Planning in Information Space for a Quadrotor Helicopter in a GPS-denied Environment

by

Ruijie He

B.S. Masssachusetts Institute of Technology (2007)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Masters of Science in Aeronautics and Astronautics

at the

Massachusetts Institute of Technology

June 2008

Author ....
Department of Aeronautics and Astronautics
May 20, 2008

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

Accepted by ....
David Darmofal
Chairman, Department Committee on Graduate Students

Planning in Information Space for a Quadrotor Helicopter in a GPS-denied Environment

by

Ruijie He

Submitted to the Department of Aeronautics and Astronautics
on May 20, 2008, in partial fulfillment of the
requirements for the degree of
Masters of Science in Aeronautics and Astronautics

Abstract

Unmanned Air Vehicles (UAVs) have thus far had limited success in flying autonomously indoors, with the exception of specially instrumented locations. In indoor environments, accurate global positioning information is unavailable, and the vehicle has to rely on on-board sensors to detect environmental features and infer its position. Given that a vehicle small enough to fly indoors can only carry a limited sensor payload, the vehicle's ability to localize itself varies across different environments, since different surroundings provide varying degrees of sensor information. Therefore, a vehicle that plans a path without regard to how well it can localize itself along that path runs the risk of becoming lost.

My research focuses on how path-planning can be performed to minimize localization uncertainty, and works towards developing a motion-planning algorithm for a quadrotor helicopter. As a starting point, I apply the Belief Roadmap (BRM) algorithm, an information-theoretic extension of the Probabilistic Roadmap algorithm, incorporating sensing during the path-planning process. I make two theoretical contributions in this research. First, I extend the original BRM to use non-linear state inference via the Unscented Kalman Filter, providing better approximation of the non-linearities of laser sensing onboard the UAV. Second, I develop a sampling strategy for the BRM, minimizing the number of samples required to find a good path. Finally, I demonstrate the BRM path-planning algorithm on a quadrotor helicopter, navigating the vehicle autonomously in an indoor environment.

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

Acknowledgments

This thesis would not have been possible without the support of a large group of people. I would like to take this opportunity to express my heartfelt gratitude to them.

- The Republic of Singapore Armed Forces for graciously funding my years of education at MIT, without which I would never have had the opportunity of experiencing the MIT education experience

- My adivsor, Professor Nicholas Roy, not only for providing much of the inspiration, direction and advice throughout my research, but also for his readiness to get his hands dirty by helping out with the numerous software and hardware issues whenever they occurred during the research

- Sam Prentice and Abe Bachrach, my partners-in-crime at the Micro Aerial Vehicle 08 (MAV08) Competition, as well as many of my labmates Matt Walter, Tom Kollar, Sooho Pak, Jared Glober, Valerie Gordeski, Emma Brunskill, Albert Huang and Finale Doshi, for teaching me all that I currently know about robotics, as well as for all the help and advice that they have rendered.

- Daniel Gurdan, Jan Stumpf and the rest of the Ascending Technologies company, for not only creating such an impressive flying machine, but also for their technical assistance with the quadrotor.

- Aero/Astro faculty and staff, for all their encouragement and support over my four years at MIT. In particular, special thanks goes to Marie Stuppard, Barbara Lechner, Col. Peter Young, William Litant, Todd Billings and Dick Perdichizzi for their indispensible help and support.

- The numerous friends who have supported me through my four years in Boston - Guo-liang, Shirleen, Wenxian, Serene Lee, Kah Seng, Rika, Shiming, Charlene, Yves, Serene Hung, Wee Lee, Charlene, Kah Keng, Sarah & Jianlong

Finally, and most importantly, to Mum, Dad, and Ruimin, for their unconditional love, care and concern, making me who I am today.

Contents

List of Figures

Chapter 1

Introduction

This chapter provides a brief motivation for the research problem tackled in this thesis, before making explicit the problem statement addressed and the hypothesis proposed in this research. We then highlight the key contributions that were made in the thesis, and finally conclude with a roadmap for the rest of this document.

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) are being used increasingly in both military and civilian applications. Today, there is widespread application of these vehicles in outdoor environments, including military surveillance operations, agriculture crops monitoring, forest fire watch, ocean and weather observation, as well as disaster relief coordination.

UAVs rely heavily on accurate knowledge of their position in a global environment for effective execution of their tasks, be it control, path planning, obstacle avoidance and/or path execution. In this regard, outdoor UAVs have leveraged off Global Positioning System (GPS) infrastructure to obtain accurate estimates of their position in the world, and unsurprisingly, the UAV community has also expended much efforts to improve the accuracy and availability of such systems in generic environments.

Unfortunately, the GPS system has not managed to be pervade our world completely; important areas such as urban canyons and indoor environments, which do not have a wide enough field-of-view of the clear sky, are still without GPS access. As such, autonomous

(a) SICK laser range-finder

(b) Miniature Camera

(c) Hokuyo Laser range-finder

Figure 1-1: Different options for onboard sensors

UAVs have thus far been incapable of flying in these environments, even though there are numerous applications for indoor autonomous UAVs. For example, during a search and rescue operation, the dispatch of an autonomous UAV will not only enable a dangerous rescue mission to be accomplished with minimal risk to first respondents, but will also allow for the operation of these vehicles in small spaces that were previously inaccessible to humans.

This is not to suggest that significant work has not been accomplished for achieving localization using onboard sensors. Sonar sensors, laser range-finder sensors and camera sensors have proven to be extremely successful for agents to recover information of their immediate environments, in order to make inferences about their position relative to the world. For instance, the SICK laser range-finder sensor (Figure 1-1(a)) is now considered commodity technology for most robotic platforms. Such sensors are now widely used onboard indoor ground robots, enabling them to perform a wide range of tasks including mapping, localization and path-planning.

Unfortunately, UAVs have thus far not had similar success in using these sensors. The key limitation is the vehicle's payload; because we desire to operate in an often-cluttered indoor environment such as an office environment, we not only have to rule out fixed-wing aircrafts that require a non-trivial minimum horizontal velocity, but the helicopters we seek to use also have an upper size limit of approximately 70cm in diameter. Together, these constraints severely limit the payload that these vehicles can carry. For example, the SICK laser range finder weighs approximately 4.5kg, making it impossible for it to be used onboard an indoor helicopter platform. In addition, due to noise constraints, the indoor environment limits us to electric-powered vehicles, whose batteries further reduce both the sensing payload capability and power capacity. In sum, a UAV that is small enough to fly through an indoor or populated urban environment safely can only carry a very limited sensor payload.

A second reason why we have not yet witnessed the widespread use of indoor UAVs is the need for consistent, active control of the vehicle just to keep it hovering in one position. In direct contrast to ground robots, which in general will remain stationary if no control is given, UAVs will not self-stabilize without active control feedback. Together with the bandwidth of the communication protocols and computational speed of the computer processors, this requirement places additional constraints on both the types of sensors that can be used, as well as the maximum computational complexity afforded to our algorithms. For example, although miniature cameras (Figure 1-1(b)) are lightweight, the post-processing required is often too computationally intensive for active control of the vehicle in an indoor environment.[1]

These constraints therefore limit the type of sensors that can be carried onboard the helicopter. Thankfully, the UAVs can still carry *some* sensing capability; they just cannot

[1]We recognize that Christopher Kemp [1] has successfully accomplished hovering of a quadrotor helicopter using a miniature camera, although significant issues remain.

carry sensors that enable them to either localize *everywhere* or do so with low uncertainty. In our problem, for instance, the Hokuyo laser range finder (Figure 1.1(c)) weighs significantly less than the SICK laser range finder, but suffers the tradeoff of a shorter range, a smaller field-of-view, and a lower update rate.

These sensor limitations imply that if we do not take the sensor model into account when planning a trajectory path for the vehicle, there is a very high probability that the vehicle would get lost easily and would not reach its desired end goal. If the UAV finds itself in the middle of wide, open space, for example, the returns from its localization sensors do not provide any meaningful information of its position in the world. Due to the noise inherent in UAV control, by the time the UAV obtains a meaningful measurement from its sensors, the UAV's uncertainty of its pose may make it impossible for it to re-localize accurately.

However, if we instead explicitly model the type of sensors that we are using for online localization of the vehicle, we can make better predictions of the likelihood that the vehicle will be able to execute the planned trajectory successfully. By using the model of our sensors to incorporate expected measurements that the UAV will obtain when it executes a particular path, we can more accurately evaluate the value of each of these individual paths, and hence plan trajectories for the UAV that are robust to the inherent sensor limitations on the vehicle.

1.2 Project Overview

In this research, we sought to develop path-planning algorithms to achieve autonomous indoor navigation for our quadrotor helicopter, developed by Ascent Technology [2], and shown in Figure 1.2. The vehicle is outfitted with a Hokuyo laser rangefinder sensor (Figure 1.1), which is capable of estimating position, heading, and altitude information from environmental features that exist within a 240° field-of-view and fall within a 4m radius [3] from the helicopter. The laser sensor returns 769 range scans at 10Hz, and these readings can then be re-projected into state space for visualization, as shown in Figure 1.2.

To date, robust and scalable methods that incorporate uncertainty into the planning

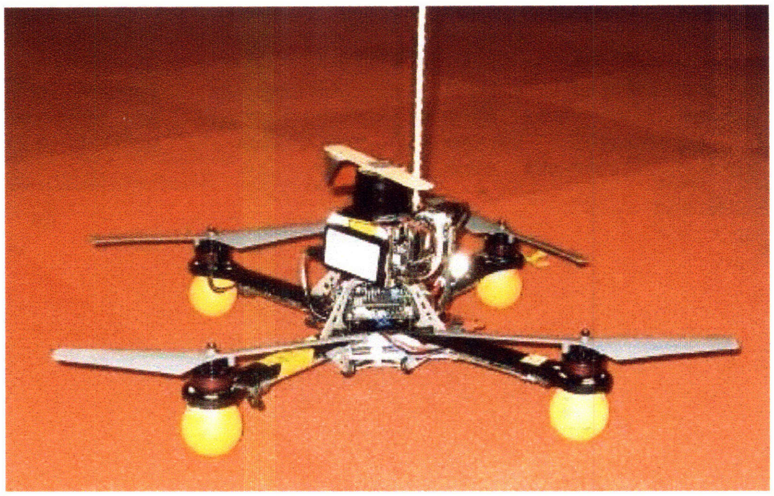


Figure 1-2: Our Quadrotor Helicopter

process is still an area of open research. As a starting point, we begin with the Belief Roadmap (BRM) Algorithm [4], an information theoretic extension of the popular Probabilistic Roadmap Algorithm (PRM) [5]. The Belief Roadmap Algorithm is a novel and efficient approach for planning in belief space, utilizing the linear-Gaussian assumption to incorporate uncertainty into planning. By planning in the space of robot beliefs, the optimal sequences of actions can be chosen based on the expected uncertainty of executing different sets of actions.

