W}}_x^T\check{\mathbf{W}}_x$, for all k. Now, consider the class of systems, and associated LQG controllers that satisfy the following property:

Property 3. *The pairs* $(\mathbf{A}_k, \mathbf{B}_k)$ *and* $(\mathbf{A}_k, \check{\mathbf{Q}}_k)$ *are controllable pairs [15], and the pairs* $(\mathbf{A}_k, \mathbf{H}_k)$ *and* $(\mathbf{A}_k, \check{\mathbf{W}}_x)$ *are observable pairs [15], for all* $k = 1, \cdots, T$.

The linearized model of a large class of systems satisfy the controllability condition in Property 3. Note that the linearization is done along the orbit (i.e., the nominal controls u_k^p are non-zero) and thus, even the linearized version of nonholonomic systems (e.g., unicycle model) satisfy this condition [73]. The observability condition on the pair $(\mathbf{A}_k, \mathbf{H}_k)$ is related to the sensor model and the sensors has to be designed in such a way to satisfy this condition.

In the following, we present three lemmas, through which we can construct pairs of periodic LQG controllers, and reachable nodes in belief space, for non-stoppable systems and systems with dynamics (possibly nonholonomic).

Lemma 8. *Consider the PLQG controller designed for the system in* (7.1) *to track the orbit* $(x_k^p, u_k^p)_{k\geq 1}$. *Given Property 3 is satisfied, in the absence of a stopping region, the belief process* b_k *under PLQG converges to a Gaussian cyclostationary process [17], i.e., the distribution over belief converges to a* T*-periodic Gaussian distribution, where we denote the mean and covariance of this process by* b_k^c *and* $\mathtt{C}_k$*, respectively:*

$$b_k \sim \mathcal{N}(b_k^c, \mathtt{C}_k) = \mathcal{N}(b_{k+T}^c, \mathtt{C}_{k+T}), \tag{7.3}$$

where $b_k \equiv (\widehat{x}_k^+, P_k)$ *and* $b_k^c \equiv (x_k^p, \check{P}_k)$*. The covariance matrices* $\check{P}_k$ *is characterized in Lemma 9 and covariance* $\mathtt{C}_k$ *is characterized in Section 2.2.3.*

Proof. See Section 2.2.3. □

Lemma 9. *Given Property 3, the following Discrete Periodic Riccati Equation (DPRE) has a unique Symmetric T-Periodic Positive Semi-definite (SPPS) solution [18], denoted by* $\check{P}_k^-$*:*

$$\begin{aligned}\check{P}_{k+1}^- &= \mathbf{G}_k\mathbf{Q}_k\mathbf{G}_k^T + \\ &\mathbf{A}_k(\check{P}_k^- - \check{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\check{P}_k^-\mathbf{H}_k^T + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T)^{-1}\mathbf{H}_k\check{P}_k^-)\mathbf{A}_k^T\end{aligned} \tag{7.4}$$

Moreover, the covariance matrix $\check{P}_k$ *introduced in Lemma 8 is computed as*

$$\check{P}_k = \check{P}_k^- - \check{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\check{P}_k^-\mathbf{H}_k^T + \mathbf{M}_k\mathbf{R}_k\mathbf{M}_k^T)^{-1}\mathbf{H}_k\check{P}_k^- \tag{7.5}$$

Proof. See Section 2.2.3 or [18]. □

Now, we state the main result, through which we can construct the proper pairs of periodic LQG controller and nodes in belief space for non-stoppable systems or systems with higher-order-dynamics.

Lemma 10. *Consider the PLQG controller* μ *designed for the system in* (7.1) *to track the orbit* $(x_k^p, u_k^p)_{k\geq 1}$*. Suppose the matrix* $\mathbf{H}_k$ *is full rank, and Property 3 is satisfied. Also, consider the sets* $B_1, B_2, \cdots, B_m$ *in belief space, such that the interior of* B_α *contains* $b_{k_\alpha}^c$ *for some* $k_\alpha \in \{1, \cdots, T\}$*. Then, under* μ*, the region* $\cup_\alpha B_\alpha$ *is reachable in a finite time with probability one.*

Proof. The intuitive idea behind the poof is that if we define a region centered at the mean value of a Gaussian distribution, and if we sample from this distribution,

in a finite number of samples we will end up with a sample in the given region. The proof is detailed in Section 2.2.3. □

As mentioned, to construct a FIRM, we first construct its underlying PNPRM, characterized by the triple $\{\{O^j\}, \{\mathbf{v}_\alpha^j\}, \{\mathbf{e}_{ij}\}\}$. Linearizing the system along the j-th orbit $O^j = (x_k^{p^j}, u_k^{p^j})_{k\geq 0}$ results in a time-varying T-periodic system $\Upsilon_k^j = (\mathbf{A}_k^j, \mathbf{B}_k^j, \mathbf{G}_k^j, \mathbf{Q}_k^j, \mathbf{H}_k^j, \mathbf{M}_k^j, \mathbf{R}_k^j)$:

$$x_{k+1} = \mathbf{A}_k^j x_k + \mathbf{B}_k^j u_k + \mathbf{G}_k^j w_k, \quad w_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k^j) \tag{7.6a}$$

$$z_k = \mathbf{H}_k^j x_k + \mathbf{M}_k^j v_k, \qquad v_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k^j). \tag{7.6b}$$

where w_k and v_k are motion and measurement noises, respectively, drawn from zero-mean Gaussian distributions with covariances $\mathbf{Q}_k^j$ and $\mathbf{R}_k^j$. The important property of the system in (7.6) is that it is a T-periodic system, i.e.,$\Upsilon_k^j = \Upsilon_{k+T}^j$. Then, we design a PLQG controller μ_k^j corresponding to the system Υ_k^j. The controller μ_k^j is called the j-th *node-controller*. Since the orbits are designed such that Property 3 is satisfied on them, based on Lemma 8 the belief converges to a Gaussian cyclostationary process, with mean $b_k^{c^j}$, which can be computed using Lemma 9, where its existence and uniqueness are also guaranteed. Knowing that $\mathbf{v}_\alpha^j$, for $\alpha = 1, \cdots, m$, lies on orbit O^j, such that $\mathbf{v}_\alpha^j = x_{k_\alpha}^{p^j}$, we choose the belief nodes B_α^j, for $\alpha = 1, \cdots, m$ such that B_α^j is an ϵ-ball in belief space, centered at $\grave{b}_\alpha^j := b_{k_\alpha}^{c^j} \equiv (x_{k_\alpha}^{p^j}, \check{P}_{k_\alpha}^j) = (\mathbf{v}_\alpha^j, \check{P}_{k_\alpha}^j)$: (See Fig.7.1(b).)

$$B_\alpha^j = \{b \equiv (x, P) : \|x - \mathbf{v}_\alpha^j\| < \delta_1, \|P - \check{P}_{k_\alpha}^j\|_m < \delta_2\}, \tag{7.7}$$

where $\|\cdot\|$ and $\|\cdot\|_m$ denote suitable vector and matrix norms, respectively. The size of the FIRM nodes is determined by δ_1 and δ_2. Based on Lemma 10, $\cup_\alpha B_\alpha^j$ is a reach-

able region under the node-controller μ_k^j. Note that δ_1 and δ_2 are sufficiently small thresholds that determine the size of FIRM node B_j that satisfy the approximation in (4.12).

7.2.2 PLQG-based Local Controllers $\mu^{\alpha,ij}$

The role of the local controller $\mu^{\alpha,ij}$ is to drive the belief from the node B_α^i to $\cup_\gamma B_\gamma^j$, i.e., to a node B_γ^j, for some $\gamma = 1, \cdots, m$. To construct the local controller $\mu^{\alpha,ij}$, we precede the node-controller μ_k^j, with a time-varying LQG controller $\overline{\mu}_k^{\alpha,ij}$, which is called the *edge-controller* here.

Consider a finite trajectory that consists of three segments: *i)* the pre-edge $\mathbf{e}^{i_\alpha j}$ as defined in Section 7.1, *ii)* the edge itself $\mathbf{e}^{ij}$, and *iii)* a part of O^j that connects the ending point of $\mathbf{e}^{ij}$ to $x_0^{p^j}$. Edge-controller $\overline{\mu}_k^{i_\alpha j}$ is a time-varying LQG controller that is designed to track this finite trajectory. The main role of the edge-controller is that it takes the belief at node B_i and drives it to the vicinity of a starting point of orbit O^j, where it hands over the system to the the node-controller, and the node-controller in turn takes the system to a FIRM node.

Thus, overall, the local controller $\mu^{\alpha,ij}$ is the concatenation of the edge-controller $\overline{\mu}_k^{\alpha,ij}$ and the node-controller μ_k^j. Note that since reachability is guaranteed by the node-controller (periodic LQG controller), by this construction, the stopping region $\cup_\gamma B_\gamma^j$ is also reachable under the local controller $\mu^{\alpha,ij}$. Hence the reachability condition is satisfied by this construction.

7.2.3 Transition Probabilities and Costs

In general, it can be a computationally expensive task to compute the transition probabilities $\mathbb{P}(\cdot|B_\alpha^i, \mu^{\alpha,ij})$ and costs $C(B_\alpha^i, \mu^{\alpha,ij})$ associated with invoking local controller $\mu^{\alpha,ij}$ at node B_α^i. However, owing to the offline construction of FIRM, it is not an issue in FIRM. We utilize sequential Monte Carlo methods [34] to compute

the collision and absorption probabilities. For the transition cost, we first consider estimation accuracy to find the paths, on which the estimator, and consequently, the controller can perform better. A measure of estimation error is the trace of estimation covariance. Thus, we use $\Phi^{\alpha,ij} = \mathbb{E}[\sum_{k=1}^{\mathcal{T}} \mathrm{tr}(P_k^{\alpha,ij})]$, where $P_k^{\alpha,ij}$ is the estimation covariance at the k-th time step of the execution of local controller $\mu^{\alpha,ij}$. The outer expectation operator is useful in dealing with the Extended Kalman Filter (EKF), whose covariance is stochastic [32, 85]. Moreover, as we are also interested in faster paths, we take into account the corresponding mean stopping time, i.e., $\widehat{\mathcal{T}}^{\alpha,ij} = \mathbb{E}[\mathcal{T}^{\alpha,ij}]$, and the total cost of invoking $\mu^{\alpha,ij}$ at B_α^i is considered as a linear combination of estimation accuracy and expected stopping time, with suitable coefficients ξ_1 and ξ_2.

$$C(B_\alpha^i, \mu^{\alpha,ij}) = \xi_1 \Phi^{\alpha,ij} + \xi_2 \widehat{\mathcal{T}}^{\alpha,ij}. \tag{7.8}$$

7.2.4 Construction of PLQG-FIRM and Planning With it

The crucial feature of FIRM is that it can be constructed offline and stored, independent of future queries. Note that based on the Algorithms 11 and 12, we still need to know the goal location. However, to be fully independent of both the start and goal location of the query, one can solve the DP in the online phase. Moreover, owing to the reduction from the original POMDP to an n-state MDP on belief nodes, the FIRM MDP can be solved using standard DP techniques such as value/policy iteration to yield the optimal policy π^g that picks the optimal local planner $\mu^* = \pi^g(B_\alpha^i)$ at each FIRM node B_α^i among all controller $\mu \in \mathbb{M}(\alpha, i)$. Algorithm 11 details the construction of FIRM. Given that the FIRM graph is computed offline, the online phase of planning (and replanning) on the roadmap becomes very efficient and thus, feasible in real time. If the given initial belief b_0 does not belong to any

B_i, we create a singleton set $B_0 = b_0$. To connect B_0 to FIRM, we first compute the expected value of the robot state, i.e. $\mathbb{E}[x_0]$ using its distribution b_0 and add $\mathbb{E}[x_0]$ to the PRM nodes, and connect it to the PRM graph. The set of newly added edges going from $\mathbb{E}[x_0]$ to the nodes on PRM is called $\mathcal{E}(0)$. We design the local controllers associated with each edge in $\mathcal{E}(0)$ and call the set of them $\mathbb{M}(0)$. Then choosing a local controller in $\mathbb{M}(0)$, the belief enters one of the FIRM nodes if no collision occurs. Thus, given the current node, we use policy π^g defined in (4.17b) over FIRM nodes to find μ^*, and pick μ^* to move the robot into $B(\mu^*)$. Algorithm 12 illustrates this procedure.

