Robust Range-Based Localization and Motion Planning Under Uncertainty Using Ultra-Wideband Radio

by

Samuel J. Prentice

B.S., Massachusetts Institute of Technology (2004)

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

September 2007

Author . . . .
Department of Electrical Engineering and Computer Science
August 21, 2007

Certified by . . . .
Nicholas Roy
Assistant Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Professor of Electrical Engineering
Chairman, Department Graduate Program

Robust Range-Based Localization and Motion Planning Under Uncertainty Using Ultra-Wideband Radio

by

Samuel J. Prentice

Submitted to the Department of Electrical Engineering and Computer Science

August 21, 2007

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

The work presented in this thesis addresses two problems: accurately localizing a mobile robot using ultra-wideband (UWB) radio signals in GPS-denied environments; and planning robot trajectories that incorporate belief uncertainty using probabilistic state estimates. Addressing the former, we improve upon traditional approaches to range-based localization by capturing non-linear sensor dynamics using a Monte Carlo method for hidden bias estimation. For the latter, we overcome current limitations of scalable belief space planning by adapting the Probabilistic Roadmap algorithm to enable trajectory search in belief space for minimal uncertainty paths. We contribute a novel solution motivated by linear least-squares estimation and the Riccati equation that provides linear belief updates, allowing us to combine several prediction and measurement steps into one efficient update. This reduces the time required to compute a plan by over two orders of magnitude, leading to a tractable belief space planning method which we call the Belief Roadmap (BRM) algorithm.

Thesis Supervisor: Nicholas Roy
Title: Assistant Professor of Aeronautics and Astronautics

Acknowledgments

I would like to thank my advisor, Nicholas Roy, for his guidance and patience, and for giving me the opportunity to learn through this project. I would also like to express my gratitude to the other lecturers of 6.141: *Robotics: Science and Systems*, Seth Teller, Daniela Rus, John Leonard, and Una-May O'Reilly, for bringing me onboard as a teaching assistant and cultivating my interest in robotics.

I would also like to thank Moe Win for sharing his knowledge of ultra-wideband radio and the Course VI administration, Anne Hunter, Vera Sayzew and Linda Sullivan, who have helped me to navigate through this place over the years.

I owe a great deal of thanks to all of my labmates for not only their help and research advice, but also for making CSAIL an enjoyable place to work.

Finally, I am indebted to my parents (and not just financially) for all of their support and encouragement.

Contents

List of Algorithms

Chapter 1

Introduction

1.1 Project Overview

The work presented in this thesis is two-fold: (1) we have explored a method for range-based localization with ultra-wideband (UWB) radio that simultaneously estimates the pose of a mobile robot and the hidden biases in UWB range measurements; (2) we have developed a novel method for planning under uncertainty that extends the Probabilistic Roadmap algorithm to Kalman belief space and enables efficient graph search for paths that obtain minimal belief uncertainty.

In this section, we introduce both problems in turn and provide motivation for our solution techniques. In Section 1.2 we highlight the key contributions of this thesis work in both problem domains. Section 1.3 describes the infrastructure and tools used in this project. The chapter is concluded in Section 1.4, where we present a roadmap for the document.

1.1.1 Range-Based Localization With UWB Radio

The Global Positioning System (GPS) provides position estimates of a user's location on the Earth to within a few meters, enabling geolocation in areas with a clear line-of-sight (LOS) to GPS satellites. However, in indoor and dense urban environments, GPS becomes unreliable or altogether unavailable. The need for accurate geolocation, however, exists in indoor and dense environments for both civil and military applications.

One approach to this problem involves placing anchor radio beacons around the target environment, enabling localization by computing range measurements from a dynamic agent to known beacons (which can geolocate via GPS). The agent emits a request signal to which the anchor beacons respond upon receipt. Ranges to each anchor beacon can be computed with a time-of-flight (TOF) calculation, measuring the round-trip time between signal emission and response signal detection. The agent can then infer its position based on range measurements from multiple sensors. Two problems limit the effectiveness of this approach when using traditional narrowband radios, namely multipath fading and LOS blockage. When a signal is transmitted, the channel properties of the environment may result in multiple signal path reflections, known as multipath components. Multipath fading occurs when interference between these path reflections results in signal degredation, or even cancellation, which inhibits the receiver's ability to successfully resolve the original signal for time-based ranging. Additionally, obstacles in the environment may occlude the radio transceivers. Such non-LOS (NLOS) scenarios may completely block signal transmission or add substantial material propagation delays, both of which are problematic for ranging applications.

The advent of ultra-wideband (UWB) technology presents a promising alternative that may overcome the limitations of traditional narrowband radio. The spread-spectrum concept employed by UWB utilizes extremely wide transmission bandwidths, which yields favorable properties for robust ranging. The wide transmission bandwidth of UWB, ranging from near-DC to tens of gigahertz, corresponds to very fine, sub-nanosecond delay resolution in the time domain. This property makes UWB pulses largely immune to multipath fading, since individual signal paths can be distinguished in resolvable intervals of time. Further, the low frequency, or base-band, components of the UWB signal enable better penetration of building materials [10], making UWB transmission robust in dense environments.

Although ultra-wideband radio is a promising candidate for reliable RF signaling, the problems of time-based ranging in dense multipath environments still exist. In such environments, the time-of-flight measurement between UWB transceivers is subject to additive delays which cause range estimates to be positively biased. Even though UWB signals are robust to obstacles blocking the direct path, the obstacles still impose a substantial transmission delay as the signal passes through building materials. Also, even though UWB signals are robust to multipath fading, the line-of-sight path can still be lost and the received signal may be from a NLOS path reflection, traveling a greater accumulated distance between the pair of radios. While UWB signal connectivity may be robust to these phenomena, they add an error to the range estimate and thus induce uncertainty in localization.

To solve this problem, probabilistic inference has been used to maintain a joint probabilistic state estimate over the robot pose and hidden range biases [24]. Using a Markov Chain Monte Carlo (MCMC) approach outperforms other state estimation methods because MCMC filters allow for non-linear characteristics of the UWB sensor model to be incorporated. However, such sampling-based filters are limited by the curse of dimensionality; the number of samples required for the filter to accurately estimate a distribution grows exponentially in the state dimension. In the context of hidden bias estimation, as the number of UWB sensors in the environment increases, the number of samples required to estimate the range bias to each sensor grows prohibitively large.

We propose an improvement to this Monte Carlo method in Chapter 3 that uses the Rao-Blackwellization technique, which enables certain forms of joint distributions to be decomposed. We show that, by employing Rao-Blackwellization, the joint distribution over the robot pose and hidden range biases in the range-based localization problem can be decoupled into two smaller estimation problems. Subsequently, sampling is only required for a distribution over robot poses in a space of constant dimension. The distribution over hidden biases can be computed analytically, and the complexity of filtering scales linearly to accommodate additional UWB sensors.

1.1.2 Planning in Belief Space

The second portion of this project addresses motion planning using probabilistic state estimates. Stochastic robot motions, noisy sensor observations, imperfect maps of the environment and dynamic obstacles increase the likelihood of collision or becoming lost. To achieve robust performance, a mobile robot must navigate safely through environments dominated by uncertainty.

Recursive filtering algorithms such as the Kalman filter make it possible to maintain an estimate of the robot's *belief* uncertainty in response to non-deterministic actions and observations, enabling mobile robots to perform tasks such as localization. However, to date, robust and scalable methods that incorporate uncertainty into the planning process are still an area of open research. In this work, we seek to integrate uncertainty predictions into the planning algorithm, thereby enabling the robot to choose actions that result in minimal belief uncertainty. Our explicit goal is to plan in the space of robot beliefs, choosing an optimal sequence of actions that is based on predictions of belief uncertainty during execution.

A traditional solution for motion planning problems is the Markov Decision Process (MDP) [6], which generates a global control policy that prescribes the best action to take in any given situation to advance towards a desired goal state. By iteratively computing the expected rewards of transitioning between states for a given set of actions, the MDP converges to a solution where each possible action transition is assigned an expected cumulative payoff. As the robot navigates towards a goal, the optimal path is easily determined by taking the pre-defined action prescribed for the current belief state of the robot.

While this approach allows for non-deterministic actions, the MDP assumes the robot can observe the full state of the environment after each action. In most real-world robot problems, however, sensor limitations have a substantial effect on the level of certainty in the robot's belief. The partially observable Markov decision process (POMDP) [46] drops the assumption that the state is fully observable and admits the problem of planning in belief space. The POMDP generates a control policy for every possible state and encompasses both probabilistic beliefs and action transitions, but with the cost of greatly increased complexity.

In high-dimensional or continuous state spaces, it becomes intractable to compute a policy for every possible state, including those with very low likelihood. Randomized motion planners have been successful in efficiently solving complicated planning problems by limiting their scope to selected regions in the state space. The Probabilistic Roadmap (PRM) algorithm [28] is one such method that approximates the topology of the configuration space with a graph structure of accessible poses in free space. A pre-processing phase selects poses according to a randomized sampling strategy and builds a visibility graph between them. The PRM is capable of efficiently planning paths to a queried goal location by searching the visibility graph for the shortest path. This limitation of this search, however, is that it does not account for the uncertainty incurred while executing the plan and ignores the potential for error.

In Chapter 4, we apply the intuition underlying the PRM to belief space planning. We use the sampling-based pre-processing phase of the PRM to overcome the complexity of the POMDP model by generating a simplified representation of belief space. We then show that by assuming Gaussian beliefs, it becomes possible to utilize Kalman filter updates in conjunction with graph search to tractably compute plans over a graph through belief space. This approach relies on the ability to simulate Kalman filter control and measurement updates to predict the evolution of uncertainty along belief trajectories. We show that this can be performed by computing a sequence of Kalman filter covariance updates at discrete points along a given trajectory to obtain the posterior covariance.

In Chapter 5, we improve the efficiency of belief trajectory search by exploiting advantageous mathematical properties of the Kalman filter. We present two alternative representations of Kalman filter updates that allow multiple covariance prediction steps to be aggregated into one simple update. We call these techniques "one-step" updates.

In Chapter 6, we apply one-step covariance prediction updates to belief trajectory search to form the Belief Roadmap (BRM) algorithm. The use of one-step updates improves the performance of trajectory search by over two orders of magnitude, making the BRM a tractable method for planning in belief space *and* efficient on-line re-planning during execution. We use the BRM to efficiently solve a very large planning problem across the MIT campus, which we believe is considerably larger than existing results.

1.2 Contributions

The following contributions were made during the course of this research project:

1. We developed a scalable method for range-based localization that is robust to hidden range biases and amenable to dense multipath environments. Our solution is based on the Rao-Blackwellization technique, which, to our knowledge, has not otherwise been applied to this problem domain.

2. We provided a general formulation for tractable belief space planning posed as search over belief trajectories. We generalized the Probabilistic Roadmap (PRM) method to a hybrid PRM, expanding the scope of the PRM from points in a discrete space to probability distributions in a hybrid discrete-continous space. Our general formulation also admits a hybrid filter-search process, which enables optimal graph search over the hybrid distribution space.

3. We derived a Kalman filter-style estimator for efficiently predicting Gaussian belief evolution in a principled manner. This estimator replaces the non-linear filter equations with linear updates in the mean and factors of the covariance, providing a powerful tool for future planning algorithms.

4. We developed two methods for aggregating Kalman filter covariance updates to form a covariance transfer function for use in robot path planning. We applied the work of Redheffer [42] and Kailath [26] to form a Hamiltonian matrix descriptor of Kalman filter updates. We utilized additional results from abstract linear algebra [1, 2] and Vaughan [51] to form a symplectic matrix descriptor, which enables linear updates in factors of the covariance. These techniques have potential for numerous future applications in the robotics domain.

5. We presented the Belief Roadmap (BRM) algorithm which enables efficient and principled planning within the Kalman belief space and represents a significant advance in planning under uncertainty. We used the BRM algorithm to solve the largest belief space planning problem to date of which we are aware. The BRM method has potential for substantial future advances in planning under uncertainty

1.3 Experimental Infrastructure

During the course of the project work, software development and hardware integration was required to perform experimental analysis. Software was developed on a Linux platform within the framework of the Carnegie Mellon Robot Navigation Toolkit (CARMEN) [37], an open-source software package for mobile robot applications. The source code for localization, navigation and ultra-wideband modules was implemented using the C and Java programming languages. Sensor and motion models were developed using a mobile wheelchair robot [13] with onboard computer, shown in Figure 1-1. The robot was outfitted with a TimeDomain UWB radio sensor [12], requiring hardware integration and the development of a socket-level software interface in Java.

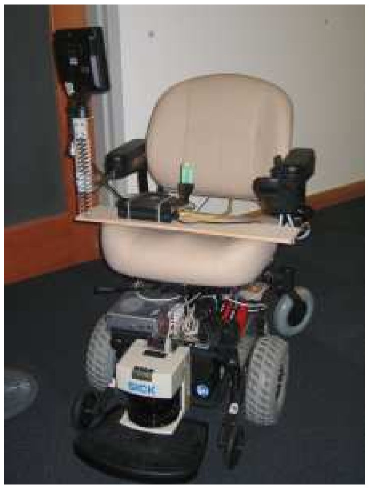


Figure 1-1: Wheelchair Robot [13] with TimeDomain UWB radio [12].

1.4 Roadmap

The layout of this document is as follows: In Chapter 2 we review the background concepts of probabilistic robotics and state estimation techniques. Both the localization and planning problems we consider in this project rely on probabilistic techniques and recursive state estimators.

We explore the range-based localization problem with ultra-wideband ranging in Chapter 3. In this chapter we develop the motion and sensor models used throughout the work. We also present the Rao-Blackwellization technique and apply it to the particle filter to simultaneously maintain distributions over the robot pose and hidden range biases in a scalable manner.

Chapters 4-6 incrementally develop a novel and efficient approach for belief space planning. In Chapter 4, we formulate the problem and develop a tractable solution by generalizing the PRM to belief space and employing the linear-Gaussian assumption for beliefs. This approach generates plans by performing graph search over belief trajectories and relies on a "multi-step" filter process to compute covariance evolution along trajectories. The multi-step Kalman filter update is a limiting factor in the efficiency of trajectory search, which motivates finding an another approach.

We derive two alternatives to the multi-step covariance update, which we call "one-step" updates. The derivation of one-step update methods merits a detailed discussion, which is the topic of Chapter 5.

We combine the general planning problem of Chapter 4 with the efficient one-step update of Chapter 5 to form the Belief Roadmap (BRM) algorithm in Chapter 6. We analyze this algorithm, demonstrate its effectiveness and discuss further applications.

The thesis is concluded in Chapter 7 with a discussion of future work and a summary of our contributions in range-based localization and belief space planning.

Chapter 2

Bayesian Filtering

Contents

2.1 Introduction

Robots cannot perceive the true state of the world; in order to make decisions and execute actions, they must rely on a *belief* that represents a hypothesis of the actual state of the world. Some traditional approaches to robotics employ a "best guess" for this belief [39], but such a limited belief representation often fails to produce reliable performance in the face of real-world uncertainty. A more recent alternative that allows for ambiguity and uncertainty in the belief uses a probability distribution over a whole space of possible guesses. This representation is the basis of *probabilistic robotics*, a modern approach to robotics which enables the uncertainty of robot perceptions and actions to be captured in the mathematically sound framework of probability theory.

This chapter introduces the key concepts of probabilistic robotics and reviews recursive state estimation, or filtering, techniques, which can be used to update probabilistic state estimates in response to uncertain control actions and noisy sensor measurements. Both the localization and path planning problems that we consider in this project rely on probabilistic techniques and recursive filters to achieve reliable performance when faced with uncertainty.

We begin by formulating the problem of mobile robot state estimation as the robot interacts with its environment in Section 2.2, and introduce the most general recursive solution, the Bayes filter, in Section 2.3. The remainder of the chapter is dedicated to specific implementations of the Bayes filter, each maintaining the belief distribution in a different form.

Section 2.4 presents the family of Gaussian filters, which maintain the belief as a multivariate normal distribution. We review the linear state-space representation in Section 2.4.1 and, in Section 2.4.2, derive the Kalman filter, which is one of the most widely used implementations of the Bayes filter. In Section 2.4.3, the Kalman filter is broadened to problems with nonlinear dynamics, leading to the extended Kalman filter (EKF). Section 2.4.4 describes the information filter, which is the complementary form of the Kalman filter using the canonical parameterization of the Gaussian distribution.

Finally, in Section 2.5 we review the particle filter, a non-parametric state estimator based on Monte-Carlo sampling methods. At the end of this chapter, Appendix 2.A presents a reference for relevant matrix mathematics and Appendix 2.B derives the fundamental equations of linear estimation.

2.2 Taxonomy of Robot State Estimation

The *state* of the robot characterizes all aspects of the configuration of the environment at a given time that can impact the future configuration. In common applications, the state is the *pose* of the robot relative to the environment, which in a 2-dimensional environment consists of the x, y and θ components of the robot's location and orientation in the world. Throughout the work, we denote the state of the robot at time t as the vector x_t.

The state of the robot is dynamic; it evolves over time as the robot interacts with its environment. We are concerned with two types of interaction: *control* actions, which describe robot actions that change the state of the environment, such as motion; and *measurement* actions, which describe information gained by the robot through sensor observations that affect perception of the environment. Both controls and measurements are modeled as data vectors, denoted u_t and z_t, respectively, in which each data element describes a feature of the interaction model. For example, control data may consist of the number of robot wheel revolutions collected by an odometer. Similarly, measurement data could be the distance observed by a range sensor between the robot and a known beacon in the environment.

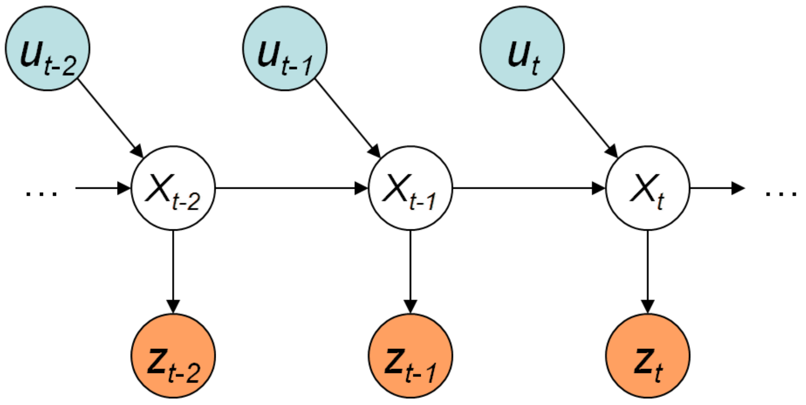


Figure 2-1: Diagram of the generative temporal model that characterizes the evolution of states x_t, controls u_t and measurements z_t.

To incorporate the effects of controls and measurements over time, the evolution of the robot state is modeled as a dynamic Bayes network (DBN). An example of this model is shown in Figure 2-1. For each control u_t and measurement z_t shown in the diagram, we use probabilistic laws to capture the stochastic evolution of the state x_t. In general, we seek to infer the state at time t as a result of all previous states, controls and measurements, which can be characterized as a conditional probability distribution $p(x_t|x_{0:t-1}, u_{1:t}, z_{1:t-1})$. To make inference computationally tractable, we employ the *Markov assumption*, which assumes that the state x_t is a sufficient summary of all previous controls and measurements up to time t. This means that the posterior state after incorporating a control input depends only on the control data u_t and the previous state x_{t-1}, yielding the simplified *state transition probability*

$$p(x_t|x_{0:t-1}, u_{1:t}, z_{1:t-1}) = p(x_t|x_{t-1}, u_t). \tag{2.1}$$

Further, the likelihood of an observation depends only on the current state, which gives rise to the *measurement probability*

$$p(z_t|x_{0:t-1}, u_{1:t}, z_{1:t-1}) = p(z_t|x_t). \tag{2.2}$$

The state cannot be directly observed; however, using the preceding model of the evolution of the state, it is possible to recursively estimate the robot's *belief* of the state.

2.3 The Bayes Filter

In this section, we present the most basic recursive state estimation algorithm, the Bayes filter. The goal is to estimate the belief distribution over the robot's state at time t in response to all control and measurement actions since the initial belief at $t = 0$. This distribution is written as

$$bel_t = p(x_t|u_{1:t}, z_{1:t}). \tag{2.3}$$

The foundation of the Bayes filter is *Bayes rule*, which relates a conditional distribution $p(a|b)$ to the reverse conditional $p(b|a)$ as

$$p(a|b) = \frac{p(b|a)p(a)}{p(b)}. \tag{2.4}$$

Bayes rule enables the posterior $p(a|b)$ to be computed from the reverse conditional $p(b|a)$ and the prior $p(a)$. This rule has powerful implications for inference, allowing the quantity a to be revised given new evidence b without explicit knowledge of the conditional $p(a|b)$. Also, note that the denominator in Equation 2.4 does not depend on a, and can therefore be replaced with a *normalizer* constant $\eta = p(b)^{-1}$.

We apply Bayes rule (Equation 2.4) to the belief posterior bel_t to begin our derivation of the Bayes filter, which gives us

$$p(x_t|u_{1:t}, z_{1:t}) = \eta p(z_t|u_{1:t}, z_{1:t-1}, x_t) \cdot p(x_t|u_{1:t}, z_{1:t-1}), \tag{2.5}$$

where η is a normalization factor. Equation 2.5 decomposes the posterior into two parts, the first of which is the measurement probability $p(z_t|x_t)$ shown in Equation 2.2. The second distribution corresponds to the predicted belief after incorporating the control u_t at time t, but before incorporating the measurement z_t. We indicate this by denoting the predicted belief with a bar as

$$\overline{bel}_t = p(x_t|u_{1:t}, z_{1:t-1}). \tag{2.6}$$

The posterior belief in Equation 2.5 can then be rewritten as follows (by using Equation 2.2 and the notation introduced in Equations 2.3 and 2.6):

$$bel_t = \eta p(z_t|x_t) \cdot \overline{bel}_t, \tag{2.7}$$

which is known as the *measurement update* of the Bayes Filter and is shown in Line 3 of Algorithm 1.

The belief prediction updates the most recent belief bel_{t-1} by accounting for the state transition at time t in response to the motion control u_t. This can be shown mathematically by applying the *law of total probability*, which is given by

$$p(a) = \int_b p(a|b)p(b)db. \tag{2.8}$$

The law of total probability in Equation 2.8 is useful because it allows us to mathematically incorporate the dependence of a on b if we only have $p(a)$. This is performed by "integrating out" or "marginalizing" b from the conditional distribution $p(a|b)$, as shown in Equation 2.8.

In the context of the Bayes Filter, the law of total probability in Equation 2.8 allows us to account for the state transition from the most recent state x_{t-1} by integrating over x_{t-1} in Equation 2.6, as follows:

$$p(x_t|u_{1:t}, z_{1:t-1}) = \int_{x_{t-1}} p(x_t|u_{1:t}, z_{1:t-1}, x_{t-1})p(x_{t-1}|u_{1:t-1}, z_{1:t-1})dx_{t-1}. \tag{2.9}$$

We can see that two distributions in the integral in Equation 2.9 correspond to the state

Algorithm 1 The Bayes Filter algorithm.

```
Require: Previous belief ($bel_{t-1}$), control ($u_t$) and measurement ($z_t$)
Ensure: Posterior belief ($bel_t$)
1: for all $x_t$ do
2:     $\overline{bel}_t = \int p(x_t|u_t, x_{t-1}) bel_{t-1} dx_{t-1}$
3:     $bel_t = \eta p(z_t|x_t) \cdot \overline{bel}_t$
4: end for
5: return $bel_t$
```

transition function (Equation 2.1) and the previous belief bel_{t-1}. Thus, Equation 2.9 simplifies to become

$$\overline{bel}_t = \int_{x_{t-1}} p(x_t|u_t, x_{t-1}) bel_{t-1} dx_{t-1}, \tag{2.10}$$

establishing the *control update* step (also known as the *prediction update* or *time update*) of the Bayes filter.

The *Bayes equation* is written by combining the above results in Equations 2.7 and 2.9, which gives us

$$bel_t = \eta p(z_t|x_t) \int p(x_t|u_t, x_{t-1}) bel_{t-1} dx_{t-1}. \tag{2.11}$$

Equation 2.11 provides a recursive form of computing the posterior belief distribution at each time step given a new control and measurement, establishing the basis of the *Bayes filter* algorithm. The full method is shown in Algorithm 1.

Many implementations of the Bayes filter are possible, each based on a particular representation of the distribution $p(x_t)$ with corresponding state transition and measurement functions. The remainder of this chapter explores derivatives of the Bayes filter technique that are used in the work.

2.4 Gaussian Filters

Gaussian filters are a widely-used class of Bayes filter derivatives that maintain the belief as a multivariate normal distribution, defined as

$$\begin{aligned} p(x) &= \left|2\pi\Sigma_{xx}\right|^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu_x)^T \Sigma_{xx}^{-1}(x-\mu_x)} && (2.12) \\ &\triangleq \mathcal{N}(\mu_x, \Sigma_{xx}). && (2.13) \end{aligned}$$

Note that the probability density of x is completely characterized by the first two moments of the distribution, namely the mean μ_x and covariance Σ_{xx}. If the state x has n dimensions, then the mean μ_x is an n-dimensional vector, and the covariance Σ_{xx} is a symmetric, positive-semidefinite $n \times n$ matrix.

In this section we focus primarily on the most notable member of the Gaussian filter family known as the Kalman filter [27]. The Kalman filter assumes the Gaussian moments parameterization of the belief and restricts the state transition and measurement functions to linear dynamics. The popularity of the Kalman filter stems from its tractability; the belief is completely maintained by two parameters that admit simple Bayesian updates. We lead up to the Kalman filter equations (shown in Section 2.4.2) by first reviewing the linear state space model in Section 2.4.1 and deriving the fundamental equations of linear estimation in Section 2.B.

Committing to a Gaussian posterior and linear dynamics has consequences that render the Kalman filter ineffective in many applications. Many real-world control and measurement models have nonlinear dynamics which cannot be captured in the restrictive linear state space model. In Section 2.4.3, we explore the extended Kalman filter, which uses a linearization technique to adapt the Kalman filter to handle nonlinear state transition and measurement functions. Further, by restricting the belief to a unimodal distribution, the Kalman filter cannot successfully maintain a posterior belief in problems with multiple distinct hypotheses. This problem can be overcome and linearization can be avoided with other models such as the particle filter (Section 2.5), but with the cost of added complexity.

Despite its shortcomings, the Kalman filter has been met with great success. While obviously restrictive, the assumptions of a Gaussian belief distribution and linear system dynamics tend to be reasonable approximations in many real-world problems, making Kalman filtering an efficient and computationally tractable approach to state estimation.

2.4.1 State-Space Representation

The dynamics of a discrete time linear system can be described by the *state-space representation.* The *state transition equation* describes the evolution of the state due to control inputs and process noise, and is given by

$$x_t = A_t x_{t-1} + B_t u_t + w_t, \tag{2.14}$$

where the components at time, t, are

x_t, the state of the system (n_x-dimensional vector)

u_t, the control input (n_u-dimensional vector)

A_t, the state transition matrix ($n_x \times n_x$ matrix)

B_t, the input gain ($n_x \times n_u$ matrix)

w_t, the zero-mean process noise term (n_x-dimensional vector).

The *measurement equation* describes the output of the system,

$$z_t = C_t x_t + q_t, \tag{2.15}$$

with

z_t, the measurement (n_m-dimensional vector)

C_t, the measurement matrix ($n_m \times n_x$ matrix)

q_t, the zero-mean measurement noise term (n_m-dimensional vector).

The stochastic elements of the system, w_t and q_t, are zero-mean noise terms with covariances denoted $\operatorname{cov}(w_t) \triangleq R_t$ and $\operatorname{cov}(q_t) \triangleq Q_t$, respectively.

2.4.2 The Kalman Filter

The Kalman filter assumes the Gaussian moments parameterization (μ_t, Σ_t) and the linear state space representation for the state transition and measurement functions. We begin our discussion of the Kalman filter by deriving the Bayes filter control and measurement updates for linear-Gaussian beliefs. This leads to the Kalman filter algorithm, which is presented at the end of the section.

Derivation of the Kalman Filter

***Control Update*: $(\mu_{t-1}, \Sigma_{t-1}) \Rightarrow (\overline{\mu}_t, \overline{\Sigma}_t)$**

The control update of the Kalman filter can be derived by examining the state transition equation of the state space representation (Equation 2.14), which is restated here for convenience

$$x_t = A_t x_{t-1} + B_t u_t + w_t.$$

This equation models the update of the state x_t from the previous state x_{t-1} as a function of the state transition $A_t x_{t-1}$, controls $B_t u_t$, and process noise w_t. Given that the belief distribution $p(x_t)$ is normally distributed, the posterior belief $\overline{bel}_t$ of the control update is completely characterized by the mean $\overline{\mu}_t$ and covariance $\overline{\Sigma}_t$ resulting from the state transition equation.

The mean $\overline{\mu}_t$ can be computed as the expectation of the state $E[x_t]$ in the following manner,

$$\begin{aligned}\overline{\mu}_t &= E[A_t x_{t-1} + B_t u_t + w_t] \\ &= A_t \mu_{t-1} + B_t u_t, \qquad (2.16)\end{aligned}$$

where we note that by definition, $E[x_{t-1}] = \mu_{t-1}$, and the process noise w_t is zero-mean Gaussian.

The covariance $\overline{\Sigma}_t$ resulting from the control update is computed from the expected deviation of the state x_t about the mean $\overline{\mu}_t$, which is given as

$$\overline{\Sigma}_t = \mathrm{cov}(x_t - \overline{\mu}_t) = E[(x_t - \overline{\mu}_t)(x_t - \overline{\mu}_t)^T]. \qquad (2.17)$$

Using the mean update from Equation 2.16, we find the state prediction error

$$\begin{aligned}x_t - \overline{\mu}_t &= (A_t x_{t-1} + B_t u_t + w_t) - (A_t \mu_{t-1} + B_t u_t) \\ &= A_t(x_{t-1} - \mu_{t-1}) + w_t. \qquad (2.18)\end{aligned}$$

Thus, we can calculate the covariance $\overline{\Sigma}_t$ of the state prediction by plugging Equation 2.18 into Equation 2.17, which yields

$$\begin{aligned}\overline{\Sigma}_t &= E[(A_t(x_{t-1} - \mu_{t-1}) + w_t)(A_t(x_{t-1} - \mu_{t-1}) + w_t)^T] \\ &= E[(A_t(x_{t-1} - \mu_{t-1}) + w_t)((x_{t-1} - \mu_{t-1})^T A_t^T + w_t^T)]. \qquad (2.19)\end{aligned}$$

Since the process noise w_t is, by definition, independent of the state prediction error, the expectation in Equation 2.19 multiplies out into two terms

$$\overline{\Sigma}_t = A_t E[(x_{t-1} - \mu_{t-1})(x_{t-1} - \mu_{t-1})^T] A_t^T + E[w_t w_t^T]. \tag{2.20}$$

Recognizing that $\Sigma_{t-1} \triangleq E[(x_{t-1} - \mu_{t-1})(x_{t-1} - \mu_{t-1})^T]$ and $R_t \triangleq E[w_t w_t^T]$, Equation 2.20 becomes

$$\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t, \tag{2.21}$$

which completes the Gaussian belief update in response to controls.

***Measurement Update*: $(\overline{\mu}_t, \overline{\Sigma}_t) \Rightarrow (\mu_t, \Sigma_t)$**

The Kalman filter measurement update stems from the *fundamental equations of linear estimation*, which compute the Gaussian posterior of state x after incorporating measurement z as

$$E(x|z) = \mu_x + \Sigma_{xz} \Sigma_{zz}^{-1} (z - \mu_z) \tag{2.22}$$

$$\mathrm{cov}(x|z) = \Sigma_{xx|z} = \Sigma_{xx} - \Sigma_{xz} \Sigma_{zz}^{-1} \Sigma_{zx}, \tag{2.23}$$

where we use $\mu_a = E[a]$ and $\Sigma_{aa} = \mathrm{cov}(a)$ to denote the mean and covariance of random variable a, respectively, and Σ_{ab} to denote the cross-covariance of random variables a and b. Note that the formal derivation of Equations 2.22-2.23 can be found in Appendix 2.B.