The BRM algorithm uses the symplectic form of the Extended Kalman Filter (EKF) to find the path for the agent that minimizes its expected cost. The EKF is a very popular variation of the Kalman Filter, performing a linearization around the mean estimate in order to provide an estimate when non-linear transition and observation functions are involved. However, the Extended Kalman Filter breaks down in highly non-linear situations, because simply linearizing the control or measurement functions leads to a poor approximation.

For instance, in our problem, the measurement vector is made up of the 769 laser range measurements that are returned with laser range scan. An EKF implementation would assume that each of the values is a separate, independent measurement, even though in reality, many of these laser returns could constitute the same obstacle in a particular environment. To overcome the independence assumption, EKF localization would require high-level features such as walls and corners to be extracted for the computation of both the expected

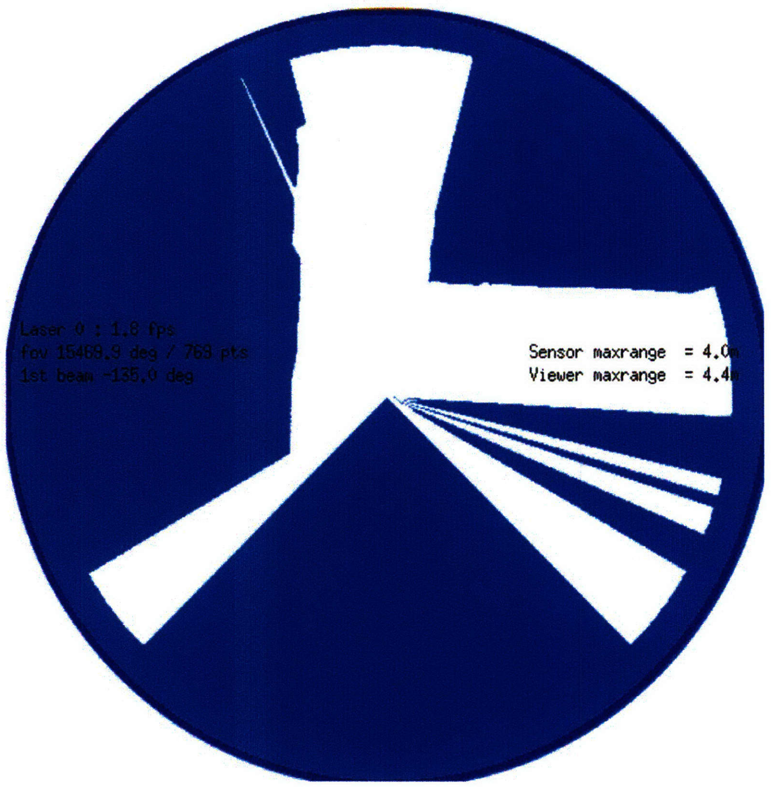


Figure 1-3: Sample return laser scan

measurement vector and its gradient. Unfortunately, since we desire to use the algorithm in any generic environment, not simply those with nice, well-defined features, the feature extraction process can become difficult. Instead, we propose using an alternative data fusion algorithm, the Unscented Kalman Filter (UKF) [6], which is able to provide a better approximation of the non-linearities in the system.

In addition, because the BRM incorporates uncertainty during the planning process by generating the expected laser measurements along each possible path from the start to the end goal, the computational complexity of the algorithm is significant. The algorithm can therefore be made a lot efficient if we minimize the size of the graph while ensuring that the optimal path is still contained in the graph. In this regard, we observe that the BRM is similar to the PRM in that it uses a sampling-based strategy for representing the collision-free positions in a given map, and that the original BRM algorithm samples the configuration space uniformly and tests if the sample is in the collision-free subset of the map. While this adequately represents the free space C_{free} in the map, many of the samples in reality do not lie on the optimal path for the UAV, if we are concerned with minimizing the uncertainty of the UAV's position. In this work, we therefore propose a sampling strategy that takes into account the expected reduction in uncertainty from sensing at each of the samples, and uses this metric to decide whether to include a sample in the graph.

1.3 Problem Statement

More formally, we seek to develop a motion planning algorithm that enables us to execute autonomous control of a quadrotor helicopter in a three-dimensional, GPS-denied indoor environment. The following is known about the environment:

- A three-dimensional map of the environment, known to accuracy of 10cm
- Onboard Inertial Measurement Unit (IMU), transmitting translational acceleration and rotational velocity data at 20Hz
- Hokuyo laser-rangefinder sensor, with 4m effective range, returning laser scans of 240° in a plane at 10hz

Because the global state of the vehicle in the environment is not known, we have to infer where the vehicle is in the global environment based solely on the local sensor information that is returned from the vehicle.

1.4 Hypothesis

It is possible to develop path-planning algorithms for a quadrotor helicopter with only limited local sensors, such as a Hokuyo laser range-finder, that minimizes the likelihood that the vehicle will get lost during path execution.

From the theoretical standpoint, we hypothesize that the Belief Roadmap Algorithm can be extended to applications where both the localization sensors that are being used, as well as the vehicle motion, contain high non-linearities in their measurement and process models respectively.

In addition, we hypothesize that we can reduce the computational complexity of the BRM algorithm significantly if we discriminate in our choice of pose samples that are used to create the BRM search graph.

1.5 Contributions

Three key contributions were made in this thesis:

1. We extended the Belief Roadmap Algorithm to use the Unscented Kalman Filter (UKF) for position tracking, thereby providing more accurate approximations of the non-linearities that exist in UAV motion and laser sensing

2. We propose a sampling strategy for the Belief Roadmap Algorithm, which we call the "Sensor Uncertainty Sampling" strategy, that is based on the expected information gain of different positions in a given map. We show that this strategy performs favorably compared to other sampling strategies in the literature

3. Finally, as a proof of concept, we test the above algorithms onboard the quadrotor helicopter and demonstrate its ability to navigate autonomously in a GPS-denied indoor environment.

The majority of the research reported in this thesis is also contained in [7], which has been accepted for presentation at the 2008 International Conference of Robots and Automation, to be held in Pasedena, CA, May 19-23, 2008.

1.6 Roadmap

The rest of the thesis describes the above three contributions in greater detail, providing the necessary background and experimental results. Chapter 2 first provides a detailed description of the original Belief Roadmap Algorithm, which is a new approach to belief space planning that incorporates uncertainty in the planning process.

Chapters 3 and 4 present the theoretical contributions of this research. Chapter 3 first shows how the BRM algorithm can be extended to use the Unscented Kalman Filter localization technique for state estimation, thereby enabling us to implement the BRM algorithm on our laser-equipped quadrotor helicopter, and also producing a better state estimate of the vehicle's position. Chapter 4 then focuses on reducing the computational complexity of the BRM algorithm by proposing a new sampling strategy, the "Sensor Uncertainty" sampling strategy, for representing the collision-free space of the map. We also compare our proposed sampling strategy with alternative sampling strategies to highlight the performance improvements that we can achieve.

Chapter 5 describes the details of implementing our algorithm onboard our quadrotor helicopter, as well as provides experimental results for experiments that were conducted onboard the helicopter.

Finally, chapter 6 summarizes the thesis and suggests future work for further research.

Chapter 2

The Belief Roadmap Algorithm: *Planning in Information Space*

In this chapter, we lay the foundations for subsequent appreciation of the contributions made in the research. In particular, we present the Belief Roadmap Algorithm (BRM), a recent algorithm proposed by Samuel Prentice and Nicholas Roy [4] that addresses the problem of trajectory planning when the full state of the vehicle is unknown. We provide a brief motivation for the algorithm, describe some of the underlying algorithms that are used in the BRM, and finally discuss some of the limitations in the algorithm that we attempt to address in subsequent chapters.

2.1 Introduction

Path-planning is one of the fundamental problems in robotics, seeking to find the minimum cost path for an agent from a start to end goal. Indeed, a wide variety of algorithms have been proposed by the research community. Very broadly speaking, the algorithms can be sub-divided into map decomposition techniques (Probabilistic Roadmap Algorithm, Voronoi diagrams, Cell decomposition, etc.), graph search techniques (Breadth-first search, Depth-first search, A* search), and those that attempt to do both simultaneously (Mixed Integer Linear Programming).

However, these algorithms assume that the full state of the agent is known. For a robot maneuvering in a given environment, this implies that at every instant in time, the robot has accurate knowledge of its pose in the world, such as its global coordinates in a map. Yet, in real-world applications, such information is often unavailable. For example, in an indoor environment, the absence of accurate GPS data prevents the robot from directly obtaining information of its location in the world. Instead, the agent has to use whatever information it can obtain, such as through the use of other local sensors, to maintain an estimate, also known as a *belief state* [8], of its position in the environment, along with some uncertainty of the accuracy of that estimate, relative to its true position in the world.

Taking the uncertainty of the agent's estimate is important for path-planning. In particular, incorporating this knowledge of uncertainty into the planning process can lead to increased robustness of an autonomous robot, thereby improving the overall performance of the system. Unfortunately, most of the algorithms that currently exist have been unable to account for this uncertainty in a manner that is both computationally scalable and does not require the discretization of the *belief space.* Although considerable progress has been made in algorithms such as the partially observable Markov Decision Process (POMDP) [8], planning optimally in the face of uncertainty has not been as successful when attempting to address large real-world problems.