Algorithm 11: Offline Construction of PLQG-FIRM

input : Free space map, $\mathbb{X}_{free}$
output : FIRM graph $\mathcal{G}$
Construct a PNPRM with T-periodic orbits $\mathcal{O} = \{O^j = (x_k^{p^j}, u_k^{p^j})_{k\geq 0}\}$, nodes $\mathcal{V} = \{\mathbf{v}_\alpha^j\}$, and edges $\mathcal{E} = \{\mathbf{e}^{ij}\}$, where $i, j = 1, \cdots, n$ and $\alpha = 1, \cdots, m$;
forall the *PNPRM orbits* $O^j \in \mathcal{O}$ **do**
    Design the node-controller (periodic LQG) μ_k^j along the periodic trajectory;
    Compute the periodic mean belief trajectory $b_k^{c^j} = (x_k^{p^j}, \check{P}_k^j)$ using (7.5);
    Construct m FIRM nodes $\mathbb{V}^j = \{B_1^j, \cdots, B_m^j\}$ using (7.7), where B_α^j is centered at $b_{k_\alpha}^{c^j}$;
Collect all FIRM nodes $\mathbb{V} = \cup_{j=1}^n \mathbb{V}^j$;
forall the $(B_\alpha^i, \mathbf{e}^{ij})$ *pairs* **do**
    Design the edge-controller $\overline{\mu}_k^{\alpha,ij}$, as discussed in Section 7.2.2;
    Construct the local controller $\mu_k^{\alpha,ij}$ by concatenating edge-controller $\overline{\mu}_k^{\alpha,ij}$ and node-controller μ_k^j;
    Set the initial belief b_0 equal to the center of B_α^i, based on the approximation in (4.12);
    Generate sample belief paths $b_{0:\mathcal{T}}$ and state paths $x_{0:\mathcal{T}}$ induced by controller $\mu^{\alpha,ij}$ invoked at B_α^i;
    Compute the transition probabilities $\mathbb{P}^g(F|B_\alpha^i, \mu^{\alpha,ij})$ and $\mathbb{P}^g(B_\gamma^j|B_\alpha^i, \mu^{\alpha,ij})$ for all γ and transition cost $C^g(B_\alpha^i, \mu^{\alpha,ij})$;
Collect all local controllers $\mathbb{M} = \{\mu^{\alpha,ij}\}$;
Compute cost-to-go J^g and feedback π^g over the FIRM by solving the DP in (4.17);
$\mathcal{G} = (\mathbb{V}, \mathbb{M}, J^g, \pi^g)$;
return $\mathcal{G}$;

Algorithm 12: Online Phase Algorithm (Planning with PLQG-based FIRM)

input : Initial belief b_0, FIRM graph $\mathcal{G}$, Underlying PNPRM graph
if $\exists B^i_\alpha \in \mathbb{V}$ *such that* $b_0 \in B^i_\alpha$ **then**
    Choose the next local controller $\mu^{\alpha,ij} = \pi^g(B^i_\alpha)$;
else
    Compute $\mathbf{v}_0 = \mathbb{E}[x_0]$ based on b_0, and connect $\mathbf{v}_0$ to the PNPRM orbits. Call the set of newly added edges $\mathcal{E}(0) = \{\mathbf{e}^{0j}\}$;
    Design local planners associated with edges in $\mathcal{E}(0)$; Collect them in set $\mathbb{M}(0) = \{\mu^{0,0j}\}$;
    forall the $\mu \in \mathbb{M}(0)$ **do**
        Generate sample belief and state paths $b_{0:\mathcal{T}}$, $x_{0:\mathcal{T}}$ induced by taking μ at b_0;
        Compute the transition probabilities $\mathbb{P}(\cdot|b_0, \mu)$ and transition costs $C(b_0, \mu)$;
    Set $\alpha, i = 0$; Choose the best initial local planner $\mu^{0,0j}$ within the set $\mathbb{M}(0)$ using (4.21);
while $B^i_\alpha \neq B_{goal}$ **do**
    while $(\nexists B^j_\gamma, s.t.,\ b_k \in B^j_\gamma)$ *and "no collision"* **do**
        Apply the control $u_k = \mu_k^{\alpha,ij}(b_k)$ to the system;
        Get the measurement z_{k+1} from sensors;
        if *Collision happens* **then return** Collision;
        Update belief as $b_{k+1} = \tau(b_k, \mu_k^{\alpha,ij}(b_k), z_{k+1})$;
    Update the current FIRM node $B^i_\alpha = B^j_\gamma$;
    Choose the next local controller $\mu^{\alpha,ij} = \pi^g(B^i_\alpha)$;

7.3 Experimental Results

In this section, we illustrate the results of FIRM construction on a simple PN-PRM. As a motion model, we consider the nonholonomic unicycle model whose kinematics are as follows:

$$x_{k+1} = f(x_k, u_k, w_k) = \begin{pmatrix} \mathsf{x}_k + (V_k + n_v)\delta t \cos\theta_k \\ \mathsf{y}_k + (V_k + n_v)\delta t \sin\theta_k \\ \theta_k + (\omega_k + n_\omega)\delta t \end{pmatrix}, \tag{7.9}$$

where $x_k = (\mathsf{x}_k, \mathsf{y}_k, \theta_k)^T$ describes the robot state (2D position and heading angle). The vector $u_k = (V_k, \omega_k)^T$ is the control vector consisting of linear velocity V_k and angular velocity ω_k. The motion noise vector is denoted by $w_k = (n_v, n_\omega)^T \sim \mathcal{N}(0, \mathbf{Q}_k)$. As the observation model, in experiments, the robot is equipped with exteroceptive sensors that provide range and bearing measurements from existing radio beacons (landmarks) in the environment. The 2D location of the j-th landmark is denoted by L_j. Denoting the vector from the robot to the j-th landmark by ${}^j\mathbf{d} = [{}^jd_1, {}^jd_2]^T := [{}^1x, {}^2x]^T - L_j$, measuring L_j can be modeled as follows:

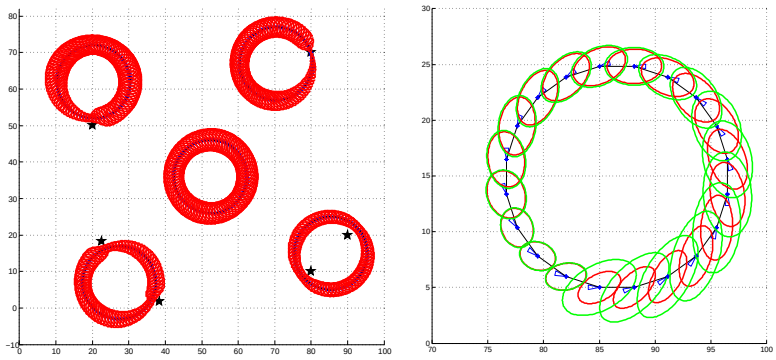
$${}^jz = {}^jh(x, {}^jv) = [\|{}^j\mathbf{d}\|, \text{atan2}({}^jd_2, {}^jd_1) - \theta]^T + {}^jv, \tag{7.10}$$

where, ${}^jv \sim \mathcal{N}(\mathbf{0}, {}^j\mathbf{R})$ and ${}^j\mathbf{R} = \text{diag}((\eta_r\|{}^j\mathbf{d}\| + \sigma_b^r)^2, (\eta_\theta\|{}^j\mathbf{d}\| + \sigma_b^\theta)^2)$. The uncertainty (standard deviation) of the sensor reading increases as the robot gets farther from the landmarks. The parameters $\eta_r = \eta_\theta = 0.3$ determine this dependency, and $\sigma_b^r = 0.01$ meter and $\sigma_b^\theta = 0.5$ degrees are the bias standard deviations. A similar model for range sensing is used in [81]. The robot observes all N_L landmarks at all times and their observation noise is independent. Thus, the total measurement vector is

denoted by $z = [{}^1z^T, {}^2z^T, \cdots, {}^{N_L}z^T]^T$ and due to the independence of measurements of different landmarks, the observation model for all landmarks can be written as $z = h(x) + v$, where $v \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ and $\mathbf{R} = \text{diag}({}^1\mathbf{R}, \cdots, {}^{N_L}\mathbf{R})$.

We first show a typical SPPS solution of DPRE on the orbits. Fig. 7.2(a) shows a simple environment with six radio beacons (black stars). For illustration purposes, we choose five large circular orbits and every orbit is discretized into 100 steps. Thus the SPPS solution of the DPRE in (7.4) on each orbit leads to a hundred covariance matrices that are superimposed on the graph in red. As is seen from Fig. 7.2(a), the localization uncertainty along the orbit is not homogeneous and varies periodically. Another important observation from the Fig. 7.2(a) is obtained by noticing the left top orbit in the Fig. 7.2(a). As can be seen, the localization uncertainty in the right hand side of the landmark, which lies close to orbit, is greater than its left hand side. In other words, this lack of symmetry emphasizes that although the phrase *The closer to the landmark, the lesser the sensing uncertainty* is true, the phrase *The closer to the landmark, the lesser the localization uncertainty* is not true, which emphasizes the role of dynamic model in filtering and its interaction with the observation model. In Fig. 7.2(b), we illustrate the covariance convergence in the periodic belief process. As can be seen in Fig. 7.2(b), the initial covariance is three times larger than the limiting covariance, and in less than one period it converges to the SPPS solution of DPRE. The convergence time is a random quantity, whose mean and variance can be estimated through simulation. However, in practical cases it usually converges in less than one full period, because the initial covariance is closer to the actual solution (due to the use of edge-controllers) and also the orbit size is much smaller, when compared to Fig. 7.2(b).