The measurement update can be obtained by adapting the fundamental equations of linear estimation (2.22-2.23) to follow the control step. This requires us to derive the quantities μ_x, μ_z, Σ_{xz}, and Σ_{zz} for the mean update $E(x|z)$, and additionally Σ_{xx} for the covariance update $\Sigma_{xx|z}$. Since this step follows the control update of the state x_t, the belief is $\mathcal{N}(\overline{\mu}_t, \overline{\Sigma}_t)$ and the following assignments are trivial

$$\mu_x = \overline{\mu}_t \tag{2.24}$$

$$\Sigma_{xx} = \overline{\Sigma}_t. \tag{2.25}$$

The expected measurement μ_z can be computed by taking the expectation of the measurement equation (2.15) and using Equation 2.24

$$\mu_z \quad = E[C_t x_t + q_t] = \quad C_t \overline{\mu}_t. \tag{2.26}$$

Using Equations 2.15 and 2.25, the cross-covariance between the state x_t and the measurement z_t is computed as

$$\begin{aligned}\Sigma_{xz} &= \operatorname{cov}(x_t, C_t x_t + q_t) \\ &= E[x_t (C_t x_t + q_t)^T] \\ &= \overline{\Sigma}_t C_t^T = \Sigma_{zx}^T.\end{aligned} \tag{2.27}$$

The measurement prediction covariance Σ_{zz} is similarly computed using the measurement equation (2.15) and Equation 2.25 to obtain

$$\begin{aligned}\Sigma_{zz} &= \operatorname{cov}(C_t \overline{x}_t + q_t, C_t \overline{x}_t + q_t) \\ &= E[(C_t \overline{x}_t + q_t)(C_t \overline{x}_t + q_t)^T],\end{aligned}$$

which simplifies, by multiplying out terms and applying the definition $E[q_t q_t^T] \triangleq Q_t$, to become

$$\Sigma_{zz} = C_t \overline{\Sigma}_t C_t^T + Q_t. \tag{2.28}$$

Now that the necessary terms have been computed, we can calculate the mean resulting from the measurement update Equation 2.22, which is restated for convenience

$$E(x|z) = \mu_x + \Sigma_{xz} \Sigma_{zz}^{-1} (z - \mu_z).$$

Plugging in for the terms (Equations 2.24, 2.26-2.28) yields

$$\begin{aligned}\mu_t &= \overline{\mu}_t + \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1} (z - C_t \overline{\mu}_t) \\ &= \overline{\mu}_t + K_t (z - C_t \overline{\mu}_t),\end{aligned} \tag{2.29}$$

where K_t is referred to as the *Kalman Gain* and is given by

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}. \tag{2.30}$$

Algorithm 2 The Kalman Filter algorithm.

```
Require: Prior belief $(\mu_{t-1}, \Sigma_{t-1})$, control $(u_t)$ and measurement $(z_t)$
Ensure: Posterior belief $(\mu_t, \Sigma_t)$
1: $\overline{\mu}_t = A_t \mu_{t-1} + B_t u_t$
2: $\overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$
3: $K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1}$
4: $\mu_t = \overline{\mu}_t + K_t(z_t - C_t \overline{\mu}_t)$
5: $\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$
6: return $\mu_t$, $\Sigma_t$
```

Similarly, we can form the measurement update of the covariance by plugging into Equation 2.23, which is restated for convenience

$$\Sigma_{xx|z} = \Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx}.$$

Plugging in for the terms (from Equations 2.25, 2.27, 2.28) yields

$$\begin{aligned}\Sigma_t &= \overline{\Sigma}_t - \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + Q_t)^{-1} C_t \overline{\Sigma}_t \\ &= \overline{\Sigma}_t - K_t C_t \overline{\Sigma}_t \\ &= (I - K_t C_t)\overline{\Sigma}_t. \qquad (2.31)\end{aligned}$$

The Kalman Filter Algorithm

The control and measurements updates derived above for linear-Gaussian beliefs give rise to the Kalman filter algorithm, shown in Algorithm 2.

The control step is shown in lines 1-2, which update the prior belief $bel_{t-1} = (\mu_{t-1}, \Sigma_{t-1})$ to the control posterior belief $\overline{bel}_t = (\overline{\mu}_t, \overline{\Sigma}_t)$. This step incorporates the controls u_t and process noise w_t into the belief according to the state transition equation (2.14). The posterior mean $\overline{\mu}_t$ is computed in line 1 using the mean update derived in Equation 2.16. The prior mean μ_{t-1} is updated by the linear state transition function A_t and the control action u_t, which is transformed into the state space using the mapping B_t. In line 2, the control update of the covariance is computed using Equation 2.21. The posterior covariance $\overline{\Sigma}_t$ is the result of a quadratic transformation of the prior covariance by the state transition matrix A_t and additive, Gaussian process noise R_t. Note that the posterior belief resulting from the control step $\overline{bel}_t$ can be considered the prior belief for the measurement step.

The measurement update is implemented in lines 3-5. The Kalman gain K_t is computed

Algorithm 3 The Extended Kalman Filter algorithm.

```
Require: Previous belief ($\mu_{t-1}, \Sigma_{t-1}$), control ($u_t$) and measurement ($z_t$)
Ensure: Posterior belief ($\mu_t, \Sigma_t$)
1: $\overline{\mu}_t = g(\mu_{t-1}, u_t)$
2: $\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
3: $K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$
4: $\mu_t = \overline{\mu}_t + K_t(z_t - H_t \overline{\mu}_t)$
5: $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
6: return $\mu_t$, $\Sigma_t$
```

in line 3 using Equation 2.30. The Kalman gain determines the extent to which a new measurement is incorporated into both the mean and covariance of the belief. In line 4, the measurement update of the mean is computed using Equation 2.29 from our derivation. The prior mean $\overline{\mu}_t$ is updated according to the *innovation*, which is defined as the difference between the actual observation z_t and the predicted observation $C_t\overline{\mu}_t$. Note that the change in the mean is determined by the Kalman gain multiplied by the innovation, which has the effect of incorporating the new observation *to the extent* that it is innovative. Similarly, in line 5 the prior covariance $\overline{\Sigma}_t$ is updated to account for the information gain of the observation z_t according to Equation 2.31. In this step, the uncertainty of the belief is reduced by a fraction determined by the Kalman gain K_t, resulting in the posterior covariance Σ_t.

2.4.3 The Extended Kalman Filter

One of the shortcomings of the standard Kalman filter is that it is unable to capture non-linear dynamics in the state transition and measurement functions. The extended Kalman filter (EKF) broadens the scope of the Kalman filter by linearizing the non-linear functions about the mean at each update step. In this case, the linear state space representation is replaced to allow arbitrary state transition and measurement functions as follows

$$x_t = g(x_{t-1}, u_t, w_t) \tag{2.32}$$

$$z_t = h(x_t, q_t). \tag{2.33}$$

As before, the filtering algorithm is decomposed into separate control and measurement steps which update the mean and covariance of the belief. In the control step, the mean is simply propagated through the state transition function g, while the covariance requires linearization of g about the mean

$$\begin{aligned} \overline{\mu}_t &= g(\mu_{t-1}, u_t) & (2.34) \\ \overline{\Sigma}_t &= G_t \Sigma_{t-1} G_t^T + V_t W_t V_t, & (2.35) \end{aligned}$$

where G_t is the Jacobian of g with respect to x_t, V_t is the Jacobian of g with respect to w_t and W_t is the process noise covariance.

Similarly, the measurement update becomes

$$\begin{aligned} \mu_t &= \overline{\mu}_t + K_t(H_t\overline{\mu}_t - z_t) & (2.36) \\ \Sigma_t &= (I - K_t H_t)\overline{\Sigma}_t, & (2.37) \end{aligned}$$

where H_t is the Jacobian of h with respect to x_t, $K_t = \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1}$ is the Kalman gain and Q_t is the measurement noise covariance.

These changes give rise to the resulting extended Kalman filter algorithm shown in Algorithm 3.

2.4.4 The Information Filter

Another important member of the Gaussian filter family is the *information filter*, which is the complementary form of the Kalman filter. The information filter is based on the canonical parameterization of the Gaussian distribution, which is given by the *information matrix* Ω and the *information vector* ξ. The canonical parameterization (ξ, Ω) is related to the moments parameterization (μ, Σ) used in the Kalman filter as follows,

$$\begin{aligned} \Omega &= \Sigma^{-1} \\ \xi &= \Sigma^{-1}\mu. \end{aligned}$$

The key point is to note that the information matrix is the inverse of the covariance matrix, which represents the natural inverse relationship between information and uncertainty. For the purposes of this project, we are primarily concerned with the *extended information filter* (EIF) and the control and measurement updates of the second order moment. The EIF follows from the information filter by linearization in the same way that the EKF

Algorithm 4 The Extended Information Filter: Information Updates

```
Require: Previous information ($\Omega_{t-1}$), state transition Jacobian ($G_t$), process noise ($R_t$),
    measurement Jacobian ($H_t$) and measurement noise ($Q_t$)
Ensure: Posterior information ($\Omega_t$)
1: $\overline{\Omega}_t = \left(G_t \Sigma_{t-1} G_t^T + R_t\right)^{-1}$
2: $\Omega_t = \overline{\Omega}_t + H_t^T Q_t^{-1} H_t^T$
3: return $\Omega_t$
```

follows from the Kalman filter.

The information matrix updates in the EIF can be derived from the EKF in a very straightforward manner by inverting the EKF control and measurement covariance updates. The EIF control update can be found by taking the inverse of Line 2 in Algorithm 3 as follows:

$$\overline{\Omega}_t = \overline{\Sigma}_t^{-1} \quad = \quad \left(G_t \Sigma_{t-1} G_t^T + R_t\right)^{-1}. \tag{2.38}$$

Similarly, we can derive the EIF measurement update for the information matrix using Lines 3 and 5 of Algorithm 3, and taking the inverse to give

$$\Omega_t = \Sigma^{-1} = \left(\overline{\Sigma}_t - \overline{\Sigma}_t H^T \left(H \overline{\Sigma}_t H^T + Q_t\right)^{-1} H_t \overline{\Sigma}_t\right)^{-1}. \tag{2.39}$$

To obtain the EIF measurement update, we apply the *matrix inversion lemma* to Equation 2.39. The matrix inversion lemma is a useful matrix relation which is given as

$$\left(X^{-1} + Y^T Z^{-1} Y\right)^{-1} = X - XY^T \left(YXY^T + Z\right)^{-1} YX. \tag{2.40}$$

Note that a derivation of the matrix inversion lemma is given in Appendix 2.A.2.

By applying the matrix inversion lemma (Equation 2.40) to Equation 2.39, we obtain the following form of the EIF measurement update

$$\Omega_t = \overline{\Omega}_t + H_t^T Q_t^{-1} H_t^T. \tag{2.41}$$

Note that this is a particularly attractive form because measurement updates are additive. Thus, multiple measurements can be incorporated in series by simply adding the associated measurement information.

Table 2.1: Duality of Kalman Filter and Information Filter

Inversion Lemma	$(A + BC^{-1})^{-1}$	$\leftrightarrow$	$CB^{-1} - CB^{-1}(A^{-1} + CB^{-1})^{-1}CB^{-1}$
Control Update	Adding Uncertainty (KF) $(R + \Omega^{-1})^{-1}$	$\leftrightarrow$	Subtracting Information (IF) $\Omega^{-1} - \Omega^{-1}(R^{-1} + \Omega^{-1})^{-1}\Omega^{-1}$
Measurement Update	Adding Information (IF) $(M + \Sigma^{-1})^{-1}$	$\leftrightarrow$	Subtracting Uncertainty (KF) $\Sigma^{-1} - \Sigma^{-1}(M^{-1} + \Sigma^{-1})^{-1}\Sigma^{-1}$

Later in the work (Chapter 5), we will see that the duality between the information and Kalman filters can be mathematically advantageous. We conclude our discussion of the information filter by presenting Table 2.1 to motivate the intuition that underlies this relationship. We restate the matrix inversion lemma and show that in the linear-Gaussian filtering problem, it describes the respective gain or loss of information or uncertainty in the filter updates. The key idea is that adding uncertainty in the Kalman filter corresponds to subtracting information in the information filter. Similarly, adding information in the information filter corresponds to subtracting uncertainty in the Kalman filter. Either filter can be used to compute a given update, but in certain cases it may be advantageous to use a specific form depending on the size of the matrices (for computing matrix inverses) or the algebra involved.

2.5 The Particle Filter

In some applications, the linear-Gaussian system underlying the Kalman filter may be too restrictive to accurately capture the belief distribution and system dynamics. In such cases, the probability density over state hypotheses may not resemble a unimodal Gaussian belief. Further, the linearization technique utilized in the EKF may yield a poor approximation of the underlying dynamics, leading to an accumulation of error that causes the belief to diverge from the true state.

A Monte-Carlo technique known as the particle filter is a powerful alternative to Gaussian filtering, enabling state estimation of arbitrary distributions with non-linear dynamics. This non-parametric implementation of the Bayes filter maintains the posterior belief distribution bel_t as a set of particles $\mathcal{X}_t$, each of which represents a state hypothesis. By sampling and assigning weight to these particles in a principled manner, the particle filter can capture the contour of any probability density with any transition dynamics.

Algorithm 5 The Particle Filter algorithm.

```
Require: Previous belief ($\mathcal{X}_{t-1}$), control ($u_t$) and measurement ($z_t$)
Ensure: Posterior belief ($\mathcal{X}_t$)
 1: $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
 2: for $i = 1$ to $N$ do
 3:     sample $x_t^{[i]} \sim p(x_t | u_t, x_{t-1}^{[i]})$
 4:     $w_t^{[i]} = p(z_t | x_t^{[i]})$
 5:     $\overline{\mathcal{X}}_t^{[i]} = \langle x_t^{[i]}, w_t^{[i]} \rangle$
 6: end for
 7: for $n = 1$ to $N$ do
 8:     draw $i$ with probability $\propto w_t^{[i]}$
 9:     add $x_t^{[i]}$ as $\mathcal{X}_t^{[n]}$ to $\mathcal{X}_t = \langle \mathcal{X}_t, \mathcal{X}_t^{[n]} \rangle$
10: end for
11: return $\mathcal{X}_t$
```

We will denote particle i in the set of N samples at time t as $\mathcal{X}_t^{[i]}$. Each particle consists of a sampled state hypothesis $x_t^{[i]}$ and an importance weight $w_t^{[i]}$, such that $\mathcal{X}_t^{[i]} = \langle x_t^{[i]}, w_t^{[i]} \rangle$. The set of all N particles at time t is denoted

$$\mathcal{X}_t = \{\langle x_t^{[1]}, w_t^{[1]} \rangle, \langle x_t^{[2]}, w_t^{[2]} \rangle, \ldots \langle x_t^{[N]}, w_t^{[N]} \rangle\}.$$

Algorithm 5 shows the key steps in the particle filtering method. This variant of the Bayes filter recursively constructs the set of particles $\mathcal{X}_t$ representing bel_t from the previous set of particles $\mathcal{X}_{t-1}$, updating the belief posterior in response to controls u_t and observations z_t. Each filter iteration consists of two key phases. In the first, the algorithm constructs a temporary set of particles $\overline{\mathcal{X}}_t$ from the previous particle set $\mathcal{X}_{t-1}$ by applying the state transition and observation functions to each particle (Lines 1-6). The likelihood of the state hypothesis $x_t^{[i]}$ represented by each particle is stored as the particle weight $w_t^{[i]}$. A second phase of the algorithm involves resampling from the temporary particle set $\overline{\mathcal{X}}_t$, drawing the final set of N particles $\mathcal{X}_t$, where the probability of particle i being selected is proportional to its weight $w_t^{[i]}$ (Lines 7-10).

Resampling is a very important part of the particle filter algorithm, for it removes hypotheses that have very low weight and focuses more particles to regions with higher probability. Before the resampling step, the particles are distributed according to $\overline{bel}_t$. After the resampling step, the particles are distributed according to the posterior belief $bel_t = \eta p(z_t | x_t^{[n]}) \overline{bel}_t$ (as shown in line 9). At this point, the particle set may contain duplicates, since particles are drawn with replacement. These duplicate "copies" of hypotheses allow for stochastic effects on the next filter iteration to be accounted for at a *higher resolution* in regions of the belief space with highest probability. The resampling technique makes the particle filter very powerful, for it enables the filter to dynamically reallocate a limited number of samples to the most pertinent contours of any belief distribution with any transition functions. For further discussion of the mathematical basis and asymptotic correctness of the resampling step, we refer the reader to [50].

We should also note the key limitation of the particle filter: the curse of dimensionality. Although the particle filter abounds with favorable properties, it requires a sufficient number of samples along *each* dimension of the state space to capture a reasonable set of possible beliefs. This means that the number of samples required for belief estimation grows exponentially in the number of states to estimate. For example, if m samples were needed to accurately represent the hypotheses over one state, then $\mathcal{O}(s^m)$ samples would be required to capture the entire distribution over s states. It is for this reason that particle filtering is most effective over a small number of states, as this method suffers from issues of scalability in higher-dimensional spaces.

2.A Matrix Mathematics Reference

2.A.1 Matrix Block Inversion

The inverse of a block matrix with blocks A, B, C and D,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}$$

is defined by the following relations:

$$\begin{aligned}
M_{11} &= A^{-1} + A^{-1}BM_{22}CA^{-1} = \left(A - BD^{-1}\right)^{-1}C && (2.42)\\
&= \left(A - BD^{-1}C\right)^{-1} && (2.43)\\
M_{12} &= -A^{-1}BM_{22} = -M_{11}BD^{-1} && (2.44)\\
M_{21} &= -M_{22}CA^{-1} = -D^{-1}CM_{11} && (2.45)\\
M_{22} &= D^{-1} + D^{-1}CM_{11}BD^{-1} = \left(D - CA^{-1}\right)^{-1}B && (2.46)\\
&= \left(D - CA^{-1}B\right)^{-1}. && (2.47)
\end{aligned}$$

2.A.2 Matrix Inversion Lemma

The *matrix inversion lemma* is a well-known identity defined as

$$\left(A - BD^{-1}C\right)^{-1} = A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1}, \tag{2.48}$$

which can be derived from the matrix block inversion formulas by plugging Equation 2.47 into Equation 2.42.

We will make heavy use of a special form of Equation 2.48 in which $X = A^{-1}$, $Y = B^T = C$ and $Z = -D^{-1}$. With these substitutions, the resulting equation becomes

$$\left(X^{-1} + Y^TZ^{-1}Y\right)^{-1} = X - XY^T\left(YXY^T + Z\right)^{-1}YX. \tag{2.49}$$

2.B Linear Least Squares Estimation

The *fundamental equations of linear estimation* [7] for computing the Gaussian posterior of state x after incorporating measurement z are as follows

$$\begin{aligned} E(x|z) &= \mu_x + \Sigma_{xz}\Sigma_{zz}^{-1}(z-\mu_z) && (2.50) \\ \text{cov}(x|z) &= \Sigma_{xx|z} = \Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx}. && (2.51) \end{aligned}$$

The derivation of these equations motivates the Kalman filter, and begins by writing a stacked vector for two jointly Gaussian random variables

$$y = \begin{bmatrix} x \\ z \end{bmatrix},$$

where

$$p(x,z) = p(y) = \mathcal{N}(\mu_y, \Sigma_{yy}).$$

The conditional pdf of x given z becomes

$$p(x|z) = \frac{p(x,z)}{p(z)} = \frac{\left|2\pi\Sigma_{yy}\right|^{-1/2} e^{-1/2(y-\mu_y)^T\Sigma_{yy}^{-1}(y-\mu_y)}}{\left|2\pi\Sigma_{zz}\right|^{-1/2} e^{-1/2(z-\mu_z)^T\Sigma_{zz}^{-1}(z-\mu_z)}}. \qquad (2.52)$$

The covariance of the conditional distribution can be found by examining the exponent in Equation 2.52, which results from the subtraction of exponents in the numerator and denominator of the fraction. Disregarding the resulting $-\frac{1}{2}$ scaling, the exponent becomes

$$\begin{aligned} \lambda &\propto (y-\mu_y)^T\Sigma_{yy}^{-1}(y-\mu_y) - (z-\mu_z)^T\Sigma_{zz}^{-1}(z-\mu_z) \\ &\propto \begin{bmatrix} x-\mu_x \\ z-\mu_z \end{bmatrix}^T \begin{bmatrix} \Sigma_{xx} & \Sigma_{xz} \\ \Sigma_{zx} & \Sigma_{zz} \end{bmatrix}^{-1} \begin{bmatrix} x-\mu_x \\ z-\mu_z \end{bmatrix} - (z-\mu_z)^T\Sigma_{zz}^{-1}(z-\mu_z). \end{aligned}$$

We can rewrite this as

$$\lambda \propto \begin{bmatrix} x-\mu_x \\ z-\mu_z \end{bmatrix}^T \begin{bmatrix} T_{xx} & T_{xz} \\ T_{zx} & T_{zz} \end{bmatrix} \begin{bmatrix} x-\mu_x \\ z-\mu_z \end{bmatrix} - (z-\mu_z)^T\Sigma_{zz}^{-1}(z-\mu_z), \qquad (2.53)$$

where the elements of T are computed using the matrix block inversion formula given in Equation 2.A.1 to obtain

$$\begin{bmatrix} T_{xx} & T_{xz} \\ T_{zx} & T_{zz} \end{bmatrix} = \begin{bmatrix} (\Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx})^{-1} & -(\Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx})^{-1}\Sigma_{xz}\Sigma_{zz}^{-1} \\ -\Sigma_{zz}^{-1}\Sigma_{zx}(\Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx})^{-1} & (\Sigma_{zz} - \Sigma_{zx}\Sigma_{xx}^{-1}\Sigma_{xz})^{-1} \end{bmatrix}.$$

The elements of T correspond to

$$\begin{bmatrix} T_{xx} & T_{xz} \\ T_{zx} & T_{zz} \end{bmatrix} = \begin{bmatrix} \Sigma_{xx|z}^{-1} & \Sigma_{xz|z}^{-1} \\ \Sigma_{zx|z}^{-1} & \Sigma_{zz|z}^{-1} \end{bmatrix}, \tag{2.54}$$

which yields our desired result for the conditional covariance of x given z (shown in Equation 2.51):

$$cov(x|z) = \Sigma_{xx|z} = T_{xx}^{-1} = \Sigma_{xx} - \Sigma_{xz}\Sigma_{zz}^{-1}\Sigma_{zx}. \tag{2.55}$$

The expected value of the state can be found by multiplying out the matrices of the exponent in Equation 2.53. For convenience, denote $\epsilon_x = x - \mu_x$ and $\epsilon_z = z - \mu_z$, yielding

$$\begin{aligned} \lambda &= \epsilon_x^T T_{xx}\epsilon_x + \epsilon_x^T T_{xz}\epsilon_z + \epsilon_z^T T_{zx}\epsilon_x + \epsilon_z^T T_{zz}\epsilon_z - \epsilon_z^T\Sigma_{zz}^{-1}\epsilon_z \\ &= (\epsilon_x + T_{xx}^{-1}T_{xz}\epsilon_z)^T T_{xx}(\epsilon_x + T_{xx}^{-1}T_{xz}\epsilon_z) + \epsilon_z^T(T_{zz} - T_{zx}T_{xx}^{-1}T_{xz})\epsilon_z - \epsilon_z^T\Sigma_{zz}^{-1}\epsilon_z. \end{aligned}$$

These last two terms cancel out, since, by the matrix block inversion equation (2.A.1)

$$\Sigma_{zz}^{-1} = T_{zz} - T_{zx}T_{xx}^{-1}T_{xz}$$

leaving

$$\lambda = \underbrace{(\epsilon_x + T_{xx}^{-1}T_{xz}\epsilon_z)^T}_{(a-\mu_a)^T}\underbrace{T_{xx}}_{\Sigma_{aa}}\underbrace{(\epsilon_x + T_{xx}^{-1}T_{xz}\epsilon_z)}_{(a-\mu_a)}.$$

It can be seen that this quadratic form matches the exponent of a Gaussian, and indeed respresents the resulting conditional normal distribution of x given z. Thus

$$\begin{aligned} x - E(x|z) &= \epsilon_x + T_{xx}^{-1}T_{xz}\epsilon_z \\ &= x - \mu_x - \Sigma_{xz}\Sigma_{zz}^{-1}(z - \mu_z) \end{aligned}$$

and the desired result in Equation 2.50 for the expected state estimate is given

$$E(x|z) = \mu_x + \Sigma_{xz}\Sigma_{zz}^{-1}(z - \mu_z).$$

Chapter 3

Rao-Blackwellized Monte Carlo Localization With UWB Ranging

Contents

3.1 Introduction

The Global Positioning System (GPS) enables an agent's location on the Earth to be estimated within a few-meter resolution. However, in regions without a clear line-of-sight

(LOS) to GPS satellites, such as indoor or urban environments, geolocation estimates from GPS become unreliable or even unavailable. One approach to the problem of geolocation in harsh environments involves outfitting the mobile agent with an RF ranging sensor and placing anchor beacons around the environment (which can geolocate via GPS). The dynamic agent can obtain range measurements to the known beacons by emitting a request signal, to which the anchor beacons respond upon receipt. Ranges are computed with a time-of-flight (TOF) calculation, measuring the round-trip time between signal emission and response signal detection. The agent can then localize by inferring its position based on odometry motion data and range measurements from multiple sensors.

While this approach is common in range-based localization problems, two key limitations are encountered when using traditional narrowband radios: *multipath fading* and *line-of-sight* (LOS) blockage. Multipath fading occurs when interference between multiple signal path reflections degrades the signal. This inhibits successful ranging as the radio receiver cannot accurately resolve the shortest path signal, rendering narrowband carriers largely ineffective. A further limitation is encountered in non-LOS (NLOS) scenarios, in which obstacles that are impenetrable by narrowband signals completely block signal reception.

Ultra-wideband radio, on the other hand, is known to overcome these limitations. The fine delay resolution of UWB pulses make it largely immune to multipath fading, and its extremely large signal bandwidth enables UWB to penetrate obstacle materials more successfully than narrowband alternatives. While this makes UWB amenable to localization, there are additional complications that must be addressed. The key issue is to account for positive biases that occur in UWB range data, which are discussed in detail in Section 3.2. It is possible to maintain a state estimate of the hidden range bias to each anchor beacon; however, the bias transitions during robot motion are described by a highly non-linear process, ruling out simple filtering techniques. Thus, the key challenge in this problem is to choose an appropriate bias state distribution and an estimation method that is capable of capturing non-linear bias transitions.

The particle filter has become a standard approach to the localization problem [17], as it allows for various distinct robot pose hypotheses and admits non-linear transition and observation functions. To solve the problem of localization amidst hidden range biasing, we follow the approach of Jourdan *et al.* [24], where a solution based on the particle filter is presented. They use this sampling-based Monte-Carlo method to estimate the joint distribution over robot poses and range biases, taking into account non-linear bias transitions. While their approach outperforms methods that do not incorporate non-linear transitions, they are limited by the particle filter's vulnerability to the curse of dimensionality. The number of samples required by the particle filter grows exponentially in the state dimension, meaning that as additional UWB sensors are added to the environment, this approach for hidden bias estimation suffers from issues of scalability.

In this project we extend their work, showing that the underlying structure of this estimation problem admits a technique known as Rao-Blackwellization, which can be used to increase the efficiency of the particle filter. This technique decomposes the joint estimation problem over poses and biases, allowing bias estimation to be marginalized out of the particle filter and computed analytically. The resulting Rao-Blackwellized particle filter (RBPF) only requires sampling from the lower-dimensional robot pose distribution and scales linearly to accommodate additional UWB sensors.

We begin in Section 3.2 by providing a brief background of ultra-wideband technology and explaining the properties that make UWB amenable to accurate localization. Section 3.2.1 presents the channel model and two-way UWB ranging protocol, and Section 3.2.2 develops the sensor model used in the work. The probabilistic robot motion model that is used during localization is shown in Section 3.3. In Section 3.4, we formally describe the particle filter-based solution to the localization problem and show that the Rao-Blackwellization technique can be applied to improve the efficiency and scalability of this approach. We present experimental results in Section 3.5 and conclude the chapter in Section 3.6.

This chapter is appended with additional material. Section 3.7 includes a discussion of related work in range-based localization and NLOS mitigation and Appendix 3.A provides additional details of the two-way UWB ranging protocol.

3.2 Ultra-Wideband Radio

3.2.1 Ultra-Wideband Background

Ultra-wideband (UWB) technology offers a promising new alternative to overcome the limitations of previous narrowband ranging systems. Recently approved by the US FCC in 2002, UWB radio is defined as having a bandwidth greater than 500 MHz or a fractional bandwidth $B_f > 20\%$ ($B_f = \frac{\text{bandwidth}}{\text{center frequency}}$), compared with narrowband counterparts having $B_f < 1\%$ [11]. The extremely wide transmission bandwidth of UWB results from generating short-duration baseband pulses using a technique known as *impulse radio* (IR). Transmission of sub-nanosecond pulses results in *spread-spectrum* signals, which distribute the signal energy across frequencies ranging from near-DC to tens of gigahertz [53].

The fine time resolution, and corresponding large bandwidth, of impulse radio makes UWB a strong candidate for ranging applications. UWB pulses are largely immune to multipath fading due to their fine delay resolution [10]. Each signal path from an emitted sub-nanosecond pulse can be distinguished upon arriving at the receiver, without substantial losses from interference or degradation in transmission. Additionally, since UWB signals span a very wide frequency range that includes low frequency components, they undergo low material penetration losses. This property makes UWB amenable to ranging applications in dense indoor or urban environments where the signal must penetrate a variety of building materials. Further, UWB pulses have a very low duty cycle ($\frac{t_{on}}{t_{on}+t_{off}} < 0.5\%$), resulting in low power consumption. This is a desirable property for the geolocation problem, as it makes UWB sensors extremely mobile and easily placed with low maintenance requirements.

These desirable properties motivate our discussion of ultra-wideband radio for range-based localization. We begin by presenting a general UWB channel model in this section. We then show how UWB radio can be used for ranging in Section 3.2.1, presenting a two-way ranging protocol that can be used to obtain very accurate range estimates in LOS conditions. In Section 3.2.2, we develop a sensor model based on experiments in LOS and NLOS scenarios.

UWB Channel

Developing an accurate characterization of the UWB channel is still an active area of research due to the nascent stage of the technology and the fact that traditional narrowband models do not accurately model the behavior of impulse-based signals. Here we present a general channel model that is commonly used to provide intuition for the relative advantages of UWB radio.

The received signal $r(t)$ can be related to the original signal $s(t)$ using a *tapped delay channel model*, which models signal transmission as a process with fading, time-shifting and noise. The tapped delay model for the UWB channel can be written as follows [5]:

$$r(t) = \sum_{l=1}^{L} \alpha_l s(t - \tau_l) + n(t), \tag{3.1}$$

where L is the number of multipath components, α_l is the fading coefficient of the lth path, τ_l is the time delay of the lth path, and $n(t)$ is zero mean Gaussian noise. The time delay τ_l can be modeled as

$$\tau_l = \frac{\sqrt{(x - x_b)^2 + (y - y_b)^2} + d_l}{c}, \tag{3.2}$$

where d_l is the added length along the lth path induced by NLOS propagation, (x, y) is the agent sensor location, (x_b, y_b) is the beacon sensor location, and $c = 3 \times 10^8$ m/s is the speed of light.

Equation 3.1 can be used to provide insight into the advantage of UWB radio over narrowband alternatives. When transmitting a narrowband signal s_t in a multipath environment, the time delays τ_l of each signal path tend to be much smaller than the wavelength of the signal. The L signal paths overlap in the time domain, meaning that in the aggregate resulting signal r_t the contribution of each path is indecipherable. Further, in the frequency domain, the overlap of signal paths causes phase interference and results in cancellation and signal degradation. Thus, the received signal r_t provides inadequate information for robust range estimation.

On the other hand, the fine temporal resolution of UWB enables each of the L multipath arrivals to be distinctly separated by the receiver, as they arrive in resolvable time intervals. The added path length of each multipath component d_l results in time delays t_l that are larger than the pulse width. This provides multipath-immunity for UWB ranging, making UWB radio a strong candidate for ranging applications in dense multipath environments.

UWB Time-Based Ranging

Ultra-wideband radio can be used to generate range estimates by measuring the time taken for a signal to travel between a pair of UWB transceivers. If this time-of-flight t_{tof} can be accurately measured, the range r (in meters) between radios can then be computed as

$$r = t_{tof} \cdot c,$$

where, again, $c = 3.0 \times 10^8$ m/s is the speed of light.

Since the local clocks on both radios are not synchronized, it is necessary to use a two-way technique known as *half-duplex ranging.* This method measures the round-trip time taken for a request-response signal transaction between the agent (requester) and the anchor beacon (responder). To compute the range to a reference sensor, a UWB requester can measure the time between transmitting a UWB pulse and receiving a response pulse. The total time t_{total} recorded by the requester is equal to twice the time-of-flight, so the range can be calculated using $t_{tof} = \frac{t_{total}}{2}$.

Note that this is a simplified description of the actual two-way ranging protocol used, which accounts for time delays internal to the sensor. A supplementary discussion of the details of half-duplex ranging is presented in Appendix 3.A.

Challenges of UWB Ranging

While this time-based approach enables accurate ranging in open LOS conditions, range measurements are often positively biased in NLOS scenarios. Overestimated ranges are due to channel delays resulting from propagation through non-free space or the receipt of a reflected multipath component. NLOS can impose a substantial propagation delay as the signal passes through building materials. Also, when a LOS path does not exist, the received signal may be a reflected multipath component, traveling a greater accumulated distance between the pair of radios. In either case, the first path pulse extracted by the leading edge detection algorithm does not correspond to the direct path. These phenomena add a positive bias to the range estimate, and thus induce uncertainty in localization.

In the next section, we develop a probabilistic ranging model to capture knowledge of this uncertainty.

3.2.2 Ultra-Wideband Measurement Model

The UWB sensor model used in the project work is based on the the behavior of range measurement biases in two scenarios: line-of-sight (LOS), in which there exists a clear path between range sensors in open space; and non-LOS (NLOS), where obstacles occlude transmission along the direct path. The general model used for range estimates can be written as

$$r_t = d_t + b_t + n_t, \tag{3.3}$$

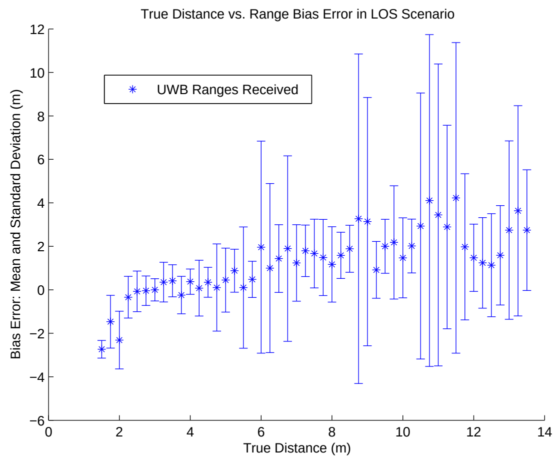


Figure 3-1: The Ultra-Wideband Ranging Model in LOS. The x-axis is the true range between sensors and the y-axis is the measured systematic error (unbiased measurements are at 0), and the error bars give the standard deviation of the random noise.

where r_t is the range, d_t is the distance between UWB sensors, b_t is the range bias, and n_t is additive noise. In the following sections we build this model in more detail by examining the two types of biases.