In light of the existing state of the research, the BRM presents an attractive method for planning in belief space that enables the efficient computation of both the reachable belief space and the path that has the minimum expected cost. Inspired by the Probabilistic Roadmap Algorithm (PRM) and the Extended Kalman Filter (EKF), the BRM demonstrates how the set of reachable belief states can be constructed in a way that allows for efficient repeated querying. This enables path searches to be found in a time that only grows linearly with the number of edges in the graph, thereby leading to greatly improved planning computation times.

2.2 Problem Formulation

Let us first formally state the problem that the BRM is attempting to solve. The traditional path-planning problem involves finding the minimium-cost collision-free path from a start state s_0 to a goal state s_g, given a map $\mathcal{M}$ of the environment. In this traditional setup, the objective cost function that is being minimized is

$$J(s_0, u_0, ...s_T, u_T) = \sum_{t=0}^{T} C(s_t, u_t) \tag{2.1}$$

where $J(...)$ refers to the cost of a particular path, and $C(s_t, u_t)$ is the cost of executing control u_t at state s_t at time t, and is often approximated by the distance travelled at each time step.

In contrast to the traditional path-planning problem, however, the belief space equivalent of the path-planning problem assumes that the state, s_t, of the agent at time t is not fully known. Instead, the agent's only knowledge about its interaction with the environment is that it receives a series of observations $z_{1:t}$ after taking a series of actions $u_{1:t}$. Given this information, the best that the agent can do is to maintain a probability of its current state, $bel(s_t) = p(s_t|u_{1:t}, z_{1:t})$, given the set of actions and observations. This belief state therefore represents the posterior probability over the state variables in the environment, after being conditioned on the available data.

In this planning problem, the task is therefore to find the minimum-cost, collision-free path through belief space starting from an initial bel_0 and ending at the posterior belief distribution bel_g that is within the goal subspace of the map, $bel_g \in G$. Here, the cost objective function J is given by

$$J(bel_0, u_0, ...bel_T, u_T) = D(bel_T) \tag{2.2}$$

where $J(...)$ refers to the cost associated with a particular path, and $D(bel_t)$ is the cost associated with the resulting uncertainty of the goal belief bel_T.

2.3 Background Algorithms

2.3.1 Probabilistic Roadmap Algorithm

We begin by describing the two algorithms, one each in the path-planning and localization domains, that form the basis of the BRM algorithm. We first present a popular path-planning technique for high-dimensional fully-observable environments, the Probabilistic Roadmap Algorithm [5]. The PRM performs efficient path-planning by creating a graph of potential trajectories through the free space of an environment, before performing a trajectory search through the graph to find the best path between a given start and goal location.

The PRM algorithm can therefore be broken down into two main steps: a pre-processing phase and a query phase. In the pre-processing phase, the PRM builds a graph of the possible trajectories that the agent can accomplish by probabilistically sampling the configuration space [9]. Configuration space refers to the space of possible positions that a physical system can attain, and in the context of our problem, is made up of all collision-free poses, C_{free}, as well as those that will result in a collision with obstacles in the map, C_{obs}, i.e. $C \equiv C_{free} \cup C_{obs}$. Each sample is then tested to determine if it lies in C_{free} or C_{obs}, and those that belong to C_{free} are added as nodes in a graph. After a desired number of samples have been added, the edges between pairs of nodes are tested for feasibility and stored in the graph if the entire edge can be traversed without collision with the obstacles in the map. In this way, a high-dimensional space can be efficiently represented as a discrete graph, containing information of which nodes in the graph can be directly traversed from a particular sample node.

In the query phase, the algorithm attempts to find a feasible, collision-free path from the start s_0 to the goal s_g. It first links s_0 and s_g to their nearest nodes in the graph, and then uses a standard graph search algorithm, such as the A* search algorithm, to find the minimum-cost path in the trajectory graph that connects the start and goal nodes. The set of edges connecting the path is then used by the agent to navigate from the start point to the end goal with a simple controller.

The PRM algorithm has been widely used within the research community for a wide-range of path-planning tasks [10, 11, 12, 13, 14, 15, 16], and many variants of the algorithm have been developed [17, 18, 19, 20, 21]. However, the algorithm assumes that the agent has accurate information of its full state at every instant in time, and can therefore accurately execute the trajectory path that has been planned. If, however, the agent has a poor estimate of its state, it may not be able to ascertain when it has reached a node in the graph, and therefore be unsure of when to follow a different edge. The PRM algorithm may therefore break down when there is uncertainty in the agent's state.

2.3.2 Bayes Filter

This inherent limitation in the PRM algorithm motivates us to turn our attention to the group of algorithms that can compensate for the lack of direct information about the agent's state, by attempting to estimate the agent's state from sensor data. In general, inferences can be made about the agent's state given the partial information that the sensors provide, after filtering out the inherent noise in the sensor measurements.

In this regard, the Bayes filter [22] represents the most general method for calculating the belief distribution $bel(s_t)$ from a given set of actions and observations. It consists of both a *process* update and a *measurement* update, which are repeated sequentially to maintain the belief state at every instant in time. Algorithm 1 presents the theoretical equations that together make up the Bayes filter.

Algorithm 1 The Bayes Filter algorithm

```
Require: Previous belief state $bel(s_{t-1})$, action $u_t$, observation $z_t$
1: for all $s_t$ do
2:     $\overline{bel}(s_t) = \int p(s_t|u_t, s_{t-1}) bel(s_{t-1}) ds_{t-1}$
3:     $bel(s_t) = \eta p(z_t|s_t)\overline{bel}(s_t)$
4: end for
5: return $bel(s_t)$
```

In each iteration, the algorithm first makes a prediction of the agent's probability distribution after taking a control action u_t. As shown in step 2, the predicted belief $\overline{bel}(s_t)$ is obtained by integrating the product of two distributions: the prior belief $bel(s_{t-1})$, and the probability that the control u_t induces a transition from s_{t-1} to s_t. In the second step, the *measurement* update, the observation z_t is incorporated as shown in Line 3. The algorithm multiplies the belief $\overline{bel}(s_t)$ by the probability that the measurement z_t has been observed, and does so for each possible posterior state s_t. The probability distribution is then normalized, returning the new belief $bel(s_t)$ after incorporating the latest recorded action and observation. These two steps are then repeated sequentially for each additional set of control and measurement information.

2.3.3 Extended Kalman Filter

Unfortunately, the Bayes filter equation is not a practical algorithm if used on its own, because it cannot be implemented on a digital computer. Instead, approximations are made to represent the probability distributions of the belief state. Here, the Kalman filter family of algorithms have proven to be a popular set of techniques for implementing the Bayes filter. The Kalman filter [23] is a form of Bayes filtering that assumes that 1) all probability distributions are Gaussian, and 2) the transition and observation functions are linearly parameterized by the state and control variables, in addition to some Gaussian noise.

However, in many real-world applications, the Kalman filter assumptions that the observations are linear functions of the state and that the state transition function is a linear function of the state and control variables often break down. For example, a simple robot making translational and rotational actions cannot be described by linear state transitions. Plain Kalman filters are therefore inapplicable to many robotics problems.

In light of these fundamental limitations of the pure Kalman filter, the Extended Kalman filter (EKF) algorithm was developed. The EKF allows the same inference algorithm to operate with non-linear transition and observation functions by linearizing these functions around the current mean estimate. More formally, the state s_t and observation z_t are given by the following functions,

$$s_t = g(s_{t-1}, u_t, w_t), \qquad w_t \sim N(0, W_t), \tag{2.3}$$

and

$$z_t = h(s_t, q_t), \qquad q_t \sim N(0, Q_t), \tag{2.4}$$

Here, u_t refers to the control action, and w_t and q_t are random, unobservable noise variables. g and h represent the non-linear control and measurement models respectively.

In the EKF, the agent's belief state is represented by a Gaussian distribution, parameter-

Algorithm 2 The Extended Kalman Filter algorithm

```
Require: Previous belief state $\mu_{t-1}, \Sigma_{t-1}$, action $u_t$, observation $z_t$
 1: $\overline{\mu}_t = g(u_t, \mu_{t-1})$
 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: $\mu_t = \overline{\mu}_t + K_t(z_t - h(\overline{\mu}_t))$
 5: $\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t$
 6: return $bel(\mu_t, \Sigma_t)$
```

ized by a mean estimate, μ_t, and a covariance matrix, Σ_t, which represents the associated uncertainty. Using the Bayes filter framework, the EKF computes the state distribution at time t in two steps: a process step that is based based only on the control input u_t and the belief state in the previous time step, (μ_t, Σ_t), as well as a measurement step that incorporates measurement z_t to obtain the new belief estimate.

The process step is calculated as follows:

$$\overline{\mu}_t = g(\mu_{t-1}, u_t), \tag{2.5}$$

$$\overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t W_t V_t^T, \tag{2.6}$$

where G_t is the Jacobian of g with respect to x and V_t is the Jacobian of g with respect to w. For convenience, we denote $R_t \triangleq V_t W_t V_t^T$. The result is the predicted belief state, represented by the predicted mean estimate $\overline{\mu}_t$ and predicted covariance $\overline{\Sigma}_t$.

Similarly, the measurement step is calculated using:

$$\mu_t = \overline{\mu_t} + K_t(H_t \overline{\mu}_t - z_t), \tag{2.7}$$

$$\Sigma_t = (I - K_t H_t)\overline{\Sigma}_t, \tag{2.8}$$

$$K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t\right)^{-1}. \tag{2.9}$$

where H_t is the Jacobian of h with respect to s and K_t is known as the Kalman gain, which represents the mapping of the measurements z_t from measurement space to state space that will yield the Minimum Mean-Square Error (MMSE) estimate. The EKF algorithm is summarized in Algorithm 2.