Figure 7.3(a) shows a sample PNPRM with 23 orbits and 67 edges. To simplify the explanation of the results, we assume $m = 1$, i.e., we choose one node on each



(a) (b)

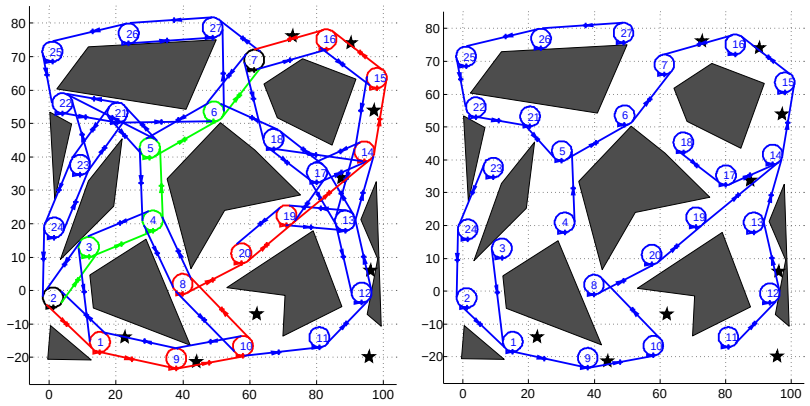
Figure 7.2: (a) Five orbits ($T = 100$) and corresponding periodic estimation covariances as the SPPS solution of DPRE in (7.4). (b) Sample covariance convergence on an orbit ($T = 20$) under PLQG. Red ellipses are the solution of DPRE and green ellipses are the evolution of estimation covariance. The initial covariance is three times bigger than the SPPS solution of DPRE, i.e., $P_0 = 3\check{P}_0$.

orbit. All elements in Fig.7.3(a) are defined in $(\mathtt{x}, \mathtt{y}, \theta)$ space but only the $(\mathtt{x}, \mathtt{y})$ portion is shown here. To construct the FIRM nodes, we first solve the corresponding DPREs on each orbit and design its corresponding node-controller (PLQG). Then, we pick the node centers $\grave{b}^j_\alpha = (\mathbf{v}^j_\alpha, \check{P}^j_{k_\alpha})$ and construct the FIRM nodes based on the component-wise version of (7.7), to handle the error scale difference in position and orientation variables:

$$B^j_\alpha = \{b \equiv (x, P) | \ \ |x - \mathbf{v}^j_\alpha| \dot{<} \epsilon, |P - \check{P}^j_{k_\alpha}| \dot{<} \Delta\}, \tag{7.11}$$

where $|\cdot|$ and $\dot{<}$ stand for the absolute value and component-wise comparison operators, respectively. We set $\epsilon = [0.8, 0.8, 5^\circ]^T$ and $\Delta = \epsilon\epsilon^T$ to quantify the B^j_α's.

After designing FIRM nodes and local controllers, the transition costs and probabilities are computed in the offline construction phase. Here, we use sequential weighted Monte Carlo based algorithms [34] to compute these quantities. In other



(a) (b)

Figure 7.3: A sample PNPRM with circular orbits. Number of each orbit is written in its center. Nine landmarks (black stars) and obstacles (gray polygons) are also shown. The moving directions on orbits and on edges are shown by little triangles with a cross in their heading direction. (a) Nodes 2 and 7 (distinguished in black) are start and goal nodes, respectively. Shortest path (green) and the most-likely path (red) under FIRM policy are also shown. (b) Assuming on each orbit, there exists a single node, the feedback π^g is visualized for all FIRM nodes.

words, for every $(B^i_\alpha, \mu^{\alpha,ij})$ pair, we perform M runs and accordingly approximate the transition probabilities $\mathbb{P}^g(B^j_\gamma | B^i_\alpha, \mu^{\alpha,ij})$, $\mathbb{P}^g(F | B^i_\alpha, \mu^{\alpha,ij})$, and costs $C^g(B^i_\alpha, \mu^{\alpha,ij})$. A similar approach is detailed in [3]. Table 7.1 shows these quantities for several $(B^i_\alpha, \mu^{\alpha,ij})$ pairs corresponding to Fig.7.3(a), where $M = 101$ and the coefficients in (7.8) are $\xi_1 = 0.98$ and $\xi_2 = 0.02$.

Table 7.1: Computed costs for several pairs of node-and-controller using 101 particles.

$(B^i_\alpha,\mu^{\alpha,ij})$ pair	$B^2_1,\mu^{1,(2,3)}$	$B^4_1,\mu^{1,(4,5)}$	$B^6_1,\mu^{1,(6,7)}$	$B^{11}_1,\mu^{1,(11,12)}$	$B^2_1,\mu^{1,(2,1)}$	$B^8_1,\mu^{1,(8,20)}$	$B^{16}_1,\mu^{1,(16,7)}$
$\mathbb{P}^g(F\|B^i_\alpha,\mu^{\alpha,ij})$	9.9010%	17.8218%	15.8416%	29.7030%	7.9208%	1.9802%	0.9901%
$\Phi^{\alpha,ij}$	2.1386	2.2834	1.9181	0.9152	2.1695	1.1857	0.4385
$\mathbb{E}[\mathcal{T}^{\alpha,ij}]$	63.6703	82.6747	62.5882	58.2000	51.7033	50.2755	35.4653

Plugging the computed transition costs and probabilities into (4.17), we can solve

the DP problem and compute the policy π^g on the graph. This process is performed only once offline, independent of the starting point of the query. Fig. 7.3(b) shows the policy π^g on the constructed FIRM in this example. Indeed, at every FIRM node B^i_α, the policy π^g decides which local controller should be invoked, which in turn aims to take the robot belief to the next FIRM node. It is worth noting that if we had more than one node on each orbit, the feedback π^g may return different controllers for each of them and for every orbit we may have more than one outgoing arrow in Fig. 7.3(b).

Thus, the online part of planning is quite efficient, i.e., it only requires executing the controller and generating the control signal. An important consequence of the feedback π^g is the efficient replanning procedure. In other words, since π^g is independent of query, if due to some unmodeled large disturbances, the system deviates significantly from the planned path, it suffices to bring the system back to the closest FIRM node and from thereon the optimal plan is already known, i.e., π^g drives the robot to the goal region as shown in Fig. 7.3(b).

We show the most likely path under the π^g, in red in Fig. 7.3(a). The shortest path is also illustrated in Fig. 7.3(a) in green. It can be seen that the "most likely path under the best policy" detours from the shortest path to a path along which the filtering uncertainty is smaller, and it is easier for the controller to avoid collisions.

8. DYNAMIC ONLINE REPLANNING IN BELIEF SPACE: APPLICATION TO PHYSICAL MOBILE ROBOTS

In this chapter, we aim at closing the loop of (i) analysis, (ii) computation, and (iii) physical realization. We aim to implement the method on low-cost physical robots equipped with low-cost sensors. To cope with discrepancies between computational models and real-world models, we utilize the proposed scheme in Chapter 4 for dynamic replanning in belief space. We investigate the performance of the method in an office-like environment and demonstrate its robustness to changing environments, sensory failures, and large deviations.

8.1 Introduction

Consider an autonomous low-cost mobile robot, working in an office environment. Each time it visits a goal location (or accomplishes a task), a new goal location (task) is assigned to it. Therefore, the robot needs to change its plan according to each goal. Moreover, although in an office-like environment most objects are stationary, there exist objects whose state may change discretely (such as office doors that may be opened or closed). Therefore, the robot needs to replan when it encounters such changes in the environment. However, low-cost robots are not able to follow the control commands exactly due to motion noise and they do not have exact measurements due to sensor noise. Therefore, this problem calls for online planning algorithms in uncertain, partially observable environments, where the state of some objects (e.g., doors) is subject to change over time. This is a typical scenario for many service and healthcare robots operating in indoor home or office-like environments. In a broader sense, this problem is an instance of the problem of decision making and control under uncertainty. However, what makes the problem more difficult is the

need for a robust solution that is able to replan online to cope with discrete changes in the environment, failures in sensory system, and large deviations from the nominal plan.

In addition to inherent challenges in solving POMDPs (such as the *curse of dimensionality* and *curse of history*), when dealing with real-world physical systems, there is another important challenge: the *discrepancy between the real models with the models used for computation*. These include discrepancies in the process model (state evolution model), the process noise model (distribution), sensor model, sensing noise model (distribution), and the environment map. In other words, the equation used for modeling the state evolution always is a simplification of the system's physics or the distribution used for the process noise never exactly matches the true noise distribution. Such discrepancies can lead to deviations of the system from the desired plan. A plausible solution for this problem is an ability to replan dynamically as the system encounters such deviations or observes signs of discrepancy during the plan execution. Moreover, as mentioned, online replanning can handle discrete changes in the environment and make the system more robust to intermittent sensing failures.

The main body of POMDP literature, in particular sampling-based methods, propose single-query solvers, i.e., the computed solution depends on the initial belief [53,81,94]. Therefore, in replanning (planning from a new initial belief) almost all the computations need to be reproduced, which limits their usage in cases where online replanning is needed. In particular, dynamic replanning schemes such as Receding Horizon Control (RHC), where the planning problem needs to be solved frequently from new start points, calls for online policy generation which restricts the usage of single-query methods. However, multi-query methods such as FIRM provide a construction mechanism, independent of the initial belief of the system. As a result, they are suitable methods to be used in an RHC framework.

The emphasis of this chapter is on the implementation of FIRM on a physical robot. We also investigate how dynamic online replanning can generate a feedback plan that is robust to discrepancies between real models and computational models as well as robust to changes in the environment, failures in the sensory system, and large deviations from the nominal plan. We believe these results lay the ground work for further moving the theoretical POMDP framework toward practical application, and achieving long-term autonomy in robotic systems.

The main goal of these experiments is to have a belief space planner that can handle the uncertainties associated with a typical low-cost robot in an office-like environment. We use an iRobot Create platform (Figure 8.2), on which a Dell Latitude laptop with an on-board camera and wireless networking capability is mounted. As will be discussed further below, landmarks are installed in the environment. The robot can get noisy measurements of the relative range and bearing to landmarks. The desired behavior for the planner is to guide the robot to the goal through the parts of the environment where the robot can better localize itself and hence better avoid collisions. However, most importantly, we need the planner to be able to replan online so that it can cope with deviations resulted from model discrepancies, changes in the environment, large disturbances, and sensor failures.

To design and evaluate a planner with the mentioned properties, we consider a scenario, where the robot needs to operate in an office environment. We conduct an experiment where the robot needs to reach a goal, and each time it reaches a goal, a new goal is submitted by the user. During this long run the robustness of the method is investigated with respect to (i) changing obstacles, such as doors, and moving people, (ii) changes in the goal location, (iii) deviations due to missing information sources, and (iv) kidnap situations (significant sudden deviation in the robot's location). During the run, there are many situations where the robot needs

to replan: It needs to replan each time a new goal is submitted and move toward the new goal. Also, the robot encounters changes in the obstacle map. For example it encounters doors that are in a different state than they were supposed to be. Similarly, it encounters moving people. Observing these changes, the robot updates its map and replans online and updates its policy accordingly. Moreover, the robot may be "kidnapped" by a person to an unknown location during the run. Thus, the robot needs to recover from this catastrophic situation. Finally, the robot may deviate from its nominal location due to temporary failures in the sensing system. In all these cases a online replanning scheme can help robot to recover from the situation and accomplish toward its goal.