Gaussian Noise in LOS Scenarios

The round-trip time calculation discussed in Section 3.2.1 is approximate in nature, leading to uncertainty in the range calculation which can be modelled as a stochastic process. In a testing campaign, we developed a Gaussian model to describe ranging uncertainty in LOS scenarios.

We attempted to characterize this Gaussian process by gathering range data between a pair of sensors at various distances with LOS visibility. The distance was increased in 0.25 meter increments from 1 to 14 meters of separation and at each point, 1,000 range samples were gathered. The resulting data is plotted in Figure 3-1, showing the mean bias μ_b and standard deviation σ_b errorbar at each distance.

This data suggests that the LOS range bias can be reasonably modeled as distance-varying Gaussian noise, with mean $\mu_b(d_t)$ and standard deviation $\sigma_b(d_t)$. Computing a linear regression yields

$$\mu_b(d_t) = \mu_b^m d_t + \mu_b^b \tag{3.4}$$

$$\sigma_b(d_t) = \sigma_b^m d_t + \sigma_b^b. \tag{3.5}$$

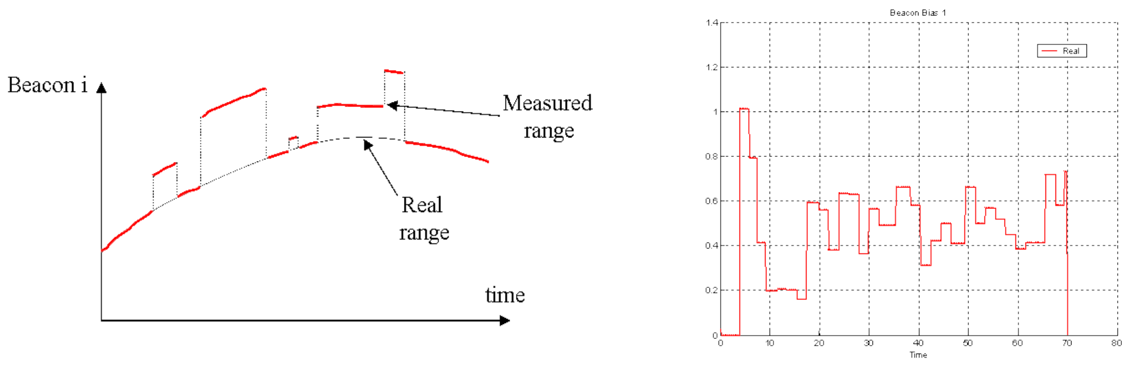


Figure 3-2: Ultra-Wideband Range Bias in NLOS [24]. Range data gathered by a mobile robot displays characteristic biasing as the robot moves around the environment. This data suggests that the bias tends to remain constant in local regions with constant UWB channel properties, and transitions abruptly when entering a new region. These effects are due to NLOS transitions, where the UWB channel in different spatial regions is defined by different multipath and propagation profiles.

The range function in Equation 3.3 then becomes

$$r_t = d_t + \mu_b(d_t) + \mathcal{N}(0, \sigma_b(d_t)^2), \tag{3.6}$$

where the bias b_t is now a linear function of the distance $\mu_b(d_t)$, and the noise term n_t is zero-mean Gaussian noise with covariance $\sigma_b(d_t)^2$.

When used in filtering problems, the range function in Equation 3.6 corresponds to the observation function $z_t = h(x_t) + v_t$, with $z_t = r_t$, $v_t = \mathcal{N}(0, \sigma_b(d_t)^2)$ and $h(x_t)$ is given as

$$\begin{aligned} h(x_t) &= d_t + \mu_b(d_t) & (3.7)\\ &= \mu_b^b + (1 + \mu_b^m)\sqrt{(x - x_b)^2 + (y - y_b)^2}, & (3.8) \end{aligned}$$

where x_t is assumed to be the robot pose $(x, y, \theta)_t$ at time t.

Stochastic Model of NLOS Biasing

In NLOS scenarios, additional positive bias is induced by two phenomena. The first type of delay occurs when the UWB signal travels through obstacles, in which the pulse is delayed and distorted during propagation through the obstacle material. The second type of bias occurs when the signal cannot penetrate obstacles occluding the direct path, and the received signal is a multipath component which is reflected around the obstacle. The path of a reflected signal requires more time for transmission, and thus induces an overestimate

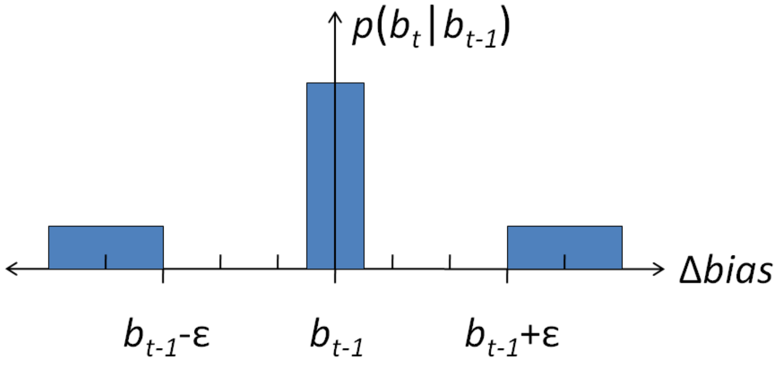


Figure 3-3: Bias Transition Model. This tri-modal distribution represents the transition probabilities for the bias at a given time step. The central mode represents the hypothesis that the bias b_t will remain relatively unchanged from the prior bias b_{t-1}. The left at right modes represent the possibility that the bias may undergo a discrete jump if the hidden UWB channel properties change.

in the TOF calculation.

We follow previous work [24] as a starting point, which develops a probabilistic model of bias transitions resembling a notch. This distribution is motivated by examining the behavior of range measurements as the robot moves around a mixed LOS/NLOS environment. Figure 3-2 shows range data gathered by the robot as it moves around an environment. In this figure it is evident that the range bias is constant for short periods of time, but undergoes discrete jumps between consecutive time periods. The intuition underlying this behavior is that the UWB channel properties vary in different spatial regions of the map due to local LOS or NLOS characteristics. Bias estimates tend to remain constant in local spaces with common UWB channel properties, but jump significantly when crossing the threshold to an adjacent spatial locality.

This behavior can be captured probabilistically as a notch-shaped distribution, shown in Figure 3-3. This density accounts for the two possible bias transitions observed at a given time step; either the bias remains relatively constant, or the robot enters a new spatial region and the bias jumps a discrete amount. The central mode in this distribution represents the probability that the bias remains the same during a new time step. The two modes on the left and right represent the probability that the bias will jump by at least ϵ to $b_t = b_{t-1} \pm \epsilon$ (with limits imposed on the extreme values of the bias).

Note that in a practical implementation, the bias transition probability distribution $p(b_t|b_{t-1})$ also depends on the existence of controls $u_t \neq \emptyset$ at time t, resulting in the transition probability $p(b_t|u_t, b_{t-1})$.

3.3 Probabilistic Robot Motion Model

In addition to UWB sensors, our robot receives odometry data from two wheels which we use to perform control updates during localization. In this section, we present the probabilistic robot motion model used to compute state transitions.

To capture the relationship between the control input and realized change in robot pose, robot motion can be decomposed into three components [16]: down-range, cross-range, and turning motions. Down-range motion, denoted D, corresponds to the distance traveled in the major axis of movement (forwards and backwards in robot coordinates). Cross-range motion, C, describes the lateral translation in the direction orthogonal to the major axis, which models shifting effects such as slippage. The turning component, T, is the turn performed by the robot during movement. The components of this model are depicted in Figure 3-4 for an example trajectory.

The state transition equation which models the state update due to robot motion over a given time interval is

$$x_t = g(x_{t-1}, u_t), \tag{3.9}$$

where the control variable corresponding to this motion model is $u_t = \begin{bmatrix} D & C & T \end{bmatrix}^T$ and the components of g corresponding to state variables x, y, and θ are

$$\begin{aligned} g_x &= x + D\cos(\theta + \frac{T}{2}) + C\cos(\theta + \frac{T+\pi}{2}) \\ g_y &= y + D\sin(\theta + \frac{T}{2}) + C\sin(\theta + \frac{T+\pi}{2}) \\ g_\theta &= \theta + T \mod 2\pi. \end{aligned}$$

This formulation lends itself to a natural probabilistic noise model in which each of the three motion components are represented as normal distributions. Each distribution is conditionally Gaussian given the amount of lateral d and rotational t movement reported by odometry over the current time interval, and is parameterized as

$$D \sim \mathcal{N}(d\mu_{D_d} + t\mu_{D_t}, d^2\sigma^2_{D_d} + t^2\sigma^2_{D_t}) \tag{3.10}$$

$$C \sim \mathcal{N}(d\mu_{C_d} + t\mu_{C_t}, d^2\sigma^2_{C_d} + t^2\sigma^2_{C_t}) \tag{3.11}$$

$$T \sim \mathcal{N}(d\mu_{T_d} + t\mu_{T_t}, d^2\sigma^2_{T_d} + t^2\sigma^2_{T_t}). \tag{3.12}$$

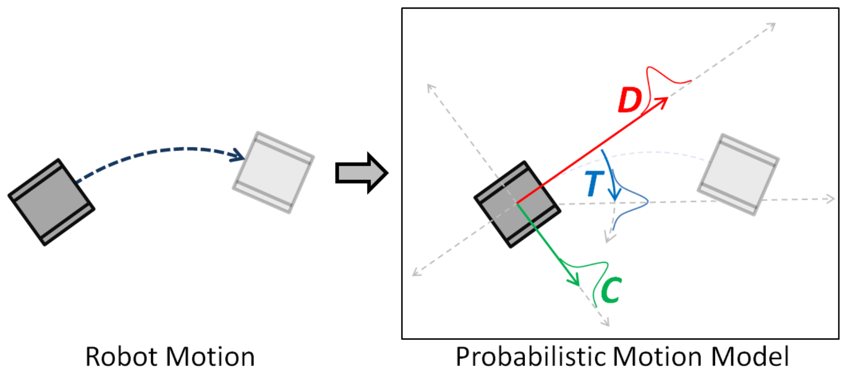


Figure 3-4: Probabilistic Motion Model. The robot motion trajectory (shown on the left) is decomposed into down-range D, cross-range C and turning T components in the probabilistic motion model (shown on the right). Each of these components is normally distributed, capturing the stochastic nature of motion outcomes.

3.4 Monte Carlo Range-Based Localization

Our goal is to accurately estimate the location of a dynamic robot given odometry motion data and UWB range measurements from multiple anchor beacons within a dense multipath environment, as in [24]. We restrict our discussion and experiments to the estimation of a two-dimensional robot pose; however, the method presented is generalizable to larger problems. We assume a harsh target environment, where GPS signals are unavailable, but geolocation can be performed by obtaining UWB range measurements in mixed LOS/NLOS conditions. As discussed in Section 3.2, these range measurements will be positively biased according to unobservable, spatially-varying UWB channel properties. Thus, to achieve accurate localization it is necessary to estimate these hidden range biases in addition to the robot location.

We seek to infer the robot pose x_t and range biases $b_t = \{b^{[1]}, b^{[2]}, \dots b^{[m]}\}$ to each of m reference sensors at time t based on all past controls $u_{1:t}$ and measurements $z_{1:t}$. Probabilistically, this corresponds to the joint distribution over poses x_t and biases b_t, which is given as

$$p(x_t, b_t | u_{1:t}, z_{1:t}).$$

Using the sensor and motion models developed in Sections 3.2.2 and 3.3, the evolution of this inference problem over time can be characterized by a dynamic Bayes network (DBN). The graphical model corresponding to the network in our localization problem is shown in Figure 3-5. The robot pose x_t at time t is stochastically dependent on the previous pose

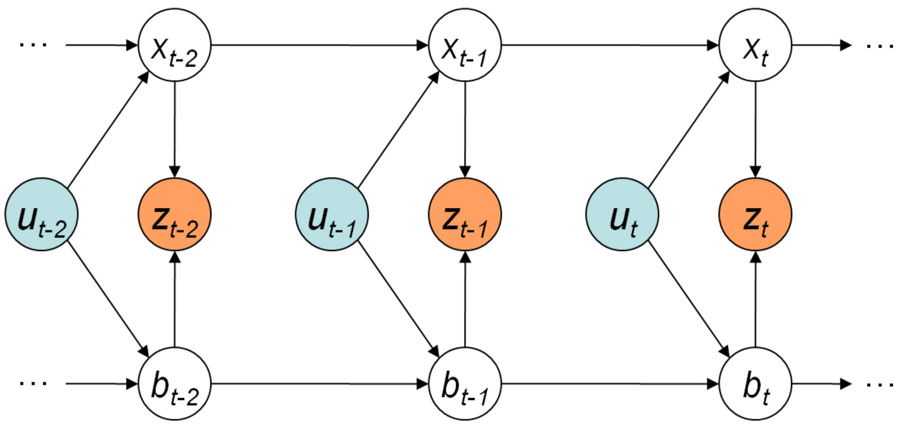


Figure 3-5: Graphical model of state inference in our range-based localization problem.

x_{t-1} and the motion u_t, which is given as the state transition probability $p(x_t|x_{t-1}, u_t)$. Similarly, the range bias b_t depends on the previous bias b_{t-1} and the robot motion u_t with bias transition probability $p(b_t|b_{t-1}, u_t)$. This corresponds to the non-linear NLOS bias transition model presented in Section 3.2.2. At each time step, the range measurement z_t depends stochastically on the robot pose x_t and the range bias b_t as the measurement probability $p(z_t|x_t, b_t)$.

Conventional parametric techniques for maintaining this DBN, such as Kalman filter approaches, have proven ineffective in capturing the nonlinearity of bias transitions [14]. This problem is addressed by Jourdan *et al.* [24], where they use a particle filter to maintain an estimate of the joint distribution over agent positions and UWB range biases. As discussed in Section 2.5, particle filters offer a nonparametric implementation of the Bayes filter which maintains the belief distribution as a finite set of sample particles. Each particle represents a belief hypothesis drawn randomly according to the prior belief. Particles can be propagated through arbitrary control and measurement functions, accumulating weight based on the likelihood of the resulting hypothesis they represent. This technique is powerful because the particle set can approximate a broader range of hypotheses and functions that cannot be accurately represented by parametric models.

This Monte Carlo approach is well-suited to our range-based localization problem for several reasons. Foremost, the particle filter admits multiple distinct state hypotheses, allowing for the distribution over robot poses and range biases to be multi-modal. This can be crucial in situations where the given information suggests that the robot could be at one of many distinct locations (e.g. in either of two identical rooms which would generate similar observations). Further, the NLOS bias model developed in Section 3.2.2 is based on the fact that bias transitions are *not* smooth, and the bias is likely to jump between distinct values represented by the tri-modal distribution shown in Figure 3-3. The particle filter also admits non-linear transition and observation functions. The motion and sensor models used in this problem are both non-linear and can be more reasonably approximated by using a Monte Carlo method than by filters that employ linearization techniques, such as the EKF.

While these powerful advantages have made the particle filter a popular approach to localization, the major drawback of this method is that sampling in high-dimensional spaces can be inefficient. For this approach to be applicable to a more general class of problems, it is necessary to accommodate a larger number of states. In our problem formulation, the belief consists of the robot pose augmented with range biases for each sensor in the environment. As the number of UWB beacon sensors grows, the particle filter must estimate additional biases and sample from a higher-dimensional state space. Additionally, if this Monte Carlo approach is used to estimate a larger robot belief (e.g. three-dimensional location), or if the sensor infrastructure requires additional state, it will suffer from scalability issues.

In the remainder of this chapter, we extend the work of Jourdan *et al.* [24] by showing that their Monte Carlo approach can be made more efficient and scalable by decomposing the joint state distribution. The resulting Rao-Blackwellized particle filter provides a more general approach to localization problems with the potential to offer increased performance on a wider class of problems.

3.4.1 Rao-Blackwellization Technique

A technique known as Rao-Blackwellization has been shown to increase the efficiency and accuracy of particle filters for DBN structures in which some of the state variables can be marginalized out exactly [15]. This technique has proven useful in numerous applications, notably in the implementation of the FastSLAM algorithm [38]. The Rao-Blackwellized particle filter (RBPF) divides the hidden state variables into two groups, sampling particles for one group as in the standard PF, while analytically computing the other. This improves overall efficiency by reducing the dimensionality of the sampling space, and can increase accuracy by exactly computing the set of tractable variables.

The motivation for the Rao-Blackwellization technique can be seen as follows. Suppose the set of state variables s_t can be divided into two partitions $s_t = \{x_t, b_t\}$. The posterior belief $p(s_t|u_{1:t}, z_{1:t})$, with controls u_t and observations z_t, can then be written as the joint distribution $p(x_t, b_t|u_{1:t}, z_{1:t})$. It is possible to decompose a joint distribution using the *law of conditional probability*, which is given as

$$p(a|c) = \frac{p(a,c)}{p(c)}, \quad p(c) > 0 \tag{3.13}$$

for the random variables a and c. This shows that if a and c are conditionally dependent, then the conditional probability $p(a|c)$ of a given c equals the joint probability $p(a, c)$ normalized over $p(c)$. Note that by multiplying Equation 3.13 on both sides by $p(c)$, we obtain the following form (sometimes referred to as the *chain rule*):

$$p(a, c) = p(a|c)p(c). \tag{3.14}$$

We can apply Equation 3.14 to the joint distribution $p(x_t, b_t|u_{1:t}, z_{1:t})$ over poses x_t and biases b_t, substituting $a = b_t$ and $c = x_t$ to obtain the following decomposition:

$$p(x_t, b_t|u_{1:t}, z_{1:t}) \;=\; p(b_t|u_{1:t}, z_{1:t}, x_t) \cdot p(x_t|u_{1:t}, z_{1:t}). \tag{3.15}$$

Equation 3.15 suggests that if $p(b_t|u_{1:t}, z_{1:t}, x_t)$ is analytically tractable, then the posterior belief corresponding to the subset of state variables x_t can be computed independently. This can have a substantial impact when applied to Monte Carlo methods, such as the particle filter, which require a number of samples that grows exponentially in the number of state dimensions. Instead of sampling from the full distribution $p(x, b| \ldots)$, we only need to sample from the lower-dimensional distribution $p(x| \ldots)$. We then separately maintain $p(b_t|u_{1:t}, z_{1:t}, x_t)$ by computing it exactly at each time step.

Put differently, to maintain the joint distribution over x_t and b_t with a particle filter, each particle would represent a sampled hypothesis of robot pose *and* range biases, such that

$$\mathcal{X}_t^{[i]} = \langle x_t^{[i]}, b_t^{[i]} \rangle, \tag{3.16}$$

where $x_t^{[i]}$ and $b_t^{[i]}$ are vectors of n and m dimensions, respectively. Thus, to maintain $p(x, b| \ldots)$ it is required to sample from a $(n + m)$-dimensional space. When the Rao-

Blackwellization technique is applied, the joint distribution over x_t and b_t is maintained as two separate conditional distributions as shown in Equation 3.15. In this case, each particle consists of a sampled robot pose hypothesis and a corresponding distribution over range biases, which is given as

$$\mathcal{X}_t^{[i]} = \langle x_t^{[i]}, p(b_t^{[i]}) \rangle, \tag{3.17}$$

Thus, it is only necessary to sample from the n-dimensional space of robot poses x_t, and subsequently maintain a bias distribution is each particle.

3.4.2 Simultaneous Pose and Bias Estimation

The Rao-Blackwellization technique can be applied to the range-based localization problem, allowing us to decompose our joint posterior over the robot state x_t and bias state b_t, as in Equation 3.15. Our task is then to derive the necessary filter update equations for maintaining the conditional distributions corresponding to each partition of the state.

Bias Distribution Estimation

We begin by examining the conditional distribution of the bias state $p(b_t|u_{1:t}, z_{1:t}, x_t)$. Applying Bayes rule (Equation 2.4), we obtain

$$p(b_t|u_{1:t}, z_{1:t}, x_t) \quad = \quad \eta p(z_t|u_{1:t}, z_{1:t-1}, x_t, b_t) \cdot p(b_t|u_{1:t}, z_{1:t-1}, x_t), \tag{3.18}$$

where η is a normalization factor. Due to conditional independence (shown in the graphical model in Figure 3-5), this simplifies to become

$$p(b_t|u_{1:t}, z_{1:t}, x_t) \quad = \quad \eta p(z_t|x_t, b_t) \cdot p(b_t|u_t). \tag{3.19}$$

We can apply the law of total probability (Equation 2.8) to capture the dependence of the bias transition model on the previous bias b_{t-1} as follows

$$p(b_t|u_{1:t}, z_{1:t}, x_t) \quad = \quad \eta p(z_t|x_t, b_t) \cdot \int p(b_t|u_t, b_{t-1}) p(b_{t-1}) db_{t-1}. \tag{3.20}$$

In Equation 3.20, it is clear to see that the conditional bias distribution admits separate control and measurement updates.

Note: In Equations 3.19-3.20 we have removed x_t from the bias transition model due to conditional independence, as in our current model the bias update does not depend on the location of the robot. However, if the bias model varies over position in the environment (e.g. if a detailed sensor coverage map is available), it is possible to incorporate x_t into the bias transition function. In that case, one would then have to perform forward-backward inference to capture the smoothed distribution $p(b_{t-1}|x_t)$ in the transition integral.

Robot Pose Distribution Estimation

Similarly, we can derive the components of the robot state conditional distribution $p(x_t|u_{1:t}, z_{1:t})$ arising in Equation 3.15. Note that this corresponds to the standard belief distribution used in the Bayes filter; however, in deriving the filter, we must account for the dependence of the observation function on both the robot state x_t and the bias state b_t. Again, we apply Bayes rule and conditional independence to obtain the standard recursion

$$\begin{aligned} p(x_t|u_{1:t}, z_{1:t}) &= \eta p(z_t|u_{1:t}, z_{1:t-1}, x_t) \cdot p(x_t|u_{1:t}, z_{1:t-1}) \\ &= \eta p(z_t|x_t) \cdot \int_{x_{t-1}} p(x_t|u_t, x_{t-1}) p(x_{t-1}|u_{1:t-1}, z_{1:t-1}) dx_{t-1}. \end{aligned} \tag{3.21}$$

We can incorporate the bias b_t by using the law of total probability (Equation 2.8), which gives

$$p(z_t|x_t) = \int_{b_t} p(z_t|x_t, b_t) p(b_t) db_t. \tag{3.22}$$

Altogether, the conditional distribution over the robot state is written by combining Equations 3.21-3.22, yielding

$$\begin{aligned} p(x_t|u_{1:t}, z_{1:t}) &= \eta \int_{b_t} p(z_t|x_t, b_t) p(b_t) db_t \\ &\quad \cdot \int_{x_{t-1}} p(x_t|u_t, x_{t-1}) p(x_{t-1}|u_{1:t-1}, z_{1:t-1}) dx_{t-1}. \end{aligned} \tag{3.23}$$

3.4.3 Rao-Blackwellized Particle Filter

The Rao-Blackwellized particle filter algorithm is shown in Algorithm 6, and is comprised of the filter updates derived in the previous section which culminated in Equations 3.20 and

Algorithm 6 The Rao-Blackwellized Particle Filter algorithm.

```
Require: Previous belief ($\mathcal{X}_{t-1}$), control ($u_t$) and measurement ($z_t$)
Ensure: Posterior belief ($\mathcal{X}_t$)
 1: $\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
 2: for $i = 1$ to $N$ do
 3:    sample $x_t^{[i]} \sim p(x_t|u_t, x_{t-1}^{[i]})$
 4:    compute $p(b_t^{[i]}, u_t) = \int_{b_{t-1}^{[i]}} p(b_t^{[i]}|u_t, b_{t-1}^{[i]})p(b_{t-1}^{[i]})db_{t-1}^{[i]}$
 5:    $w_t^{[i]} = \int_{b_t^{[i]}} p(z_t|x_t^{[i]}, b_t^{[i]})p(b_t^{[i]})db_t^{[i]}$
 6:    compute $p(b_t^{[i]}|u_t, z_t, x_t) = \eta p(z_t|x_t^{[i]}, b_t^{[i]}) \cdot p(b_t^{[i]}|u_t)$
 7:    $\overline{\mathcal{X}}_t^{[i]} = \langle x_t^{[i]}, p(b_t^{[i]}), w_t^{[i]} \rangle$
 8: end for
 9: for $i = 1$ to $N$ do
10:    draw $i$ with probability $\propto w_t^{[i]}$
11:    add $\langle x_t^{[i]}, p(b_t^{[i]}) \rangle$ to $\mathcal{X}_t$
12: end for
13: return $\mathcal{X}_t$
```

3.23. We utilize the RBPF for maintaining the bias-augmented belief state in our range-based localization problem as a particle set, with particle i given as

$$\mathcal{X}_t^{[i]} = \langle x_t^{[i]}, p(b_t^{[i]})^{[1]}, p(b_t^{[i]})^{[2]}, \dots p(b_t^{[i]})^{[M]}, w_t^{[i]} \rangle,$$

where $x_t^{[i]}$ is a sampled robot pose, $p(b_t^{[i]})^{[j]}$ is the range bias distribution corresponding to sensor j, M is the number of sensors, and $w_t^{[i]}$ is the importance weight assigned to the particle. For convenience in notation, we will assume there is one sensor $M = 1$ in the environment and refer to the bias distribution as $p(b^{[i]})$, without loss of generality.

In this section, we discuss in detail the steps of the algorithm and their implementation in our solution to the localization problem.

Choice of Representation

In Chapter 2 we presented the Bayes filter, and showed that any implementation thereof requires a representation of the state distribution $p(s_t)$, a state transition model $p(s_t|u_t, s_{t-1})$, and an observation model $p(z_t|s_t)$. When decomposing the state variables with the Rao-Blackwellization technique, one must then choose these three distributions for *each* partition of the hidden variables.

As we have stated, our goal is to utilize Rao-Blackwellization to improve the efficiency of particle filtering for localization, allowing us to maintain the robot pose with the traditional sampling-based approach and exactly compute the range bias distribution in each update. In this case, the density of the robot pose $p(x_t)$ is represented as a sampled particle set and the transition function due to motion $p(x_t|u_t, x_{t-1})$ can be computed in the typical manner. The observation function $p(z_t|x_t)$, however, is dependent on the range bias b_t.

Thus, the remaining task is to choose a representation for the hidden range bias distribution and the corresponding transition and observation models. In order to analytically compute the bias distribution for each particle at each timestep, the representation must admit a closed-form update. Further, it must enable the observation function $p(z_t|x_t)$, which integrates over the bias density (Equation 3.22), to be computed. Typical choices would include exponential family functions, such as the normal distribution; however, given the NLOS sensor model and non-linear notch transition model (Section 3.2.2), the selection of reasonable representations is more limited. We chose to model the bias density with a discrete multinomial distribution, as this admits closed-form updates and naturally captures the notch transition model. This representation also enables simple computation of the importance weight integration (line 5) by iterating over discrete bins in the multinomial. The implementation is discussed side-by-side with the algorithm in more detail below.

Discussion of RBPF Implementation

As in the standard particle filter (Chapter 2.5), the RBPF method consists of two main steps. The first step is in lines 1-8 of Algorithm 6, which creates a temporary set of particles $\overline{\mathcal{X}}_t$ based on the controls u_t and observations z_t in the filter iteration at time t. The second step produces the resulting particle set $\mathcal{X}_t$ by resampling the temporary set $\overline{\mathcal{X}}_t$ according to the particle weights w_t. We begin by discussing the control and measurement update steps used to generate $\overline{\mathcal{X}}_t$.

The control updates for the conditional distributions of the robot pose and bias are performed in lines 3 and 4 as follows:

- **Line 3**: Implements the state transition of the robot pose corresponding to robot motion. This is performed by sampling controls from the motion model (Section 3.3) and applying them to the previous robot pose x_{t-1}. The raw controls, distance d and rotation t, are used to sample from the downrange D, crossrange C, and turn T distributions defined in Equations 3.10-3.12.

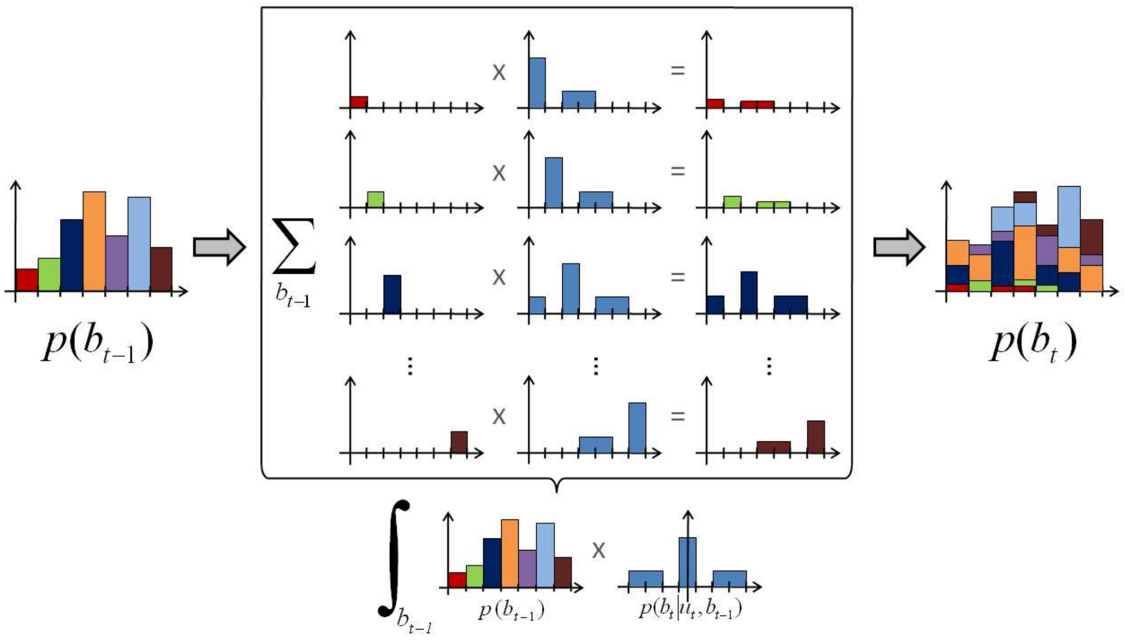


Figure 3-6: Bias Transition Update. The bias multinomial distribution $p(b_{t-1})$ is updated in response to controls u_t by computing the weighted sum of the transition probability of each multinomial bin. This computation is depicted graphically in the box in the center of the figure, showing the contribution of each bin transition to the resulting distribution $p(b_t)$, shown on the right.

- **Line 4**: Implements the bias transition model discussed in Section 3.2.2, which updates the bias distribution $p(b_t^{[i]})$ of particle i in response to controls u_t. To incorporate the dependence on the previous bias, we integrate over the previous bias distribution $p(b_{t-1}^{[i]})$. Since the bias is maintained as a discrete multinomial density, this is implemented as a summation over each bias bin convolved with a corresponding notch transition model. This is depicted in Figure 3-6 and can be described as

$$p(b_t|u_t) = \sum_{j=1}^{B} \underbrace{p(b_t|u_t, \beta_{j-1} < b_{t-1} \le \beta_j)}_{\text{notch transition}} \underbrace{p(\beta_{j-1} < b_{t-1} \le \beta_j)}_{\text{bias bin}}, \tag{3.24}$$

where B is the number of bins in the bias multinomial and β_j is the upper boundary of the bin interval j. For a given bin $(\beta_{j-1} < b_{t-1} \le \beta_j)$, the contribution to the updated density $p(b_t|u_t)$ is computed by multiplying by a corresponding notch transition multinomial located about the center of the bin $\frac{\beta_{j-1}+\beta_j}{2}$. The sum over all of these resultants yields the updated distribution $p(b_t|u_t)$, and is a well defined probability density (no normalization is required).

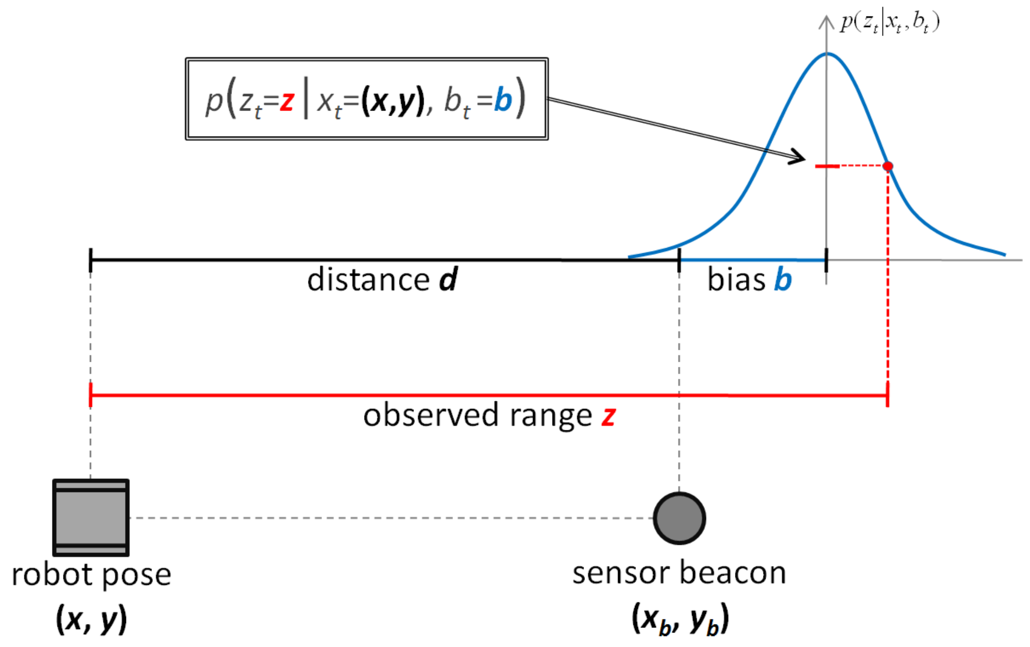


Figure 3-7: Range Measurement Update. The robot, at pose (x, y), obtains a range measurement to the sensor beacon located at (x_b, y_b). The predicted observation is given by the euclidean distance d between the robot and sensor plus the bias b with Gaussian noise (shown in blue). The importance weight w_t of a particle is computed as the observation probability $p(z_t|x_t, b_t)$, whose computation is depicted in red. The probability of obtaining the observed range z is calculating by evaluating the sensor model Gaussian $p(z_t|x_t, b_t)$ at range $z_t = z$.