An alternate form of the EKF is the Extended Information Filter, which represents the EKF covariance by its inverse, the information matrix [24]. Hence, instead of the EKF covariance matrix update equations (2.6) and (2.8), the information matrix updates can be written as

$$\overline{\Omega}_t = \overline{\Sigma}_t^{-1} = \left(G_t \Sigma_{t-1} G_t^T + R_t\right)^{-1} \tag{2.10}$$

$$\Omega_t = \overline{\Omega}_t + H_t^T Q_t^{-1} H_t. \tag{2.11}$$

For convenience, we denote $M_t \triangleq H_t^T Q_t^{-1} H_t$, such that $\Omega_t = \overline{\Omega}_t + M_t$. M_t therefore represents the information gained by incorporating the measurements into the probability distribution. In the information form, the distribution $p(s_t|u_{1:t}, z_{1:t})$ can instead be represented by the information vector ξ_t and the information matrix $\Omega_t = \Sigma_t^{-1}$. It has been shown that the EIF is a more efficient update method in domains where the information matrix is sparse.

2.4 Belief Roadmap Algorithm

Thus far in this chapter, we have described the two underlying algorithms that provide inspiration for the BRM algorithm. Recall from Section (2.3.1) that the PRM algorithm allows for efficient path-planning when the full state of the agent is known at every instant in time. However, when only incomplete information of the agent's state is available, as discussed in the problem formulation of Section (2.2), planning must instead be carried out in belief space. Belief space planning allows for decisions to be made not just based on the best mean estimate of the agent's state, but instead can be based on additional statistics that describe the full probability distribution of the agent's state. For instance, if we adopt the EKF form to represent our belief state, the probability distribution of our agent at every instant in time can be represented by a mean estimate, μ_t, and a covariance matrix, Σ_t. In particular, if we could incorporate both the mean and covariance into our planning process, we can generate plans that take into account the associated expected uncertainty of the agent's state as it traverses through the path, and ignore those paths that result in states with a large uncertainty.

In the original BRM algorithm, the primary goal, as described in section (2.2), is for an agent with incomplete information of its state to successfully navigate from a given start position to a desired goal location. This is in contrast to traditional path-planners, which focus on searching for collision-free paths that minimize the cost of reaching the goal location. However, such a performance breaks down in belief space, since every belief at every instant in time has some probability that it is at the goal state. Hence, a different objective function was chosen for belief space planning, seeking instead to find a path that maximizes the probability of being at the goal state.

If we wanted to extend the PRM algorithm to belief space, one naive method of doing this would be to sample directly from the large space encompassed by all possible combinations of (μ, Σ), and test the edges created by each pair of samples (bel_i, bel_j) for feasibility, so as to create a discrete representation of the belief space. Once the graph has been constructed, a search through the graph could then reveal the best path for the agent.

Unfortunately, it has been shown in [4] that the likelihood of actually sampling any beliefs in the large belief space that is actually reachable by the initial belief ($b_0 = (\mu_0, \Sigma_0)$) is zero. This is because the control problem is actually underactuated, and only a small subset of the entire belief space is actually realizable. Furthermore, the covariance of the agent's motion is often a highly non-linear function of its existing state, actions and observations, and hence the reachable subset of the belief space cannot be easily partitioned from the rest of the belief space.

Instead, the BRM algorithm uses the EKF representation that parameterizes the belief state into two components, μ_t and Σ_t. Under some mild assumptions of unbiased motion and sensor models, μ_t can be made equivalent to s_t in the fully observable path-planning problem, and thus the reachability of μ is only dependent on constraints in the vehicle dynamics and obstacles in the map environment. Having computed the reachable subset of μ, μ_{feas}, we can then compute the associated reachable set of covariance matrices. Given an initial mean estimate and covariance (μ_0, Σ_0) and the reachable set of mean estimates μ_{feas}, the corresponding covariances can be predicted by propagating the initial covariance Σ_0 along any path that connects the initial mean estimate μ_0 to the particular mean estimate by using the EKF update equations (2.5) - (2.8), as well as the known motion and sensing models.

Therefore, to construct a graph of the reachable belief space, the planner would first sample a set of mean poses $\{\mu_i\}$ from $\mathcal{C}_{free}$ using the standard pose sampling of the PRM algorithm, and place an edge e_{ij} between pairs (μ_i, μ_j) if the straight line between the two poses is collision-free. This creates a graph that initially looks identical to a graph that would have been generated by using the PRM algorithm. The graph is then used to search for a path from start estimate μ_0 to desired goal μ_g, but for each node, the associated posterior covariance, rather than the standard cost-to-go, is computed and stored.

Unfortunately, the significant reduction in the size of the graph's state-space is still insufficient for making the process of planning under uncertainty a tractable problem, especially in high dimensional spaces. First, unlike the simple calculation of the cost-to-go value in a deterministic PRM algorithm, the calculation of the posterior covariance at the end of each edge requires multiple EKF updates along each edge e_{ij}, and is dependent on the length of each edge. While this operation is a constant multiplier of the asymptotic search complexity, a significant additional computation cost is nevertheless incurred to calculate the posterior covariance of an edge, given a prior covariance.

More importantly, performing the graph construction in this manner prevents us from making the graph building process a one-time operation. Had this been the case, we would have been able to get away with the large computation cost of performing the multiple EKF updates, since we can perform this in an offline fashion and retain the online efficiency. Instead, even with a fixed start point, there are multiple paths that will be explored to get to node i, implying that there will be a variety of prior covariances for edge e_{ij}. These prior covariances will have to be propagated separately through edge e_{ij}, because the posterior covariance is not a linear function of the prior covariance, and it becomes computationally intractable to try and propagate all the possibilities offline. In addition, we would also like to be able to perform successive trajectory searches without having to recompute the entire graph each time we start from a different start node.

Ideally, we would therefore like to represent the covariance propagation in a form similar to the cost-to-go function in the traditional PRM. This would allow us to find the cost

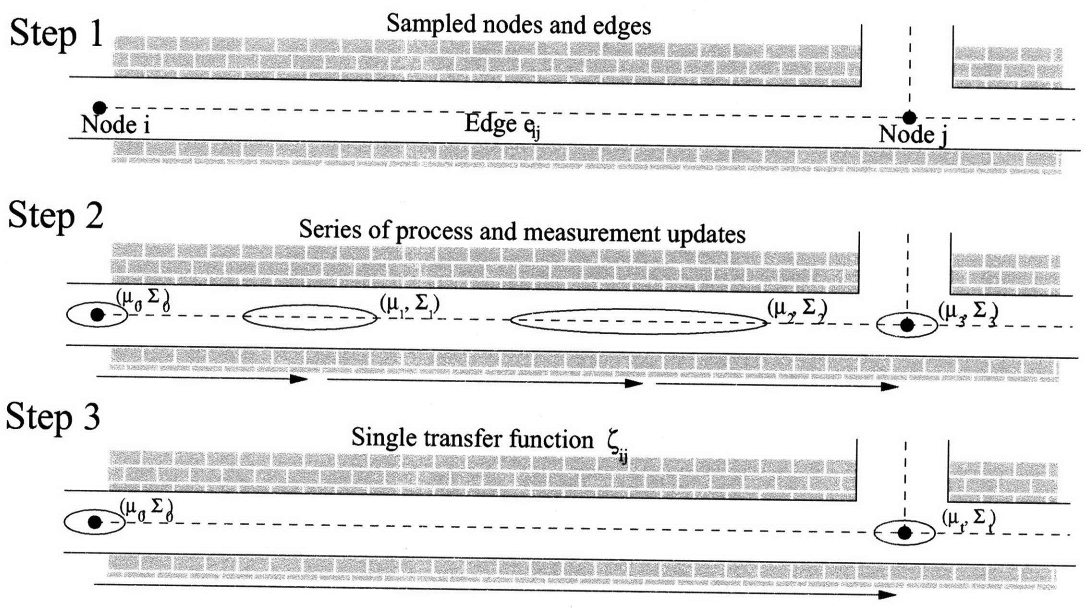


Figure 2-1: Illustration of the Belief Roadmap one-step process. In step 1, the graph of mean poses is constructed, and mutually visible nodes are connected with edges. In step 2, the posterior covariance is calculated through a series of process and measurement updates. In step 3, the one-step covariance transfer function is calculated from the individual multi-step updates.

of a candidate path without having to tediously propagate an initial covariance through every individual EKF update, thereby allowing us to compare the costs of alternative paths. Furthermore, once the graph has been built, it can then be used successively for quick replanning.

2.4.1 Belief Updating as a one-step operation

The BRM algorithm does just that by using an alternate representation of the covariance, allowing multiple EKF updates to be compiled into a single linear transfer function. Given an initial covariance at one end i of an edge e_{ij}, the posterior covariance at j can be found by simply performing a few simple matrix operations. Thus, by pre-computing the transfer function for each edge, the search complexity for planning in belief space becomes comparable to traditional configuration space planning.

Figure 2-1 provides a pictorial representation of the general BRM process. In the first step, the PRM-like graph is computed by sampling the configuration space to create a graph of mean poses, and poses that are mutually visible are connected with nodes. After the graph has been constructed, the second step simulates the series of process and measurement updates that the agent is expected to experience when it moves along an edge, and a transfer function is built for each pair of process and measurement updates. Finally, in step three, the individual pairs of updates are compiled into a single one-step covariance transfer function, returning a transfer function for each edge in the graph.

We now describe the alternative representation of the covariance that enables us to perform the one-step update. It has been shown previously in [4] that for a Kalman filter-based state estimator, such as the EKF and the UKF, the covariance, Σ, can be factored as $\Sigma = BC^{-1}$, where the separate process and measurement updates in a Kalman-filter based technique enables B_t and C_t to be written as linear functions of B_{t-1} and C_{t-1}. Thus, even though the posterior covariance of an edge is not a linear function of the prior covariance, the factorized form can be represented as such.

In particular, given an initial prior covariance, Σ_{t-1}, the EKF process update (Eqn. 2.6), previously described in Algorithm 2, can be represented as

$$\Sigma_{t-1} = B_{t-1}C_{t-1}^{-1} \tag{2.12}$$

$$\Rightarrow \overline{\Sigma}_t = G_t B_{t-1} C_{t-1}^{-1} G_t^T + R_t \tag{2.13}$$

$$= (G_t B_{t-1})(G_t^{-T} C_{t-1})^{-1} + R_t \tag{2.14}$$

$$= \left(\overline{D}_t \overline{E}_t^{-1}\right)^{-1} \tag{2.15}$$

where $\overline{D}_t = G_t^{-T} C_{t-1}$ and $\overline{E}_t = G_t B_{t-1} + R_t(G_t^{-T} C_{t-1})$ and Equation 2.15 follows from a matrix inversion lemma, shown in [25].