8.1.1 Environment

The specific environment for conducting experiments is the fourth floor of the Harvey Bum Bright (HRBB) building at the Texas A&M University campus in College Station, TX. A floor-plan can be seen in Fig. 8.1. The floor spans almost 40 meters of hallways whose width are almost 2 meters, which is distinguished in yellow and blue in Fig. 8.1. The main experiments are conducted in the region which is highlighted in blue in Fig. 8.1, part of which contains a large cluttered office (room 407). This area has interesting properties that makes the planning more challenging: (i) As is seen in Fig. 8.1, there are several doors in this office (407) which may be opened or closed. Two of these doors (front-door and back-door) are shown in Fig. 8.1. In addition, (ii) there are objects such as trash-cans in this environment which usually get displaced and block some of the landmarks in the environment. This is an instance of missing information sources. Finally, (iii) people are moving in this area. Therefore, a reactive behavior may displace the robot from its planned path, which introduces another challenge for the high level planner.

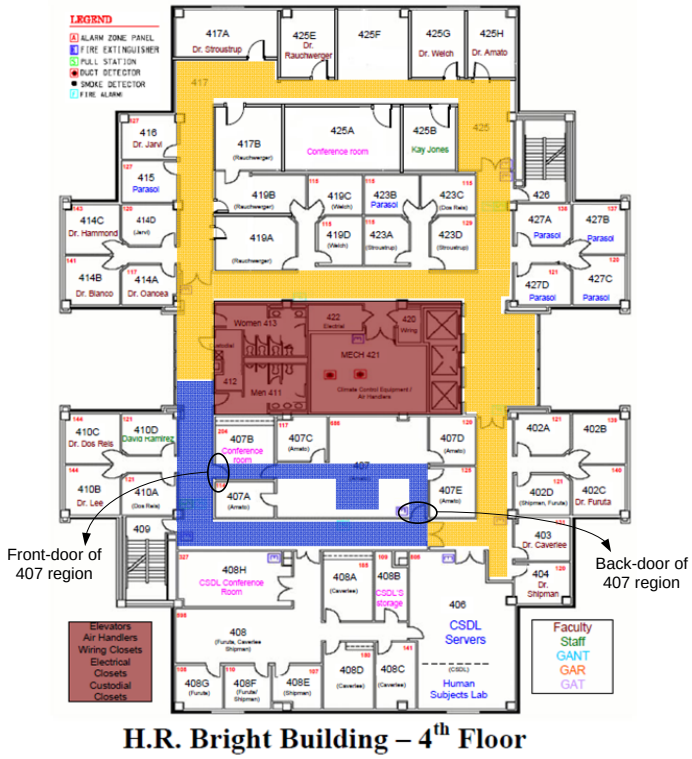


Figure 8.1: Floor-plan of the environment, in which experiments are conducted.

8.1.2 Robot Model

The robot utilized in our experiments is the iRobot Create mobile robot (See Fig. 8.2). The robot can be modeled as a unicycle whose kinematics is as follows:

$$x_{k+1} = f(x_k, u_k, w_k) = \begin{pmatrix} \mathsf{x}_k + (V_k \delta t + n_v \sqrt{\delta t}) \cos\theta_k \\ \mathsf{y}_k + (V_k \delta t + n_v \sqrt{\delta t}) \sin\theta_k \\ \theta_k + \omega_k \delta t + n_\omega \sqrt{\delta t} \end{pmatrix}, \tag{8.1}$$

where $x_k = (\mathsf{x}_k, \mathsf{y}_k, \theta_k)^T$ describes the robot state, in which $(\mathsf{x}_k, \mathsf{y}_k)^T$ is the 2D position of the robot and θ_k is the heading angle of the robot, at time step k. Control commands are the linear and angular velocities $u_k = (V_k, \omega_k)^T$. We use the Player

robot interface [37] to send these control commands to robot.

The motion noise vector is denoted by $w_k = (n_v, n_\omega)^T \sim \mathcal{N}(0, \mathbf{Q}_k)$, which mostly arose from uneven tiles on the floor, wheel slippage, and inaccuracy in the length of time control signals need to be applied. Experimentally, we found 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 k-th time step as

$$\mathbf{Q}_k = \begin{pmatrix} (\eta V_k + \sigma_b^V)^2 & 0 \\ 0 & (\eta \omega_k + \sigma_b^\omega)^2 \end{pmatrix} \tag{8.2}$$

where, in our implementations we have $\eta = 0.03$, $\sigma_b^V = 0.01$m/s, $\sigma_b^\omega = 0.001$rad $= 0.057$deg.

Figure 8.2: A picture of the robot (an iRobot Create) in the operating environment. Landmarks can be seen on the walls.

The connection between the planner and the robot hardware is established through

the Player robot interface [37].

8.1.3 Sensing Model

For sensing purposes, we use the on-board camera existing on the laptop. We perform a vision-based landmark detection based on ArUco (a minimal library for Augmented Reality applications) [63]. 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. 8.2) at the same height with the camera (robot is moving in a 2D space). The absolute position and orientation of each landmark in the environment is known. The ArUco library performs the detection process and presents the range and bearing relative to each visible landmark along with its id. Therefore, if we denote the j-th landmark position in the 2D global coordinates as ${}^{j}L$, we can model the observation as a range-bearing sensing system:

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

where ${}^{j}\mathbf{d}_k = [{}^{j}d_{1_k}, {}^{j}d_{2_k}]^T := [\mathtt{x}_k, \mathtt{y}_k]^T - L_j$.

A random vector ${}^{j}v$ models the measurement noise associated with the measurement of the j-th landmark. Experimentally, we found 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 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

$$ {}^{j}\mathbf{R}_k = \begin{pmatrix} (\eta_{r_d}\|{}^{j}\mathbf{d}_k\| + \eta_{r_\phi}|\phi_k| + \sigma_b^r)^2 & 0 \\ 0 & (\eta_{\theta_d}\|{}^{j}\mathbf{d}_k\| + \eta_{\theta_\phi}|\phi_k| + \sigma_b^\theta)^2 \end{pmatrix} \tag{8.3} $$

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 $\sigma_b^\theta = 2.0$deg.

At every step the robot observes a subset of the landmarks, which fall into its field of view. Suppose at a particular step the robot can see r landmarks $\{L_{i_1}, \cdots, L_{i_r}\}$. The concatenation of visible landmarks is the total measurement vector that is denoted by $z = [{}^{i_1}z^T, \cdots, {}^{i_r}z^T]^T$ and due to the independence of measurements of different landmarks, the observation model for all landmarks can be written as $z = h(x)+v$, where $v = [{}^{i_1}v^T, \cdots, {}^{i_r}v^T]^T$. Thus, the full measurement noise vector is drawn from $v \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, where $\mathbf{R} = \text{diag}({}^{i_1}\mathbf{R}, \cdots, {}^{i_r}\mathbf{R})$.

8.2 FIRM Elements

In this section, we discuss the concrete realization of the FIRM constructed for conducting the experiments.

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

$$ c(b_k, u_k) = \zeta_p \text{tr}(P_k) + \zeta_u \|u_k\| + \zeta_T. \tag{8.4} $$

where $tr(P_k)$ is the trace of estimation covariance as a measure of localization uncertainty, i.e., $P_k = \int_{\mathbb{X}} x^2 b_k dx - (\int_{\mathbb{X}} x b_k dx)^2$. 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 to combine these

costs to achieve a desirable behavior. In the presence of constraints (such as obstacles in the environment), we first 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.

To construct a FIRM graph, we first need to sample a set of stabilizers. Each stabilizer is a feedback controller (whose role is to drive the belief to a FIRM node), and thus it consists of a filter and a separated controller [50]. Similar to the case in Chapter 5, we first sample a set of points $\mathcal{V} = \{\mathbf{v}^i\}$ in the problem's state space and then associated with each point we construct a stabilizer. Since during stabilization the system is around the target point and aims to reach the target point $\mathbf{v}$, we adopt the Stationary Kalman Filter. As discussed in the previous chapter, an important observation is that if the system is observable the covariance under this filter converges to a stationary covariance P_s^+ that can be obtained through solving a corresponding Discrete Algebraic Riccati Equation (DARE), whose solution can be computed efficiently [11]. It is important to note that the stationary covariance P_s^+ does not depend on the choice of the separated-controller as long as the separated controller can keep the system close enough to the target node such that the linearized model about the target point is valid. However, if due to a large noise or any other reason the system goes far from the target point, the dynamic replanning will take care of the deviation as will be discussed in later sections.

A separated-controller μ is responsible for generating the control signal based on the available belief, i.e., $u_k = \mu(b_k)$. The iRobot Create is a nonholonomic robot and is modeled as a unicycle (see Section 8.1.2). Since the covariance is already approaching its stationary value, the separated-controller needs to only act on the mean value and drive it toward the target point $\mathbf{v}$. Therefore, we can use a variety

of controllers designed for stabilizing nonholonomic systems, e.g., [73], [64], and [83]. However, it is worth noting that the mean value gets affected by random observations, and thus it is impossible to take it to an exact point in the belief space. However, defining a ball around the target point $\mathbf{v}$, the controller can take the mean value into this ball in a finite time (note that the observation noise has a zero mean). As a result, the adopted controller needs to perform well under such uncertainties, which limits our choices. In our experiments, we implemented different controllers such as polar coordinate-based controller [33], or Dynamic Feedback Linearization-based controller (see Chapter 6). Observing their behavior on a physical robot, the best results were obtained using a variant of the Open-Loop Feedback Control (OLFC) scheme [15]. 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. Only one control (i.e., $l = 1$) is not enough due to the nonholonomicity of the system.

Therefore, similar to previous chapters, we construct FIRM node $B = \{b : \|b - \grave{b}\| \leq \epsilon\}$ associated with each sample $\mathbf{v}$, where $\grave{b} \equiv (\mathbf{v}, P_s^+)$. Characterizing a FIRM node for each sampled node $\mathbf{v}^i$, we get a set of FIRM nodes $\{B^i\}$. We denote the edge (controller) between nodes i and j by μ^{ij} and the set of edges by $\mathbb{M} = \{\mu^{ij}\}$. Computing costs and transition probabilities associated with each edge, we solve the DP in Equation 4.17 to get the optimal graph policy π^g (optimal mapping from graph nodes to edges); i.e., $\pi^g : \mathbb{V} \rightarrow \mathbb{M}$. We denote the set of all possible policies as Π^g.

The particular tracker and stabilizer we have adopted in our implementation are the same as the tracker and stabilizer used along the edges (described in Section 8.2). If the current belief is $b_0 = (\widehat{x}_0^+, P_0)$ we fix $x_0^d = \widehat{x}_0^+$. Then, for each FIRM node B^j, we

retrieve the corresponding state node $\mathbf{v}^j$ (based on $\grave{b}^j = (\mathbf{v}^j, P^j)$ (see Section 8.2)) and fix $x_n^d = \mathbf{v}^j$. Accordingly, aim to find the optimal $\mathbf{v}^j$ and intermediate deterministic states and controls $x_{0:n}^d, u_{0:n}^d$ that satisfy the system equations, $x_{k+1}^d = f(x_k^d, u_k^d, 0)$.