The measurement updates for each distribution take place in lines 5 and 6 as follows:

- **Line 5**: Implements the importance weight $w_t^{[i]}$ calculation for particle i by computing the observation probability $p(z_t|x_t^{[i]})$. The observation model depends on the range bias $b_t^{[i]}$, which is incorporated by integrating over the bias distribution $p(b_t^{[i]})$. For the discrete multinomial bias density, this is computed as a summation, given as

$$p(z_t|x_t^{[i]}) = \sum_{j=1}^{B} \underbrace{p(z_t|x_t^{[i]}, \beta_{j-1} < b_t^{[i]} \leq \beta_j)}_{\text{sensor model}} \underbrace{p(\beta_{j-1} < b_t^{[i]} \leq \beta_j)}_{\text{bias bin}}, \tag{3.25}$$

where B is the number of bins in the bias multinomial and β_j is the upper boundary of the bin interval j. This computation is depicted in Figure 3-7 for one bias bin $(\beta_{j-1} < b_t^{[i]} \leq \beta_j)$, where the bias b used in the calculation is the center of the corresponding bin $b = \frac{\beta_{j-1}+\beta_j}{2}$. The distance d between the robot $x_t^{[i]} = (x, y)$ and a known sensor (x_b, y_b) is computed as

$$d = \sqrt{(x - x_b)^2 + (y - y_b)^2}. \tag{3.26}$$

In the figure, omitting the indices of the particle, the predicted measurement distribution $p(z_t|x_t, b_t)$ is a Gaussian (Section 3.2.2) centered about a mean predicted range of $d + b$. The likelihood of obtaining the actual range z is computed by evaluating $z_t = z$ in the predicted observation function.

- **Line 6**: Implements the bias measurement update, which calculates the probability of each possible bias given the observed range measurement. The bias distribution is updated by iterating through the discrete multinomial distribution and computing the measurement probability for each bin. Each bin in the multinomial distribution can be updated as

$$p(\alpha < b_t^{[i]} \leq \beta | u_t, z_t, x_t^{[i]}) = \eta p(z_t | \alpha < b_t^{[i]} \leq \beta, x_t^{[i]}) \cdot p(\alpha < b_t^{[i]} \leq \beta | u_t, x_t^{[i]}), \quad (3.27)$$

  where α and β are the lower and upper bin boundaries, respectively. The right-hand term $p(\alpha < b_t^{[i]} \leq \beta | u_t, x_t^{[i]})$ is simply the current bin value obtained during the control update. The measurement probability $p(z_t | \alpha < b_t^{[i]} \leq \beta, x_t^{[i]})$ is computed as in the importance weight computation described above and shown in Figure 3-7. After iterating through all regions in the multinomial, the distribution is then normalized.

The resampling phase of Algorithm 6 in lines 9-12 is identical to that of the standard particle filter. The final particle set $\mathcal{X}_t$ is obtained by sampling particles from $\overline{\mathcal{X}}_t$, where each particle i is drawn according to its importance weight $w_t^{[i]}$.

3.5 Experiments and Results

In order to evaluate the Rao-Blackwellized particle filter localization algorithm, we performed two experiments in a small indoor environment in simulation. In both experiments, a mobile robot traversed a path approximately 50 m in length turning to follow three corridors in an office-like scenario. The map was 50 m on a side and UWB beacons were placed at randomized locations. The environment was filled with walls and obstacles to generate mixed LOS and NLOS conditions. To quantify the accuracy of localization, we measured the realized positional error of the robot at the goal location. In each evaluation, we compared the RBPF approach presented in Section 3.4 to a particle filter estimating the full joint distribution over robot poses and range biases.

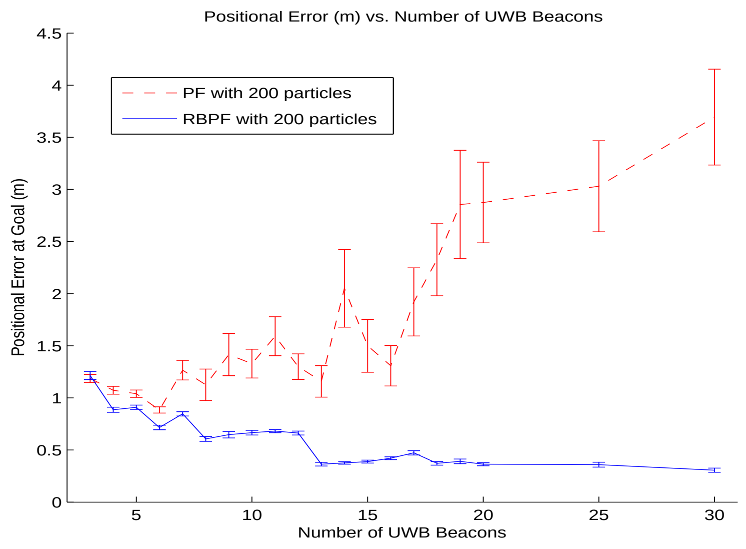


Figure 3-8: Scalability of Localization Performance. This graph shows error bars for the positional accuracy of the Particle Filter (PF) and Rao-Blackwellized Particle Filter (RBPF) with a constant number of particles and variable number of UWB beacons. The RBPF scales well to estimate an increasing number of UWB range biases, using the additional measurement information to improve localization performance. The PF, on the other hand, cannot accurately represent the belief distribution with only 200 particles as the dimension of the belief space increases. Positional accuracy of the PF improves from 3 to 6 beacons, but thereafter diverges and realizes worse performance than using odometry alone (which obtains an average positional error of 1.3 m).

Evaluation of Algorithm Scalability

The first evaluation tested the scalability of the localization algorithms by increasing the number of UWB beacons in the environment, while limiting the filters to a constant number of particles. We varied the number of beacons from 3 to 30, limited the particle filters to 200 samples and performed 40 trials for each environment.

The results of these trials are shown in Figure 3-8. As the number of UWB beacons is increased, the RBPF scales well to estimate additional biases and use the information gain to achieve better localization performance. The PF, however, cannot maintain an accurate position estimate as the dimension of the belief space increases with additional beacon biases. The particle filter is able to utilize sensor information to improve localization performance from 3 to 6 beacons, but diverges as the state space if further increased with additional beacons. This divergence is somewhat counterintuitive; one would expect additional sensor information to improve the quality of localization. Since the particle filter is limited to 200 particles, it is overwhelmed with increasing numbers of states and left to choose from a set of poor hypotheses. As a result, the robot becomes nearly lost and realizes

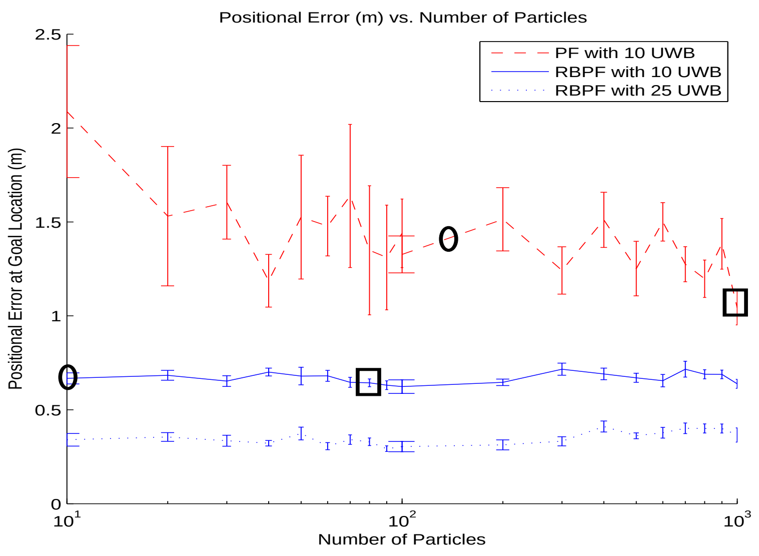


Figure 3-9: Comparison of Positional Accuracy. This graph shows error bars for the positional accuracy of the Particle Filter (PF) and Rao-Blackwellized Particle Filter (RBPF) over a varying number of particles. In an environment with 10 UWB beacons, the RBPF consistently realizes more accurate and reliable positional accuracy than the PF. The RBPF requires very few particles to achieve reasonable localization, and scales well to utilize additional sensor information as shown in an additional plot with 25 UWB beacons. The ellipses and rectangles in the diagram indicate points of equivalent computational time between methods.

worse performance using sensor measurements than with odometry alone (which obtains an average positional error of approximately 1.3 m).

This behavior hints at an underlying insight into the advantage of decomposing the joint distribution through Rao-Blackwellization; it is in the nature of this problem to maintain separate estimates of the robot pose and hidden biases. The RBPF ensures that particles are distributed to best represent the space of robot pose hypotheses. In the particle filter, on the other hand, the distribution of samples in the robot pose subspace of a high-dimensional state space can deplete to a sparse and incorrect representation without a sufficient number of particles.

Comparison of Positional Accuracy

The second evaluation tested the positional accuracy of each algorithm as the number of particles available to the filter was varied. During this experiment, we maintained a constant number of UWB beacons and increased the number of particles in each filter, performing twenty trials for each data point. The motivation for this test was to demonstrate that the RBPF uses particles more efficiently, achieving better performance with a smaller number of particles.

The results of the experimental trials are shown in Figure 3-9. In an environment with 10 UWB beacons, the average positional error of the particle filter generally decreases as more particles are available to represent additional hypotheses. This downward trend is very slow, indeed even with 10^4 particles (not shown in graph), the particle filter is still unable to localize as well as the RBPF with only 10 particles.

It is interesting to note that the positional accuracy of the RBPF does not noticeably improve with additional particles, as one would assume. This suggests that the RBPF makes very efficient use of particles and can accurately identify salient robot pose hypotheses. This also suggests that maintaining a separate, even coarse, representation of range biases as a multinomial is very effective in capturing non-linear bias transitions. Also note that the graph contains an additional plot of the average positional error of the RBPF in an environment with 25 UWB beacons, which demonstrates similar results.

There is an important issue to address regarding the comparisons in these two evaluations. We have not discussed the additional space and computation requirements required by the RBPF, which suggest that the number of particles is not a straightforward comparison. While this is the case, these evaluations provide insight for the behavior of both algorithms. Additionally, we have measured running times for each filter update during execution to provide comparison on the basis of computation. In Figure 3-9, we have shown two points of equivalent computation time between the RBPF and the PF indicated by the ellipses and rectangles. The average time per filter update required for the RBPF with 10 UWB beacons and 10 particles, is equivalent to the average time per filter update required by the PF with 10 UWB beacons and 150 particles (shown with ellipses). Similarly, the rectangles shown in Figure 3-9 indicate a computational time equivalence between the RBPF at 80 particles, and the PF at 1000 particles, for 10 UWB sensors. With this added dimension, it becomes clear that the RBPF algorithm produces superior results in typical operating conditions.

3.6 Conclusion

In this section, we have presented an improvement of the Monte Carlo approach to UWB range-based localization by applying the Rao-Blackwellization technique. This technique enables the joint distribution over robot poses and range sensor biases to be decomposed into two simpler distributions: a distribution over robot poses; and, a conditional distribution of sensor biases with respect to robot poses. Rather than sampling particles from the full joint distribution, it is only necessary to sample robot poses and, subsequently, analytically compute bias estimates for each particle. We have shown that the Rao-Blackwellized particle filter outperforms a standard particle filter approach to state estimation of a joint posterior over robot poses and range sensor biases. The RBPF results in both greater scalability to environments with more UWB beacons and better positional accuracy during localization.

3.7 Related Work

In this section we discuss related work in the problems of range-based localization and NLOS mitigation.

Previous Work in Range-Based Localization

Our work was primarily based on the Monte Carlo approach to UWB-based localization presented in [24], however other approaches have been presented using other carriers and different localization techniques. In [44], an EKF-based approach is used for indoor localization with the Cricket system, in which each sensor transmits on both RF and ultrasonic channels. While this hybrid sensor avoids most bias issues, many beacons are required per room and their EKF approach is subject to periodic failures. A positioning system is developed in [30] uses signal measurements between IEEE 802.11b wireless Ethernet nodes. While they are able to overcome biasing and infer the location of a mobile agent, they require a substantial training phase to build a sensor map of the environment. An interesting alternative is presented in [41], where magnetic field sensors are used to overcome the NLOS biasing limitations of RF sensors. They do not use an explicit sensor noise model, but rather use a least-squares method to estimate the location of a mobile robot. For further reading on localization in wireless networks we refer the reader to [21, 20].

Previous Work in NLOS Mitigation

Numerous techniques have been explored to cope with the biases induced by NLOS signaling conditions. Statistical information about the NLOS error is used in [54], where a time history of the variance of ToA measurements is used to identify NLOS observations. Subsequently, they reconstruct the equivalent LOS measurement after detecting NLOS. This technique was applied to obtain rough position estimates using mobile GSM telephones, but it is unclear if this method is applicable to UWB measurements.

Similarly, biased and unbiased Kalman filters are employed in [19, 31] to determine if measurements correspond to LOS or NLOS signal paths. Then position estimates at consecutive time intervals are smoothed by an additional filter to ensure a continuous robot trajectory. Also, by building a scattering model of the environment, ToA statistics can be constructed and used with techniques such as maximum likelihood (ML) to mitigate the effects of NLOS error [3, 4]. Both of these techniques, however, require a substantial training period to model the target environment.

3.A Ultra-Wideband Ranging

There are a number of techniques that enable positioning using RF sensors. Conventional approaches are based on angle of arrival (AOA), received signal strength (RSS), and time of arrival (ToA) calculations. Time-based schemes such as ToA are well-suited for ultra-wideband signals, because they provide great accuracy by exploiting the fine time resolution of UWB. In this section we discuss a time-based ranging method that we utilize for robot localization.

Communication and Ranging Protocol

Ultra-wideband radio can be used to generate range estimates by measuring the time taken for a signal to travel between a pair of UWB transceivers.

In a typical RF communication scheme, a transmitter emits waveform symbols within constant time segments called frames. The receiver can use a correlation-based approach to identify and decode the transmitted data within the frame. Shifted versions of a template signal are used to compute the correlation with a received signal, resulting in correlation peaks at the locations of matching waveforms. The time delay which results in the largest correlator output can then be used as the ToA estimate for ranging.

Conventional serial-based searching over each bin becomes intractable as the set of possible delay positions is very large in comparison to the UWB bin size. Further, it is impractical to continuously sample the high resolution UWB signal at the Nyquist rate, especially at beacon sensor locations where low power consumption is very desirable.

To overcome this challenge, the template correlation search is typically computed using a coarse-to-fine method, which consists of an *acquisition* phase followed by a *payload* phase. Acquisition is the process of synchronizing the radios. The goal is to identify the region in the signal frame, or *lockspot*, where the receiver can expect the payload data to arrive. A UWB link is created by sending several data frames called the *acquire preamble*. The receiver performs signal strength integration over coarse-sized windows within the frame, identifying the sub-frame region that contains the most signal energy. By placing the lockspot at this location, the uncertainty region for the payload arrival is drastically reduced.

The payload phase involves a fine resolution correlation-based search over the region surrounding the lockspot. The correlation output can then be processed to find the first path pulse, corresponding to the ToA of the ranging payload. Once this ToA is accurately measured, it is straightforward to compute the distance between sensors by using the velocity of the signal (in this case the speed of light c). However, such a precise measurement requires the clocks on both sensors to be synchronized, which is not the case in practical UWB systems.

Half-Duplex Ranging

A common ranging approach that alleviates the need for prior synchronization is known as *half-duplex ranging* [32]. This method measures the round-trip time taken for a request-response signal transaction between the agent (requester) and the anchor beacon (responder). In simplistic terms, this can be likened to sonar ranging, which measures the time elapsed between emitting a sonar ping and receiving an echo from the target object. The distance to the object is then computed using half of the round-trip time. Similarly, to compute the range to a reference sensor, a UWB requester can measure the time between transmitting a UWB pulse and receiving a response pulse. The challenge, however, is to account for positive biasing of the two-way time-of-flight due to internal delays in the sensor.

Three types of internal delays create errors in measuring the round trip time for a signal transaction between sensors:

1. *Electrical Delay* (t_{elec}): Two forms of electrical delay add to the round trip time: time in transmission $t_{elec-out}$ required to create a pulse in internal logic and send it

to the antenna; and time during reception $t_{elec-in}$ for the pulse to travel from the antenna back through internal logic. Since each transceiver undergoes both delays during ranging, we can lump them together as $t_{elec-A} = t_{elec-in-A} + t_{elec-out-A}$ for sensor A. Note that the each radio pre-computes its own constant electrical delay on startup.

2. *Acquisition Delay* (t_{acq}): The radios use a separate coding scheme to transmit the acquire preamble for synchronization, which adds a number of frames to the transmitted signal. This process identifies a lockspot within the frame window in which the payload signal is expected to arrive.

3. *Leading Edge Delay* (t_{led}): When a payload ranging pulse is received, the time window surrounding the lockspot must be processed and searched to detect the location of the first pulse component. A *leading edge* detection algorithm identifies the first multipath component received, which corresponds to the most direct pulse path. The time offset from this peak to the lockspot t_{led} is an additional bias.

These delays and the resulting round-trip time calculation between radios A and B are depicted in Figure 3-10. In this diagram, the independent timers in each radio are shown along the respective time lines t_A and t_B. The ranging process begins when radio A emits a request at $t_A = 0$. The request signal undergoes an electrical delay $t_{elec-out-A}$ before being transmitted from the antenna into open air. The signal then incurs a time-of-flight delay t_{tof} before reception by the antenna of radio B. Note that this delay t_{tof} corresponds to the exact range we wish to calculate.

The signal undergoes an electrical delay in radio B $t_{elec-in-B}$, traveling from the antenna to internal logic. At this point, the radio timers t_A and t_B are not synchronized, and a delay of t_{acq} is required to correct for this in the acquisition phase, which processes the acquire preamble frames of the signal packet. Radio B sets its clock to $t_B = 0$ at the coarse estimation of the payload arrival time, and at this point the radios are synchronized. The first path pulse is found using a leading edge detection algorithm and the corresponding offset t_{led-B} occurs before the lockspot $t_B = 0$. The response signal is emitted from the antenna of radio B at $t_B = t_{elec-out-B}$. Note that the response signal is appended with information packets that contain the timing delays incurred on radio B, namely t_{elec-B}, t_{acq} and t_{led-B}.

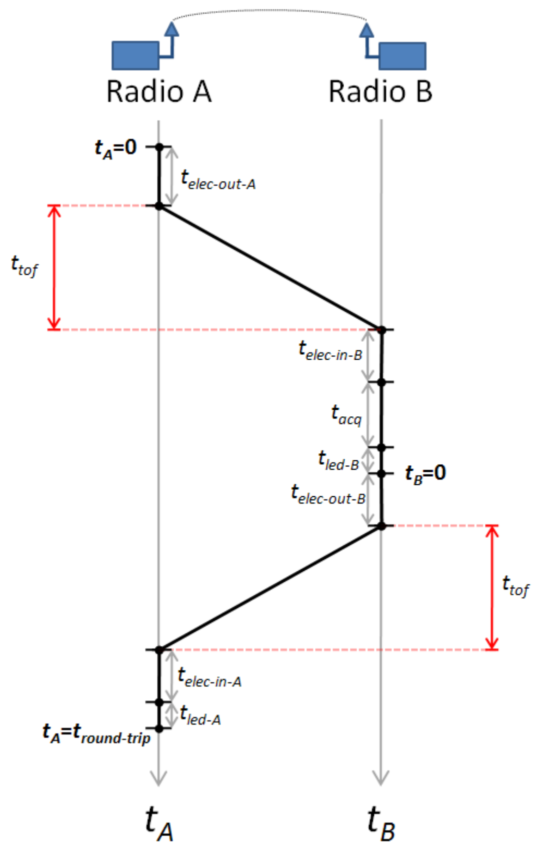

Radio A
Radio B
t_A=0
$t_{elec\text{-}out\text{-}A}$
t_{tof}
$t_{elec\text{-}in\text{-}B}$
t_{acq}
$t_{led\text{-}B}$
t_B=0
$t_{elec\text{-}out\text{-}B}$
t_{tof}
$t_{elec\text{-}in\text{-}A}$
$t_{led\text{-}A}$
t_A=$t_{round\text{-}trip}$
t_A
t_B

Figure 3-10: Half-Duplex Ranging Method: Radio A computes the range to radio B by using a two-way time-of-flight measurement and accounting for internal time delays.

The response signal undergoes the second time-of-flight t_{tof} delay, before reception by radio A's antenna, and then propagates through an additional electrical delay $t_{elec-in-A}$. Since radios A and B are synchronized, the lockspot of radio A at $t_A = t_{round-trip}$ represents the coarse estimation time of the ranging pulse arrival. The leading edge detection algorithm then finds the time offset t_{led-A} before $t_A = t_{round-trip}$ at which the direct path pulse occurs.

Thus, the round-trip time $t_{round-trip}$ is the total of all of the delays

$$\begin{aligned} t_{round-trip} &= t_{elec-out-A} + t_{tof} + t_{elec-in-B} + t_{acq} + t_{led-B} + t_{elec-out-B} \\ &\quad + t_{tof} + t_{elec-in-A} + t_{led-A}. \end{aligned}$$

The desired time-of-flight t_{tof} can then be determined as

$$t_{tof} = \frac{t_{round-trip} - t_{elec-A} - t_{elec-B} - t_{acq} - t_{led-A} - t_{led-B}}{2}.$$

Finally, the range r (in meters) between radios A and B can be computed as

$$r = t_{tof} \cdot c,$$

where $c = 3.0 \times 10^8$ m/s is the speed of light.

Chapter 4

Planning in Belief Space With Trajectory Search

Contents

In the following three chapters we incrementally develop a novel and efficient approach for planning in belief space. To date, methods for planning an optimal sequence of actions that consider robot belief distributions have been met with limited success due to the prohibitive size of the robot belief space. In this work, we show that by employing the linear-Gaussian assumption for robot beliefs and using a sampling-based approach to construct a representative subset of this belief space, planning in belief space can be performed tractably in a principled manner.

4.1 Introduction

For mobile robots to autonomously perform complex tasks, they must have the ability to plan a viable sequence of actions that achieves a desired goal with safe and reliable execution. One of the key challenges faced in attaining robust autonomy is to cope with the uncertainty that pervades real-world interactions. The perceptual limitations of a robot leads to incomplete information about the true state of the world; as the robot becomes more uncertain of its state, it is much more prone to failure. Since the level of reliability in performance is often directly linked to uncertainty in the robot's belief of the world, it is paramount that the planning algorithm choose actions that minimize belief uncertainty.

In this work, we are concerned with the problem of motion planning under uncertainty for mobile robots. To navigate safely and achieve robust performance, it is essential that robots choose paths that result in minimal belief uncertainty, thereby mitigating the chance of colliding with objects or becoming lost. The intuition underlying this approach is that it may be preferable to detour from paths with higher expected uncertainty, even if they are much shorter. If the robot biases its plans toward trajectories with better sensor coverage and higher expected information gain, the robot can achieve higher reliability in lieu of faster, but more risky, performance.

The maturing field of probabilistic robots has made it possible to capture the uncertainty inherent in robot actions, observations and beliefs with probability theory, enabling the evolution of robot-environment interactions to be modelled as a network of stochastic processes. Recursive filtering algorithms, such as the Kalman filter, make it possible to maintain an estimate of the robot's belief uncertainty as a probability distribution, which is updated in response to non-deterministic actions and observations. Methods using probabilistic state estimates have achieved great success in applications such as robot localization; however, to date, robust and scalable methods that incorporate uncertainty into the planning process are still to be desired.

Traditional solutions to motion planning rely on simplifying assumptions of the robot's belief to exploit the efficiency of well-established optimization and search algorithms. If it is assumed that perfect knowledge of the robot's pose is available, a routine search over possible states can be used to find a solution path; however, such techniques are not robust in real-world scenarios where the robot pose is estimated from incomplete information of observations and dynamics. The Markov Decision Process (MDP) [6] does admit non-deterministic action outcomes while generating a global control policy, but assumes the robot can observe the full state of the environment after each action. While this may be reasonable if the pose uncertainty remains small, in most real-world robot problems sensor limitations play a key role in determining the certainty of the robot's belief state.

The restrictive assumptions and limited performance of traditional planning algorithms motivate the development of methods that create plans utilizing the knowledge of uncertainty available in the probabilistic state estimate. Modern approaches to planning with incomplete state information are typically based on the partially observable Markov decision process (POMDP) model [46] or cast as a graph search through belief space [9]. Computing POMDP solutions requires finding an optimal action for each possible belief in the entire belief space. While the POMDP provides a general framework for belief space planning, as the size of the belief space grows POMDP techniques become computationally intractable. The complexity of the solution grows exponentially in both the number of possible control outcomes and the number of potential observations. Numerous approximation algorithms continue to mitigate the problem of scalability [40, 45, 47], but to date POMDP techniques still face computational issues in addressing large problems.

A promising alternative for belief space planning involves graph search over trajectories through belief space. In general, trajectory search is subject to similar scalability issues as POMDP techniques, wherein solutions become computationally infeasible as the belief space grows larger. While the formal POMDP model is only tractable for very small problems, the key insight for efficient trajectory search is that we are only concerned with the subset of beliefs the robot may actually experience. Randomized motion planners, such as the Probabilistic Roadmap (PRM) [28], have proven successful in solving complicated planning problems by limiting their scope to selected regions in the state space. The PRM algorithm approximates the topology of the configuration space with a graph structure of trajectories between accessible poses in free space.

Our goal is to apply the intuition underlying the PRM to belief space planning. We seek to use PRM methodology to overcome the POMDP complexity in the same way that the PRM has successfully solved high-dimensional motion planning problems; namely, by generating a compact representation of the belief space. Ideally, the PRM would do this by first limiting the scope of search to robot beliefs that are realizable, and then building a trajectory graph exclusively within this free space.

Even with a simplified representation of the belief space, scalability remains a limiting factor, as the branching factor of search through belief space is equal to the number of potential observations. For a robot using ranging sensors, incorporating all possible measurements at a given belief state yields a set of posterior beliefs that becomes intractably large. It is possible to overcome this limitation, however, by choosing a suitable belief representation. If one employs the common assumption that the belief state can be maintained as a single-mode Gaussian distribution, the effects of stochastic controls and potential measurements can be predicted in closed form using Kalman filter updates. We show that by also assuming the maximum likelihood observation at each belief, the Kalman filter mean updates become a linear process. This helps to isolate the search process to mean-belief trajectories, while also guaranteeing that straight-line trajectories represent well-defined filter updates.

In this chapter, we present a belief space variant of the PRM algorithm that utilizes the linear-Gaussian assumption to incorporate uncertainty into planning. Our general approach based on trajectory search is developed here and then further optimized in Chapter 5, culminating in the Belief Roadmap (BRM) algorithm in Chapter 6.

We begin in Section 4.2 by presenting the Probabilistic Roadmap (PRM) algorithm, which enables efficient motion-planning in robot configuration space by creating a simplified graph representation of the space. Our goal in Section 4.3 is to extend the PRM methodology to belief space to develop a tractable approach to planning with incomplete states. While the PRM cannot generally be applied to belief space, we show in Section 4.3.2 that by using Gaussian beliefs, a linear form of Kalman filter updates can be derived to admit PRM graph generation and search in the Kalman belief space. The resulting trajectory search algorithm is shown in Section 4.3.3, which finds a goal path of minimal uncertainty by propagating Kalman filter predictions of belief uncertainty through the trajectory graph. The chapter is concluded with a discussion in Section 4.4.

4.2 PRM-Based Planning in Configuration Space

The Probabilistic Roadmap (PRM) algorithm [28] can be used to produce a collision-free path between start and goal positions given a map and robot dimensions. The PRM method creates a graph of potential trajectories, then performs trajectory search through this graph to find a valid path between the start and goal. We begin this section by motivating the PRM algorithm from the formulation of the general motion planning problem.

The set of all possible robot poses within the map is known as the *configuration space* [34], which is denoted $\mathcal{C}$. This set can be partitioned such that $\mathcal{C} \equiv \mathcal{C}_{free} \cup \mathcal{C}_{obst}$, where $\mathcal{C}_{free}$ is the subset of all collision-free poses and $\mathcal{C}_{obst}$ is the subset of all poses resulting in collision with obstacles. In motion planning problems, the task is to find a path between start x_{start} and goal x_{goal} locations by searching for an optimal series of trajectories that lie exclusively within $\mathcal{C}_{free}$. In real-world scenarios, it may be intractable or unreasonable to compute an exact representation of $\mathcal{C}_{free}$; the PRM algorithm overcomes this challenge by approximating $\mathcal{C}_{free}$ using a sampling-based technique.

4.2.1 PRM Algorithm

The Probabilistic Roadmap provides a general framework for efficiently solving fully observable motion planning problems in two stages, as follows [28, 8]:

1. **Pre-processing phase**: The PRM builds and maintains a trajectory graph, or roadmap, that is a simplified representation of $\mathcal{C}_{free}$. Robot poses are sampled from $\mathcal{C}$ according to a suitable probabilistic measure and an inexpensive test is performed to determine if the pose lies in $\mathcal{C}_{free}$ or $\mathcal{C}_{obst}$. Desirable poses within $\mathcal{C}_{free}$ are retained and added as nodes in the trajectory graph. Edge trajectories are then created by computing the visibility to other nodes in the graph and retaining collision-free trajectories to the k nearest neighbors.

2. **Query phase**: Given a start and goal pose, a graph search algorithm finds a path through the trajectory graph that connects the corresponding start and goal nodes. If the start or goal location is not in the trajectory graph, an attempt is made to first add and link a corresponding node to the graph, as in the pre-processing phase.

The power of the PRM algorithm lies in the pre-processing phase, which captures salient features of the configuration space, resulting in a compact representation amenable to search. For planning problems in high-dimensional spaces, $\mathcal{C}_{free}$ can be approximated as a discrete graph by sampling poses from $\mathcal{C}$, retaining desirable samples and computing the visibility between them.

The PRM is also versatile, for it provides a very general framework that can be applied to a large class of problems and customized for specific tasks. The pre-processing phase allows considerable freedom in choosing *sampling strategies*, and the query phase can be implemented with any searching algorithm. Sampling strategies can be used to capture different topological features of $\mathcal{C}_{free}$. Such strategies can be used to bias samples towards regions of interest and to determine whether a sample should be retained or discarded. The most basic sampling strategy is uniform sampling, which samples a pose with uniform probability from $\mathcal{C}$. Other strategies can be used to find samples near obstacles or in corridors by sampling an initial pose and then directing subsequent samples to probe a region. Good sampling strategies are not only useful in identifying regions of interest, but also in reducing the number of samples needed to represent $\mathcal{C}_{free}$, resulting in increased search efficiency. For further discussion of sampling strategies, we refer the reader to [23, 36].

4.2.2 PRM Issues and Considerations

The PRM algorithm is a compromise: it can overcome the intractability of complex planning problems by sacrificing exactness and optimality for improved efficiency through approximation. As such, there are issues inherent to this tradeoff that must be considered when applying the PRM. Some general considerations are listed below:

- *Search Efficiency*: The strength of the PRM lies in its ability to perform simple graph search through a complex space by using a compact representation of the space. Since the complexity of search algorithms grows exponentially in the number of states that must be considered, the efficiency of the query phase is strongly dependent on the number of samples needed to represent $\mathcal{C}_{free}$. For the PRM to be tractable, it is essential that the graph representation of $\mathcal{C}_{free}$ be sufficient, yet minimal.

- *Visibility of Free Space*: The configuration space $\mathcal{C}$ must have favorable visibility properties in order to find a simple and efficient solution. If $\mathcal{C}_{free}$ has limited visibility, it may require an unreasonable number of samples or complex sampling strategies to obtain a solution, if any. In [23], however, it is shown that in practice many motion planning problems do have suitable visibility properties, despite high algebraic complexity.

- *Quality of Sampling Strategies*: The quality of PRM solutions depends on the ability of sampling strategies to accurately capture the topology of free space, while limiting the number of unnecessary samples. While sampling strategies do not guarantee accurate coverage of the space, good sampling strategies decrease uncertainty in the quality of the generated graph. Thus, it is important for sampling strategies to allocate a higher density of samples to regions of free space with poorer visibility properties.

In general, these considerations reflect that the PRM technique is beneficial in problems where the free space can be represented by a roadmap that is small enough, yet complete enough, to successfully answer queries.

4.3 Extending the PRM to Kalman Belief Space

The quality of a plan strongly depends on a robot's ability to reliably execute the plan. While the Probabilistic Roadmap algorithm has proven successful in generating solutions to complicated planning problems, it does not consider the quality of these plans when faced with the uncertainty of real-world execution. As with other traditional planning algorithms, the PRM deals only with trajectories in state space, ignoring the error inherent in the robot's incomplete state knowledge which results in deviations from planned trajectories. This limitation can be overcome by generating plans in the belief space of the robot instead of the state space, which the robot cannot directly observe. In the same way that the PRM (Section 4.2) has been used to solve high dimensional motion planning problems that were previously considered intractable [18, 29], we seek to utilize the PRM methodology to overcome the complexity of the formal POMDP for belief space planning. In this section, we improve upon the general PRM by adapting the algorithm to incorporate probabilistic state estimates and generate robust plans in belief space.

4.3.1 Formulation of Belief Space Planning

While the PRM algorithm itself is not robust to uncertainty, it provides a general framework and intuition for approaches in belief space. In this section, we show that by constraining the robot to realizable states along the PRM state trajectories, the size of the corresponding belief space can also be drastically reduced with a set of suitable assumptions.