The covariance update in the information form (Eqn. 2.11) can similarly be factored as

$$\Sigma_t = (\overline{\Sigma}_t^{-1} + H_t^T Q_t^{-1} H_t^T)^{-1} \tag{2.16}$$

$$= (\overline{D}_t \overline{E}_t^{-1} + M_t)^{-1} \tag{2.17}$$

And using the same matrix inversion lemma,

$$= \overline{E}_t(\overline{D}_t + M_t\overline{E}_t)^{-1} \tag{2.18}$$

$$\Rightarrow \Sigma_t = B_t C_t^{-1}, \tag{2.19}$$

where $B_t = \overline{E}_t = G_t B_{t-1} + R_t(G_t^{-T}C_{t-1})$ and $C_t = (\overline{D}_t + M_t\overline{E}_t) = G_t^{-T}C_{t-1} + M_t G_t B_{t-1} + R_t(G_t^{-T}C_{t-1})$.

In both cases, B_t and C_t are linear functions of B_{t-1} and C_{t-1}. Collecting the above terms, we can write the complete step of a single EKF update linearly, such that

$$\Psi_t = \begin{bmatrix} B \\ C \end{bmatrix}_t = \begin{bmatrix} 0 & I \\ I & M \end{bmatrix}_t \begin{bmatrix} 0 & G^{-T} \\ G & RG^{-T} \end{bmatrix}_t \begin{bmatrix} B \\ C \end{bmatrix}_{t-1}, \tag{2.20}$$

where Ψ_t is the stacked block matrix $\left[\begin{smallmatrix} B \\ C \end{smallmatrix}\right]_t$ consisting of the covariance factors and $\zeta_t = \left[\begin{smallmatrix} 0 & I \\ I & M \end{smallmatrix}\right]_t \left[\begin{smallmatrix} 0 & G^{-T} \\ G & RG^{-T} \end{smallmatrix}\right]_t$ is the one-step transfer function for the covariance factors, and is a linear combination of G_t, R_t and M_t.

All the elements in ζ are directly controllable, except for M_t. Recall that in the EKF, G_t refers to the Jacobian of the control model, R_t is the associated process noise, and thus both matrices can be determined based on prior knowledge of the agent's model. On the other hand, $M_t \triangleq H_t^T Q_t^{-1} H_t$ represents the total amount of information that the measurement provides at time t, and is a function of the Jacobian of the measurement model, H_t, and the associated measurement noise, Q_t (which is usually held constant). The measurement Jacobian H_t is dependent on the sensor model and the particular map environment, and therefore the accuracy of the EKF approximation assumes that the measurement function is locally linear. However, in situations when the EKF algorithm is used, the locally linear measurement model is implicitly assumed, and hence we can assume that M_t is constant and known *a priori.*

Thus, all the terms required in the transfer function are known, and we can pre-compute the transfer function of each EKF update along a particular trajectory. Furthermore, because the update is a linear function of the covariance factors in the previous timestep, we can combine multiple ζ_t matrices into a single, one-step update for the covariance propa-

Algorithm 3 The Belief Roadmap (BRM) algorithm.

```
Require: Start belief $(\mu_0, \Sigma_0)$, goal $\mu_{goal}$ and map $C$
 1: Sample poses $\{\mu_i\}$ from $C_{free}$ to build belief graph node set $\{n_i\}$ such that $n_i = \{\mu = \mu_i, \Sigma = \emptyset\}$
 2: 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
 3: Build one-step transfer functions $\{\zeta_{ij}\} \quad \forall \quad e_{ij} \in \{e_{ij}\}$
 4: Augment node structure with best path $p$=$\emptyset$, such that $n_i$=$\{\mu, \Sigma, p\}$
 5: Create search queue with initial position and covariance $Q \leftarrow n_0 = \{\mu_0, \Sigma_0, \emptyset\}$
 6: while $Q$ is not empty do
 7:    Pop $n \leftarrow Q$
 8:    if $n = n_{goal}$ then
 9:       Continue
10:    end if
11:    for all $n'$ such that $\exists e_{n,n'}$ and not $n' \ni n[p]$ do
12:       Compute one-step update $\Psi' = \zeta_{n,n'} \cdot \Psi$, where $\Psi = \begin{bmatrix} n[\Sigma] \\ I \end{bmatrix}$
13:       $\Sigma' \leftarrow \Psi'_{11} \cdot {\Psi'_{21}}^{-1}$
14:       if $tr(\Sigma') < tr(n'[\Sigma])$ then
15:          $n' \leftarrow \{n'[\mu], \Sigma', n[p] \cup \{n'\}\}$
16:          Push $n' \rightarrow Q$
17:       end if
18:    end for
19: end while
20: return $n_{goal}[p]$
```

gation. Therefore, for each edge e_{ij} in the BRM graph, we can pre-compute the transfer function ζ_t for each update along a single edge using the relevant Jacobians, and by multiplying all of them together, return a single transfer function ζ_{ij} that will propagate an initial covariance (in factored form) along the length of an edge in a single matrix multiplication. The BRM graph that is computed off-line is therefore made up of a series of mean poses s_i, s_j, edges e_{ij} linking visible poses, and the associated transfer function ζ_{ij} of each edge. Using this BRM graph, the expected posterior covariance from a given path trajectory and initial covariance can then be easily computed by first multiplying, in sequential order, all the transfer functions ζ_{ij} associated with the edges that make up the path, thereby creating a single transfer function $\zeta_{1:T}$, before recovering the posterior covariance by multiplying the initial covariance with the transfer function, and then performing the un-factorization.

Algorithm 3 describes the complete Belief Roadmap algorithm, which requires searching for the path from start s_0 to goal s_g that results in the smallest possible uncertainty of the agent's position at the goal. Steps 1 and 2 represent the creation of the graph that represents the C_{free} space of the map. Step 3 contains the pre-processing phase where each edge is labeled with the transfer function ζ_{ij}, allowing any covariance to be propagated in a single step. Step 6-19 then performs the search process to find the path with the smallest covariance. Starting from the initial start node, nodes that are linked by edges are expanded and the posterior covariances computed. Using the trace of the covariance matrix as a measure of the associated uncertainty, a node is only re-expanded if the trace of the new covariance is smaller than the covariance that had already been associated with the node in previous paths. This is repeated until all useful nodes have been expanded, and the queue of nodes becomes empty.

2.5 Conclusion and BRM Limitations

In this chapter, we have formulated the general problem of planning in belief space, and presented the original Belief Roadmap algorithm, first presented in [4]. It presents an effective solution to the motion planning problem in belief space, and hence provides the theoretical basis for the rest of the research presented in this thesis. Nevertheless, this new technique has opened up new areas of research, and a number of limitations with the original algorithm can be immediately identified:

1. The algorithm leverages the Extended Kalman Filter as the filtering technique for localization. While the EKF has become immensely popular amongst the robotics and signal processing communities, the BRM implicitly assumes that the linearization of the measurement and control models is a reasonable assumption to make. This is valid when the measurement model does not yield large deviations in potential measurement information in a local region, and when the belief uncertainty is small enough that the mean predictions will similarly not deviate much. However, if there is actually a large uncertainty in the vehicle's state, or if the measurement model is highly non-linear, then the linearization assumption actually breaks down, resulting in the BRM not accurately accounting for the uncertainty associated with a path.

2. Being an extension of the PRM algorithm, the BRM algorithm is a sampling-based algorithm, and in the existing formulation, the uniform sampling strategy is used. However, the uniform sampling strategy is a naive sampling technique that does not take into account additional information available in a given map, such as likelihood of successful localization and other sensor information etc. A sampling technique that biases the search towards regions with more information will likely lead to the creation of trajectories with lower goal uncertainty in a more efficient manner.

3. The algorithm employs a breadth-first search strategy to iterate through all the possible paths from start s_0 to goal s_g. Unfortunately, in the current implementation, more efficient search techniques, such as the A* search algorithm, cannot be used because admissible heuristics have not been found. Unlike traditional configuration space techniques that employ the straight line distance between a node and the goal as an admissible heuristic, the transfer functions cannot be used as a direct measure of the quality of an edge. The development of admissible heuristics will therefore lead to a significant reduction in search time.

4. Finally, the existing BRM algorithm adopts the expected posterior covariance of the agent at the goal s_g, after executing a particular trajectory, to ascertain the performance of the path. This unfortunately fails to take into account other metrics that are equally critical to the agent's performance, including the uncertainty along the path (intermediate covariance), the likelihood of colliding with an obstacle during the path (overlap of obstacles and covariance along the path) and the energy cost of navigating the path (path distance and control actions). Some combination of these cost metrics will lead to more robust performance for autonomous navigation and control.

Some of the limitations described above become critical if we return to our motivating problem of trying to achieve autonomous path-planning and control of a quadrotor helicopter, equipped with a $4m$ laser range-finder sensor, in a GPS-denied indoor environment. We therefore seek to extend the BRM algorithm in the rest of the thesis. In particular, we address the first two limitations described above in the subsequent two chapters - extending the BRM to apply the Unscented Kalman Filter in Chapter 3, and introducing a information-based sampling strategy in Chapter 4.

Chapter 3

Extending BRM to UKF

3.1 Introduction

In Chapter 2, we presented the BRM algorithm as a solution to the path-planning problem in *belief* space, where the uncertainty of an agent's state needs to be taken into account when performing the trajectory planning. Unfortunately, we encounter difficulties when trying to directly apply this algorithm to our motivating problem, which is to enable a quadrotor helicopter, equipped only with a $4m$ laser range-finder sensor and an IMU, to fly autonomously in an indoor, GPS-denied environment. The original BRM algorithm was verified with experiments using Ultra-Wide Band (UWB) sensors, each of which returns a single range measurement of where the agent is relative to the sensor. Because each of these measurements are independent with respect to all other measurements, the UWB sensor model can be linearized for use in the EKF, returning the measurement Jacobian H.