To also handle changes in the environment, changes in the goal location, or large deviations in its location, we adopt the FIRM-based rollout policy. Therefore, we aim to find a sequence of policies that ends up in a FIRM node and minimizes the cost in (4.59). To find this optimal policy, we parametrize the policy space $\widetilde{\Pi}$ and perform the minimization over the parameter space. For our particular system, the policy is considered to be a concatenation of a tracker μ_k (a controller to track a nominal trajectory) and a stabilizer μ^s (a controller that stabilize the belief to a fixed belief):

$$\pi(\cdot; \{x_{0:n}^d, \mathbf{v}\}) = \{\mu_k(\cdot; x_{0:n}^d), \mu^s(\cdot; \mathbf{v})\} \tag{8.5}$$

The tracker is parametrized by a nominal deterministic trajectory $x_{0:n}^d$. The role of tracker is to drive the belief to the vicinity of a belief node (FIRM node). The stabilizer is parametrized by a nominal state $\mathbf{v}$. The role of stabilizer is to drive the belief into the FIRM node.

8.3 Experimental Results on Planning with PRM and FIRM

In this section, we discuss results of PRM and FIRM-based motion planning on a low-cost iRobot Create equipped with a laptop. The integrated web-camera (monocular) is used to observe the landmarks. The goal of this section is to show how FIRM, as a belief space planner, guides the robot through regions with reduced collision probability, and more information to better localize the robot.

The solution of the dynamic programming 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 give graph node. Therefore, for each node, the feedback tree contains only one outgoing edge ($\mu = \pi^g(B^i)$). The feedback tree is rooted at the goal node.

The environment is shown in Fig. 8.3. 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. 8.3. The goal location is inside room 407 (see Fig. 8.1) and the start is close to the front door. In the following, we compare the performance (success probability) of MAPRM (Medial-Axis PRM), a conventional configuration space planner, with FIRM.

MAPRM builds the roadmap on the medial axis of the obstacles in the environment. In other works, the MAPRM nodes have maximum clearance (distance) from the obstacles. We construct Medial-Axis PRM (MAPRM) in this environment. The resulting roadmap is shown in Fig. 8.3. As can be seen, there exists a homotopy class of paths between the start and goal nodes through the front door of room 407 as well as a homotopy class of paths through the back door of the room. From Fig. 8.3, 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 the DP in this case is shown in Fig. 8.4. As expected, the DP guides the robot to go through the front door.

To execute the plan generated from PRM, we use time-varying LQG controllers to keep the robot close to the nominal path returned as the solution of the PRM-based planning. 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 along the path as the robot is prone to drift. The success probability along the nominal path is computed by a Monte Carlo simulation (100 runs) and is equal to 27% (27 runs out of 100 runs were successful).

As can be seen in Fig. 8.3, the distribution of information is not uniform in the environment. The density of landmarks (information sources) along the path through the back door is more than the landmarks along the path going through the front door. Incorporating the information distribution in the environment in planning leads to a better judgement of the narrowness of passages. For example, in this experiment, the path through the front door seems to be shorter than the path through the back door. However, considering the information sources the success probability of going through the back door is more than the success probability of going through the front door. Such knowledge about the environment is reflected in the FIRM cost-to-go and success probability in a principled rigorous framework. As a result, it generates a policy that suits the application, taking into account the uncertainty, and available information in the environment. Solving a DP problem on the FIRM graph gives a feedback as shown in Fig. 8.5, which results in an 88% success probability.

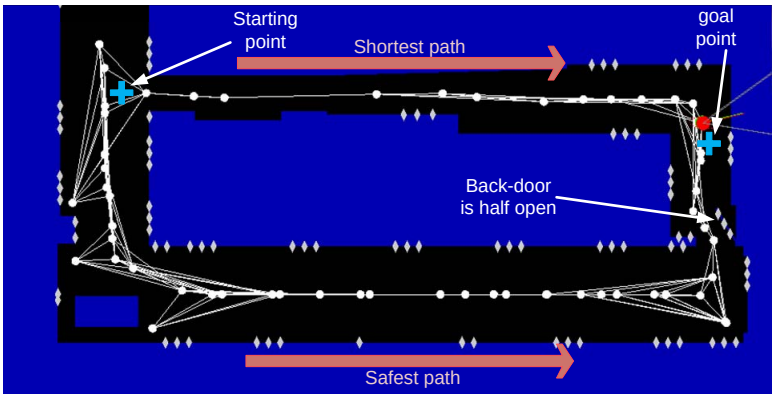


Figure 8.3: The environment including obstacles (blue regions), free space (black region), and landmark (white diamonds) are shown in this figure. An MAPRM graph approximating the connectivity of free space is also shown. The start and goal points (for the experiments in this section) are distinguished by crosses in the light blue.

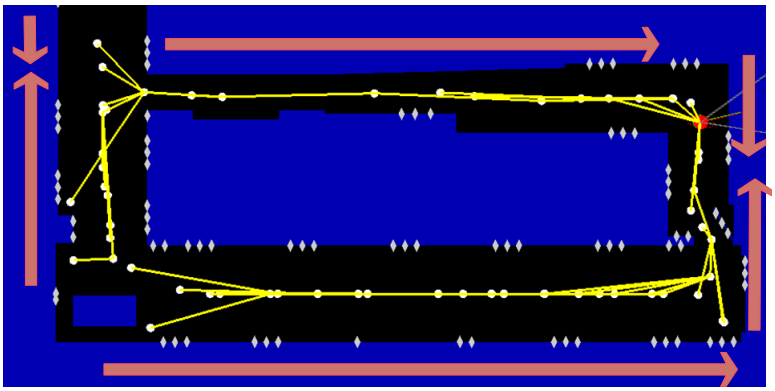


Figure 8.4: The feedback tree generated by solving DP on MAPRM is shown in yellow. From each node there is only one outgoing edge (in yellow), computed by DP, guiding the robot toward the goal (See Fig. 8.3). Arrows in pink coarsely represent the direction on which the feedback guides the robot.

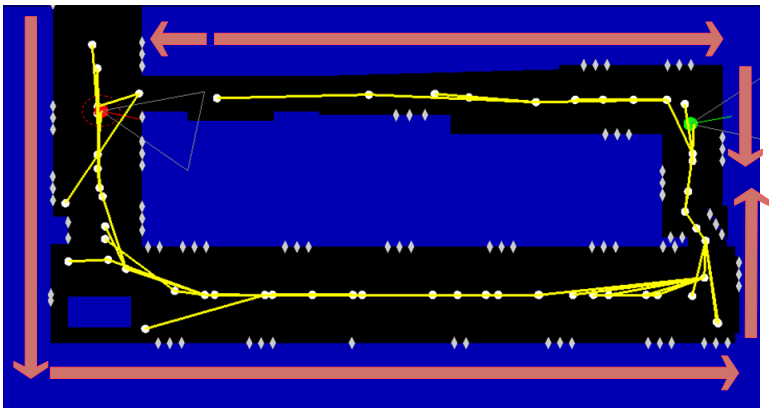


Figure 8.5: The feedback tree generated by solving DP on FIRM is shown. As can be seen the computed feedback guides robots through the more informative regions that leads to more accurate localization and less collision probabilities. Arrows in pink coarsely represent the direction on which the feedback guides the robot.

8.4 Robustness Experiments

In this section, we examine and discuss the robustness properties of the FIRM-based RHC framework for stochastic systems. We first look into the case where the obstacle map is subject to change. Then, we investigate the robustness to large deviations. Finally, we consider repeated changes in the goal location, along with the types of changes mentioned above and demonstrate the method's performance on a more complex scenario.

8.4.1 Robustness to Changes in the Environment: Obstacles and Information Sources

In our experiments, we consider three types of obstacles. Most obstacles are static such as walls. The second class of obstacles include those that discretely change their state such as doors in the environment (open and closed state). The last class of objects consists of people standing or moving in the environment. It is worth noting that dealing with a fully dynamic environment is not a goal of this research. To handle moving obstacles (people) we assume there exists a reactive planner at a lower level that suppresses the belief space planner in the vicinity of obstacles. Accordingly, after moving away from the moving obstacle, the robot may have deviated from its nominal plan and thus the belief space planner has to replan to recover from such deviations. The main focus of the following experiments is to demonstrate how our method can replan online when encountered with changes in the environment map. Dealing with agile obstacles or a fully dynamic environment requires development of more sophisticated reactive planners and is beyond the scope of this research.

As the first experiment, we consider the environment shown in Fig. 8.1. The start and goal locations are shown in Fig. 8.6(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 a corresponding FIRM and solve dynamic programming on it. As a result, we get the feedback tree shown in Fig. 8.6(a) that guides the robot to go 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 people moving in the environment. Moreover, for different reasons (such as blur in the image or blocked landmarks by people or different objects), the robot may miss detecting landmarks temporarily during the run.

We assume that the robot is equipped with a sensor that detects the obstacles in the vicinity of the robot. Such perception can be performed by a Laser Range Finder (LRF). However, designing the perception module is not a concern of this research, and since our robot is not equipped with a LRF, we use a simple method in our experiments to detect objects. We stick a small marker with a specific ID on moving objects (doors or people's shoes). When the robot observes these landmarks, it realizes that there is an obstacle in the vicinity of the robot. To handle such a change in the obstacle map and replan accordingly, we use the "lazy feedback evaluation" algorithm outlined below.

To adapt the proposed framework to the changing environment, we rely on lazy evaluation methods. Inspired by the lazy evaluation methods for PRM frameworks [20], we propose a variant of the lazy evaluation methods for evaluating the generated feedback tree. The basic idea is that at every node the robot re-evaluates *only* the next edge (or the next few edges up to a fixed horizon) that the robot needs to take. By re-evaluation, we mean it needs to re-compute the collision probabilities along these edges. If there is a significant change in the collision probabilities, 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. 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 13.

Algorithm 13: Lazy Feedback Evaluation (Lazy Replanning)

input : Feedback tree π^g, current belief $b_{current}$
output : Updated feedback tree, π^g
Update the obstacles map;
if *there is a change in map* **then**
    $\mathcal{F} \leftarrow$ Retrieve the sequence of nominal edges returned by feedback up to horizon l;
    forall the *edges* $\mu \in \mathcal{F}$ **do**
        Re-compute the collision probabilities $\mathbb{P}_{new}(B, \mu)$ from the start node B of edge;
    if *exists* $\mu \in \mathcal{F}$ *such that* $|\mathbb{P}_{new}(B, \mu) - \mathbb{P}(B, \mu)| > \alpha$ **then**
        $\mathbb{P}(B, \mu) \leftarrow \mathbb{P}_{new}(B, \mu)$;
        $\pi^g \leftarrow$ Replan$(b_{current})$;
return π^g;

Imagine a case where the robot is in a room with two doors. Suppose after checking both doors, the robot realizes they are closed. In such cases to persuade the robot to recheck the state of doors, we reset the door state to "open" after a specific amount of time as if the robot forgets that the door was "closed". In our experiments, the forgetting time for doors is 10 minutes, and the forgetting time for other moving obstacles is about 10 seconds.