To highlight the key issues involved in this problem, we begin by formally stating the belief space planning problem.

Problem Formulation

The traditional planning problem requires finding a path from an initial state x_{start} to a goal state x_{goal}. However, since the robot cannot observe the actual state of the world, our task is to plan in the space of its beliefs, which is denoted $\mathcal{B}$. The belief space $\mathcal{B}$ consists of all possible belief distributions b, which represent probability densities over states $b = p(x)$.

We assume the robot begins with a known initial belief b_{start}, and seeks a path to a belief in the goal subspace $\mathcal{G}$, which denotes the set of belief distributions whose maximum-likelihood state is the goal state $\mathcal{G} \triangleq \{b|E[b] = x_{goal}\}$. Each path is a series of belief actions that map the current belief distribution to a posterior distribution in belief space. In motion planning, each belief action corresponds to executing some motion control, and obtaining some observation to realize a posterior belief.

Our task is to find the path through belief space from b_{start} that results in a posterior belief distribution b_{goal} within the goal subspace $b_{goal} \in G$ with the minimum expected cost, where the cost objective function J is given as

$$J(b_0, u_0, \ldots, b_T, u_T) = \sum_{t=0}^{T} C(b_t, u_t) + D(b_T), \tag{4.1}$$

where $J(\ldots)$ is the cost of a path, $C(b_t, u_t)$ is the cost of executing control u_t from belief b_t, and $D(b_T)$ is the cost associated with the resulting uncertainty of the goal belief b_T.

Preliminary Discussion

In the general POMDP model of this problem, the graph of potential belief trajectories is too large to admit a tractable search. For each state trajectory, the corresponding belief trajectories must represent the continuum of posterior belief distributions that result from stochastic control outcomes and incomplete observations. In a discretized version of the general POMDP, a search tree with nodes at each state branches in the number of possible control outcomes and the number of possible measurements. In other words, the planner must consider performing each of N controls $u_{1:N}$ from a given state x. Executing a specific control u_i could result in a set of M stochastic outcomes $\{u_i^{[1]}, \ldots, u_i^{[M]}\}$, each with an associated probability, and therefore M possible resulting states $\{x_i^{[1]}, ..., x_i^{[M]}\}$. Similarly, for each state $x_i^{[j]}$, the observation $z_i^{[j][k]}$ obtained could be any of a set of L noisy observations $\{z_i^{[j][1]}, \ldots, z_i^{[j][L]}\}$ with corresponding probabilities. The effective branching factor of the corresponding search tree would then be $N \cdot M \cdot L$.

We can overcome the intractability of the POMDP, however, by using a PRM-based approach with a suitable choice of belief representation and specific assumptions about the controls and observations during planning. The intuition behind the PRM can be extended in a straightforward manner to reduce the size of the belief space. The set of beliefs that is actually realizable by the robot is the subset of belief distributions whose maximum likelihood state is in free space $\mathcal{B}_{free} \triangleq \{b | E[b] \in C_{free}\}$, where C_{free} corresponds to robot configurations (states) in free space.

This suggests that, as in the standard PRM, one could sample from $\mathcal{C}_{free}$ to generate potential trajectory arcs in $\mathcal{B}_{free}$; however, in belief space such trajectories would only capture the topology of the first-order moment of belief distributions. One could attempt to sample the expected uncertainty as well, but it is not clear that there exists a suitable sampling source or probability measure to account for higher order moments of the belief distribution. Even if one were to sample complete belief distributions (perhaps by uniform sampling), it would then be necessary to compute the visibility between belief distributions. This is problematic in belief space, because without a suitable sampling source the notion of visibility is ill-defined. In the next section, we show that a reasonable alternative can be found through compromise; by parting with the PRM's discrete representation of the trajectory space and standard search algorithm for a hybridized approach.

General Form of Solutions: Hybrid PRM

Since this problem is posed as a search over belief trajectories, recursive filtering techniques can be used to compute posterior beliefs in lieu of explicitly computing the visibility between belief distributions. This suggests that we can use the PRM in belief space by formally noting the correspondence between the state space $\mathcal{C}$ and the mean-belief subspace $\mathcal{B}^{\mu}$ of the belief space $\mathcal{B}$. Thus, we can use the pre-processing phase of the PRM to generate a discrete graph of trajectories through $\mathcal{B}^{\mu}_{free} \equiv \mathcal{C}_{free}$, reducing the size of the belief space representation with the mean-beliefs constrained to discrete values. The remainder of the belief distribution is left unconstrained and continuous, forming a hybrid search space[1].

The problem with this formulation, however, is that we have yet to show that mean-belief trajectories in $\mathcal{B}^{\mu}_{free}$ ensure visibility in the belief space $\mathcal{B}$. In fact, in general they do not; however, if we use a suitable class of belief distributions whose corresponding filter admits a linear update of the mean, then straight-line trajectories in $\mathcal{B}^{\mu}_{free}$ become well-defined.

Thus, a hybrid version of the PRM can be used to approach belief space planning, as follows:

- **Pre-processing phase**: A simplified representation of the belief space is generated as a hybrid of discrete mean-belief trajectories and continuous belief distribution space. Mean-belief samples have well-defined straight-line visibility *only* if the belief distribution admits linear filter updates of the mean.

- **Query phase**: The query phase consists of a corresponding search-filter hybrid process: a graph search is performed over mean-belief trajectories; and recursive filter updates analytically compute the belief distribution along the trajectory.

While at first glance this representation is not as sparse as in the standard PRM, note that only mean-belief trajectories are used for explicit search. There is, however, and added cost: the efficiency of search is now constrained by the computational load of simultaneous recursive filtering.

As we have stated, in general recursive filtering algorithms do not admit a linear mean update. However, in the next section we show that by representing beliefs in a linear-Gaussian system with a specific assumption about observations, Kalman filter updates take on a suitable linear form[2].

4.3.2 Predicting Trajectory Uncertainty with the Kalman Filter

The Gaussian distribution is a natural choice for the belief representation in this problem for a number of reasons. First, the normal distribution is completely characterized by its

[1]The hybrid space representation makes intuitive sense, for the only assumption we have made to limit the scope of the problem is that $\mathcal{B}_{free}$ consists of the subset of $\mathcal{B}$ whose maximum likelihood is in $\mathcal{C}_{free}$. $\mathcal{B}_{free}$ is not constrained in any other way, meaning that the remainder of the corresponding belief distributions, and all moments other than the mean, can take on any value.

[2]In the next chapter we show that, by factoring the covariance matrix, that Kalman filter also provides linear updates in the numerator and denominator of this matrix fraction. This has further implications in the context of the general solution form developed in this section.

first and second moments, providing a simple representation of the parameters of interest in this problem. The mean is used to identify PRM trajectories and the covariance as a measure of uncertainty for decision-making during planning. Additionally, the domain of applications can be easily extended to allow well-behaved non-linear systems with the EKF. Most importantly, Gaussian beliefs allow the evolution of uncertainty to be captured in closed form with Kalman filter updates, which have advantageous mathematical properties. In this section we show that the Kalman filter mean update can be linearized with suitable assumptions, enabling the hybrid PRM approach to be used in a principled manner.

Linear Mean Update

To apply the Kalman filter to belief trajectory search, we must first make specific assumptions about the controls and observations during planning. We assume that the controller can adhere to a given trajectory, executing motions that drive the mean of the belief distribution to specified points. This is not to say that controls are deterministic; rather, the stochastic nature of controls is accounted for by Kalman filter covariance updates. The transition function of the motion model $g(u_t, \mu_{t-1})$ is typically decomposed into a series of rigid transformations and, with the given assumption, the control update of the mean vector (line 1 of Algorithm 3) can be written linearly using homogeneous coordinates, such that

$$\begin{bmatrix} \overline{\mu}_t \\ 1 \end{bmatrix} = \begin{bmatrix} g(u_t, \mu_{t-1}) \\ 1 \end{bmatrix} = G_t(u_t) \begin{bmatrix} \mu_{t-1} \\ 1 \end{bmatrix}. \tag{4.2}$$

We also assume that the observation received at a given belief is the maximum-likelihood observation,

$$z_t = h(\overline{\mu}_t),$$

which results in a simple measurement update (line 4 of Algorithm 3) of the mean vector as

$$\begin{aligned} \mu_t &= \overline{\mu}_t + K_t(h(\overline{\mu}_t) - h(\overline{\mu}_t)) \\ &= \overline{\mu}_t. \end{aligned}$$

While this assumption is clearly an approximation, it allows us to predict the expected information gain from likely observations in different regions of the environment.

Algorithm 7 Multi-Step Covariance Update Along Trajectory procedure.

```
Require: Start covariance $\Sigma$, edge trajectory $e_{\mu,\mu'}$ and step length $\epsilon$
Ensure: Posterior covariance $\Sigma'$
    COMPUTE_MULTISTEP($\Sigma, e_{\mu,\mu'}, \epsilon$)
 1: Initialize $\Sigma' = \Sigma$
 2: for $\mu_i = \mu$ to $\mu'$ step $\epsilon_\mu = \epsilon \cdot \delta e_{\mu,\mu'}$ do
 3:     $\overline{\Sigma}_i = G_i \Sigma' G_i^T + R_i$
 4:     $\Sigma' = \overline{\Sigma}_i - \overline{\Sigma}_i H_i^T (H_i \overline{\Sigma}_i H_i^T + Q_i)^{-1} H_i \overline{\Sigma}_i$
 5: end for
 6: return $\Sigma'$
```

The implication of this linear form of the mean update is very important; it means that linear trajectories through the mean-belief subspace have well-defined visibility properties, and the hybrid PRM formulation can be used. In other words, these assumptions restrict the mean of our belief distribution exclusively to states along the trajectory, while allowing the uncertainty of controls and measurements to be incorporated in closed form by Kalman filter covariance updates. In this formulation, the complexity of searching is thus greatly reduced, for we avoid branching in belief space.

Covariance Propagation Along Trajectories

Given a linear form of the Kalman filter mean update, it is possible to approximate the evolution of expected belief uncertainty along a trajectory by performing covariance updates at N discrete points $\{\mu_1, \ldots, \mu_N\}$ along the trajectory. Beginning with an initial covariance Σ_0 at the start of the trajectory, filter updates are performed by simulating robot motion u_t and observations z_t along the trajectory as prescribed by the assumptions in the mean update. For each point along the trajectory, a filter iteration is computed by performing separate control and measurement updates.

The EKF control update incorporates the state transition dynamics $g(u_t, \mu_{t-1})$ and process noise R_t corresponding to the motion u_t of the robot from the previous trajectory point. This is computed as in line 2 of Algorithm 3, which is restated here

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t, \tag{4.3}$$

where G_t is the state transition function $g(u_t, \mu_{t-1})$ linearized about the mean state μ_t and R_t is the process noise of the motion model linearized about the simulated control u_t.

Algorithm 8 Build Belief Trajectory Graph procedure.

```
Require: Map $\mathcal{C} \equiv \mathcal{B}^{\mu}$
Ensure: Belief trajectory graph $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}\}$ with belief node set $\{n_i\}$ and edge trajectory set $\{e_{ij}\}$
    BUILD_TRAJECTORY_GRAPH ($\mathcal{B}^{\mu}$)
1: Sample poses $\{\mu_i\}$ from $\mathcal{B}^{\mu}_{free}$ using a standard PRM sampling strategy.
2: Build graph node set $n_i = \{\mu = \mu_i, \Sigma = \emptyset\}$
3: Create edge set $\{e_{ij}\}$ between nodes $(n_i, n_j)$ if the straight-line path between $(n_i[\mu], n_j[\mu])$ is collision-free
4: return $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}\}$
```

Expected sensor measurements z_t are computed at each point along the trajectory as the maximum likelihood observation. In the case of UWB sensors, range measurements are simulated from the mean robot pose to the known sensor locations. The measurement update incorporates the expected uncertainty of these range measurements into the belief distribution. The measurement covariance update is computed from lines 3 and 5 of Algorithm 3, which is given as

$$\Sigma_t = \overline{\Sigma}_t - \overline{\Sigma}_t H_t^T (H_t \overline{\Sigma}_t H_t^T + Q_t)^{-1} H_t \overline{\Sigma}_t, \tag{4.4}$$

where H_t is the measurement function linearized about the mean state $\overline{\mu}_t$ and Q_t is the expected process noise of the maximum likelihood observation z_t.

The expected belief uncertainty Σ_N at the end of a trajectory is the result of propagating the initial uncertainty Σ_0 through multiple filter updates at discretized positions along the trajectory, as shown in Algorithm 7.

4.3.3 Planning with Trajectory Search

It is now possible to implement the hybrid PRM formulation shown in Section 4.3.1 for the Kalman belief space by using the EKF variant developed in Section 4.3.2. In this section, we present the resulting trajectory search algorithm for a linear-Gaussian system. This algorithm consists of a build phase and a query phase, which are presented in turn.

Belief Trajectory Graph Build Process

We have shown in Section 4.3.2 that linear EKF mean updates admit well-defined straight-line trajectories through the mean-belief subspace $\mathcal{B}^u$ of the Kalman belief space

$\mathcal{B}$. Therefore, it is possible to generate a mean-belief trajectory graph by implementing the pre-processing phase of the hybrid PRM for a linear-Gaussian system. In Algorithm 8, we show the trajectory graph build process which creates a hybrid representation of the Kalman belief space.

This BUILD_TRAJECTORY_GRAPH routine is very similar to the standard PRM pre-processing phase, noting again the equivalence between the configuration space $\mathcal{C}$ and the mean-belief subspace $\mathcal{B}^u$. As in the PRM, poses μ_i are sampled from free space $\mathcal{B}^{\mu}_{free}$ according to a sampling strategy in line 1. A trajectory graph is then formed by creating straight-line edges e_{ij} between poses, which is shown in line 3. However, there are some important distinctions to be made. First, the node structure has been adapted to the Kalman belief space, as shown in line 2. Instead of using only poses μ_i to represent nodes in the graph, each belief node n_i also includes a covariance matrix $n_i[\Sigma]$. In accordance with the hybridized belief space representation, the mean of belief nodes $n_i[\mu]$ is assigned a discrete value while the covariance is continuous and initially unconstrained $n_i[\Sigma] = \emptyset$. Similarly, an edge e_{ij} between nodes n_i and n_j is a well-defined straight-line trajectory between mean-beliefs $n_i[\mu]$ and $n_j[\mu]$, while the corresponding covariance evolution between these nodes is unconstrained.

After building a node set and edge set in lines 1-3, the resulting belief trajectory graph structure $\mathcal{G}$ is returned at the end of the build process in line 4.

Belief Trajectory Graph Search Process

In the general PRM, the query phase consists of a standard algorithm for computing the shortest path in a state visibility graph. Since planning in belief space takes the form of the hybrid PRM shown in Section 4.3.1, our query phase now involves a hybrid search process over discrete mean-belief trajectories, while analytically computing uncertainty updates along these trajectories. With the ability to predict uncertainty over mean-belief trajectories in Kalman belief space (Section 4.3.2), it is possible to implement the search-filter hybrid process using EKF covariance updates when branching during graph search. The search can find paths of minimal uncertainty by making decisions based on the cumulative uncertainty of different paths.

A simple implementation of this search process is shown in Algorithm 9, which takes the form of a uniform cost (UC) search through the trajectory graph. The primary modification

Algorithm 9 The Uniform-Cost Belief Trajectory Search algorithm.

```
Require: Start belief $(\mu_0, \Sigma_0)$, goal location $\mu_{goal}$, map $\mathcal{B}^\mu$ and edge step size $\epsilon$
 1: $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}\} \leftarrow$ BUILD_TRAJECTORY_GRAPH $(\mathcal{B}^\mu)$
 2: Append $\mathcal{G}$ with nodes $\{n_0, n_{goal}\}$ and edges from start node $\{e_{0,j}\}$ and to goal node
    $\{e_{i,goal}\}$
 3: Augment node structure with best parent node $n^p = \emptyset$, such that $n_i = \{\mu, \Sigma, n^p\}$
 4: Create search queue with initial position and covariance $Q \leftarrow n_0 = \{\mu_0, \Sigma_0, \emptyset\}$
 5: while $Q$ is not empty do
 6:   Pop $n \leftarrow Q$
 7:   if $n = n_{goal}$ then
 8:     Continue
 9:   end if
10:   for all $n'$ such that $\exists e_{n,n'}$ and not IN_PATH($n$, $n_0$, $n'$) do
11:     $\Sigma' \leftarrow$ COMPUTE_MULTISTEP($n[\Sigma], e_{n,n'}, \epsilon$)
12:     if $tr(\Sigma') < tr(n'[\Sigma])$ then
13:       $n' = \{n'[\mu], \Sigma', n\}$
14:       Push $n' \rightarrow Q$
15:     end if
16:   end for
17: end while
```

to the standard UC algorithm is that the notion of *cost* is no longer cumulative distance, but rather expected uncertainty. We compute this cost using a standard measure of uncertainty, the trace of the covariance matrix $tr(\Sigma)$. The trajectory search algorithm is modified from the standard UC search accordingly:

- The best cost at each node $n_i[\Sigma]$ and the search state cost $n[\Sigma] \in Q$ are covariance matrices.

- Where a cumulative distance calculation would normally take place during node expansion, this search process propagates uncertainty along the corresponding trajectory to estimate cumulative uncertainty. This is shown in line 11, where the COMPUTE_MULTISTEP procedure (Algorithm 7) is called to compute a series of EKF covariance updates along a trajectory.

- The node relaxation step in lines 12-15 similarly consists of a comparison between the current search state cost and the best cost achieved thusfar. If the trace of the search state covariance is less that the best cost at a given node (i.e., if the uncertainty of the current search path is *better* than the best found so far), the node is updated with the lower covariance. Also, this node is then pushed back onto the queue to propagate this new covariance to neighboring nodes.

Algorithm 10 IN_PATH Boolean function.

```
Require: End node $n$, start node $n_0$ and target node $n'$
Ensure: Return true if target node $n'$ is in path defined by best parent structure between
    $\{n[n^p], \ldots, n_0\}$, otherwise return false
    IN_PATH($n, n_0, n'$)
1: while $n \neq n_0$ do
2:   $n \leftarrow n[n^p]$
3:   if $n = n'$ then
4:     return true
5:   end if
6: end while
7: return false
```

There are several points that should be addressed in regards to this algorithm. First, in line 3, the node structure of the trajectory graph is augmented with a best parent node n^p, which is used to maintain a best-path policy. The best parent node $n_i[n^p]$ of node n_i corresponds to the neighboring node n_j whose outgoing edge e_{ji} resulted in the minimum achievable covariance $n_i[\Sigma]$ at node n_i. At the end of the search process, the best path to the goal node can be found as the *reverse* policy in the best parent $n_{goal}[n^p]$ configuration of the search graph state. In other words, the best path policy can be obtained by starting at the goal node n_{goal} and recursively following the best parent nodes $n[n^p]$ until reaching the start node n_0.

Second, the trajectory search algorithm assumes a queue function Q that orders the nodes in a first-in, first-out manner. Third, the UC search requires a boolean IN_PATH function (used in line 10) which acts as a visited list to prevent loops during search. A neighbor node n' should not be considered during the expansion of node n if it is already a member of the best path to n. An example IN_PATH procedure is shown in Algorithm 10, which searches the best parent configuration from an end node n to the start node n_0 in the graph to see if a target node n' is already in the current best path. This is a valid concern because unlike cumulative distance searches, where the cost grows monotonically during search, covariance updates may cause the uncertainty to increase or decrease as the search progresses. This could result in a cyclic path, which although might be desirable in some circumstances, we omit in our work. Fourth, it is assumed that when a node n' is improved and pushed onto the queue in lines 13-14, it replaces any current member n' on the queue.

Fifth, it should be noted that the objective function J (shown in Section 4.3.3) is not specifically addressed in Algorithm 9. We do not explicitly include the cost per trajectory $C(\ldots)$, since this parameter is application-dependent. Algorithm 9 is intended to be general and could be easily adapted to include an additional distance cost, for example, in the search state at each node. Instead, we focus on the problem of minimizing the uncertainty at the goal node, which is associated with cost function $D(n_{goal}[\Sigma])$. Sixth, we assume a discretization for trajectories, specified by the step size ϵ, that approximates the actual motion of the robot. This should be determined for each specific application by approximating the typical operating characteristics of robot controls. Finally, while more efficient, heuristic-driven search processes may exist, we leave the construction of admissible heuristics for future work and focus on improving the efficiency of filter predictions.

4.3.4 Trajectory Search Issues

The method we developed in this chapter, which culminated in the trajectory search algorithm (Algorithm 9), presents a general solution to the problem of belief space planning. Although this algorithm is tractable, three key side-effects result from the multistep covariance update that impose efficiency limitations and scalability concerns:

The first issue is that branching during the hybrid search process requires performing *multiple* EKF updates to compute the posterior covariance prediction along a trajectory (line 11). Since these multistep covariance updates must be performed for each outgoing trajectory of the state node during each iteration of the search, a key strength of the general PRM algorithm is lost: its ability to answer queries online by performing search over the trajectory graph very efficiently.

The second issue is inherent in the mechanism for uncertainty propagation along trajectories. For a given trajectory, each novel initial covariance at the start node must be fully propagated through a multistep filter update to obtain the posterior covariance. This is problematic because each time a node is improved during the search process, the new "best cost" covariance is propagated to neighboring nodes, as seen in lines 12-15. This new initial condition requires a multistep covariance update to each neighboring node, meaning that the search process may be required to perform redundant multistep updates for a given trajectory.

Redundant updates also create an additional side-effect: they make re-planning inefficient. Although some applications of the PRM are designed to answer a single query, motion planning typically requires re-planning online to account for deviations from the predicted execution. This means that our algorithm should be tailored to handle multiple queries from the same PRM graph. Since trajectory updates must be recomputed for each novel covariance, each re-planning query process must start from scratch, performing a redundant search since the entire search process is completely dependent on the initial belief.

As can be seen, the current multistep method for covariance propagation induces a substantial computational penalty during the search process. Its sensitivity to initial conditions and the vast number of redundant computations that result motivate the development of an alternative.

4.4 Conclusion

We have presented a general method for robot motion planning under uncertainty, which takes the form of belief space planning using probabilistic state estimates. While solving the general formulation of this problem is computationally intractable, we show that by limiting the problem to a pertinent subspace, choosing a suitable belief distribution and making reasonable assumptions about controls and observations, it becomes possible to perform trajectory search in belief space.

Our solution takes the form of a PRM-Kalman filter hybrid. The pre-processing phase builds PRM trajectories in the mean-belief subspace of Gaussian distributions, which has favorable visibility properties. The query phase adapts a standard search algorithm to incorporate multistep EKF covariance updates along each branch of the search tree.

Although our trajectory search algorithm provides a tractable solution in belief space, this approach is limited by the computational costs of repeated multistep updates. This problem motivates the focus of the next chapter, which involves deriving an efficient one-step alternative to the redundant multi-step uncertainty update.

Chapter 5

One-Step Update for Trajectory Uncertainty

Contents

5.1 Introduction

In the last chapter, we showed that using Kalman filter predictions along trajectory arcs enables search within Gaussian belief space; however, this approach suffers from issues of scalability and computational feasibility due to the computational burden of performing redundant trajectory updates during search. This motivates deriving an alternate update to avoid the cost of computing repetitive multi-step Kalman filter updates over trajectories during searching and replanning.

In this chapter we focus on developing a one-step update to efficiently propagate uncertainty along a robot trajectory in a single computational step. Since we have shown that the mean of the belief distribution can be made to satisfy a linear Kalman filter update, in this problem we can assume it is predetermined and focus solely on the evolution of the second moment. We have also shown that Kalman filter covariance updates can applied in succession at discrete points to propagate an initial covariance along a trajectory and obtain the posterior covariance. At each point, we incorporate the uncertainty effects of robot motion, the state transition G_t and process noise $R_t \triangleq V_t W_t^{-1} V_t^T$, and the expected measurement information $M_t \triangleq H_t^T Q_t^{-1} H_t$. We will assume that these quantities are also predetermined for each of N time steps. With these assumptions, the problem we are solving can be formally stated as follows:

Given: an initial covariance Σ_0 and the mean vectors $\mu_{0:N}$, state transition matrices $G_{1:N}$, control uncertainty matrices $R_{1:N}$ and measurement information matrices $M_{1:N}$ for each of N time steps.

Find: posterior covariance $\Sigma_N = \mathit{onestep}(\Sigma_0, \mu_{0:N}, G_{1:N}, R_{1:N}, M_{1:N})$, where *onestep* is a simple computation that avoids explicitly computing the Kalman filter update at all time steps for each novel initial covariance Σ_0.

The solution to this problem is not straightforward, because the recursive covariance update in the Kalman filter is a non-linear function. Combining the control and measurement updates for one time step (lines 2, 3 and 5 of Algorithm 3), the full update in one filter iteration is given as

$$\begin{aligned}\Sigma_t &= (G_t\Sigma_{t-1}G_t^T + R_t) \\ &\quad -(G_t\Sigma_{t-1}G_t^T + R_t)H_t^T\left(H_t(G_t\Sigma_{t-1}G_t^T + R_t)H_t^T + Q_t\right)^{-1}H_t(G_t\Sigma_{t-1}G_t^T + R_t),\end{aligned}$$

or alternatively, by basing the recursion around the control posterior, as

$$\overline{\Sigma}_t = R_t + G_t\overline{\Sigma}_{t-1}\left(I + M_t\overline{\Sigma}_{t-1}\right)^{-1}G_t^T.$$

In either case, it is clear to see that the form of this equation is fractional and that for multiple recursive steps it becomes a sequence of nested non-linear equations. This does not immediately lend itself to solving for the posterior covariance as a simple, non-repetitive function of the initial covariance.

We will show in this chapter that the key to computing a one-step uncertainty update lies in functional composition of Kalman filter covariance updates. If each covariance update can be represented in a form that admits composition into a function of the same form, then multiple covariance updates can be represented as one update. In this chapter we present two such forms that admit functional composition of Kalman filter covariance updates. One is based on the Redheffer "star" product and provides an intuitive representation of individual KF updates and composition of multiple KF updates. The other method enables the non-linear Riccati equation corresponding to covariance updates to be decoupled into a linear system by factoring the covariance matrix as a matrix fraction. This is potentially very powerful because it linearizes the covariance update, resulting in a form of the Kalman filter that is completely linear.

The layout of this chapter is as follows: We begin in Section 5.2 by presenting an algebraic solution that is motivated by properties of the Kalman filter covariance updates. In Section 5.3 we provide an introduction to solutions motivated by the Riccati equation. We present a theoretical background of the Riccati equation, showing both the form of the equation and the two associated systems that pertain to the one-step problem. In Section 5.4 we show the explicit correspondence between Kalman filtering and the Riccati equation, and then describe the one-step solutions resulting from each characteristic system. Redheffer's "star product" and parallels to scattering theory, which yield a versatile solution for simultaneously solving the forward and backward onestep problem, are discussed in Section 5.4.1. A solution based on symplectic systems is presented in Section 5.4.3, which linearizes the Kalman filter update by factoring the covariance.

5.2 KF Interpretation and Algebraic Solution

In this section we show that the mathematical properties of the Kalman filter motivate an algebraic solution to the OS problem by factoring the covariance matrix. Each covariance update can be expressed in fractional form, and by decomposing the initial covariance matrix as a fraction, each update becomes a linear operator in the numerator and denominator of this fraction.

To demonstrate this, we rely on the following relation for matrix fractions:

$$AB^{-1} + C = AB^{-1} + CBB^{-1} = (A + CB)B^{-1}, \tag{5.1}$$

which can be interpreted, through a slight abuse of notation, as

$$\frac{A}{B} + C = \frac{A + CB}{B}.$$

To see how this can be applied to linearize EKF covariance updates, let us begin by assuming that the covariance matrix can be factored as a matrix fraction, such that

$$\Sigma = XY^{-1} \rightarrow \frac{X}{Y}, \tag{5.2}$$

which can be trivially attained using $X = \Sigma$ and $Y = I$. The control and measurement covariance updates can then be expressed as linear functions of X and Y, using Equation 5.1 and the dual Kalman and information forms of the filter. The steps of this derivation are shown in the following sections.

5.2.1 Linear Fractional Control Update

We begin with the EKF control update, restated here as

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t. \tag{5.3}$$

Applying the decomposition made in Equation 5.2, the control update in Equation 5.3 becomes

$$\overline{X}_t\overline{Y}_t^{-1} = G_t X_{t-1} Y_{t-1}^{-1} G_t^T + R_t. \tag{5.4}$$

By re-grouping terms on the right-hand side of Equation 5.4, we can see that it matches the form of Equation 5.1 as

$$G_t X_{t-1}(G_t^{-T} Y_{t-1})^{-1} + R_t \quad \equiv \quad AB^{-1} + C$$

with $A = G_t X_{t-1}$, $B = G_t^{-T} Y_{t-1}$ and $C = R_t$. Thus, applying Equation 5.1 gives us

$$\overline{X}_t\overline{Y}_t^{-1} = (G_t X_{t-1} + R_t G_t^{-T} Y_{t-1})(G_t^{-T} Y_{t-1})^{-1}, \tag{5.5}$$

which, in fractional form could be expressed as

$$\frac{\overline{X}_t}{\overline{Y}_t} = \frac{G_t X_{t-1} + R_t G_t^{-T} Y_{t-1}}{G_t^{-T} Y_{t-1}}.$$

Thus, Equation 5.5 demonstrates a control update $\Sigma_{t-1} \to \overline{\Sigma}_t$ that is linear in the numerator and denominator of the fraction as

$$\begin{bmatrix} \overline{X} \\ \overline{Y} \end{bmatrix}_t = \begin{bmatrix} G & RG^{-T} \\ 0 & G^{-T} \end{bmatrix}_t \begin{bmatrix} X \\ Y \end{bmatrix}_{t-1}. \tag{5.6}$$

We can thus capture the control update of the covariance at time t as a transfer function, denoted

$$T_t^C \triangleq \begin{bmatrix} G & RG^{-T} \\ 0 & G^{-T} \end{bmatrix}_t. \tag{5.7}$$

5.2.2 Linear Fractional Measurement Update

The measurement update can also be expressed as a linear fractional update by beginning with the information form of the update (line 2 of Algorithm 4), which is given as

$$\Omega_t \quad = \quad \overline{\Omega}_t + M_t. \tag{5.8}$$

Applying the decomposition of Equation 5.2, noting that $\Omega = \Sigma^{-1} = (XY^{-1})^{-1}$, we can express Equation 5.8 as the *inverse* of the desired fractional form in Equation 5.1, such that

$$X_t Y_t^{-1} \;=\; \left(\overline{Y}_t \overline{X}_t^{-1} + M_t\right)^{-1} \equiv \left(AB^{-1} + C\right)^{-1},$$

where $A = \overline{Y}_t$, $B = \overline{X}_t$ and $C = M_t$. Using these values and the relation in Equation 5.1, we obtain

$$\begin{aligned} X_t Y_t^{-1} &= \left((\overline{Y}_t + M_t \overline{X}_t)\overline{X}_t^{-1}\right)^{-1} \\ &= \overline{X}_t (\overline{Y}_t + M_t \overline{X}_t)^{-1}. \end{aligned} \tag{5.9}$$

The fractional interpretation of this update can be expressed as

$$\frac{X_t}{Y_t} = \frac{\overline{X}_t}{\overline{Y}_t + M_t \overline{X}_t},$$

which represents a linear update in the numerator and denominator of the fraction. The linear system corresponding to Equation 5.9 is given as

$$\begin{bmatrix} X \\ Y \end{bmatrix}_t = \begin{bmatrix} I & 0 \\ M & I \end{bmatrix}_t \begin{bmatrix} \overline{X} \\ \overline{Y} \end{bmatrix}_t . \tag{5.10}$$

Thus, we can represent the measurement covariance update at time t as a transfer function, denoted

$$T_t^M \triangleq \begin{bmatrix} I & 0 \\ M & I \end{bmatrix}_t . \tag{5.11}$$

5.2.3 Complete Linearized Covariance Update

Given the linear forms of the control and measurement covariance updates derived above, we can represent the update of one full iteration of the filter as a linear system by combining

Equations 5.6 and 5.10, resulting in

$$\begin{bmatrix} X \\ Y \end{bmatrix}_t = \begin{bmatrix} I & 0 \\ M & I \end{bmatrix}_t \begin{bmatrix} G & RG^{-T} \\ 0 & G^{-T} \end{bmatrix}_t \begin{bmatrix} X \\ Y \end{bmatrix}_{t-1} \tag{5.12}$$

$$= \begin{bmatrix} G & RG^{-T} \\ MG & MRG^{-T} + G^{-T} \end{bmatrix}_t \begin{bmatrix} X \\ Y \end{bmatrix}_{t-1}, \tag{5.13}$$

where we now have a linear matrix operator for the filter update at time t as

$$T_t \triangleq T_t^M \cdot T_t^C = \begin{bmatrix} G & RG^{-T} \\ MG & MRG^{-T} + G^{-T} \end{bmatrix}_t. \tag{5.14}$$

5.2.4 One-Step Covariance Propagation

With a linear update in the factored form of the covariance, it is now possible to build a one-step update matrix that exactly represents multiple filter updates. To compute the one-step matrix for N updates, $T^\star$, the individual updates for each time step can be combined by matrix multiplication as follows:

$$\begin{bmatrix} X \\ Y \end{bmatrix}_N = \underbrace{T_N \cdot T_{N-1} \cdots T_2 \cdot T_1}_{T^\star} \cdot \begin{bmatrix} X \\ Y \end{bmatrix}_0. \tag{5.15}$$

The resulting one-step representation is a block matrix of the form

$$T^\star \triangleq \begin{bmatrix} T_{11}^\star & T_{12}^\star \\ T_{21}^\star & T_{22}^\star . \end{bmatrix} \tag{5.16}$$

With this update matrix, it is now possible to compute the posterior covariance Σ_N that results from performing multiple filter updates from an initial covariance Σ_0. The first step is to factor the initial covariance as a matrix fraction $\Sigma_0 = X_0 Y_0^{-1}$, which can be trivially fulfilled by setting $X_0 = \Sigma_0$ and $Y_0 = I$. Using this initial condition and the linear equations for X_N and Y_N shown in Equation 5.13, the posterior covariance can be computed as

$$\Sigma_N = X_N Y_N^{-1} = (T_{11}^\star \Sigma_0 + T_{12}^\star)(T_{21}^\star \Sigma_0 + T_{22}^\star)^{-1}. \tag{5.17}$$

5.2.5 A Note Concerning Initial Conditions

We have not been very rigorous in motivating our choice of initial conditions. To provide a more formal justification, we show that our assumed initial condition $\Sigma_0 = X_0 Y_0^{-1} = \Sigma_0 I^{-1}$ is an achievable result of performing a *boundary update* T_0 from each of two possible boundary conditions (which we will denote with a minus subscript as $\Sigma_- = \frac{X_-}{Y_-}$, or equivalently $\Omega_- = \frac{Y_-}{X_-}$). The first boundary condition we will consider is that of infinite uncertainty, or equivalently zero information. The second is that of zero uncertainty, or equivalently infinite information. The linear system corresponding to the boundary update is given as

$$\begin{bmatrix} X \\ Y \end{bmatrix}_0 = \begin{bmatrix} G & RG^{-T} \\ MG & MRG^{-T} + G^{-T} \end{bmatrix}_0 \begin{bmatrix} X \\ Y \end{bmatrix}_- \tag{5.18}$$

We consider each boundary condition in turn, by solving the system in Equation 5.18 and imposing the constraint $\Sigma_0 = X_0 Y_0^{-1}$.