Unfortunately, the EKF is not always a feasible form of Bayesian filtering, and this becomes apparent when we consider the problem in the context of our laser range-finder sensor. Given an agent's pose, each individual laser range reading (approx. 750) in a single range scan can be projected into a grid world to represent the obstacles that the laser sensor detects. However, such a grid map representation contains a strong independence assumption about each grid cell, assuming that the measurements of neighboring grid cells are uncorrelated. This is in spite of the fact that in reality, adjacent grid cells can actually be highly correlated, resulting in the breakdown of the EKF measurement model. To get around this problem, a technique that is frequently used is to extract high-level features such as walls and corners in order to compute the respective Jacobians and the amount of information available from the measurements. Nevertheless, feature extraction is difficult in a generic environment, where there are often no clear distinct features such as walls and corners, especially when there is a significant level of measurement noise.

Instead, to address the limitations of linearization, alternate forms of the Bayes filter have been developed. The Unscented Kalman Filter [6] is a recent extension of the Kalman-filter family of algorithms, and seeks to improve upon the linearization technique employed by the EKF by capturing the variability of observations and controls over a local region about the prior mean. By being able to capture the non-linearities of the process and measurement models, the UKF algorithm produces better localization performance for an agent in an uncertain environment.

In this chapter, we show how the BRM algorithm can be extended to incorporate the UKF localization technique. We first provide a description of the Unscented Kalman Filter algorithm, before showing how the UKF can be used in place of the EKF in the BRM algorithm. In particular, we show experimentally that despite the fact that the UKF covariance propagation is directly coupled with the prior covariance at each update step, the error induced in the transfer function from making the approximation of a constant prior is actually small, implying that we can still apply the BRM framework to the UKF algorithm.

3.2 Unscented Kalman Filter

The Unscented Kalman Filter uses an alternative method for approximating the non-linearities in the process and measurement models. Known as the unscented transform, it avoids the linearization of the models through a Taylor series expansion, and hence is able to address some of the limits of linearization. For an n-dimensional Gaussian distribution with mean μ and covariance Σ, a set of $2n + 1$ "sigma points" are deterministically chosen

Algorithm 4 The Unscented Kalman Filter algorithm

```
Require: Previous belief state $\mu_{t-1}, \Sigma_{t-1}$, action $u_t$, observation $z_t$
 1: $\mathcal{X}_{t-1} = (\mu_{t-1} \quad \mu_{t-1} + \gamma\sqrt{\Sigma_{t-1}} \quad \mu_{t-1} - \gamma\sqrt{\Sigma_{t-1}})$
 2: $\overline{\mathcal{X}}_t^{\star} = g(u_t, \mathcal{X}_{t-1})$
 3: $\overline{\mu}_t = \sum_{i=0}^{2n} w_m^i \overline{\mathcal{X}}_t^{\star,i}$
 4: $\overline{\Sigma}_t = \sum_{i=0}^{2n} w_c^i (\overline{\mathcal{X}}_t^{\star,i} - \overline{\mu}_t)(\overline{\mathcal{X}}_t^{\star,i} - \overline{\mu}_t) + R_t$
 5: $\overline{\mathcal{X}}_t = (\overline{\mu}_t \quad \overline{\mu}_t + \gamma\sqrt{\overline{\Sigma}_t} \quad \overline{\mu}_t - \gamma\sqrt{\overline{\Sigma}_t})$
 6: $\overline{\mathcal{Z}}_t = h(\overline{\mathcal{X}}_t)$
 7: $\hat{z}_t = \sum_{i=0}^{2n} w_m^i \overline{\mathcal{Z}}_t^i$
 8: $S_t = \sum_{i=0}^{2n} w_m^i (\overline{\mathcal{Z}}_t^i - \hat{z}_t)(\overline{\mathcal{Z}}_t^i - \hat{z}_t) + Q_t$
 9: $\overline{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^i (\overline{\mathcal{X}}_t^i - \overline{\mu}_t)(\overline{\mathcal{Z}}_t^i - \hat{z}_t)$
10: $K_t = \overline{\Sigma}_t^{x,z} S_t^{-1}$
11: $\mu_t = \overline{\mu}_t + K_t(z_t - \hat{z}_t)$
12: $\Sigma_t = \overline{\Sigma}_t - K_t S_t K_t$
13: return $bel(\mu_t, \Sigma_t)$
```

to represent the probability distribution, according to the following distribution:

$$\mathcal{X}_t^0 = \mu_{t-1}, \tag{3.1}$$

$$\mathcal{X}_t^i = \mu_{t-1} + \left(\sqrt{(n+\lambda)\Sigma_t}\right)^i, \quad i=1,\ldots,n \tag{3.2}$$

$$\mathcal{X}_t^i = \mu_{t-1} - \left(\sqrt{(n+\lambda)\Sigma_t}\right)^i, \quad i=n+1,\ldots,2n \tag{3.3}$$

where $\left(\sqrt{(n+\lambda)\Sigma_t}\right)^i$ is the ith column of the root of the matrix. Here, $\lambda = \alpha^2(n+\kappa)-n$, and α and κ are parameters that can be tuned to determine how far the sigma points are spread from the mean. For our implementation, we fixed the values $\alpha = 0.1$ and $\kappa = 0$. Given a probability distribution, the sigma points are therefore deterministic samples that are used to represent the distribution. In addition, each sigma point $\mathcal{X}^i$ has an associated weight, w_m^i, that is used when computing the mean, and a slightly different one, w_c^i, that is used for computing the covariance. Specifically,

$$w_m^0 = \frac{\lambda}{n+\lambda} \tag{3.4}$$

$$w_c^0 = \frac{\lambda}{n+\lambda} + (1 - \alpha^2 + \beta) \tag{3.5}$$

$$w_m^i = w_c^i = \frac{1}{2(n+\lambda)} \quad , i = 1, 2, ..., 2n \tag{3.6}$$

Unsurprisingly, the mean and covariance weights respectively sum to 1, i.e. $\sum_{i=0}^{2n} w_c^i = \sum_{i=0}^{2n} w_m^i = 1$. Here, the β parameter can be chosen to take into account additional, higher order, knowledge of the distribution underlying the Gaussian representation. For our purposes, we have assumed that the underlying distribution is also a Gaussian distribution, and have fixed $\beta = 2$.

The UKF performs the state estimation, which includes both the process and measurement update, by propagating these $2n + 1$ sigma points through the relevant models to capture the predicted mean and uncertainty after each step, thereby avoiding the linearization process. Algorithm 4 formally presents the Unscented Kalman Filter algorithm. After generating the sigma points from the prior distribution with equations (3.1)-(3.3), each of the samples is propagated according to the non-linear process model

$$\overline{\mathcal{X}}_t^i = g(\mathcal{X}_t^i, u), \tag{3.7}$$

generating the process mean and covariance

$$\overline{\mu}_t = \sum_{i=0}^{2n} w_m^i \overline{\mathcal{X}}_t^i \tag{3.8}$$

$$\overline{\Sigma}_t = \sum_{i=0}^{2n} w_c^i (\overline{\mathcal{X}}_t^i - \overline{\mu}_t)(\overline{\mathcal{X}}_t^i - \overline{\mu}_t) + R_t. \tag{3.9}$$

Given the mean $\overline{\mu}_t$ and covariance $\overline{\Sigma}_t$ after the process update, a new set of sigma points is generated based on this new probability distribution, as shown in step 5 in Algorithm 4. The sigma points are then propagated through the measurement update as follows

$$\overline{\mathcal{Z}}_t^i = h(\overline{\mathcal{X}}_t^i) \tag{3.10}$$

The observation sigma points $\overline{\mathcal{Z}}_t$ are then used to compute the predicted observation $\hat{z}_t$, and the associated uncertainty matrix S_t is calculated according to

$$\hat{z}_t = \sum_{i=0}^{2n} w_m^i \overline{\mathcal{Z}}_t^i \tag{3.11}$$

$$S_t = \sum_{i=0}^{2n} w_m^i (\overline{\mathcal{Z}}_t^i - \hat{z}_t)(\overline{\mathcal{Z}}_t^i - \hat{z}_t) + Q_t \tag{3.12}$$

As in the EKF update, Q_t represents the covariance matrix of the additive measurement noise, and S_t is equivalent to $H_t \overline{\Sigma}_t H_t^T + Q_t$ of the EKF algorithm, presented in step 3 of Algorithm 2. As in all other Kalman-filter family algorithms, the Kalman gain K_t is then computed, by first computing the cross-covariance $\overline{\Sigma}_t^{x,z}$ between the state and observations, using the following equations:

$$\overline{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^i (\overline{\mathcal{X}}_t^i - \overline{\mu}_t)(\overline{\mathcal{Z}}_t^i - \hat{z}_t) \tag{3.13}$$

$$K_t = \overline{\Sigma}_t^{x,z} S_t^{-1} \tag{3.14}$$

Recall that the Kalman gain K_t represents the degree to which the measurement should be incorporated into the new mean estimate. We can therefore finally recover the posterior mean and covariance from the updates with

$$\mu_t = \overline{\mu}_t + K_t(z_t - \hat{z}_t) \tag{3.15}$$

$$\Sigma_t = \overline{\Sigma}_t - K_t S_t K_t. \tag{3.16}$$

The key advantage of the UKF formulation, especially in comparison to the EKF algorithm, is that the process and measurement functions are not projected into the state space by a linearization. Instead, the unscented transform computes the moments of the process and measurement distributions directly in the state space. This not only allows the UKF to be a *derivative-free* filter by avoiding the computation of the Jacobians, which can be difficult in certain domains, but also allows the UKF to produce as good, if not better, results than the EKF, because it is able to capture the distribution up to the second order accurately, in contrast to the first order fidelity of the EKF.

In addition, the asymptotic complexity of the UKF algorithm is the same as the EKF, though it is found that the EKF is usually slightly faster than the UKF [26]. Finally, it has also been found that because the sigma points are deterministically chosen to represent the probability distribution with a Gaussian approximation, the UKF performs very well if the underlying distribution is approximately Gaussian. However, if the actual belief is actually highly non-Gaussian, then the UKF, despite being an improvement on the EKF, is still too restrictive and will perform poorly.