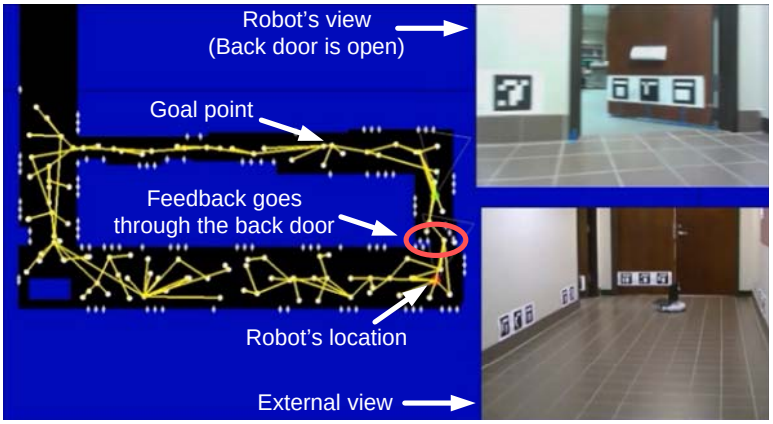
Figure 8.6(b) shows a snapshot of our run when the robot detects the change signal, i.e., detects the door is in a different state than its recorded situation in the map. As a result, the robot updates the obstacle map as can be seen in Fig. 8.6(b) (Door is closed). Accordingly, the robot replans; Figure 8.6(b) shows the feedback tree resulting from the replanning. As can be seen, 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 detail and is available in [67].

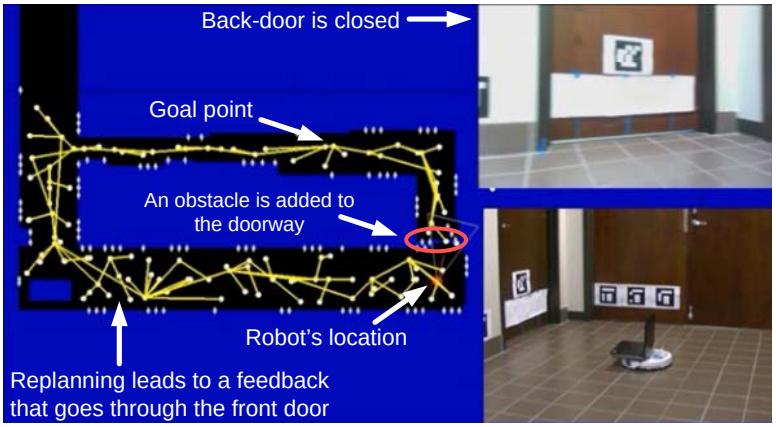
It is important to note that it is the graph structure of FIRM that makes such a replanning feasible online. The graph structure of FIRM allows us to *locally* change the collision probabilities in the environment without affecting the collision probability of the rest of the graph (i.e., properties of different edges on the graph are independent of each other). It is important to note that such a property is not present in any other state-of-the-art belief space planner, such as BRM (Belief Roadmap Method) [81], or LQG-MP [94]. In those methods, collision probabilities and costs on *all* edges (the number of possible edges is exponential in the size of the underlying PRM) need to be re-computed.

We may also miss detecting some information sources. For example, people may block the landmarks temporarily. Also, in our physical experiments, we observed that the rugged parts of the floor, which leads to a lot of jitter in the robot's motion, can make the captured images blurry. Thus, in those regions we may miss some landmarks intermittently (this is an example of discrepancy in computational models and physical models). Also, we constantly encountered objects (such as trash cans) that have been moved and block some of the landmarks. This phenomenon can also be a common issue for service robots.

Experimentally, we found that the effect of missing information sources in the environment is usually manifest in two ways: (i) an increase in stabilization time, or (ii) a deviation from the underlying nominal PRM edge. Therefore, we check both these conditions at each step and if either of them is satisfied, we use the replanning algorithm from the current belief. The current belief (initial belief for replanning) usually has a larger uncertainty due to the missing information sources, and thus replanning can take into account this growth in uncertainty.



(a)



(b)

Figure 8.6: (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.

8.4.2 Robustness to Large Deviations

In this subsection, 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*.

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 [29]. This problem is commonly used to test a robot's ability to recover from catastrophic localization failures. 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 accomplish its goal. Our main focus, here, is on the third part, i.e., how to replan in belief space from the new point in the belief space after recovering from being kidnapped.

To detect the kidnapped situation, we constantly monitor the innovation signal $\widetilde{z}_k = z_k - z_k^-$ (the different between the actual observations and predicted observation). To define the specific measure of innovation we use in our implementation, recall that 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:

$$\widetilde{r}_k = \max_j(|{}^j r_k - {}^j r_k^-|), \quad \widetilde{\theta}_k = \max_j(d^\theta({}^j \theta_k, {}^j \theta_k^-)) \tag{8.6}$$

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 are satisfied, we follow the FIRM feedback (i.e., we are in the *Feedback Following Mode* (FFM)). However, violation of any of these conditions 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 first replace the estimation

covariance 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 very small and conservative steps (e.g., turning in place or taking random actions with small velocities) to get some known measurements. Once the robot obtains a few measurements, the localization module corrects the estimation value and innovation signal reduces. When conditions $\overline{r}_k < r_{max}$ and $\overline{\theta}_k < \theta_{max}$ are satisfied again, we quit the information gathering mode.

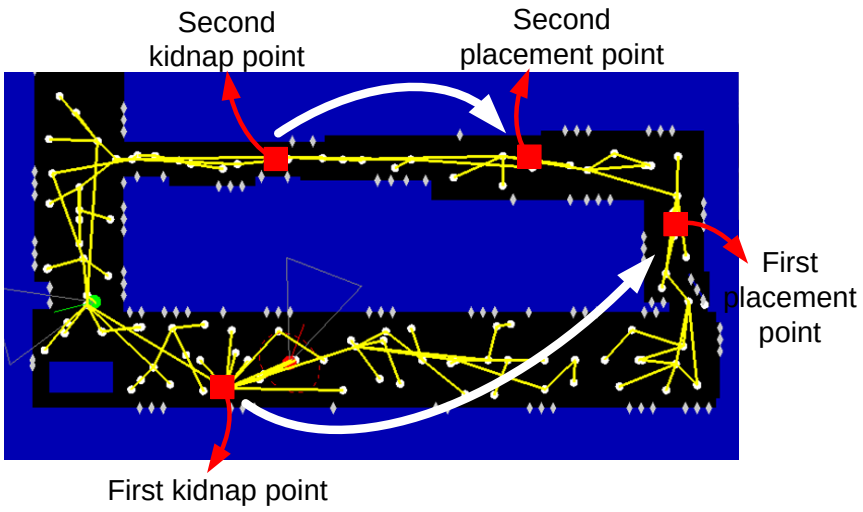


Figure 8.7: This figure shows the set up for the experiment containing two kidnapping.

After recovering from being kidnapped, controlling the robot in belief space is a significant challenge as the system can be far from where it was supposed to be. However, using FIRM, the robot just needs to go to a neighboring node from this new point. Since the FIRM graph is spread in the belief space, there is no need for 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.

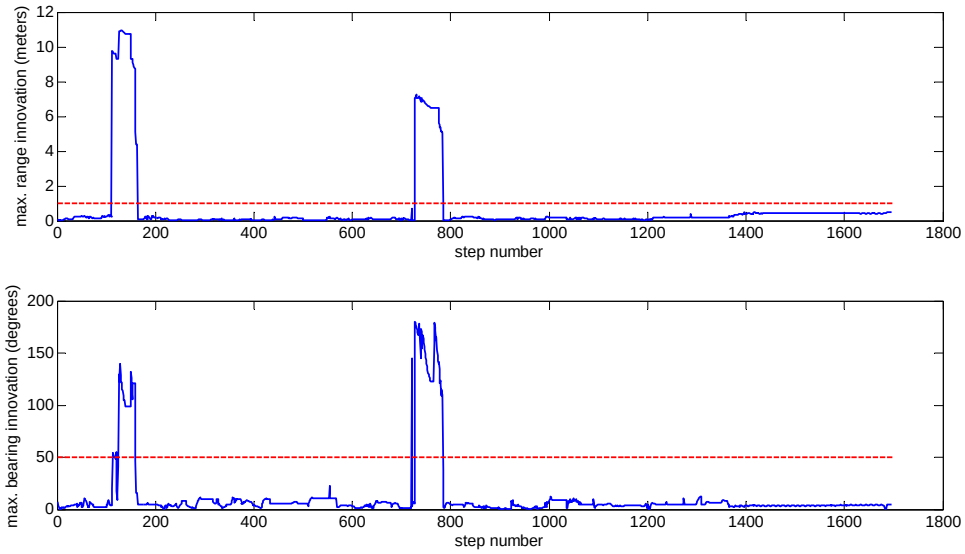


Figure 8.8: This figure shows the innovation signals $\widehat{r}_k$ and $\widehat{\theta}_k$ during this run. When both of the signals are below their specified thresholds r_{max} and θ_{max} (dashed red lines), robot follows the FIRM feedback. Otherwise, the system enters the information gathering mode.

Figure 8.7 shows a snapshot of a run that contains two kidnapping and illustrates the robustness of the planning algorithm to the kidnapping situation. The start and goal positions are shown in Fig. 8.7. 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 (see Fig. 8.7) and placed it in an unknown location within the 407 office (see Fig. 8.7). In our implementations, we consider $r_{max} = 1$ (meters) and $\theta_{max} = 50$ (degrees). The first jump in 8.8 shows this deviation. Once the robot recovers from being kidnapped (i.e., when both innovation signals in Fig. 8.8 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 room 407. However, again before the robot reaches the goal point, it is kidnapped and placed in an unknown location (see Fig. 8.7). The second jump

in the innovation signals in Fig. 8.8 corresponds to this kidnapping.

8.4.3 A Longer and More Complex Experiment: Robustness to Changing Goals, Obstacles, and Landmarks and to Large Deviations

In this section, we emphasize the ability of the system to perform long-term tasks that consist of visiting several goals. The replanning ability allows us to change the plan online as the goal location changes. In this experiment, we consider a scenario in which users submit a new goal for robot to reach after it reaches its currently assigned 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 missing information and kidnapped robot situations. Thus, the robot needs to perform many and frequent online replannings in belief space to cope with these changes. The video in [67] shows robots performance in this long and complex scenario.

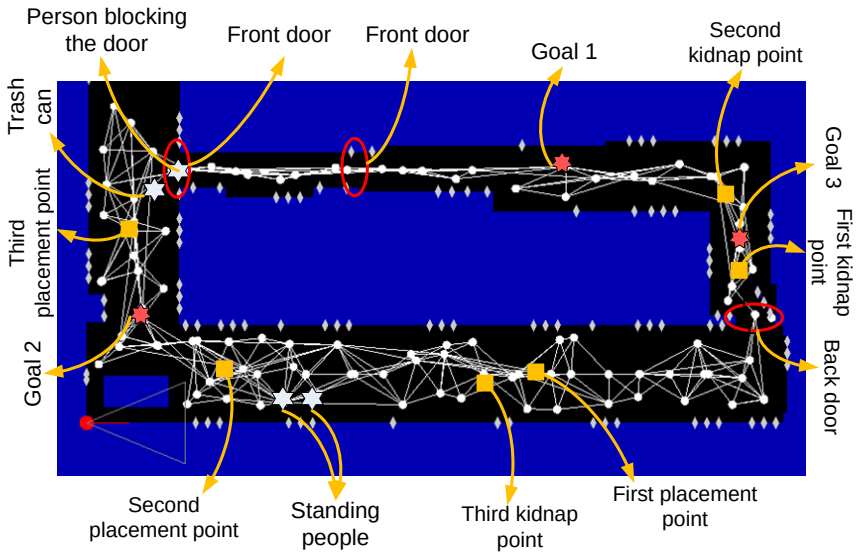


Figure 8.9: This figure shows the set up for the longer experiment with a sequence of goals as well as intermediate events and changes in the environment map.