Boundary Case: Infinite Uncertainty, Zero Information

The boundary condition of infinite uncertainty $\Sigma_- = \infty$, or zero information $\Omega_- = 0$, corresponds to the case where $Y_- = 0$, which is shown as follows:

$$\Sigma_- = \frac{X_-}{Y_-} = \frac{X_-}{0} = \infty, \qquad \Omega_- = \frac{Y_-}{X_-} = \frac{0}{X_-} = 0, \qquad X_- \neq 0.$$

Using Equation 5.18, the covariance factors are written as

$$X_0 = G_0 X_- + 0 = G_0 X_- \tag{5.19}$$

$$Y_0 = M_0 G_0 X_- + 0 = M_0 G_0 X_-. \tag{5.20}$$

Solving for the initial condition using Equations 5.19-5.20, we obtain,

$$X_0 Y_0^{-1} = G_0 X_- \cdot X_-^{-1} G_0^{-1} M_0^{-1} = M_0^{-1},$$

which implies the following constraint:

$$\Sigma_0 = M_0^{-1}. \tag{5.21}$$

By denoting $\mathcal{A} = G_0 X_-$ and applying the constraint in Equation 5.21 to Equations 5.19-5.20, we obtain the following solution set:

$$\begin{bmatrix} X \\ Y \end{bmatrix}_0 = \begin{bmatrix} \mathcal{A} \\ \Sigma_0^{-1}\mathcal{A} \end{bmatrix}, \quad \mathcal{A} \neq 0. \tag{5.22}$$

Note that our trivial initial condition of $X_0 = \Sigma_0$ and $Y_0 = I$ is valid with $\mathcal{A} = \Sigma_0$.

Note: The result shown above is intuitive when considering EKF/EIF control and measurement updates from the boundary condition of zero information $\Omega_- = 0$ and, equivalently, infinite uncertainty $\Sigma_- = \infty$. The EKF control update is irrelevant, since adding any process noise to infinite covariance results in infinity:

$$\overline{\Sigma}_0 = G_0 \Sigma_- G_0^T + R_0 = G_0 \infty G_0^T + R_0 = \infty. \tag{5.23}$$

The measurement update, however, shows that an update from this boundary condition corresponds to receiving a measurement $M_0 = \Omega_0$:

$$\Omega_0 = \Omega_- + M_0 = 0 + M_0 = M_0. \tag{5.24}$$

Thus, this boundary update is equivalent to beginning in a state with zero information and increasing certainty by incorporating a measurement of value $M_0 = \Omega_0 = \Sigma_0^{-1}$.

Boundary Case: Zero Uncertainty, Infinite Information

The boundary condition of zero uncertainty $\Sigma_- = 0$, or infinite information $\Omega_- = \infty$, corresponds to the case where $X_- = 0$, which is shown as follows:

$$\Sigma_- = \frac{X_-}{Y_-} = \frac{0}{Y_-} = 0, \qquad \Omega_- = \frac{Y_-}{X_-} = \frac{Y_-}{0} = \infty, \qquad Y_- \neq 0.$$

Plugging into the system equations (Equation 5.18), the covariance factors are written as

$$\begin{aligned} X_0 &= G_0 \cdot 0 + R_0 G_0^{-T} Y_- = R_0 G_0^{-T} Y_- & (5.25) \\ Y_0 &= M_0 G_0 \cdot 0 + (M_0 R_0 G_0^{-T} + G_0^{-T}) Y_- = (M_0 R_0 + I) G_0^{-T} Y_-. & (5.26) \end{aligned}$$

Computing the initial covariance corresponding to Equations 5.25-5.26 gives us

$$\begin{aligned} X_0 Y_0^{-1} &= R_0 G_0^{-T} Y_- \cdot Y_-^{-1} G_0^T (M_0 R_0 + I)^{-1} \\ &= R_0 (M_0 R_0 + I)^{-1}, \end{aligned}$$

implying the following constraint on R_0 and M_0:

$$\Sigma_0 = R_0 (M_0 R_0 + I)^{-1}. \tag{5.27}$$

We make the substitution $\mathcal{B} = G_0^{-T} Y_-$ for the free variables, and apply the constraint in Equation 5.27 to Equations 5.25-5.26, yielding the solution set:

$$\begin{bmatrix} X \\ Y \end{bmatrix}_0 = \begin{bmatrix} R_0 \mathcal{B} \\ (M_0 R_0 + I)\mathcal{B} \end{bmatrix}, \quad \mathcal{B} \neq 0, \quad \{R_0, M_0 | \Sigma_0 = R_0 (M_0 R_0 + I)^{-1}\}. \tag{5.28}$$

This result is also intuitive, although not as straightforward as the previous boundary update in which the control update had no effect. Due to the ordering of control and measurement updates, the initial covariance can result from a combination of both adding uncertainty R_0 to the boundary state of perfect information, and then subsequently adding measurement information M_0. It is for this reason that the constraint set is a function of both R_0 and M_0. However, our assumed initial condition of $X_0 = \Sigma_0$ and $Y_0 = I$ is the trivial result of adding only process noise $R_0 = \Sigma_0$ and zero measurement information $M_0 = 0$, with $\mathcal{B} = I$.

5.3 Riccati Equation Theory

The covariance update function of the Kalman filter obeys a form of the Riccati equation, which is a non-linear difference function with a broad theoretical background. An exploration of Riccati theory yields insight into the one-step filtering problem; in this section we develop the theoretical basis for two solutions to the one-step problem. We show that the discrete Riccati equation is associated with two characteristic systems, each of which allows us to capture the EKF covariance update by encoding it in matrix form. Both systems admit functional composition of these matrices, allowing us to take multiple update descriptor matrices and combine them into one descriptor matrix, thereby facilitating one-step solutions.

5.3.1 Characteristic Systems of the Riccati Equation

The mathematical roots of the Riccati equation stem from non-linear, ordinary differential equations and the equation comes in many forms that are applicable to both continuous and discrete problems. In our work we will be concerned with the *discrete Riccati difference equation* (DRDE), which takes the form

$$X_{t+1} = B + AX_t(I - CX_t)^{-1}D. \tag{5.29}$$

This difference equation computes the value of X_{t+1} at a new time step as a non-linear function of the previous value of X_t. We will refer to X_t as the *Riccati variable* and in our problem this will be the covariance matrix.

The DRDE is associated with two characteristic systems: one is called *Hamiltonian* and the other is called *symplectic* [1]. The corresponding matrices for these systems are defined as follows:

Definition of Hamiltonian and Symplectic Matrices

- We denote J as the $2n \times 2n$ matrix

  $$J = \begin{bmatrix} 0 & -I_n \\ I_n & 0 \end{bmatrix} \tag{5.30}$$

  and $J^T = J^{-1} = -J$.

- A $2n \times 2n$ block matrix $\mathcal{S} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ is called *Hamiltonian* if

  $$J\mathcal{S} = (J\mathcal{S})^T = -\mathcal{S}^T J, \tag{5.31}$$

  which implies that the $n \times n$ blocks B and C are *symmetric*, or *self-adjacent*.

- A $2n \times 2n$ matrix $\mathcal{T} = \begin{bmatrix} E & F \\ G & H \end{bmatrix}$ is called *symplectic* if

  $$\mathcal{T}^T J \mathcal{T} = J, \tag{5.32}$$

which implies that $\mathcal{T}$ is symplectic *iff* $\mathcal{T}$ is invertible and

$$\mathcal{T}^{-1} = J^T \mathcal{T}^T J = \begin{bmatrix} H^T & -F^T \\ -G^T & E^T \end{bmatrix}. \tag{5.33}$$

□

Hamiltonian System

The first characteristic system associated with the DRDE is a *Hamiltonian system* of the form

$$\begin{aligned} x_t &= Ax_{t-1} + By_t & (5.34) \\ y_{t-1} &= Cx_{t-1} + Dy_t. & (5.35) \end{aligned}$$

Written in matrix form, the equations become

$$\begin{bmatrix} x_t \\ y_{t-1} \end{bmatrix} = \underbrace{\begin{bmatrix} A & B \\ C & D \end{bmatrix}}_{\mathcal{S}_t} \begin{bmatrix} x_{t-1} \\ y_t \end{bmatrix}, \tag{5.36}$$

where $\mathcal{S}_t$ is a Hamiltonian matrix corresponding to the DRDE. As can be seen, the Hamiltonian system is a forward-backward system in time, similar to the *filtering* and *smoothing* problems in linear estimation, where x_t is filtered from x_{t-1} at the previous time step, and y_{t-1} is smoothed from y_t in the next time step.

Symplectic System

The DRDE is also associated with a *symplectic system* of the form:

$$\begin{bmatrix} Y \\ Z \end{bmatrix}_t = \underbrace{\begin{bmatrix} E & F \\ G & H \end{bmatrix}}_{\mathcal{T}_t} \cdot \begin{bmatrix} Y \\ Z \end{bmatrix}_{t-1}. \tag{5.37}$$

The symplectic formulation represents a *linear-fractional* or *Mobius* transfer function $\mathcal{F}(Y,Z)$, which is given as

$$\mathcal{F}(Y,Z) = (EY + FZ)(GY + HZ)^{-1} \rightarrow \frac{EY + FZ}{GY + HZ} \tag{5.38}$$

and defines a matrix fraction whose numerator and denominator are linear functions encoded in the symplectic matrix $\mathcal{T}$.

The linear-fractional nature of the DRDE can be exposed in the following series of manipulations:

$$\begin{aligned}
X_{t+1} &= B + AX_t(I - CX_t)^{-1}D \\
&= B + AX_t(D^{-1} - D^{-1}CX_t)^{-1} \\
&= \Big(B(D^{-1} - D^{-1}CX_t) + AX_t\Big)\Big(D^{-1} - D^{-1}CX_t\Big)^{-1} \\
&= \Big((A - BD^{-1}C)X_t + BD^{-1}\Big)\Big(-D^{-1}CX_t + D^{-1}\Big)^{-1} && (5.39) \\
&\equiv (\overline{A}X_t + \overline{B})(\overline{C}X_t + \overline{D})^{-1}, && (5.40)
\end{aligned}$$

where we have simplified notation with the substitution of $\overline{A}$, $\overline{B}$, $\overline{C}$ and $\overline{D}$ from Equation 5.39 to 5.40. Note that the system has been decoupled into a matrix fraction, where both the numerator and denominator are linear functions of X_t.

Since an initial condition X_0 will lead to a fractional result $X_1 = (\overline{A}X_0 + \overline{B})(\overline{C}X_0 + \overline{D})^{-1}$, this suggests factoring X_t as a matrix fraction, $X_t = Y_tZ_t^{-1}$. Substituting this fractional form of X_t into Equation 5.40 yields the linear system of Equations 5.37-5.38 in the following manner:

$$\begin{aligned}
X_{t+1} = Y_{t+1}Z_{t+1}^{-1} &= (\overline{A}Y_tZ_t^{-1} + \overline{B})(\overline{C}Y_tZ_t^{-1} + \overline{D})^{-1} \\
&= (\overline{A}Y_t + \overline{B}Z_t)Z_t \cdot Z_t^{-1}(\overline{C}Y_t + \overline{D}Z_t)^{-1} \\
&= (\overline{A}Y_t + \overline{B}Z_t)(\overline{C}Y_t + \overline{D}Z_t)^{-1} && (5.41) \\
&\equiv (EY_t + FZ_t)(GY_t + HZ_t)^{-1}. && (5.42)
\end{aligned}$$

Equation 5.41 indicates that factoring $X_t = Y_tZ_t^{-1}$ admits a well-defined linear-fractional update and satisfies a linear system in Y_t and Z_t as in Equation 5.37. Thus, we have shown that the DRDE corresponds to a linear symplectic system, which is given as

$$Y_{t+1} = EY_t + FZ_t \tag{5.43}$$

$$Z_{t+1} = GY_t + HZ_t, \tag{5.44}$$

where $X_t = Y_t Z_t^{-1}$ and $\{E, F, G, H\}$ are defined by Equation 5.39 (using the relations in Equations 5.40-5.42) as

$$\mathcal{T}_t = \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} A - BD^{-1}C & BD^{-1} \\ -D^{-1}C & D^{-1} \end{bmatrix}. \tag{5.45}$$

Hamiltonian-Symplectic Correspondence

Note that the symplectic system corresponds directly to the Hamiltonian system, which can be derived from Equations (5.34-5.35) by rewriting the Hamiltonian system as a forward system. We can rearrange the terms in Equation 5.35 as follows:

$$\begin{aligned} y_{t-1} &= Cx_{t-1} + Dy_t \\ Dy_t &= y_{t-1} - Cx_{t-1} \\ y_t &= D^{-1}y_{t-1} - D^{-1}Cx_{t-1}. \end{aligned} \tag{5.46}$$

Then, by plugging y_t from Equation 5.46 into Equation 5.34 we obtain

$$\begin{aligned} x_t &= Ax_{t-1} + By_t \\ &= Ax_{t-1} + B(D^{-1}y_{t-1} - D^{-1}Cx_{t-1}) \\ &= (A - BD^{-1}C)x_{t-1} + BD^{-1}y_{t-1}, \end{aligned} \tag{5.47}$$

which yields the same symplectic system shown in Equation 5.45

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \underbrace{\begin{bmatrix} A - BD^{-1}C & BD^{-1} \\ -D^{-1}C & D^{-1} \end{bmatrix}}_{\mathcal{T}_t} \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix}, \tag{5.48}$$

where $\mathcal{T}$ is a symplectic matrix associated with the DRDE (5.29).

Case of Hermitian Riccati Variable

When X_t is Hermitian, $D = A^T$ and B and C are also Hermitian. In this case, the Riccati equation is known as the Hermitian discrete Riccati difference equation (HDRDE), which is adapted from Equation 5.29 to become

$$X_{t+1} = B_t + A_t X_t (I - C_t X_t)^{-1} A_t^T. \tag{5.49}$$

In the Kalman filtering problem, the covariance matrix is real and symmetric, and therefore Hermitian. In this chapter we will be concerned with the Hamiltonian and symplectic descriptor matrices corresponding to the HDRDE, which are updated from Equations 5.36 and 5.48 to respectively become

$$\mathcal{S}_t = \begin{bmatrix} A & B \\ C & A^T \end{bmatrix}_t, \tag{5.50}$$

$$\mathcal{T}_t = \begin{bmatrix} E & F \\ G & H \end{bmatrix}_t = \begin{bmatrix} A - BA^{-T}C & BA^{-T} \\ -A^{-T}C & A^{-T} \end{bmatrix}_t. \tag{5.51}$$

For additional theoretical background regarding the Riccati equation, Hamiltonian systems and symplectic systems, we refer the reader to [1, 2].

5.3.2 Functional Composition of the Riccati Equation

We will now show that the Hamiltonian and symplectic matrix representations of the HDRDE (Equations 5.50-5.51) admit functional composition, which as we have stated is the key to solving the one-step filtering problem.

Hamiltonian Composition: Redheffer Star Product

Hamiltonian descriptor matrices can be composed by an operator known as the Redheffer "star" product (denoted with a '$\star$') [42] . We can formally derive this method of composition by starting with descriptor matrices of a Hamiltonian system at two adjacent timesteps as follows:

$$\begin{bmatrix} x_2 \\ y_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_1 \\ y_2 \end{bmatrix}, \tag{5.52}$$

$$\begin{bmatrix} x_3 \\ y_2 \end{bmatrix} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \begin{bmatrix} x_2 \\ y_3 \end{bmatrix}. \tag{5.53}$$

(Note: for clarity it should be stated that these two system matrices have the same block structure, where each block actually corresponds to a time-varying quantity.) Our goal is to determine the Hamiltonian matrix corresponding to the aggregate system that represents both Equations 5.52 and 5.53. We will show that the resulting system can be computed using the star product in the following manner:

$$\begin{bmatrix} x_3 \\ y_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \star \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \begin{bmatrix} x_1 \\ y_3 \end{bmatrix}. \tag{5.54}$$

Our explicit goal is to derive equations for the variables x_3 and y_1 in terms of x_1 and y_3 to determine the star product operation in Equation 5.54. We begin by writing the given equations from the systems in Equations 5.52 and 5.53, as follows:

$$\begin{aligned} x_2 &= Ax_1 + By_2 & (5.55) \\ y_1 &= Cx_1 + Dy_2 & (5.56) \\ x_3 &= Wx_2 + Xy_3 & (5.57) \\ y_2 &= Yx_2 + Zy_3. & (5.58) \end{aligned}$$

Substituting y_2 from Equation 5.58 into Equation 5.55 we solve for x_2 as follows:

$$\begin{aligned} x_2 &= Ax_1 + B(Yx_2 + Zy_3) \\ (I - BY)x_2 &= Ax_1 + BZy_3 \\ x_2 &= (I - BY)^{-1}Ax_1 + (I - BY)^{-1}BZy_3 & (5.59) \end{aligned}$$

We also substitute x_2 from Equation 5.55 into Equation 5.58 to solve for y_2 as follows:

$$\begin{aligned} y_2 &= Y(Ax_1 + By_2) + Zy_3 \\ (I - YB)y_2 &= YAx_1 + Zy_3 \\ y_2 &= (I - YB)^{-1}YAx_1 + (I - YB)^{-1}Zy_3. \end{aligned} \tag{5.60}$$

Now with x_2 and y_2 both written in terms of x_1 and y_3, it is possible to similarly solve for x_3 and y_1. To solve for x_3, we substitute x_2 from Equation 5.59 into Equation 5.57:

$$\begin{aligned} x_3 &= Wx_2 + Xy_3 \\ &= W\Big((I - BY)^{-1}Ax_1 + (I - BY)^{-1}BZy_3\Big) + Xy_3. \end{aligned}$$

which is simplified to give the desired result

$$\boxed{x_3 = W(I - BY)^{-1}Ax_1 + (X + W(I - BY)^{-1}BZ)y_3}. \tag{5.61}$$

We also substitute y_2 from Equation 5.60 into Equation 5.56 to solve for y_1 as follows:

$$\begin{aligned} y_1 &= Cx_1 + Dy_2 \\ &= Cx_1 + D(I - YB)^{-1}YAx_1 + D(I - YB)^{-1}Zy_3, \end{aligned}$$

which simplifies to become

$$\boxed{y_1 = (C + D(I - YB)^{-1}YA)x_1 + D(I - YB)^{-1}Zy_3}. \tag{5.62}$$

Now, with Equations 5.61 and 5.62, our solution is obtained as the aggregate system in Equation 5.54, which can now be written in terms of one matrix as

$$\begin{bmatrix} x_3 \\ y_1 \end{bmatrix} = \begin{bmatrix} W(I - BY)^{-1}A & X + W(I - BY)^{-1}BZ \\ C + D(I - YB)^{-1}YA & D(I - YB)^{-1}Z \end{bmatrix} \begin{bmatrix} x_1 \\ y_3 \end{bmatrix}, \tag{5.63}$$

where we have now derived the star product as a set of matrix block operators, shown formally in the following definition.

Definition of Star Product

The star product of two block matrices is a set of matrix block operators given as

$$\boxed{\begin{bmatrix} A & B \\ C & D \end{bmatrix} \star \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} = \begin{bmatrix} W(I-BY)^{-1}A & X + W(I-BY)^{-1}BZ \\ C + D(I-YB)^{-1}YA & D(I-YB)^{-1}Z \end{bmatrix}}. \tag{5.64}$$

Note: It is sometimes useful to employ an alternate form of the operators in the $1, 2$ and $2, 1$ entries of the resultant in Equation 5.64, which is derived by using the following relation:

$$(I - EF)^{-1}E = (E^{-1} - E^{-1}EF)^{-1} = E(I - FE)^{-1}. \tag{5.65}$$

Applying Equation 5.65 to the $1, 2$ and $2, 1$ entries of Equation 5.64 yields the following alternate forms:

$$X + W(I-BY)^{-1}BZ = X + WB(I-YB)^{-1}Z \tag{5.66}$$

$$C + D(I-YB)^{-1}YA = C + DY(I-BY)^{-1}A. \tag{5.67}$$

Symplectic Composition

Composition of symplectic descriptor matrices is given by a much simpler operator, matrix multiplication. This can be trivially demonstrated by noting that a forward symplectic system at two consecutive time steps

$$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

can be aggregated by matrix multiplication as follows:

$$\begin{bmatrix} x_3 \\ y_3 \end{bmatrix} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix} \cdot \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}. \tag{5.68}$$

5.4 Riccati-Based One-Step Update Solutions

With the theoretical background in place, we now show the explicit correspondence between Kalman filter covariance updates and the Riccati equation. A full iteration of the Kalman filter involves two update steps: a control update and measurement update. In mobile robot filtering applications, it is typically assumed that in a given time step, the control update precedes the measurement update. For the EKF update at time t, we denote the posterior covariance resulting from control and measurement updates as $\overline{\Sigma}_t$ and Σ_t, respectively. Note that for ease of readability, we will omit the time subscripts from all quantities other than the covariance Σ_t in the subsequent derivations; however, all quantities are assumed to be time-varying.

The EKF control update (line 2 of Algorithm 3) is given as

$$\overline{\Sigma}_t = G\Sigma_{t-1}G^T + R, \tag{5.69}$$

where $R \triangleq VWV^T$ denotes the process noise in state space. We will use two forms of the measurement update. The standard EKF form of the measurement update (line 5 of Algorithm 3) is given by

$$\Sigma_t = \overline{\Sigma}_t - \overline{\Sigma}_t H^T (H\overline{\Sigma}_t H^T + Q)^{-1} H\overline{\Sigma}_t. \tag{5.70}$$

We also use the covariance version of the EIF measurement update (line 2 of Algorithm 4) with $\Omega_t = \Sigma_t^{-1}$, which is given as

$$\Sigma_t = (\overline{\Sigma}_t^{-1} + M)^{-1}, \tag{5.71}$$

where $M \triangleq H^T Q^{-1} H$ denotes the measurement information in state space.

It is possible to formulate the Riccati equation corresponding to a full filter update in two ways. The KF recursion may be written to produce the posterior covariance directly following either the measurement Σ_t or control $\overline{\Sigma}_t$ update, as shown below.

Measurement-Based Recursion

The measurement-based recursion can be formed by plugging Equation 5.69 into Equation 5.70, which yields

$$\begin{aligned} \Sigma_{t+1} &= (G\Sigma_t G^T + R) \\ &\quad -(G\Sigma_t G^T + R)H^T \Big[H(G\Sigma_t G^T + R)H^T + Q\Big]^{-1} H(G\Sigma_t G^T + R). \end{aligned} \tag{5.72}$$

It requires a significant amount of work to show that the Kalman filter recursion based on the measurement update in Equation 5.72 obeys the Riccati equation form in Equation 5.49. For this reason, the result is stated here and the proof can be found in [2].

Equation 5.72 corresponds to the HDRDE (restated from Equation 5.49)

$$\Sigma_{t+1} = B + A\Sigma_t(I - C\Sigma_t)^{-1}A^T,$$

where

$$\begin{aligned} A &= G - RH^T(HRH^T + Q)^{-1}HG \\ B &= R - RH^T(HRH^T + Q)^{-1}HR \\ C &= G^T H^T(HRH^T + Q)^{-1}HG. \end{aligned}$$

Control-Based Recursion

Basing the recursion on the control update, $\overline{\Sigma}_t$ is produced by plugging Equation 5.71 into Equation 5.69, yielding the Riccati equation

$$\begin{aligned} \overline{\Sigma}_t &= R + G\left(\overline{\Sigma}_{t-1}^{-1} + M\right)^{-1}G_t^T \\ &= R + G\overline{\Sigma}_{t-1}\left(I + M\overline{\Sigma}_{t-1}\right)^{-1}G^T. \end{aligned} \tag{5.73}$$

The control recursion in Equation 5.73 has a straightforward correspondence to the HDRDE (Equation 5.49) with $A = G$, $B = R$, and $C = -M$.

Equivalence of Recursions

We are typically interested in the resulting filtered covariance for a given time step, which suggests basing the recursion on the measurement posterior Σ_t. However, for clarity in the subsequent derivations in this chapter, it is advantageous to base the covariance recursion on the control posterior $\overline{\Sigma}_t$ at each "half"-timestep, for algebraic reasons shown above.

It turns out that this is very reasonable, because the same basic results can be obtained using either form. This is possible the HDRDE is well-defined for both the individual control and measurement updates. We demonstrate this here, without proof, by showing the HDRDE form of each update.The measurement update (using Equation 5.71) corresponds to the HDRDE as

$$\boxed{\Sigma_t = 0 + I\overline{\Sigma}_t(I + M\overline{\Sigma}_t)^{-1}I} = (\overline{\Sigma}_t^{-1} + M)^{-1}, \tag{5.74}$$

with $A = I$, $B = 0$, $C = -M$, and the EKF/EIF measurement update shown on the right. The control update corresponds to the HDRDE as

$$\boxed{\overline{\Sigma}_t = R + G\Sigma_{t-1}(I - 0\Sigma_{t-1})^{-1}G_t^T} = G\Sigma_{t-1}G^T + R, \tag{5.75}$$

with $A = G$, $B = R$, $C = 0$, and the standard EKF control update is shown on the right.

5.4.1 Hamiltonian Filtering Formulation

In this section we demonstrate that the star product composition of Hamiltonian descriptor matrices shown in Section 5.3.2 can be used to aggregate Kalman filter covariance updates. Using the Kalman filter correspondence to the HDRDE provided in Equation 5.73, it is straightforward to build the Hamiltonian matrix that describes a Kalman filter update. We restate this explicitly as follows:

$$\begin{array}{rrcl} \text{HDRDE:} & X_t & = & B + AX_{t-1}(I - CX_{t-1})^{-1}A^T \\ \text{KF-HDRDE:} & \overline{\Sigma}_t & = & R + G\overline{\Sigma}_{t-1}(I + M\overline{\Sigma}_{t-1})^{-1}G^T. \end{array}$$

With $A = G$, $B = R$, and $C = -M$, it directly follows that the Hamiltonian matrix (Equation 5.50) corresponding to the HDRDE is given as

$$\mathcal{S}_t \quad = \quad \begin{bmatrix} A & B \\ C & A^T \end{bmatrix}_t = \begin{bmatrix} G & R \\ -M & G^T \end{bmatrix}_t .$$

The power of this representation of Kalman filter updates is that these matrices $\mathcal{S}_t$ encode the uncertainty effects of each update step at time t. Further, we showed in Section 5.3.2 that Hamiltonian descriptor matrices $\mathcal{S}_t$ at adjacent time steps can be aggregated using the star product. Therefore, the net EKF covariance update for all updates from time $t = 1$ to N can be captured by star-producing consecutive updates as

$$\mathcal{S}_{1:N} = \mathcal{S}_1 \star \mathcal{S}_2 \star \cdots \star \mathcal{S}_{N-1} \star \mathcal{S}_N. \tag{5.76}$$

We now have a method for aggregating EKF updates into the matrix $\mathcal{S}_{1:N}$, but for the moment we defer to an alternate interpretation. This process can be described much more intuitively with an analogy in the context of Redheffer's original work, scattering theory.

5.4.2 Scattering Theory Parallel

The Hamiltonian formulation of Kalman filter covariance updates provides a very intuitive parallel with *scattering/transmission-line theory*, which was developed in the 1940s through 1960s to study the propagation of waves through a medium. Stemming from the work of Redheffer [42], a scattering theory framework was developed for filtering by Kailath and Ljung in [33, 26, 52, 49], which we make use of in this section. The key to this analogy lies in the forward-backward Hamiltonian system associated with the HDRDE: in filtering, this corresponds to a system which simultaneously produces filtered and smoothed estimates; in scattering theory it is interpreted as waves traveling through a medium in opposite directions with forward and backward transmission and reflection operators, whose interactions are determined by the state space parameters $\{G, R, M\}$. Given these parameters for a set of consecutive scattering medium layers, or equivalently a set of consecutive Kalman filter updates, the descriptor matrices for each update can be combined using the star product to produce *one* descriptor matrix. This resulting descriptor matrix represents the aggregate scattering medium, or equivalently the aggregate filter update, which exactly captures the effects of all individual layers, or filter steps.

The underlying HDRDE for the scattering model is represented as follows:

$$\Sigma_t \quad = \quad \mathcal{P}_t + \Phi_t \Sigma_{t-1}(I - \mathcal{M}_t \Sigma_{t-1})^{-1} \Phi_t^T, \tag{5.77}$$

where at filtering time t (*or scattering layer* t)

$$\begin{aligned}
\Phi_t &= \text{state transition matrix } (\textit{or scattering transmission matrix}) \\
\mathcal{P}_t &= \text{error covariance } (\textit{or right reflection coefficient}) \\
\mathcal{M}_t &= \text{measurement information } (\textit{or left reflection coefficient})
\end{aligned}$$

and the associated Hamiltonian matrix (Equation 5.50) is called the *scattering matrix* and has the form:

$$\mathcal{S}_t = \begin{bmatrix} \Phi & \mathcal{P} \\ -\mathcal{M} & \Phi^T \end{bmatrix}_t = \begin{bmatrix} G & R \\ -M & G^T \end{bmatrix}_t . \tag{5.78}$$

Multiple layers can be composed using the star product. Here, through a slight abuse of notation, we explicitly demonstrate the star product operator by adding layer $\mathcal{S}_{t+1}$ to layer $\mathcal{S}_t$, computing $\mathcal{S}_{t:t+1} = \mathcal{S}_t \star \mathcal{S}_{t+1}$ as follows:

$$\begin{bmatrix} \Phi & \mathcal{P} \\ -\mathcal{M} & \Phi^T \end{bmatrix}_{t:t+1} = \begin{bmatrix} \Phi & \mathcal{P} \\ -\mathcal{M} & \Phi^T \end{bmatrix}_t \star \begin{bmatrix} G & R \\ -M & G^T \end{bmatrix}_{t+1} , \tag{5.79}$$

which yields the set of recursions (using Equations 5.64, 5.66 and 5.67):

$$\Phi_{t:t+1} = G_{t+1}(I + \mathcal{P}_t M_{t+1})^{-1}\Phi_t \tag{5.80}$$

$$\mathcal{P}_{t:t+1} = R_{t+1} + G_{t+1}\mathcal{P}_t(I + M_{t+1}\mathcal{P}_t)^{-1}G_{t+1}^T \tag{5.81}$$

$$\mathcal{M}_{t:t+1} = \mathcal{M}_t + \Phi_t^T M_{t+1}(I + \mathcal{P}_t M_{t+1})^{-1}\Phi_t. \tag{5.82}$$

Intuitive Individual Updates

It is useful to note that Hamiltonian descriptor matrices can be used for each individual measurement and control update. In Equations 5.74 and 5.75, we showed the HDRDE corresponding to measurement and control updates, respectively. These translate directly into descriptor matrices as

$$\mathcal{S}_t^M = \begin{bmatrix} I & 0 \\ -M & I \end{bmatrix}_t \quad \text{and} \quad \mathcal{S}_t^C = \begin{bmatrix} G & R \\ 0 & G^T \end{bmatrix}_t , \tag{5.83}$$

where $\mathcal{S}_t^C$ is a control update matrix and $\mathcal{S}_t^M$ is a measurement update matrix. The full update matrix for one iteration of the control-based recursion is then given as

$$\mathcal{S}_t = \mathcal{S}_t^M \star \mathcal{S}_t^C. \tag{5.84}$$

Initial Conditions

The initial conditions in this formulation are handled in similar fashion to our discussion in Section 5.2.5. The insight in applying the initial covariance is to create a *boundary layer*, which is a scattering layer that is attached to the *null* scattering layer.

First we consider the interpretation of a null layer, or equivalently a null filter update, which is denoted $\mathcal{S}_\emptyset$. In terms of wave propagation, this null layer can be likened to a vaccuum with no interaction effects: no reflections $\mathcal{P}_\emptyset = \mathcal{M}_\emptyset = 0$ and unity transmission $\Phi_\emptyset = I$. In filtering, a null filter update would correspond to control and measurement updates that have no effect on the covariance: there would be no process noise $R = 0$ or measurement information $M = 0$, and the state transition matrix would be unity $G = I$. The null layer would thus be represented by a descriptor matrix as follows:

$$\mathcal{S}_\emptyset = \begin{bmatrix} \Phi & \mathcal{P} \\ -\mathcal{M} & \Phi^T \end{bmatrix}_\emptyset = \begin{bmatrix} G & R \\ -M & G^T \end{bmatrix}_\emptyset = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}. \tag{5.85}$$

This is easily verified, as star-producing any matrix with the null scattering matrix results in the original matrix.