3.3 Extending BRM to UKF

The UKF algorithm provides an attractive technique for our helicopter to localize itself in a global environment, using only a laser range-finder sensor and inertial sensors for localization. Instead of having to perform feature extraction from the laser scans in the case of the EKF, we can directly simulate the laser measurements at each of the sigma points so as to attain the $\overline{\mathcal{Z}}_t^i$, $\hat{z}_t$ vectors, thereby creating the S_t and K_t matrices.

We would therefore like to apply the BRM algorithm to the situation where the UKF localization technique is used. Recall that the BRM algorithm requires us to calculate the transfer function ζ_t for each pair of process and measurement update, and in the EKF form, is expressed as

$$\zeta_t = \begin{bmatrix} 0 & I \\ I & M \end{bmatrix}_t \begin{bmatrix} 0 & G^{-T} \\ G & RG^{-T} \end{bmatrix}_t \tag{3.17}$$

To create the equivalent transfer function for the UKF, we first assume that the EKF form of the process update, which requires calculating the Jacobian G_t of the control model $g(\mathcal{X}, u)$ and the associated process noise R_t, is a valid approximation for the helicopter's dynamics, and that the non-linearities inherent in the process model can be mitigated as such. Unfortunately, the same cannot be said for the measurement model, and we therefore cannot find the $M_t \triangleq H_t^T Q_t^{-1} H_t$ matrix directly by calculating the Jacobian H_t of the measurement model. Instead, we have to rely upon the UKF to find an equivalent form of the M_t matrix, which represents the total amount of information provided by the measurement

Algorithm 5 Build BRM Graph using UKF updates

```
Require: Prior mean $\mu_t$, covariance $\overline{\Sigma}_t$ after process update, map $\mathcal{C}$,
1: Calculate $G_t$, the Jacobian of the process model
2: Calculate $R_t$, the associated process noise
3: $\overline{\mathcal{X}}_t = (\overline{\mu}_t \quad \overline{\mu}_t + \gamma\sqrt{\overline{\Sigma}_t} \quad \overline{\mu}_t - \gamma\sqrt{\overline{\Sigma}_t})$
4: $\overline{\mathcal{Z}}_t = h(\overline{\mathcal{X}}_t)$
5: $\hat{z}_t = \sum_{i=0}^{2n} w_m^i \overline{\mathcal{Z}}_t^i$
6: $S_t = \sum_{i=0}^{2n} w_m^i (\overline{\mathcal{Z}}_t^i - \hat{z}_t)(\overline{\mathcal{Z}}_t^i - \hat{z}_t) + Q_t$
7: $\overline{\Sigma}_t^{x,z} = \sum_{i=0}^{2n} w_c^i (\overline{\mathcal{X}}_t^i - \overline{\mu}_t)(\overline{\mathcal{Z}}_t^i - \hat{z}_t)$
8: $K_t = \overline{\Sigma}_t^{x,z} S_t^{-1}$
9: $\Sigma_t = \overline{\Sigma}_t - K_t S_t K_t$
10: $M_t = \Sigma_t^{-1} - \overline{\Sigma}_t^{-1}$
11: $\zeta_t = \begin{bmatrix} 0 & I \\ I & M \end{bmatrix}_t \begin{bmatrix} 0 & G^{-T} \\ G & RG^{-T} \end{bmatrix}_t$
12: return $\zeta_t$
```

update at time t.

However, despite not having the Jacobian of the measurement model H_t, we can still recover M_t from the UKF update by working in the information form and recovering the information gained from the measurement z_t. We recall from our discussion of the Extended Information Filter in Section (2.3.3) that the information gain from a measurement update can be written as

$$\Omega_t = \overline{\Omega}_t + M_t \tag{3.18}$$

$$M_t = \Omega_t - \overline{\Omega}_t \tag{3.19}$$

$$= \Sigma_t^{-1} - \overline{\Sigma}_t^{-1} \tag{3.20}$$

$$= (\overline{\Sigma}_t - K_t S_t K_t)^{-1} - \overline{\Sigma}_t^{-1} \tag{3.21}$$

where the information matrix $\Omega_t = \Sigma_t^{-1}$ is the inverse of the covariance matrix. Therefore, to pre-compute the M matrix for a path between two points, we can generate a prior covariance $\overline{\Sigma}_t$, and calculate the posterior covariance, Σ_t, using step 12 of the UKF algorithm (Algorithm 4). It is important to note that although the UKF algorithm is used for the calculation of the M_t matrix, the UKF covariance update for calculating Σ_t does not depend on the actual measurement received, just as in the case of the EKF covariance update. Hap-

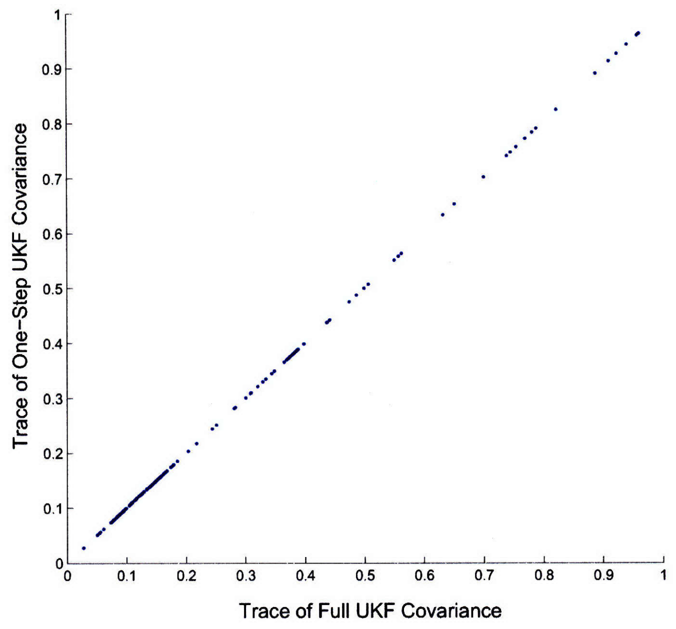


Figure 3-1: Comparison of trace of covariance from full UKF filtering with and trace of covariance from one-step BRM-UKF transfer function

pily, we can therefore pre-compute the M_t matrices in an offline fashion, and furthermore create a BRM graph that can be used for multiple re-queries. Algorithm 5 summarizes the algorithm for building the BRM transfer function under the UKF formulation.

3.4 Theoretical Verification

We now provide experimental verification for some of the approximations made in the algorithm presented in this chapter. Ultimately, for any given trajectory and initial conditions, we require the BRM algorithm to return a covariance from the one-step transfer function that is identical to the resultant covariance if we had propagated the initial covariance through each non-linear UKF update individually along the path. We therefore generated a series of random trajectories and initial conditions and compared the resultant trace between the one-step BRM-UKF update with the full UKF update. As shown in Figure 3-1, the trace of the covariances from the two methods are closely matched. Thus, even though the M matrix is approximated in creating the one-step transfer function, the induced error

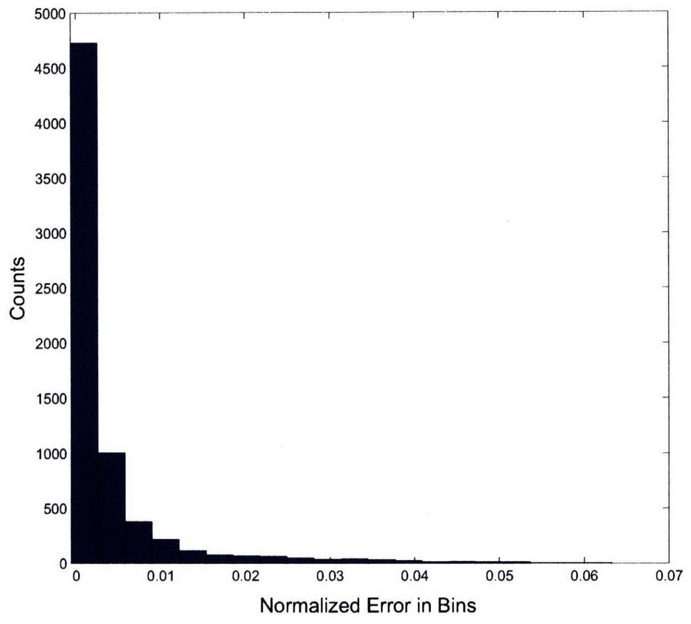


Figure 3-2: Distribution of ratio of error induced by computing the M matrix for the one-step transfer function using a constant prior.

remains low, and thus the one-step transfer function is a reasonable approximation even in the case of the UKF.

Nevertheless, we do recognize that the calculation of the UKF form of the information gain, M_t, *does* depend on the specific prior matrix $\overline{\Sigma}_t$. Thus, one may argue that the different choices of prior in Equation 3.21 may result in different one-step transfer functions, making it incorrect for us to make the constant prior assumption. We therefore test this accusation by comparing the trace of the full UKF covariance with the resultant trace of the one-step update when we used different priors to calculate the M matrix. We conducted a total of 7000 trials using 100 different priors and again randomized the trajectories and initial conditions. We then compared the error in the trace of the one-step update with that of the full UKF update. Figure 3-2 shows the results of the trials, and suggests that the error induced from the approximation of the constant prior, regardless of what the constant prior actually is, is small. The error induced in the one-step transfer function for using a constant M is less than 2% with a significance of $p = 0.955$, indicating low sensitivity to the choice of prior over a range of operating conditions.

3.5 Conclusion

In this chapter, we have shown that the Belief Roadmap Algorithm can be extended to the Unscented Kalman Filter localization algorithm, even though a closed form solution does not exist. We have shown that just as in the original BRM algorithm, we can similarly create the BRM-UKF transfer function by approximating the UKF measurement update with a constant prior covariance, thus allowing us to perform covariance propagations in an efficient manner.

Chapter 4

BRM Sampling Strategies

4.1 Introduction

As we have described in Chapter 2, the BRM algorithm is an extension of the Probabilistic Roadmap algorithm and therefore employs a sampling-based technique to represent the configuration $\mathcal{C}$ space. In the original BRM algorithm presented in [4], the uniform sampling strategy is employed to produce a set of nodes, from which a belief graph is built. In this chapter, we present a new sampling strategy, the Sensor Uncertainty (SU) sampling strategy, which seeks to sample Gaussian means only from the most useful portions of the $\mathcal{C}_{free}$ space to minimize the number of samples needed to express paths with low uncertainty. We first describe the need for a more efficient sampling strategy, before presenting the sensor uncertainty sampling algorithm. We then compare the performance of SU sampling against a variety of other sampling strategies that are found in the literature.