In the following, we provide an itemized description of the specific steps involved in this run based on Fig. 8.9. Also, we discuss different changes in the environment with which the robot needs to cope along the way to accomplishing its goals. All of the following steps can be seen more clearly in the accompanying video [67].

1. The robot starts at the starting point shown in Fig. 8.9 and aims to reach goal 1 as shown in Fig. 8.9. Goal 1 is inside room 407. FIRM returns a feedback tree that guides the robot through the back door of 407 (cf. Fig. 8.9).

2. The robot goes through the narrow passage introduced by the back door (it is half-open). However, before reaching the goal it gets kidnapped (the first kidnap point as shown in Fig. 8.9). The robot is placed in an unknown location (shown in Fig. 8.9 by first placement point.)

3. Observing new landmarks, the robot detects that it has been kidnapped. Accordingly it adds a new node to the graph and replans online. As a result, the feedback guides the robot toward the goal point through the back door again.

4. However, in the meantime the back door has closed and when the robot reaches the vicinity of the back door, it detects that the door is closed. Therefore, it updates its map by closing the door (i.e., putting an obstacle at the doorway). Note that the robot will open the door (remove the obstacle) in its map after the forgetting time of 10 minutes. Accordingly, the robot replans a feedback tree that guides the robot through the front door toward the goal point.

5. Along the way, people are moving in the hallway and inside the 407 office. Thus, the robot replans accordingly as it encounters the people. Moving people are ignored but the standing people and static obstacles such as a trash-can (see Fig. 8.9) temporarily get added to the map as obstacles. Replanning several

times to cope with such changes, the robot goes through the front and inner doors and reaches the goal point inside the 407 office.

6. After reaching the goal point, another goal (second goal in Fig. 8.9) is assigned to the robot.

7. Replanning for reaching this goal leads to a feedback tree that guides the robot through the inner door, and front door, toward goal 2.

8. However, as the robot reaches the vicinity of the inner door, it detects the door has been closed. Therefore, it updates its map and replans accordingly. The replanning leads to a feedback tree that guides the robot toward goal 2 through the back door. Again, along the way robot encounters moving people in the office 407 and in the hallway.

9. However, before reaching the goal point, the robot gets kidnapped at the "second kidnap point" as shown in Fig. 8.9. The robot is placed at a really far-off point (the "second placement point"). Once the robot detects it is kidnapped, it replans and moves slower to gather information. Detecting landmarks, it reduces its uncertainty and continues going toward the goal point.

10. After reaching the goal point, the next goal (i.e., third goal) is assigned to the robot (see Fig. 8.9). Replanning (Re-query) for this goal, leads to a feedback that guides the robot through the front door.

11. However, when the robot reaches the front door, it encounters a person standing in the doorway. Accordingly, it replans and decides to go through the back door.

12. On the way to the back door, it is again displaced at the "third kidnap point" and placed at the "third placement point".

13. This time, due to the forgetting time, the replanning leads to a path through the front door (the person is not there any more).

14. Again, the robot follows the feedback and achieves its goal.

This long and complicated scenario demonstrates the robustness of the method to model discrepancies, changes in the environment, and large deviations in the robot's location. Such robustness stems from the ability to replan online in belief space. It is worth noting that online replanning in belief space is a challenge for state-of-the-art methods in belief space as they mainly rely on structures that depend on the system's initial belief. Hence, when the system's belief encounters a significant deviation, replanning from the new belief requires the structure to be re-built and it is a not a feasible operation online. However, constructing a query-independent graph in FIRM allows us to embed it in a replanning scheme such as the rollout policy technique and perform online replanning dynamically.

9. CONCLUSION AND FUTURE WORK

In this chapter, we review the contributions of this work and discuss future work that can utilize or extend this research.

In this work, we proposed the Feedback-based Information RoadMap (FIRM) framework for solving the motion planning problem under motion and sensing uncertainties. FIRM is the first multi-query graph-based method for planning in belief space, and hence it can be viewed as the extension of the celebrated Probabilistic Roadmap Method (PRM) to belief space. The results and contributions of the work can be discussed in three parts: (i) the abstract FIRM framework, (ii) concrete instantiations of FIRM for three main classes of robotic systems, and (iii) practical impact of FIRM.

The abstract FIRM framework proposes a method to reduce the original computationally intractable POMDP problem to a computationally tractable problem on a representative graph in belief space [2, 8]. The abstract FIRM utilizes feedback controllers to steer the belief toward graph nodes and establishes assumptions for these feedback controllers to ensure the belief node reachability in finite time. Hence, FIRM preserves the optimal substructure property on the roadmap and overcomes the curse of history in the original POMDP problem. Another important contribution of FIRM is its ability to seamlessly integrate constraints, such as collision probabilities, into the planning framework. An important feature of the solution returned by FIRM is that once the graph edge costs and transitions are formed, the success probability associated with the returned solution can be characterized analytically, which provides theoretical guarantees on the performance of the method. Moreover, we have extended the probabilistic completeness concept to planners un-

der uncertainty and showed that FIRM is PCUU (probabilistically complete under uncertainty) [4]. Therefore, for any given set of sampled nodes one can compute the success probability analytically, and increase the number of nodes to achieve a desired success probability (if such a policy exists in the class of graph policies) Finally, the computational complexity of the algorithm designed for the offline construction of FIRM, is a constant multiplier of the computational complexity of constructing the underlying PRM, i.e.; it is linear in the number of graph nodes. Therefore, it offers a scalable structure compared to the main body of belief space planners that are exponential in the number of underlying samples.

In this work, we constructed concrete instantiations of the abstract FIRM framework for three main classes of robotic systems: holonomic, nonholonomic, and non-point-stabilizable. SLQG-FIRM [2, 8] proposes a method to satisfy the assumptions established in the abstract framework for holonomic systems. SLQG-FIRM utilizes SLQG controllers as the belief stabilizers. The method characterizes the reachable beliefs under SLQG controllers and proposes a belief sampling and connecting techniques accordingly. Presenting algorithms for constructing the graph and planning with it, we demonstrated the performance of SLQG-FIRM on different scenarios, such as omni-directional mobile robots and an 8-arm manipulator. We analyzed the computational complexity of the method and provided concrete numbers on the method construction speed. Additionally, this work proposed an instantiation of the abstract FIRM for nonholonomic systems using dynamic feedback linearization-based controllers. This instantiation is referred to as the DFL-based FIRM [6]. Finally, we generalized the method from "point stabilization" to "periodic maneuver stabilization" to handle non-stoppable (or non-point-stabilizable) systems such as fixed-wing aircraft. The Periodic-Node PRM (PNPRM) is introduced whose nodes lie on periodic trajectories, called orbits. Exploiting the properties of periodic LQG controllers

on the orbits, the local planners in the PNPRM-based FIRM framework are realized by periodic LQG controllers, such that the distribution over the belief converges to a periodic distribution. Accordingly, it is shown that by suitably choosing the belief node regions along the orbits, the belief node reachability and hence the established assumptions in the abstract FIRM framework are achieved.

Equally important, the FIRM framework offers powerful tools for practical purposes. First, its multi-query (or query-independent) nature makes it robust to large deviations. In other words, since the optimal feedback on the graph is computed from all graph node offline, in the online phase large deviations can be compensated by driving the belief to a neighboring graph node and following the feedback from thereon. Second, FIRM provides more reliable solutions in the sense that the failure probabilities are accurately incorporated into the planning framework. Third, its scalability allows us to consider larger planning domains for POMDP problems. Finally, it is a suitable framework to be embedded in dynamic replanning schemes such as the rollout policy framework [1]. This is a key ability to handle (i) discrepancies between real world models and computational models, (ii) changes in the environment and obstacles, and (iii) large deviations. By implementing this belief space planner on a physical robot and demonstrating its robustness to the above-mentioned discrepancies, this method takes an important step in making POMDP methods applicable to real world robotic applications.

FIRM opens up new directions to follow for planning under uncertainty. The concrete FIRM instantiations proposed in this dissertation are limited to the Gaussian beliefs, and thus it is a subject of future work to design a non-Gaussian instantiation of the abstract FIRM framework. Analyzing stationary behavior of belief under non-Gaussian filters such as particle filters combined with an appropriate choice of a separated controller might be the first step in this direction. Similarly, devising in-

stantiations of the FIRM framework that can handle the systems with discrete state, control, or observations spaces is another future research direction. In other words, one can come up with belief stabilizers that work in discrete state space settings to design a discrete-state variant of FIRM.

REFERENCES

[1] Aliakbar Aghamohammadi, Saurav Agarwal, Aditya Mahadevan, Suman Chakravorty, Daniel Tomkins, Jory Denny, and Nancy Amato. Robust online belief space planning in changing environments: Application to physical mobile robots. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, Hong Kong, China, 2014.

[2] Aliakbar Aghamohammadi, 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)*, SanFrancisco, 2011.

[3] Aliakbar Aghamohammadi, Suman Chakravorty, and Nancy Amato. Motion planning in belief space using sampling-based feedback planners. *Technical Report: TR11-007, Parasol Lab., CSE Dept., Texas A&M University*, 2011.

[4] Aliakbar Aghamohammadi, Suman Chakravorty, and Nancy Amato. On the probabilistic completeness of the sampling-based feedback motion planners in belief space. In *IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, 2012.

[5] Aliakbar Aghamohammadi, Suman Chakravorty, and Nancy Amato. Periodic-feedback motion planning in belief space for non-holonomic and/or non-stoppable robots. *Technical Report: TR12-003, Parasol Lab., CSE Dept., Texas A&M University*, 2012.

[6] Aliakbar Aghamohammadi, Suman Chakravorty, and Nancy Amato. Sampling-based nonholonomic motion planning in belief space via dynamic

feedback linearization-based firm. In *International Conference on Intelligent Robots and Systems (IROS)*, Vilamoura, Portugal, 2012.

[7] Aliakbar Aghamohammadi, Suman Chakravorty, and Nancy Amato. Sampling-based stochastic control with constraints: A unified approach in state and information spaces. In *the American Control Conference (ACC)*, Washington DC, 2013.

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

[9] Aliakbar Aghamohammadi, Sandip Kumar, and Suman Chakravorty. *Motion Planning under Uncertainty.* Book Chapter, J. Valasek (Ed.), Advances in Intelligent and Autonomous Aerospace Systems, Progress in Astronautics and Aeronautics, American Institute of Aeronautics and Astronautics (AIAA), Reston, VA, 2012.

[10] Ron Alterovitz, Thierry Siméon, and Ken Goldberg. The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In *Proceedings of Robotics: Science and Systems (RSS)*, Atlanta, GA, June 2007.

[11] William F Arnold III and Alan J Laub. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proceedings of the IEEE*, 72(12):1746–1754, 1984.

[12] K Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.