Based on the interpretation of the null layer, it is straightforward to see that the initial covariance Σ_0 could be represented as a null layer with additional process noise $R = \mathcal{P} = \Sigma_0$, or alternatively with additional measurement information $M = \mathcal{M} = \Sigma_0^{-1} = \Omega_0$. It could also be represented as a combination of R and M whose net contribution is Σ_0. These are identical to the cases derived in Section 5.2.5. The first two cases would be boundary layers described as

$$\mathcal{S}_0 = \begin{bmatrix} I & \Sigma_0 \\ 0 & I \end{bmatrix}, \quad \text{or} \quad \mathcal{S}_0 = \begin{bmatrix} I & 0 \\ -\Omega_0 & I \end{bmatrix}, \tag{5.86}$$

where this boundary layer can be attached to a scattering medium to impose an initial condition.

The scattering matrix interpretation is very intuitive, but this can also be shown formally by star-producing the initial condition matrix with the filter update descriptor matrix at $t = 1$. For algebraic simplicity in the derivation, we concern ourselves with the initial condition matrix $\mathcal{S}_0$ on the left in Equation 5.86, which gives us:

$$\mathcal{S}_0 \star \mathcal{S}_1 = \begin{bmatrix} I & \Sigma_0 \\ 0 & I \end{bmatrix} \star \begin{bmatrix} G_1 & R_1 \\ -M_1 & G_1^T \end{bmatrix} = \begin{bmatrix} - & \mathcal{P}_{0:1} \\ - & - \end{bmatrix}. \tag{5.87}$$

Note that at the moment we are only concerned with the 2, 1 entry of the resultant, which is the effective error covariance in state space of the combined filter update $S_{0:1}$. Using the recursion for this term $\mathcal{P}_{0:1}$, defined in Equation 5.81, we obtain

$$\mathcal{P}_{0:1} = R_1 + G_1\Sigma_0(I + M_1\Sigma_0)^{-1}G_1^T \qquad \Rightarrow \overline{\Sigma}_1. \tag{5.88}$$

This result is very important, because it is identical to computing $\overline{\Sigma}_1$ using the Kalman filter HDRDE in Equation 5.73. Equation 5.88 shows that by adding the boundary layer in Equation 5.73, the filtered covariance after the Kalman filter update $\{G_1, R_1, M_1\}$ is exactly computed as $\mathcal{P}_{0:1} = \overline{\Sigma}_1$. In general, this means that the state of uncertainty in the filter can be maintained as a descriptor matrix, and the posterior covariance of a subsequent filter update results from the star product block operator for $\mathcal{P}$.

Computing the Hamiltonian One-Step Update

As shown in Equation 5.76, the scattering matrices for multiple layers $t = 1$ to N can be combined by using the star product to generate the aggregate layer descriptor $\mathcal{S}_{1:N}$, which in general we will denote as $\mathcal{S}^\star$.

The key to the filtering interpretation of an aggregate filter descriptor $\mathcal{S}^\star$ lies in the associative property of the star product operation [42], which ensures that the star product of any scattering matrices yields another scattering matrix of the same form, such that

$$\mathcal{S}^\star = \begin{bmatrix} \Phi^\star & \mathcal{P}^\star \\ -\mathcal{M}^\star & {\Phi^\star}^T \end{bmatrix} = \begin{bmatrix} G^\star & R^\star \\ -M^\star & {G^\star}^T \end{bmatrix}. \tag{5.89}$$

The implication of this is powerful and extends beyond the scope of our desired one-step operator; not only will this enable us to compute a multi-step covariance update in one-step, but we also generate an *exact* description of the aggregate filter. We now have the aggregate state transition $G^\star$, process noise $R^\star$ and measurement information $M^\star$ for a trajectory, which is independent of initial conditions.

As shown in Equations 5.87-5.88, a one-step update of the covariance can be easily computed for a novel initial condition Σ_0. After $\mathcal{S}^*$ has been computed, the posterior covariance Σ is exactly computed as

$$\Sigma = R^\star + G^\star\Sigma_0(I + M^\star\Sigma_0)^{-1}{G^\star}^T. \tag{5.90}$$

5.4.3 Symplectic Filtering Formulation

It is straightforward to formulate the KF covariance update as symplectic system by once again recognizing the correspondence between the Kalman filter and HDRDE in Equation 5.73, where $A = G$, $B = R$, and $C = -M$. The symplectic descriptor matrix for the HDRDE (shown in Equation 5.51) is given as

$$\mathcal{T}_t = \begin{bmatrix} A - BA^{-T}C & BA^{-T} \\ -A^{-T}C & A^{-T} \end{bmatrix}_t = \begin{bmatrix} G + RG^{-T}M & RG^{-T} \\ G^{-T}M & G^{-T} \end{bmatrix}_t,$$

which represents a forward linear system shown in Equation 5.45 as

$$\begin{bmatrix} Y \\ Z \end{bmatrix}_t = \underbrace{\begin{bmatrix} G + RG^{-T}M & RG^{-T} \\ G^{-T}M & G^{-T} \end{bmatrix}}_{\mathcal{T}_t} \cdot \begin{bmatrix} Y \\ Z \end{bmatrix}_{t-1}, \tag{5.91}$$

where Σ_t is decomposed as the matrix fraction $\Sigma_t = Y_t Z_t^{-1}$.

Intuitive Individual Updates

We note again that descriptor matrices can be formed from the HDRDE corresponding to each individual measurement and control update in Equations 5.74 and 5.75. These equations translate into symplectic descriptor matrices as

$$\mathcal{T}_t^M = \begin{bmatrix} I & 0 \\ M & I \end{bmatrix}_t \quad \text{and} \quad \mathcal{T}_t^C = \begin{bmatrix} G & RG^{-T} \\ 0 & G^{-T} \end{bmatrix}_t, \tag{5.92}$$

where $\mathcal{T}_t^C$ is a control update matrix and $\mathcal{T}_t^M$ is a measurement update matrix.

Note that this is the same result derived from the Kalman filter equations in Section 5.2, shown in Equations 5.11 and 5.7. The resulting solution in that section (Equation 5.14) was indeed an equivalent symplectic solution. The only difference is that Equation 5.14 was the result of a measurement-based recursion, which formed $\mathcal{T}_t$ as $\mathcal{T}_t = \mathcal{T}_t^M \cdot \mathcal{T}_t^C$. Our current solution in Equation 5.91 is the result of a control-based recursion, which is given as

$$\mathcal{T}_t = \mathcal{T}_t^C \cdot \mathcal{T}_t^M = \begin{bmatrix} G + RG^{-T}M & RG^{-T} \\ G^{-T}M & G^{-T} \end{bmatrix}. \tag{5.93}$$

Algorithm 11 Build One-Step Hamiltonian Matrix procedure.

```
Require: Edge trajectory $e_{\mu,\mu'}$ and step length $\epsilon$
Ensure: One-step aggregate star descriptor matrix $\mathcal{S}^\star$
    BUILD_ONESTEP_HAMILTONIAN($e_{\mu,\mu'}, \epsilon$)
1: Initialize $\mathcal{S}^\star = I^{2n\times 2n}$
2: for $\mu_i = \mu$ to $\mu'$ step $\epsilon_\mu = \epsilon \cdot \delta e_{\mu,\mu'}$ do
3:     Compute $G_i^{n\times n}$ and $R_i^{n\times n}$ from motion control $\mu_{i-1} \rightarrow \mu_i$
4:     Build control update descriptor $\mathcal{S}_i^C = \begin{bmatrix} G_i & R_i \\ 0 & G_i^T \end{bmatrix}$
5:     Compute $M_i^{n\times n}$ from maximum likelihood measurement at $\mu_i$
6:     Build measurement update descriptor $\mathcal{S}_i^M = \begin{bmatrix} I & 0 \\ M_i & I \end{bmatrix}$
7:     $\mathcal{S}^\star = \mathcal{S}^\star \star \mathcal{S}_i^C \star \mathcal{S}_i^M$
8: end for
9: return $\mathcal{S}^\star$
```

Algorithm 12 Compute One-Step Hamiltonian Update procedure.

```
Require: Start covariance $\Sigma$ and OS star descriptor $\mathcal{S}^\star = \begin{bmatrix} G^\star & R^\star \\ M^\star & G^\star \end{bmatrix}$
Ensure: Posterior covariance $\Sigma'$
    COMPUTE_ONESTEP_HAMILTONIAN($\Sigma, \mathcal{S}^\star$)
1: Compute $\Sigma' = R^\star + G^\star \Sigma (I + M^\star \Sigma)^{-1} {G^\star}^T$
2: return $\Sigma'$
```

Recovering the Posterior Covariance

The solution to the one-step problem with symplectic matrices is identical to the Kalman filter-based solution presented in Section 5.2. One must first build an aggregate symplectic matrix that accounts for all measurement and control updates from time $t = 1$ to N, which is given by matrix multiplication as

$$\mathcal{T}^\star = \mathcal{T}_{1:N} = \mathcal{T}_N^M \cdot \mathcal{T}_N^C \cdots \mathcal{T}_1^M \cdot \mathcal{T}_1^C. \tag{5.94}$$

Then, given a novel initial covariance Σ_0, the resulting posterior covariance after N updates is easily computed as

$$\Sigma_N = (\mathcal{T}_{11}^\star \Sigma_0 + \mathcal{T}_{12}^\star)(\mathcal{T}_{21}^\star \Sigma_0 + \mathcal{T}_{22}^\star)^{-1}, \tag{5.95}$$

where $\mathcal{T}_{ij}^\star$ is the i, j block of the aggregate symplectic matrix $\mathcal{T}^\star$.

Algorithm 13 Build One-Step Symplectic Matrix procedure.

```
Require: Edge trajectory $e_{\mu,\mu'}$ and step length $\epsilon$
Ensure: One-step aggregate symplectic descriptor matrix $\mathcal{T}^\star$
    BUILD_ONESTEP_SYMPLECTIC($e_{\mu,\mu'}, \epsilon$)
1: Initialize $\mathcal{T}^\star = I^{2n\times 2n}$
2: for $\mu_i = \mu$ to $\mu'$ step $\epsilon_\mu = \epsilon \cdot \delta e_{\mu,\mu'}$ do
3:     Compute $G_i^{n\times n}$ and $R_i^{n\times n}$ from motion control $\mu_{i-1} \to \mu_i$
4:     Build control update descriptor $\mathcal{T}_i^C = \begin{bmatrix} G_i & R_i G_i^{-T} \\ 0 & G_i^{-T} \end{bmatrix}$
5:     Compute $M_i^{n\times n}$ from maximum likelihood measurement at $\mu_i$
6:     Build measurement update descriptor $\mathcal{T}_i^M = \begin{bmatrix} I & 0 \\ M_i & I \end{bmatrix}$
7:     $\mathcal{T}^\star = \mathcal{T}_i^M \cdot \mathcal{T}_i^C \cdot \mathcal{T}^\star$
8: end for
9: return $\mathcal{T}^\star$
```

Algorithm 14 Compute One-Step Symplectic Update procedure.

```
Require: Start covariance $\Sigma$ and OS symplectic descriptor $\mathcal{T}^\star = \begin{bmatrix} \mathcal{T}_{11}^\star & \mathcal{T}_{12}^\star \\ \mathcal{T}_{21}^\star & \mathcal{T}_{22}^\star \end{bmatrix}$
Ensure: Posterior covariance $\Sigma'$
    COMPUTE_ONESTEP_SYMPLECTIC($\Sigma, \mathcal{T}^\star$)
1: Compute $N = \mathcal{T}_{11}^\star \Sigma + \mathcal{T}_{12}^\star$
2: Compute $D = \mathcal{T}_{12}^\star \Sigma + \mathcal{T}_{22}^\star$
3: return $\Sigma' = ND^{-1}$
```

5.5 Summary of One-Step Solutions

In this chapter, we presented two forms of one-step solutions that allow multiple EKF covariance updates to be aggregated into *one* covariance transfer function. The motivation for these one-step methods was to improve the efficiency of the belief trajectory search presented in Chapter 4. We thus conclude this chapter by summarizing the one-step solutions in terms of covariance propagation along trajectories. For each method, we present a build procedure and update procedure in algorithmic form. The build procedure constructs an aggregate EKF descriptor matrix for a trajectory which represents the one-step transfer function. The update procedure computes the posterior covariance Σ' at the end of a trajectory for a novel initial condition Σ by applying the one-step transfer function. For the Hamiltonian solution, the corresponding build and update procedures are shown in Algorithms 11 and 12, respectively. For the symplectic solution, the corresponding build and update procedures are shown in Algorithms 13 and 14, respectively.

Each build procedure computes an aggregate one-step descriptor matrix by iterating through a sequence of discrete points along a trajectory $e_{\mu,\mu'}$. At each point μ_i, the state transition G_i and process noise R_i matrices are computed according to the incremental motion control from the previous point $u_i = \mu_{i-1} \rightarrow \mu_i$. A control update descriptor matrix is then created with G_i and R_i. Similarly, the expected measurement information M_i is computed at each point μ_i from the maximum likelihood measurement. A measurement update descriptor matrix is then formed using M_i. Finally, for each discrete point μ_i the aggregate one-step descriptor is updated, via functional composition, to account for the uncertainty effects encoded in the control and measurement update descriptors.

Each update procedure implements the respective one-step computation. For the Hamiltonian method, this computation is shown in Equation 5.90. For the symplectic method, the update computation is shown in Equation 5.95.

Chapter 6

The Belief Roadmap Algorithm

Contents

6.1 Introduction

In this chapter, we combine the general trajectory search algorithm from Chapter 4 with the efficient one-step update from Chapter 5 to form the Belief Roadmap (BRM) algorithm. As we discussed in Chapter 4, the key limitation of the trajectory search planning algorithm is the computational penalty of performing redundant multi-step updates during graph search.

The BRM method mitigates this burden by replacing the multi-step trajectory update in the hybrid search process with a simple, one-step transfer function. In experiments, we show that this improves search times by over two orders-of-magnitude. While it is still necessary to compute the uncertainty effects along each trajectory in multiple steps, this cost is amortized to the one-time BRM build process. Altogether, the BRM algorithm harnesses the tractable foundation for belief space planning presented in the trajectory search formulation, while realizing substantial on-line speed gains with one-step uncertainty updates.

The efficiency of the BRM algorithm is owed to the decoupling of multi-step uncertainty propagation from initial conditions that is performed by one-step methods. Specifically, the one-step insight enables us to move the repetitive computational burden of MS updates from the on-line search process into a one-time procedure in the off-line build process. Section 6.2 shows the manifestation of this trade-off as we modify both the trajectory graph build phase of Algorithm 8 and the hybrid search process in Algorithm 9 to construct the BRM algorithm. In Section 6.2.3, we propose an alternate form of the search process which uses a modified objective function to limit the maximum uncertainty encountered along the solution path. In Section 6.3, we present our experimental setup and results, beginning with a derivation of the linearized motion and sensor models used in this work in Sections 6.3.1-6.3.2. We assess the algorithmic and localization performance of the BRM and present an example solution to a large planning problem across the MIT campus in Sections 6.3.4-6.3.6. In Section 6.4, we conclude the work presented in this chapter and defer discussion of further applications and future work to Chapter 7.

6.2 The Belief Roadmap Algorithm

In this section, we develop the Belief Roadmap algorithm by adapting the trajectory graph planning algorithm presented in Chapter 4 to use one-step trajectory updates. We first present the modified BRM build process in Section 6.2.1, leading to the complete BRM planning algorithm in Section 6.2.2.

6.2.1 Building the Belief Roadmap

With one-step trajectory updates, it is possible to isolate the cost of performing multiple

Algorithm 15 Build Belief Roadmap procedure.

```
Require: Map $\mathcal{C} \equiv \mathcal{B}^{\mu}$ over mean-belief locations
Ensure: BRM graph $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}, \{\mathcal{S}_{ij}\}\}$ with belief node set $\{n_i\}$, edge trajectory set $\{e_{ij}\}$ and one-step descriptor set $\{\mathcal{S}_{ij}\}$
    BUILD_BRM_GRAPH($\mathcal{B}^{\mu}$)
1: Sample poses $\{\mu_i\}$ from $\mathcal{B}^{\mu}_{free}$ using a standard PRM sampling strategy.
2: Build graph node set $\{n_i\}$ for each sample such that $n_i = \{\mu = \mu_i, \Sigma = \emptyset\}$
3: Create edge set $\{e_{ij}\}$ between nodes $(n_i, n_j)$ if the straight-line path between $(n_i[\mu], n_j[\mu])$ is collision-free
4: for all edges $e_{ij} \in \{e_{ij}\}$ do
5:     $\mathcal{S}_{ij} \leftarrow$ BUILD_ONESTEP($e_{ij}, \epsilon$)
6: end for
7: return $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}, \{\mathcal{S}_{ij}\}\}$
```

Kalman filter updates along each trajectory to the off-line pre-processing phase of the planning algorithm. We adapt the graph building method in Algorithm 8 to construct one-step descriptor matrices as shown in Algorithm 15.

Two modifications have been made to Algorithm 8. First, the underlying structure of the trajectory graph $\mathcal{G}$ has been augmented with a set of one-step descriptor matrices $\{\mathcal{S}_{ij}\}$ corresponding to each edge e_{ij} in the trajectory edge set $\{e_{ij}\}$. Each one-step descriptor matrix $\mathcal{S}_{ij}$ represents the uncertainty transfer function between nodes n_i and n_j along the trajectory e_{ij}. The second modification is the addition of a build step for the one-step matrices, shown in lines 4-6. For each edge in the graph, the BUILD_ONESTEP procedure is used to construct the corresponding descriptor matrix. Either of the build methods, BUILD_ONESTEP_HAMILTONIAN (Algorithm 11) or BUILD_ONESTEP_SYMPLECTIC (Algorithm 13), may be used, depending on the choice of one-step update.

Note that the graph structure and uncertainty transfer functions are all created without knowledge of the initial covariance used during search. This is a substantial improvement over the original build method in Algorithm 8, for we have amortized the cost of sequential Kalman filter trajectory updates to the off-line pre-processing phase. While the computational burden of the BUILD_ONESTEP routine for a given trajectory is roughly equivalent to that of the conventional COMPUTE_MULTISTEP (Algorithm 7), the BUILD_ONESTEP method is independent of initial conditions and, thus, computed only *once* during the entire course of planning and re-planning.

Algorithm 16 The Belief Roadmap (BRM) algorithm.

```
Require: Start belief $(\mu_0, \Sigma_0)$, goal location $\mu_{goal}$, map $\mathcal{B}^{\mu}$ and edge step size $\epsilon$
Ensure: Path $p$ resulting in minimal covariance at $\mu_{goal}$
 1: $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}, \{\mathcal{S}_{ij}\}\} \leftarrow$ BUILD_BRM_GRAPH $(\mathcal{B}^{\mu})$
 2: Append $\mathcal{G}$ with nodes $\{n_0, n_{goal}\}$, edges $\{\{e_{0,j}\}, \{e_{i,goal}\}\}$, and one-step descriptors
    $\{\{\mathcal{S}_{0,j}\}, \{\mathcal{S}_{i,goal}\}\}$
 3: Augment node structure with best parent node $n^p = \emptyset$, such that $n_i = \{\mu, \Sigma, n^p\}$
 4: Create search queue with initial position and covariance $Q \leftarrow n_0 = \{\mu_0, \Sigma_0, \emptyset\}$
 5: while $Q$ is not empty do
 6:   Pop $n \leftarrow Q$
 7:   if $n = n_{goal}$ then
 8:     Continue
 9:   end if
10:   for all $n'$ such that $\exists e_{n,n'}$ and not IN_PATH($n$, $n_0$, $n'$) do
11:     $\Sigma' \leftarrow$ COMPUTE_ONESTEP($n[\Sigma], \mathcal{S}_{n,n'}$)
12:     if $tr(\Sigma') < tr(n'[\Sigma])$ then
13:       $n' \leftarrow \{n'[\mu], \Sigma', n\}$
14:       Push $n' \rightarrow Q$
15:     end if
16:   end for
17: end while
```

6.2.2 BRM Trajectory Search

The complete BRM algorithm is shown in Algorithm 16. The only modification to the search process from Algorithm 9 is that we have replaced the multi-step trajectory uncertainty update with a one-step transfer function. This is shown in line 11 of Algorithm 16, where the posterior covariance along a given trajectory is obtained from the COMPUTE_ONESTEP routine. Again, this one-step update can be performed by using either the COMPUTE_ONESTEP_HAMILTONIAN or COMPUTE_ONESTEP_SYMPLECTIC procedure depending on which one-step method is chosen.

In general, the issues noted in the discussion of the trajectory search algorithm in Section 4.3.2 apply to the BRM algorithm. However, there are additional considerations which we explore below.

Discussion of Search Complexity

For a trajectory consisting of k actions, the multi-step update in Algorithm 7 requires $\mathcal{O}(k)$ EKF control and measurement updates to compute a posterior covariance. This means that the asymptotic complexity of the overall search process in Algorithm 9 is $\mathcal{O}(kb^d)$, with branching factor b and search depth d. The efficiency improvement of modifying the search process to use one-step updates, such that $k = 1$, is not immediately apparent, since the difference in complexity is a constant multiplier k of exponential growth. However, the computational cost of performing specific EKF updates, consisting of relatively expensive matrix operations (multiplications and inversions) and measurement simulations, has a significant effect on the overall time to plan. It is for this reason that using a one-step update during the hybrid search process, with complexity $\mathcal{O}(b^d)$, results in a substantial improvement in efficiency.

There are additional points to make note of regarding our search process. In order to guarantee an optimal solution of the objective function, it is necessary to perform an exhaustive search of the BRM graph $\mathcal{G}$ for acyclic paths between the start and goal locations. Note, however, that in practice the depth of this exhaustive search does not traverse the entire graph $\mathcal{G}$; rather, the depth is a function of the largest connected component $\mathcal{G}_{cc}$ between the start and goal nodes. This point has two implications. First, the average complexity of the search process may be significantly less than the worst case complexity if the nature of the graph is such that $\mathcal{G}_{cc}^{avg} << \mathcal{G}_{cc}^{worst}$. Second, the added complexity of computing one-step descriptors during the build process need only be limited to performing BUILD_ONESTEP for trajectories within the connected component. While $\mathcal{G}_{cc}$ is determined after the build process for a specific on-line query, this suggests that the one-time cost of building a one-step trajectory can be arbitrarily moved between the build and search processes to suit a given application. For example, the COMPUTE_ONESTEP routine could be easily modified to build a one-step descriptor on-demand during the search process if the given matrix has not yet been computed.

It should also be noted that one of the key benefits of the BRM is its applicability to on-line planning. If the search phase was only to be performed once to generate a static plan, the one-step technique would lead to little improvement in the overall planning process. However, in typical motion planning problems it is desirable to re-plan during the course of execution to account for deviations from the original plan and also to incorporate dynamic constraints. The BRM provides an very suitable approach for handling multiple queries because it amortizes the significant cost of filtering over trajectories to the one-time build phase. As a result, planning and re-planning with novel initial conditions are performed very efficiently by the BRM's stream-lined search process.

Algorithm 17 The Min-Max Belief Roadmap (minmax-BRM) algorithm.

```
Require: Start belief $(\mu_0, \Sigma_0)$, goal location $\mu_{goal}$, map $\mathcal{B}^{\mu}$ and edge step size $\epsilon$
Ensure: Path $p$ from $\mu_0$ to $\mu_{goal}$ with minimum maximum covariance.
 1: $\mathcal{G} = \{\{n_i\}, \{e_{ij}\}, \{\mathcal{S}_{ij}\}\} \leftarrow$ BUILD_BRM_GRAPH $(\mathcal{B}^{\mu})$
 2: Append $\mathcal{G}$ with nodes $\{n_0, n_{goal}\}$, edges $\{\{e_{0,j}\}, \{e_{i,goal}\}\}$, and one-step descriptors
    $\{\{\mathcal{S}_{0,j}\}, \{\mathcal{S}_{i,goal}\}\}$
 3: Augment node structure with best path $p = \emptyset$ and maximum covariance $\Sigma^{p}_{max} = \infty$
    along path $p$, such that $n_i = \{\mu, \Sigma, p, \Sigma^{p}_{max}\}$
 4: Create search queue with initial position and covariance $Q \leftarrow n_0 = \{\mu_0, \Sigma_0, \emptyset, \infty\}$
 5: while $Q$ is not empty do
 6:   Pop $n \leftarrow Q$
 7:   if $n = n_{goal}$ then
 8:     Continue
 9:   end if
10:   for all $n'$ such that $\exists e_{n,n'}$ and $n' \ni n[p]$ do
11:     $\Sigma' \leftarrow$ COMPUTE_ONESTEP$(n[\Sigma], \mathcal{S}_{n,n'})$
12:     if $max(tr(\Sigma'), tr(n[\Sigma^{p}_{max}])) < tr(n'[\Sigma^{p}_{max}])$ then
13:       $n' \leftarrow \{n'[\mu], \Sigma', \{n[p], n'\}, max(\Sigma', n[\Sigma^{p}_{max}])\}$
14:       Push $n' \rightarrow Q$
15:     end if
16:   end for
17: end while
```

An additional advantage of the BRM in on-line applications stems from planning within belief space, wherein the actual robot belief during execution can be directly utilized by the planner. For example, when the robot reaches a waypoint during trajectory execution, it can efficiently re-plan using the realized belief covariance as the initial condition of the search process. This ability makes the BRM an attractive method for robust planning *and* execution under uncertainty.

6.2.3 Modified BRM for MinMax Path Uncertainty

In the BRM formulation shown in Algorithm 16, the search process finds the series of trajectories that results in minimal uncertainty at the goal location; however, it may be desirable to instead limit the maximum uncertainty encountered along an entire path. One approach could be to impose bounds on the maximum allowable uncertainty during the BRM search $tr(\Sigma) < tr_{max}$ to discard undesirable goal paths. In a more principled approach, one could modify the BRM search process to optimize an alternative objective function that minimizes the maximum predicted uncertainty along the entire path. Within the context of the BRM graph, this approach would consider the posterior covariance predicted at each intermediate belief node in a series of trajectories. The goal would be to minimize the objective function K, which is given as

$$K(\mu_0, \Sigma_0, u_0, \ldots, \mu_T, \Sigma_T, u_T) = \sum_{t=0}^{T} C(\mu_t, u_t) + D(max(\Sigma_0, ..., \Sigma_T)), \tag{6.1}$$

where $K(\ldots)$ is the cost of a path, $C(\mu_t, u_t)$ is the cost of executing control u_t from pose μ_t, and $D(...)$ is the cost associated with the maximum uncertainty of all discrete belief nodes along the path.

The BRM search process is adapted to minimize the objective function K in Algorithm 17, which we have named the Min-Max Belief Roadmap (minmax-BRM) algorithm. There are two key changes from the standard BRM. First, the augmented search node structure in line 3 stores the best path p to the given node and the maximum covariance Σ^p_{max} along p. The best path $n_i[p]$ to node n_i corresponds to the series of nodes beginning at the start node n_0 that collectively has the minimum maximum (min-max) covariance of all such paths considered to node n_i. The maximum covariance $n_i[\Sigma^p_{max}]$ along this best path $n_i[p]$ is also stored in the search state for computing the associated cost D in the objective function K (Equation 6.1) and for decision-making during search. Note that the covariance $n_i[\Sigma]$ stored at node n_i is no longer the minimum achievable covariance, but rather to posterior covariance resulting from the best path $n_i[p]$.

The second change is a series of modifications to the node expansion step in lines 10-16. First, path cycles are disallowed in line 10, where a neighboring node n' is only considered if it is not already in the search state path $n[p]$. Then, as before, for each valid neighboring node n' the current search state covariance is propagated along the corresponding edge in line 11. The primary decision-making step in line 12 is modified for the new objective function. In this case, the path being considered in the current search state $\{n[p], n'\}$ is deemed better than the existing path $n'[p]$ to node n' if its maximum uncertainty is less than that of the existing path. Note that the maximum uncertainty of the current search path $\{n[p], n'\}$ is computed by taking the max function of the associated uncertainty of each portion of this path, which is $tr(n[\Sigma^p_{max}])$ for $n[p]$ and $tr(\Sigma')$ for n'. If the current search path is better than the existing path, then the node n' is updated accordingly in line 13 and placed on the queue in line 14.

A key consideration of the minmax-BRM algorithm is that it can only guarantee optimality for a specific resolution of the uncertainty evolution along a path. In Algorithm 17, we only consider the covariance at node locations within the entire path. While the COMPUTE_ONESTEP method exactly computes the posterior covariance of multiple KF updates along a trajectory, the underlying multi-step process is not monotonic. This means that it is possible for the covariance at an intermediate point along a trajectory to be larger than both the prior covariance and posterior covariance for the full trajectory. It is possible to revert to some form of multi-step approach to this problem, but, without further assumptions, the guarantee of optimality will always be limited to the resolution of discretization. We leave the analysis of this problem for future work, and place our focus on the standard BRM for experiments.

It is important to note the generality of the BRM formulation, which was demonstrated in this section by modifying the search process to optimize an alternative objective function. The BRM technique presents a general approach to planning in belief space that can be adapted to solve a broad class of planning problems.

6.3 Experiments and Results

In order to evaluate the BRM algorithm, we performed a series of evaluations on a small planning domain in simulation. In this section, our experimental setup and the results of each evaluation are presented in turn. We begin by presenting the motion and sensor models used in our experiments, which are linearized versions of the models developed in Sections 3.3 and 3.2.2, respectively, for use in our EKF-based formulation. Note that, for readability, we omit time index subscripts in the next two sections; however, all matrices derived are time-varying quantities.

6.3.1 Linearized Motion Model

The non-linear probabilistic motion model presented in Section 3.3 can be approximated by linearization for use by filtering techniques that require a linear system. For convenience, we restate the state transition function $x_t = g(x_{t-1}, u_t)$ for this motion model (from

Equation 3.10):

$$\begin{aligned} g_x &= x + D\cos(\theta + \frac{T}{2}) + C\cos(\theta + \frac{T+\pi}{2}) \\ g_y &= y + D\sin(\theta + \frac{T}{2}) + C\sin(\theta + \frac{T+\pi}{2}) \\ g_\theta &= \theta + T \mod 2\pi, \end{aligned}$$

where g_x, g_y and g_θ are the components of g corresponding to each state variable, and the control variable u_t is given by $u_t = \begin{bmatrix} D & C & T \end{bmatrix}^T$ with down-range D, cross-range C and turn T components.

In the EKF, the state transition matrix G is the Jacobian of the motion model with respect to the state, and is computed by linearizing the state transition function g about the mean state μ as follows:

$$G = \begin{bmatrix} \frac{\delta g_x}{\delta x} & \frac{\delta g_x}{\delta y} & \frac{\delta g_x}{\delta \theta} \\ \frac{\delta g_y}{\delta x} & \frac{\delta g_y}{\delta y} & \frac{\delta g_y}{\delta \theta} \\ \frac{\delta g_\theta}{\delta x} & \frac{\delta g_\theta}{\delta y} & \frac{\delta g_\theta}{\delta \theta} \end{bmatrix}_\mu = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix},$$

where

$$\begin{aligned} a &= -D\sin(\mu_\theta + \frac{T}{2}) - C\sin(\mu_\theta + \frac{T+\pi}{2}) \\ b &= D\cos(\mu_\theta + \frac{T}{2}) + C\cos(\mu_\theta + \frac{T+\pi}{2}). \end{aligned}$$

The linearized process noise in state space is computed as $R \triangleq VWV^T$ where W is the process noise covariance in control space

$$W = \begin{bmatrix} \sigma_D^2 & 0 & 0 \\ 0 & \sigma_C^2 & 0 \\ 0 & 0 & \sigma_T^2 \end{bmatrix}$$

and V is the mapping from control to state space, computed as the Jacobian of the motion model with respect to the control space components

$$V = \begin{bmatrix} \frac{\delta g_x}{\delta D} & \frac{\delta g_x}{\delta C} & \frac{\delta g_x}{\delta T} \\ \frac{\delta g_y}{\delta D} & \frac{\delta g_y}{\delta C} & \frac{\delta g_y}{\delta T} \\ \frac{\delta g_\theta}{\delta D} & \frac{\delta g_\theta}{\delta C} & \frac{\delta g_\theta}{\delta T} \end{bmatrix}_{\mu,\mu^{[u]}} .$$

Computing these partial derivatives leads to

$$V = \begin{bmatrix} \cos(\theta + \frac{T}{2}) & \cos(\theta + \frac{T+\pi}{2}) & -\frac{1}{2}\Big(D\sin(\theta + \frac{T}{2}) + C\sin(\theta + \frac{T+\pi}{2})\Big) \\ \sin(\theta + \frac{T}{2}) & \sin(\theta + \frac{T+\pi}{2}) & \frac{1}{2}\Big(D\cos(\theta + \frac{T}{2}) + C\cos(\theta + \frac{T+\pi}{2})\Big) \\ 0 & 0 & 1 \end{bmatrix}_{\mu,\mu^{[u]}} ,$$

where the components of V are evaluated at the mean state μ and the mean control $\mu^{[u]}$ (D, C, and T take on the mean values of the respective normal distributions specified in equations 3.10-3.12).