[13] Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A Ngo. Monte carlo value iteration for continuous-state POMDPs. In David Hsu, Volkan Isler, Jean-Claude Latombe, and MingC. Lin, editors, *Algorithmic foundations of robotics IX*, volume 68 of *Springer Tracts in Advanced Robotics*, pages 175–191. Springer Berlin Heidelberg, 2011.

[14] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

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

[16] Dimitri P Bertsekas. *Dynamic programming and stochastic control*. Academic Press (New York), 1976.

[17] S. Bittanti, P. Bolzern, G. De Nicolao, and L. Piroddi. Representation, prediction, and identification of cyclostationary processes: a state-space approach. *in: W.A.Gardner (Ed.), Cyclostationarity in Communications and Signal Processing, IEEE Press, New York, NY*, 1994, pp. 267-295 (Chapter 5).

[18] Sergio Bittanti and Patrizio Colaneri. *Periodic Systems: Filtering and Control*. London: Springer London, 2009.

[19] R. Bohlin. *Robot Path Planning*. PhD thesis, Chalmers University of Technology, Goteborg, Sweden, 2002.

[20] Robert Bohlin and Lydia E Kavraki. Path planning using lazy PRM. In *IEEE International Conference on Robotics and Automation*, San Francisco, CA, 2000.

[21] Roger W. Brockett. Asymptotic stability and feedback stabilization. In R. S. Millman R. W. Brockett and H. J. Sussmann, editors, *Differential Geo-*

metric Control Theory, pages 181–191. Birkhauser, Boston, 1983.

[22] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 723–730, Shanghai, China, 2011.

[23] Andrea Censi, Daniele Calisi, Alessandro De Luca, and Giuseppe Oriolo. A Bayesian framework for optimal motion planning with uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, CA, May 2008.

[24] I. Chadès, E. McDonald-Madden, M. McCarthy, B. Wintle, M. Linkie, and H. Possingham. When to stop managing or surveying cryptic threatened species. *National Academy of Sciences of the USA*, 105(37):13936–13940, 2008.

[25] Suman Chakravorty and R.S. Erwin. Information space receding horizon control. In *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL)*, pages 302–309, Paris, France, April 2011.

[26] Suman Chakravorty and Sandip Kumar. Generalized sampling based motion planners with application to nonholonomic systems. In *Proceedings of the IEEE Internation Conference on Systems, Man, and Cybernetics*, San Antonio, TX, October 2009.

[27] Suman Chakravorty and Sandip Kumar. Generalized sampling-based motion planners. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 41(3):855–866, 2011.

[28] 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.

[29] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of robot motion: theory, algorithms, and implementations*. MIT Press, Cambridge, MA, 2005.

[30] P. Colaneri, R. Scattolini, and N. Schiavoni. LQG optimal control for multirate samled-data systems. *IEEE Transactions on Automatic Control*, 37(5):675–682, 1992.

[31] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Second Edition*. MIT Press, Cambridge, MA, 2001.

[32] John Crassidis and John Junkins. *Optimal Estimation of Dynamic Systems*. Chapman & Hall/CRC, Boka Raton, FL, 2004.

[33] Alessandro De Luca, Giuseppe Oriolo, and Marilena Vendittelli. Control of wheeled mobile robots: An experimental overview. In *Ramsete*, pages 181–226. Springer Berlin Heidelberg, 2001.

[34] A. Doucet, J.F.G. de Freitas, and N.J. Gordon. *Sequential Monte Carlo methods in practice*. New York: Springer, 2001.

[35] 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*, Catalina Island, California, 2010.

[36] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: theory and practicea survey. *Automatica*, 25(3):335–348, 1989.

[37] Brian Gerkey, Richard T Vaughan, and Andrew Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *11th Inter-*

national Conference on Advanced Robotics (ICAR), volume 1, pages 317–323, Coimbra, Portugal, 2003.

[38] L. Guibas, D. Hsu, H. Kurniawati, and E. Rehman. Bounded uncertainty roadmaps for path planning. In *International Workshop on Algorithmic Foundations of Robotics*, Guanajuato, Mexico, 2008.

[39] R. He, E. Brunskill, and N. Roy. PUMA: Planning under uncertainty with macro-actions. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI)*, Atlanta, GA, 2010.

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

[41] J. Hoey, A. von Bertoldi, P. Poupart, and A. Mihailidis. Assistiong persons with dementia during handwashing using a partially observable Markov decision process. In *International conference on computer vision systems (ICVS)*, Bielefeld, Germany, 2007.

[42] D. Hsu. *Randomized single-query motion planning in expansive spaces*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, 2000.

[43] V. Huynh and N. Roy. icLQG: combining local and global optimization for control in information space. In *IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, 2009.

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

[45] Tamás Kalmár-Nagy, Raffaello D'Andrea, and Pritam Ganguly. Near-optimal dynamics trajectory generation and control of an omnidirectional vehicle. *Robotics and Autonomous Systems*, 46(1):47–64, January 2004.

[46] L.E. Kavraki, M.N. Kolountzakis, and J.C. Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14:166–171, February 1998.

[47] L.E. Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot motion planning. In *Proc. ACM Symp. Theory of Computing*, pages 353–362, 1995.

[48] L.E. Kavraki, P. Švestka, J.C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[49] James P. Keener. *Principles of Applied Mathematics: Transformation and Approximation, 2nd Edition*. Westview Press, Boulder, CO, 2000.

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

[51] H. Kurniawati, T. Bandyopadhyay, and N.M. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, 33(3):255–272, 2012.

[52] 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*, Zurich, Switzerland, 2008.

[53] Hanna Kurniawati, Yanzhu Du, David Hsu, and Wee Sun Lee. Motion planning under uncertainty for robotic tasks with long time horizons. *International*

Journal of Robotics Research, 30:308–323, 2010.

[54] Hanna Kurniawati, Yanzhu Du, David Hsu, and Wee Sun Lee. Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research*, 30(3):308–323, 2011.

[55] A.M. Ladd and L.E. Kavraki. Measure theoretic analysis of probabilistic path planning. *IEEE Transactions on Robotics and Automation*, 20(2):229–242, April 2004.

[56] 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.

[57] Z. Li and J.F. Canny, editors. *Nonholonomic motion planning*. Kluwer Academic Press, Norwell, MA, USA, 1993.

[58] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 100(2):108–120, 1983.

[59] O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the Sixteen Conference on Artificial Intelligence (AAAI)*, pages 541–548, Orlando, FL, 1999.

[60] Nik Melchior and Reid Simmons. Particle RRT for path planning with uncertainty. In *International Conference on Intelligent Robots and Systems (IROS)*, San Diego, CA, 2007.

[61] Sean Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability: 2nd Edition*. Cambridge University Press, Cambridge, UK, 2009.

[62] P. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *ICRA*, Orlando, FL, 2006.

[63] R. Munoz-Salinas. http://sourceforge.net/projects/aruco/.

[64] Richard M Murray and Sosale Shankara Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38(5):700–716, 1993.

[65] A. Nakhaei and F. Lamiraux. A framework for planning motions in stochastic maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, Pasadena, California, 2008.

[66] James R. Norris. *Markov Chains*. Cambridge Univeristy Press, Cambridge, United Kingdom, 1997.

[67] Video of FIRM plan execution on a physical robot in a changing environment. https://parasol.tamu.edu/groups/amatogroup/movies/small_sizev_22.mp4.

[68] Video of FIRM plan execution under large disturbances (with annotation). https://parasol.tamu.edu/groups/amatogroup/movies/extensiontwo_v6.mp4.

[69] Video of FIRM plan execution under large disturbances (with no annotation). https://parasol.tamu.edu/groups/amatogroup/movies/replanning4timesfaster.mp4.

[70] Video of FIRM plan execution (with annotation). https://parasol.tamu.edu/groups/amatogroup/movies/extensionone_v5.mp4.

[71] Video of FIRM plan execution (with no annotation). https://parasol.tamu.edu/groups/amatogroup/movies/largeenvironmentdoublespeed.mp4.

[72] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *International Journal of Robotics Research*, 29(8):1053–1068, 2010.

[73] Giuseppe Oriolo, Alessandro De Luca, and Marilena Vandittelli. WMR control via dynamic feedback linearization: design, implementation, and experimental validation. *IEEE Transactions on Control Systems Technology*, 10(6):835–851, 2002.

[74] C. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

[75] Sachin Patil, Jur van den Berg, and Ron Alterovitz. Estimating probability of collision for safe planning under Gaussian motion and sensing uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, Minneapolis, MN, 2012.

[76] 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, Acapulco, Mexico, 2003.

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

[78] R. Platt, L. Kaelbling, T. Lozano-Perez, , and R. Tedrake. Efficient planning in non-Gaussian belief spaces and its application to robot grasping. In *Proc. of International Symposium of Robotics Research, (ISRR)*, Flagstaff, AZ, 2011.

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

[80] Josep M. Porta, Nikos Vlassis, Matthijs T. J. Spaan, and Pascal Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine*

Learning Research, 7:2329–2367, November 2006.

[81] 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.

[82] Stéphane Ross, Joelle Pineau, Sébastien Paquet, and Brahim Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[83] Claude Samson and K Ait-Abderrahim. Feedback control of a nonholonomic wheeled cart in cartesian space. In *IEEE International Conference on Robotics and Automation*, pages 1136–1141, Sacramento, CA, 1991.

[84] 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)*, Minneapolis, MN, 2012.

[85] Dan Simon. *Optimal State Estimation: Kalman, H-infinity, and Nonlinear Approaches*. John Wiley and Sons, Inc, Malden, MA, 2006.

[86] 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.

[87] Moshe Sniedovich. Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics*, 35(3):599–620, 2006.

[88] Richard Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112:181–211, 1999.

[89] P. Švestka and M. Overmars. Motion planning for car-like robots using a probabilistic learning approach. *International Journal of Robotics Research*, 16(2):119–143, 1997.

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

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

[92] J. van den Berg and M. Overmars. Kinodynamic motion planning on roadmaps in dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pages 4253–4258, San Diego, CA, 2007.

[93] Jur van den Berg, Pieter Abbeel, and Ken Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. In *Proceedings of Robotics: Science and Systems (RSS)*, Zaragoza, Spain, June 2010.

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

[95] Jur van den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using differential dynamic programming in belief space. In *Proc. of International Symposium of Robotics Research, (ISRR)*, Flagstaff, AZ, 2011.

[96] Jur van den Berg, Sachin Patil, and Ron Alterovitz. Efficient approximate value iteration for continuous Gaussian POMDPs. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, Toronto, Canada, 2012.

[97] Jur Van Den Berg, Sachin Patil, Ron Alterovitz, Pieter Abbeel, and Ken Goldberg. LQG-based planning, sensing, and control of steerable needles. In *Algorithmic Foundations of Robotics IX*, pages 373–389. Springer Berlin Heidelberg, 2011.

[98] D.H. 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 *Proceedings of the the Conference on Decision and Control (CDC)*, pages 929–934, Maui, HI, December 2003.

[99] Michael P. Vitus and Claire J. Tomlin. Closed-loop belief space planning for linear, Gaussian systems. In *ICRA*, pages 2152–2159, Shanghai, China, 2011.

[100] Robert J. Webster, Jin Seob Kim, Noah J. Cowan, Gregory S. Chirikjian, and Allison M. Okamura. Nonholonomic modeling of needle steering. *IJRR*, 25(5-6):509–525, 2006.