6.3.2 Linearized LOS Sensor Model

Similarly, the UWB sensor model developed in Section 3.2.2 can be linearized for use in the EKF. For convenience we restate the distance-varying Gaussian noise model of the bias $\mathcal{N}(\mu_b(d), \sigma_b(d))$ and the observation function z. The noise model is given by,

$$\begin{aligned} \mu_b(d) &= \mu_b^m d + \mu_b^b \\ \sigma_b(d) &= \sigma_b^m d + \sigma_b^b \end{aligned}$$

and the observation function $z = h(x) + v$ is determined by,

$$\begin{aligned} h(x) &= \mu_b^b + (1 + \mu_b^m)\sqrt{(x - x_b)^2 + (y - y_b)^2} \\ v &= \mathcal{N}(0, \sigma_b(d)^2), \end{aligned}$$

where (x, y) is the robot pose, (x_b, y_b) is the beacon location, d is the euclidean distance and the parameters of the Gaussian bias distribution are linear functions of the distance.

The linearized transformation from measurement space to state space is computed as the measurement Jacobian H, computed as the partial derivatives of the measurement function with respect to each component of the state:

$$H = \begin{bmatrix} \frac{\delta h}{\delta x} & \frac{\delta h}{\delta y} & \frac{\delta h}{\delta \theta} \end{bmatrix},$$

which becomes:

$$H = \begin{bmatrix} \frac{(1+\mu_b^m)(x-x_b)}{\sqrt{(x-x_b)^2+(y-y_b)^2}} & \frac{(1+\mu_b^m)(y-y_b)}{\sqrt{(x-x_b)^2+(y-y_b)^2}} & 0 \end{bmatrix}. \tag{6.2}$$

Note that $(x - x_b) = d \cos\theta_m$ and $(y - y_b) = d \sin\theta_m$, where $\theta_m = \text{atan2}(y - y_b, x - x_b)$ is the angle of measurement relative to the robot pose. Thus, Equation 6.2 becomes

$$\begin{aligned} H &= \begin{bmatrix} \frac{(1+\mu_b^m)d\cos\theta_m}{d} & \frac{(1+\mu_b^m)d\sin\theta_m}{d} & 0 \end{bmatrix} \\ &= \begin{bmatrix} (1+\mu_b^m)\cos\theta_m & (1+\mu_b^m)\sin\theta_m & 0 \end{bmatrix}. \end{aligned} \tag{6.3}$$

As can be seen, the range measurements yield no information on bearing, and thus only affect the estimation of the x and y components of the robot state.

The measurement noise covariance $Q \triangleq cov(q, q)$ for a given beacon is the 1×1 matrix

$$Q = \begin{bmatrix} (\sigma_b^m d + \sigma_b^b)^2. \end{bmatrix} \tag{6.4}$$

6.3.3 Experimental Setup

Our evaluations of the BRM algorithm consisted of two objectives: (1) to quantify the computational advantage of using the linearized one-step EKF covariance update during the search process; and (2), to assess the quality of plans produced by the BRM algorithm in terms of minimizing uncertainty at a goal location.

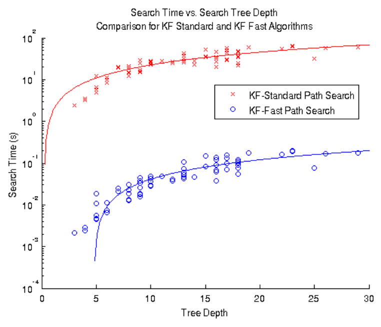
We used various maps of sizes ranging between $30 - 100$ m on a side with UWB sensor beacons placed at random locations. These environments were assumed to be free of obstacles with line-of-sight (LOS) conditions for UWB sensors. This test design was chosen for two primary reasons. Firstly, LOS conditions reduce the experimental bias that could result from environment-specific artifacts in NLOS scenarios with random trajectory generation. Secondly, for the purposes of generality, the UWB sensor model in LOS conditions provides a less sensor-specific noise model. By using only the linear-Gaussian portion of the noise model, our results are more general and provide greater intuition for applying the BRM to other sensor domains.

The BRM graph was built by sampling poses according to a medial-axis sampling strategy [22]. In the LOS environments used, all poses are visible; however, we limit the creation of edges from each node to the k-nearest neighbors. The resulting BRM graph was used with beacon sensor locations from the map for EKF covariance propagation.

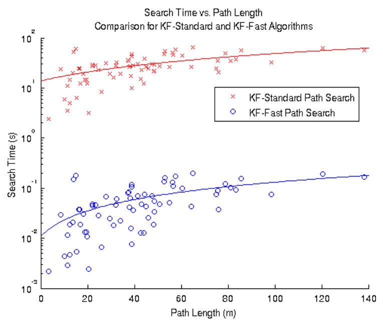
In the EKF trajectory updates during testing, motion controls were simulated between discrete points along the given trajectory arc. The length of the sub-arcs corresponding to each control action was chosen to emulate the actual length of discrete motion updates during real-world plan execution. An EKF control update is performed for each simulated control action u_t by computing the corresponding state transition G_t and process noise R_t matrices linearized about the mean-belief location μ_{t-1} at the beginning of the sub-arc. Further, at each mean-belief location μ_t, UWB range measurements were simulated to beacons within a specified radius. For each beacon, the maximum-likelihood observation was computed by measuring distance d_t relative to the mean-belief location μ_t and calculating the mean $\mu_b(d_t)$ and standard deviation $\sigma_b(d_t)$ of the distance-varying Gaussian bias. The linearized measurement Jacobian H_t and the measurement noise covariance Q_t were computed using the values of $\mu_b(d_t)$ and $\sigma_b(d_t)^2$, respectively.

6.3.4 Algorithmic Performance

To assess the speed improvement of employing one-step EKF updates over trajectories, we compared the time required by the BRM search process (Algorithm 16) using one-step covariance updates to that of the standard trajectory search (Algorithm 9) with MS covariance updates. The experiments were performed over maps of varying size with UWB sensors placed at randomized locations. To reduce the variability in speed results, the number of belief nodes and the number of UWB sensors were constrained as follows. In each trial, the number of nodes in the BRM graph was sampled in proportion to the area of the environment. This served to limit the variability of trajectory lengths in the BRM graph, allowing for reasonable comparison of speed results measured over different graphs. Additionally, a constant number of UWB sensors was used throughout all trials. Similarly, this limited the variability of the time required for measurement updates in different environments.



(a) Time vs. Tree Depth



(b) Time vs. Path Length

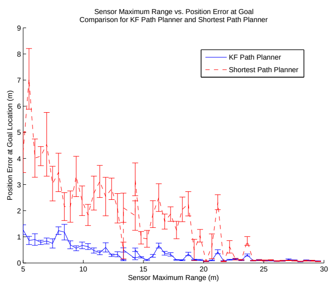
Figure 6-1: Algorithmic Performance. (a) Time to Plan vs. Tree Depth (b) Time to Plan vs. Path Length. Note that these are semi-log graphs, indicating two orders of magnitude increase in speed.

The results of the speed comparison over search tree depth and resulting path length are shown in Figure 6-1. Figure 6-1(a) shows the time required by the planning search process with respect to the depth of the search tree for both planners. As evidenced in this semi-logarithmic plot, the BRM demonstrates a consistent improvement of over two orders of magnitude in search time. Further, the cost of searching grows logarithmically with increasing tree depth, indicating that the efficiency of the BRM search process scales well to accommodate larger trajectory graphs.

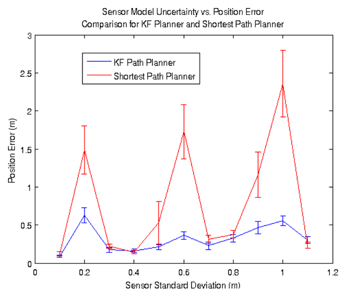
Similar results are shown in Figure 6-1(b), which plots the time required to search with respect to the length of the resulting path. Again, the one-step update employed by the BRM consistently outperforms the multi-step planner by over two orders of magnitude in search time. This comparison reiterates the significant scalable improvement that is enabled by the one-step technique.

Note that both planning algorithms also incur a one-time build cost to generate the trajectory graph, in which the BRM incurs the additional cost of computing one-step update descriptors for each edge in the graph. As a result, the time required to perform the build phase and *one* query is comparable in either planner. However, the key point to note is that the BRM amortizes the one-time cost of aggregating multi-step updates over the entire planning and re-planning process, whereas the standard trajectory search planner incurs this multi-step cost for *every* query. The BRM thus realizes substantial speed gains in on-line planning tasks, as demonstrated in these experiments, enabling belief space planning to not only be computationally tractable, but also efficient in real-world execution.

6.3.5 Localization Performance



(a) Varied Sensor Range



(b) Varied Sensor Noise

Figure 6-2: Characterization of Planner Positional Accuracy. (a) Accuracy vs. Sensor Range. The localization performance of the shortest path algorithm increased as more sensors were in range. The BRM was able to find paths with sufficient sensor information for localization even with very short range. (b) Accuracy vs. Sensor Noise. The BRM consistently outperforms the shortest path planner. The positional accuracy of the shortest path planner shows unreliability and that it suffers with increased noise.

To evaluate the quality of paths produced by the BRM algorithm, we compared the performance of plan execution for paths generated by the BRM search process to those resulting from a standard shortest path search. The aim of this evaluation was to test the underlying intuition of the BRM design; that detouring from the shortest path for trajectories with greater potential information gain may result in less positional error at the goal location. We quantified the quality of plans in these experiments by measuring the average positional error obtained at the goal location after executing paths prescribed by each planner.

We performed two evaluations of localization performance by artificially changing parameters of the sensor model. The intuition for this test design stems from the objective function J that is minimized by the BRM, which is restated here as

$$J(b_0, u_0, \ldots, b_T, u_T) = \sum_{t=0}^{T} C(b_t, u_t) + D(b_T),$$

where $J(\ldots)$ is the cost of a path, $C(b_t, u_t)$ is the cost of executing control u_t from belief b_t, and $D(b_T)$ is the cost associated with the resulting uncertainty of the goal belief b_T. We seek to test the hypothesis that as the quality of the sensor is *improved*, there is less difference in potential information gain between trajectories (as they all become equally *good*) and, thus, the cost D associated with goal uncertainty becomes less significant in differentiating paths. Then, as the cost C of executing controls becomes dominant, shorter paths consisting of less cumulative control uncertainty will be favored, and the BRM should perform similarly to a shortest path planner. Conversely, decreasing the quality of the sensor should make trajectories with greater potential sensor information favorable. In randomized environments, such trajectories should tend to deviate from the shortest path and their execution should result in more reliable performance.

In the first analysis, we artificially limited the range of the UWB sensors by discarding measurements beyond a maximum range r_{max}. Figure 6-2(a) shows the realized positional error after executing plans for different maximum sensor ranges. It is clear that when r_{max} is small, the BRM is able to find trajectories in close proximity to sensors and remain localized, attaining reasonable positional accuracy. At the same time, the shortest path receives less range measurements and results in greater positional error. However, as r_{max} increases, trajectories farther from sensors provide sufficient information for localization and the positional errors converge to similar values for both planners.

In the second analysis, we artificially modified the stochastic, distance-varying noise model of the UWB range sensors. The resulting positional error obtained at the goal location is shown in Figure 6-2(b) for the BRM and shortest path planners over varying sensor noise. We see that the BRM is able to select trajectories with more favorable sensor measurements and achieve greater positional accuracy. As the sensor noise increases, both algorithms result in greater positional error, but the performance of the shortest path algorithm is less reliable and degrades more quickly. Note that the spikes in this graph indicate sensitivity to the randomly generated environment for each noise level. In future experiments we seek to isolate this variability.

Overall, these two evaluations indicate that, by incorporating predictions of uncertainty, the BRM choses higher-quality paths that yield improved reliability in performance.

6.3.6 Example: Large Planning Problem

Our final experiment was a demonstration of the BRM algorithm in a very large planning problem. Figure 6-3 shows paths generated through a trajectory graph which spans the MIT campus. The robot must travel in free-space (shown in white) outside of buildings from a starting location in the lower right corner to a goal location in the upper left corner. UWB ranging beacons are scattered throughout the environment and shown as small blue circles. Figure 6-3(a) shows the path generated by the shortest path planner, which finds

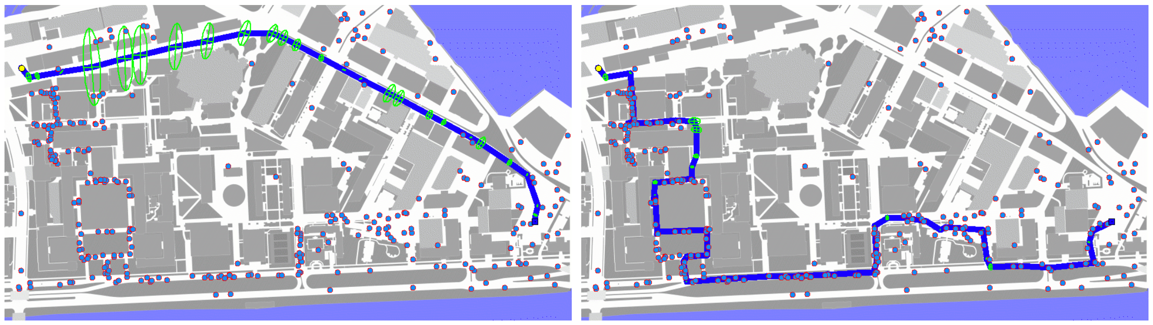

(a) Shortest Path (b) Lowest Expected Cost

Figure 6-3: Example motion plans for navigation across MIT campus (from lower right to upper left). Paths are computed by a shortest path planner in (a) and by the BRM planner in (b). In each case, the solid blue line shows the prescribed path, small blue circles are UWB sensor beacons and the green ellipses are the predicted covariances of the robot position estimate during execution. The shortest path is direct, but grows very uncertain. The BRM path enables the robot to stay well localized at the cost of slightly increased length.

a direct route but does not consider uncertainty. As a result, the uncertainty grows very large, as indicated by the green covariance ellipses. The BRM algorithm, on the other hand, chooses a path that results in less uncertainty, as shown in Figure 6-3(b). While this path is less direct, it traverses areas of high sensor density which are amenable to localization and more likely to yield reliable performance.

6.4 Conclusion

In this chapter we have presented the Belief Roadmap (BRM) algorithm, which is a belief space variant of the Probabilistic Roadmap algorithm that substantially improves planning performance and positional accuracy during execution. We have shown that the computational cost of EKF predictions during search can be reduced by pre-computing "one-step" covariance transfer functions that combine multiple EKF update steps into a single process. We demonstrated that this insight decreases the time required to search during planning by over two orders of magnitude. We have also shown that, by incorporating uncertainty into planning, the BRM chooses higher quality paths than a traditional shortest path planner and yields improved reliability during execution. Finally, we demonstrated the BRM algorithm on a large-scale planning problem and showed that we could plan efficiently in very large spaces.

Chapter 7

Conclusion and Future Work

Contents

This chapter concludes the thesis with a discussion of future work and a summary of the contributions made during this project. In Section 7.1 we suggest future exploration into improvements and extensions of the range-based localization algorithm presented in Chapter 3. Section 7.2 consists of further discussion of belief space planning, in which we consider immediate applications and extensions of the Belief Roadmap algorithm and also propose areas for future work. In Section 7.3, we conclude the work presented in the thesis.

7.1 Future Work in Range-Based Localization

In Chapter 3, we demonstrated that the Rao-Blackwellized particle filter can be used to improve the scalability and performance of Markov Chain Monte Carlo methods for range-based localization with hidden bias estimation. The promising results obtained with this approach suggest future exploration in several domains:

- **Improved Bias Transition Model:** In our experiments we implemented the range bias belief as a multinomial distribution in an attempt to capture the notch-shaped transition probabilities (Section 3.2.2) observed in sensor measurements on a moving robot. The control update of this distribution requires updating each bias bin of each multinomial maintained in a given particle. The computational complexity of this update could be reduced by finding an alternative bias belief model which admits closed form updates. In future work, we seek to explore distributions from the exponential family which may fulfill this requirement.

- **Sensor Coverage Map:** While our range-based localization algorithm is designed to operate in environments with limited map information, in some applications it may be possible to build an *a priori* sensor coverage map with regional bias characterization. With this additional information, it is likely that our approach could yield positioning results with greatly increased accuracy.

- **Multi-agent Localization:** In future research, our approach could be extended to accommodate dynamic sensor locations in problems such as multi-agent localization using a cooperative UWB network.

- **Real-World Testing:** We seek to perform future tests in real-world scenarios and to explore alternative sensor models based on further experimentation. Additionally, we intend to pursue real-world system integration tests, using the BRM planning algorithm for navigation.

7.2 Future Work in Belief Space Planning

The BRM algorithm not only presents an effective solution to the motion planning problem in belief space, but it also exposes new avenues for research and has great potential for further applications. In this section, we explore some additional considerations of the BRM algorithm, propose immediate applications that would require very little modification and discuss potential extensions for future work.

Immediate Applications to Other Problem Domains

There appear to be several immediate applications and extensions of the BRM technique:

- **Dynamic Environments:** The efficiency of search makes on-line re-planning very feasible. One advantage of this efficiency is the ability to consider certain forms of dynamic constraints during search. Dynamic obstacles that create trajectory occlusions could be accounted when considering possible trajectories during search. It is possible to account for other constraints that affect the sensor or motion models, however this would require recomputing one-step descriptors for affected trajectories, which is more costly.

- **High-Dimensional Belief Space and General PRM Replacement:** In this work, we limited the scope of our exploration to the 2-dimensional motion planning problem. We also were primarily concerned with finding trajectories with greater potential for information gain from observations, since in the motion model for a 2D mobile robot most atomic control actions have a similar incremental effect on uncertainty. However, the BRM algorithm is generally applicable to higher-dimensional problems *and* problems where control uncertainty is dominant. For example, the BRM could be applied to the motion planning problem for a n-DOF robot arm with no observation feedback. If uncertainty costs vary significantly in the dimensions of the control space, the BRM could generate much more reliable plans than a naive PRM-based alternative.

- **Choosing Between Goals:** Given a set of multiple goals $\{\mu_1, ..., \mu_j\}$, the BRM could be modified to simultaneously find the minimum achievable covariance $\{\Sigma_1^{min}, ..., \Sigma_j^{min}\}$ at each goal and then choose the minimum of this set as $min(\{\Sigma_1^{min}, ..., \Sigma_j^{min}\})$. This would correspond to performing Algorithm 16 on the largest connected component $\mathcal{G}_{cc}^{\star}$ of the trajectory graph $\mathcal{G}$ that contains the start location and all goals $\{\mu_1, ..., \mu_j\}$, such that $\mathcal{G}_{cc}^{\star} \equiv \mathcal{G}_{cc}^{1} \cup \cdots \cup \mathcal{G}_{cc}^{j}$.

- **Finding Goals:** The results of a BRM search can be used to identify "safe" goal locations, or nodes within the BRM graph that have low expected uncertainty.

- **Finding Uncertain Regions:**

  Identifying Points for Exploration: Instead of finding "safe" goals, the results of a BRM search can be used to identify uncertain locations for further exploration.

  Graph Pruning: After applying the BRM, we could improve efficiency during re-planning by pruning regions of the graph with high uncertainty that are likely to be undesirable during re-planning. By reducing the size of the trajectory graph to areas we would likely use, search during re-planning could be made more efficient.

Guide Sampling to Uncertainty: After applying the BRM algorithm, we can actively identify regions of the solution path in which the robot is likely to encounter the greatest uncertainty. Further applications of the BRM on a more local level at a higher sampling resolution could be used to generate better plans for avoiding uncertain regions. This leads directly to the hierarchical BRM approach suggested below.

- **Trajectory Optimization via Hierarchical BRM:** A hierarchical BRM could be used to address two issues of the standard BRM: a fundamental limitation of PRM-based approaches is that the efficiency of search depends on the number of nodes in the graph; and, BRM solutions represent coarse, straight-line trajectories that suggest further optimization. By applying a coarse-to-fine methodology to the BRM, the BRM can be broken into modular subproblems (e.g. individual trajectory arcs) that can be each solved efficiently with additional applications of the BRM at higher resolution. A coarse trajectory search could first be employed to identify the best (or n best) high-level path suggestions. Further applications of the BRM could be employed sequentially on smaller regions of the belief space, effectively solving modular POMDP's with boundary constraints. Additionally, these fine-grained searches at higher resolution could be prioritized to regions of the path with the highest expected uncertainty.

  One consideration of a hierarchical BRM is that guarantees of search optimality in the BRM are limited to the resolution of a given search. It is not immediately clear what global guarantees or bounds could be made in a modular approach of varying resolution.

Future Work

There appear to be several natural domains for improving the BRM in future work. We begin by listing some considerations of the BRM algorithm that should be addressed:

- *Generalizing the BRM to Other Sensor Models:* The following considerations must be taken into account when attempting to use the BRM algorithm with other sensor models:

  $\mapsto$ is the sensor model linearizable?

$\mapsto$ does the maximum likelihood observation make sense?

$\mapsto$ is it reasonable to simulate measurements?

- *Assumption of Maximum-Likelihood Observation:* A crucial assumption in the BRM formulation is that of the maximum-likelihood observation, which is necessary to ensure a linear EKF mean update. This assumption is reasonable when: (a) the sensor model does not yield large deviations in potential measurement information in local regions; and (b), when the expected belief uncertainty is small enough that error in the state estimate does not cause the mean-predicted-observation to have much error. Ideally we would like to consider potential observations based on the belief uncertainty of the robot. If the belief uncertainty is large, it becomes increasingly likely that the actual state of the robot is farther away from the mean. Thus, we should consider the likelihood of receiving observations in a spatial region of the belief space that accounts for this uncertainty.

- *Assumption of Linear Controls:* While the linear EKF control update of the mean is reasonable for a mobile robot with differential-drive, it may not be reasonable in other applications with nonholonomic kinematic constraints. Further, the linear-Gaussian assumption of controls may not be reasonable in some problem domains, such as UAVs.

UKF Extension of the BRM

A natural BRM extension to address these issues would involve incorporating the unscented Kalman filter (UKF) [25] to capture the variability of observations and controls over a local spatial region defined by the covariance. The UKF improves upon the linearization technique with a compromise between sampling-based methods and the linear-Gaussian assumption. The UKF creates two *Sigma points* for each dimension of the state, which are placed on each side of the mean at a distance and weight that are functions of the prior covariance Σ_{t-1}. The non-linear control and measurement steps update each individual Sigma point, capturing regional non-linear effects about the mean. At the end of a UKF iteration, the filter reconstructs the resulting mean u_t and covariance Σ_t from the updated Sigma points.

Our goal in future work would be to apply the UKF to the BRM algorithm to capture the variability of observations (and potentially controls) over a local spatial region. Ideally, we would like to use the UKF instead of the EKF; however, the UKF formulation is completely coupled with the prior covariance at each step. This is problematic, because the efficiency of the BRM algorithm relies on the ability of EKF updates to be decoupled from initial conditions. In future work, we seek to explore in detail the UKF formulation in search of a reasonable compromise between EKF and UKF theory for application to the BRM.

Additional Areas for Future Work

Other areas for future work could include the following:

- **Extend BRM to Uncertain Maps** It may be possible to extend the BRM to uncertain maps for the purposes of exploration with SLAM. Similar techniques currently use the entropy as a measure of the expected information gain of taking specific actions. In [48], a cumulative measure of entropy is computed along potential action trajectories to choose actions that simultaneously maximize the expected information gain in terms of unexplored map territory and robot localization. In another integrated exploration approach [35], average entropy is computed at each location in a grid-map for use as one component of the overall exploration objective function. In an EKF-SLAM-based technique, it may be possible to sample and build trajectories from the map distribution and then use a variant of the BRM algorithm to predict uncertainty and identify goals of interest.

- **Trajectory Optimization:** In addition to the hierarchical BRM method suggested above, trajectory optimization could also take the form of a greedy algorithm along local trajectories. Such an algorithm could follow the gradient of information and deviate from the straight-line trajectory, subject to arriving at the node location at the end of the trajectory. It may be possible to determine this gradient using the linear, symplectic descriptor matrix for each filter update.

- **Sensor-Specific Sampling Strategies:** It is likely that belief space planning would be improved by developing sampling strategies that are aware of the sensor model. Current sampling strategies are naive to sensor information, meaning that belief uncertainty is only considered during the search phase. It may be possible to use the sensor model to bias the build phase toward regions of the belief space that are likely to have better information properties. As a result, "better" trajectories would be considered during search and the overall algorithm might find paths that obtain less goal uncertainty.

- **Stability Analysis of Trajectories:** The one-step methods developed for use in the BRM algorithm present advantageous mathematical properties for stability analysis of trajectories. The eigenvalues of the Hamiltonian and Symplectic one-step matrices can be used directly to characterize the stability of covariance growth along trajectories. In terms of control, the one-step descriptors represent a transfer function of covariance along a trajectory. It may be possible to borrow tools from control theory to analyze and make use of the properties of this transfer function. Potential applications could include assessing the stability of a cyclic trajectory or using the eigenvalues as a heuristic during search to bias towards trajectories that are more stable.

- **Improve Efficiency of Search Process:** In this work, we limited our focus to a breadth-first search strategy, but it is possible that more efficient searches exist. Future research could lead to admissible heuristics that would allow a more efficient, $A^{\star}$-based search to be used.

7.3 Conclusion

In Chapter 3, we presented an improvement for Markov Chain Monte Carlo approaches to range-based localization with hidden bias estimation by using the Rao-Blackwellization technique. We showed that this method enables the joint estimation problem over robot poses and range biases to be decomposed into two simpler estimation problems. Consequently, the number of samples required to maintain an accurate state estimate is limited to exponential dependence on only the constant dimensional space of robot poses; the range bias distributions can be analytically computed for each robot pose hypothesis. We demonstrated in experiments that use of Rao-Blackwellization substantially improves the scalability and performance of a standard particle filter approach to UWB range-based localization.

In the Chapters 4-6, we incrementally developed a solution to the problem of belief space planning for linear-Gaussian POMDP's. In Chapter 4, we presented a tractable formulation of the belief space planning problem posed as search over belief trajectories. We showed that it is possible to isolate the mean-belief to well-defined trajectories in belief space by using a linearized version of the Kalman-filter for belief tracking. We then developed a hybrid Probabilistic Roadmap with a corresponding search process that was capable of finding a series of belief trajectories that obtain minimal covariance at a goal location. The efficiency of this search process was limited by the multi-step Kalman filter updates required to propagate an initial covariance along each branch of the search tree. In Chapter 5, we showed that the EKF admits one-step covariance updates, which enables the multi-step cost to be shifted to the build phase, leading the drastically reduced on-line search times. This culminated in the Belief Roadmap (BRM) algorithm presented in Chapter 6, which we have shown substantially improves planning performance and positional accuracy over traditional planners that ignore uncertainty. We demonstrated the BRM algorithm with UWB ranging in a very large planning problem, which we believe to be considerably larger than existing results [43].

Bibliography

[1] H. Abou-Kandil. *Matrix Riccati Equations in Control and Systems Theory*. Birkhauser, 2003.

[2] C.D. Ahlbrandt and A.C. Peterson. *Discrete Hamiltonian Systems: Difference Equations, Continued Fractions, and Riccati Equations*. Kluwer Academic Publishers, 1996.

[3] S. Al-Jazzar and J. Caffery Jr. ML and Bayesian TOA location estimators for NLOS environments. *Vehicular Technology Conference, 2002. Proceedings. VTC 2002-Fall. 2002 IEEE 56th*, 2, 2002.

[4] S. Al-Jazzar, J. Caffery Jr, and H.R. You. A scattering model based approach to NLOS mitigation in TOA location systems. *Vehicular Technology Conference, 2002. VTC Spring 2002. IEEE 55th*, 2, 2002.

[5] H. Arslan, Z. N. Chen, and M.G.D. Benedetto. *Ultra Wideband Wireless Communication*. John Wiley and Sons, 2006.

[6] KJ Astrom. Optimal control of Markov decision processes with incomplete state estimation II. *Journal of Mathematical Analysis and Applications*, 26:403–406, 1969.

[7] Y. Bar-Shalom and X.R. Li. *Estimation and Tracking: Principles, Techniques, and Software*. Artech House, 1993.

[8] R. Bohlin and LE Kavraki. Path planning using lazy PRM. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, 1, 2000.

[9] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. *Proc. AIPS*, pages 52–61, 2000.

[10] D. Cassioli, MZ Win, and AF Molisch. The ultra-wide bandwidth indoor channel: from statistical model to simulations. *Selected Areas in Communications, IEEE Journal on*, 20(6):1247–1257, 2002.

[11] FC Commision. First report and order. *FCC 02*, 48, 2002.

[12] T. Domain. Pulson technology overview. Technical report, TimeDomain Inc., 2001.

[13] F. Doshi. Wheelchair robot. http://people.csail.mit.edu/finale/wiki/index.php/robotic_wheelchair.

[14] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice.* Springer-Verlag, 2001.

[15] A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 176–183, 2000.

[16] A.I. Eliazar and R. Parr. Learning probabilistic motion models for mobile robots. *ACM International Conference Proceeding Series*, 2004.

[17] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte Carlo localization: Efficient position estimation for mobile robots. *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 113:114, 1999.

[18] E. Frazzoli, M.A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[19] S. Gezici, H. Kobayashi, and H.V. Poor. A new approach to mobile position tracking. *Proc. IEEE Sarnoff Symp. Advances in Wired and Wireless Communications*, pages 204–207, 2003.

[20] S. Gezici, Z. Tian, GB Giannakis, H. Kobayashi, AF Molisch, HV Poor, and Z. Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *Signal Processing Magazine, IEEE*, 22(4):70–84, 2005.

[21] F. Gustafsson and F. Gunnarsson. Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements. *Signal Processing Magazine, IEEE*, 22(4):41–53, 2005.

[22] C. Holleman and LE Kavraki. A framework for using the workspace medial axis in PRM planners. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, 2, 2000.

[23] D. Hsu, J.C. Latombe, and H. Kurniawati. On the Probabilistic Foundations of Probabilistic Roadmap Planning. *The International Journal of Robotics Research*, 25(7):627, 2006.

[24] Damien B. Jourdan, John J. Deyst Jr., Moe Z. Win, and Nicholas Roy. Monte carlo localization in dense multipath environments using uwb ranging. In *Proceedings of the IEEE International Conference on Ultra-Wideband (ICU 2005)*, Zurich, Switzerland, September 2005.

[25] S.J. Julier and J.K. Uhlmann. A new extension of the Kalman filter to nonlinear systems. *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, 3, 1997.

[26] T. Kailath, B. Friedlander, and L. Ljung. Scattering theory and linear least squares estimation—II discrete-time problems. *J. Franklin Institute*, 301:77–82, 1976.

[27] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[28] LE Kavraki, P. Svestka, J.C. Latombe, and MH Overmars. Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[29] J. Kuffner, K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Motion planning for humanoid robots. *Int. Symp. Rob. Res., Siena, Italy*, 2003.

[30] AM Ladd, KE Bekris, AP Rudys, DS Wallach, and LE Kavraki. On the feasibility of using wireless ethernet for indoor localization. *Robotics and Automation, IEEE Transactions on*, 20(3):555–559, 2004.

[31] B.L. Le, K. Ahmed, and H. Tsuji. Mobile location estimator with NLOS mitigation using Kalman filtering. *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 3, 2003.

[32] J.Y. Lee and RA Scholtz. Ranging in a dense multipath environment using an UWB radio link. *Selected Areas in Communications, IEEE Journal on*, 20(9):1677–1683, 2002.

[33] L. Ljung, T. Kailath, and B. Friedlander. Scattering theory and linear least squares estimation—Part I: Continuous-time problems. *Proceedings of the IEEE*, 64(1):131–139, 1976.

[34] T. Lozano-Perez. Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

[35] A.A. Makarenko, S.B. Williams, F. Bourgault, and H.F. Durrant-Whyte. An experiment in integrated exploration. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 534–539, 2002.

[36] P.E. Missiuro and N. Roy. Adapting probabilistic roadmaps to handle uncertain maps. *IEEE Int. Conf. Rob. Aut*, 2006.

[37] M. Montemerlo, N. Roy, S. Thrun, D. Haehnel, C. Stachniss, and J. Glover. Carnegie mellon robot navigation toolkit (CARMEN). http://carmen.sourceforge.net.

[38] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 593–598, 2002.

[39] I.R. Nourbakhsh, R. Powers, and S. Birchfield. DERVISH - An Office-Navigating Robot. *AI Magazine*, 16(2):53–60, 1995.

[40] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. *International Joint Conference on Artificial Intelligence (IJCAI)*, 13, 2003.

[41] EA Prigge and JP How. Signal architecture for a distributed magnetic local positioning system. *Sensors Journal, IEEE*, 4(6):864–873, 2004.

[42] R. Redheffer. On the relation of transmission-line theory to scattering and. *J. Math. Phys.*, 41:1–41, 1962.

[43] N. Roy and S. Thrun. Coastal navigation with mobile robots. *Advances in Neural Processing Systems*, 12:1043–1049, 1999.

[44] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha. Tracking moving devices with the cricket location system. *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 190–202, 2004.

[45] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 520–527, 2004.

[46] E.J. Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford, 1971.

[47] M.T.J. Spaan and N. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, 2005.

[48] C. Stachniss, G. Grisetti, and W. Burgard. Information Gain-based Exploration Using Rao-Blackwellized Particle Filters. *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[49] B. Hassibi T. Kailath, A.H. Sayed. *Linear Estimation*. Prentice-Hall, 2000.

[50] S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

[51] D. Vaughan. A nonrecursive algebraic solution for the discrete Riccati equation. *Automatic Control, IEEE Transactions on*, 15(5):597–599, 1970.

[52] G. Verghese, B. Friedlander, and T. Kailath. Scattering theory and linear least-squares estimation, part III: The estimates. *Automatic Control, IEEE Transactions on*, 25(4):794–802, 1980.

[53] MZ Win and RA Scholtz. Impulse radio: how it works. *Communications Letters, IEEE*, 2(2):36–38, 1998.

[54] MP Wylie and J. Holtzman. The non-line of sight problem in mobile location estimation. *Universal Personal Communications, 1996. Record., 1996 5th IEEE International Conference on*, 2, 1996.