4.2 BRM Sampling

The PRM algorithm, which is effectively the fully-observable form of the BRM, is widely used for path-planning because it is able to represent the $\mathcal{C}_{free}$ space of a given map very efficiently. Few samples are needed to find a path from the start s_0 to the goal s_g, although the greater the number of samples used to probe the $\mathcal{C}_{free}$ space, the higher the likelihood that the path found will converge to the optimal path through the given environment.

Similarly, as the number of samples and the density of the graph grows, the BRM path-planning algorithm will find increasingly low-covariance paths. By having a greater number of edges to choose from the graph, a larger set of alternative paths from start to goal can be generated, and the path with the least-covariance can then be chosen.

Unfortunately, as more samples are added to the graph, the density of the graph increases, and the time cost incurred in building the graph will also grow. This is particularly problematic for the BRM algorithm, especially in comparison to the PRM. The build phase of the BRM has a complexity of $\mathcal{O}(lb^d)$, while the PRM has a complexity of $\mathcal{O}(b^d)$, where l is the average length of the edges, b is the number of edges per node, and d is the search depth of the graph. In addition, while the PRM only requires the simple addition of the cost-to-go of a particular edge, the BRM incurs the computational cost of l EKF or UKF updates for each additional edge in the map when building the BRM graph. Although this is a constant multiplier to the computational cost of the algorithm, it nevertheless adds a significant cost to the time incurred for the graph building process. We therefore seek to minimize the size of the graph, without compromising the expected performance of the computed path.

4.3 Sensor Uncertainty Field

In an ideal case, we would minimize the number of samples needed by only generating samples that lie on the optimal path to the goal. Unfortunately, this would require knowing the optimal path beforehand, which then makes the path-planning problem redundant. Despite not having prior knowledge of the optimal path, however, we can make inferences based on the information we have available to choose samples that are more likely to be used in an optimal path. Given that the BRM attempts to generate a path that minimizes the expected uncertainty at the goal, we observe that vehicle poses that generate measurements with high information of the vehicle's position relative to the environment is much more likely to lie on the optimal path, as compared to vehicle poses that generate measurements with little or no information. If we are able to bias our sampling more heavily towards poses that provide greater expected information gain when the sensor measurement is taken at that point, we should then have a graph that places nodes where we expect to achieve the maximum localization accuracy for the vehicle.

This provides the motivation for our sampling strategy, which we call *sensor uncertainty sampling*, drawing inspiration from the "Sensor Uncertainty Field" (SUF) first termed by Takeda and Latombe in [27]. At every pose x in C_{free}, the SUF is an estimate of the uncertainty in the "sensed configuration" that the vehicle will experience, and is computed by matching the expected data at x against the environment model. In our context, the sensor uncertainty is similarly a mapping of location x to its expected information gain, $x \rightarrow \mathcal{I}(x)$ [28], where locations with high information gain correspond to locations where the expected sensor measurements at those locations would enable us to maximize the localization accuracy of the vehicle.

In the original SUF algorithm, the entire sensor uncertainty field is first generated and used to find paths that minimize expected uncertainty [29]. Unfortunately, explicitly building this field is computationally expensive in practice. Instead, we simply sample from this field to build the BRM graph, and thus gain the benefits of creating nodes in locations which have high information gain, without having to explicitly build the sensor uncertainty field.

Algorithm 6 Sensor Uncertainty Sampling Algorithm

```
Require: Map $C$, Number of samples $N$, Constant prior $P_0$
 1: while size of graph < N do
 2:     Sample a pose, $c_1$, from $C$, with equal probability
 3:     if $c_1 \in C_{free}$ then
 4:         Simulate expected sensor measurement, $z$, at $c_1$
 5:         Generate sigma points, $\chi_i$, about $c_1$ according to constant prior $P_0$, creating prior
            distribution $p(x)$
 6:         Calculate information gain $\mathcal{I}(x) = P_0 - H(p(z|x))$
 7:         Normalize $\mathcal{I}(x)$ such that $\mathcal{I}(x) \in [0, 1]$
 8:         Add $c_1$ to graph with probability $\mathcal{I}(x)$
 9:     end if
10: end while
11: return graph
```

Algorithm 6 describes the sensor uncertainty sampling algorithm. We first sample from C and test if it is in C_{free}. The expected information gain associated with the sample is then calculated by taking the difference in entropy of the prior and the posterior distributions (after measurement z has been incorporated) [30],

$$\mathcal{I}(x) = H(p(x)) - H(p(x|z)) \tag{4.1}$$

where entropy is $H(p(x)) = -\int p(x) \log p(x)$. Entropy is a measure of the uncertainty associated with a probability distribution, and the greater the entropy, the higher the associated uncertainty. Put another way, the information gain is a description of the reduction in uncertainty from the initial probability distribution after the expected sensor measurement is incorporated.

The sensor uncertainty sampling strategy is meant to operate directly with the BRM algorithm to provide samples that capture the locations in $\mathcal{C}_{free}$ that have the highest information gain. Here, we recall that for our motivating problem of autonomous navigation and control of a laser-equipped quadrotor helicopter, we are applying the UKF implementation of the BRM algorithm, as described in Chapter 3. In particular, we showed in Section 3.4 and Figure 3-2 that the measure of the information gain, represented by the M matrix in the BRM UKF algorithm, is statistically insensitive to the choice of prior. We have therefore used a constant covariance prior $p(x) = \Sigma_0$ as an approximation, such that $H(p(x)) = P_0$. In addition, to simulate the UKF localization process, we generate sigma points about the sample according to the prior covariance Σ_0. Finally, we use Bayes' rule to compute $p(x|z) = p(z|x).p(x)$. These two modifications result in

$$\mathcal{I}(x) = P_0 - H(p(z|x)) \tag{4.2}$$

where $z = \arg max_z p(z|x)$, and $p(z|x)$ is calculated according to the UKF algorithm. This is done by simulating the sensor measurement at the sample's location and finding the probability of the observing the sensor measurement at each of the sigma points. In general, the lower the probability of observation at the neighboring sigma points, the smaller the entropy of the posterior distribution, and therefore the greater the information gain.

Finally, we normalize the information gains $\mathcal{I}$ so that it lies in the range [0,1] by dividing by P_0. This is done because although we want to bias our sampling to locations with high information gain, we similarly want to ensure that we have sufficiently represented the $\mathcal{C}_{free}$ space so that a path can always be found regardless of the start s_0 and s_g locations. By treating the information gain as a probability term between 0 and 1, we can then use it to probabilistically determine if we should accept or reject the sample.

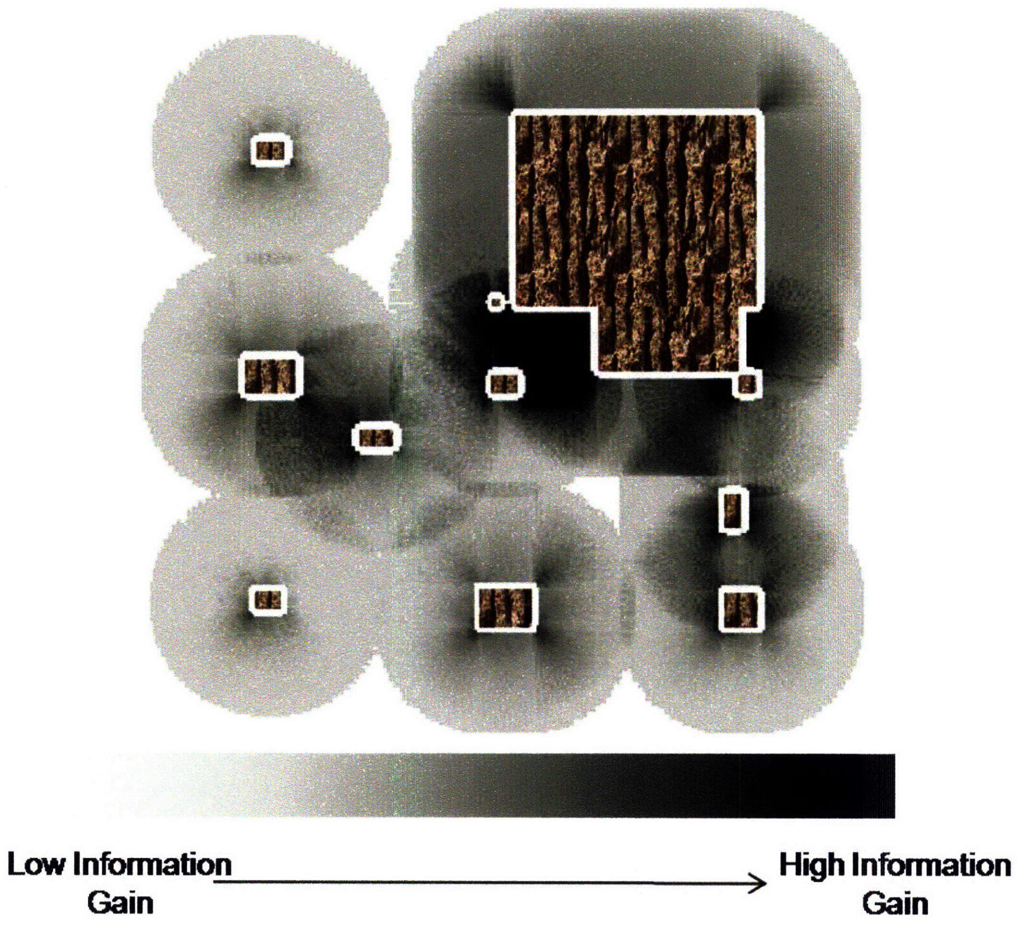


Figure 4-1: Sample map of Sensor Uncertainty Field. The darker pixels indicate regions with greater information gain

Figure 4-1 shows the sensor uncertainty field for a sample map - the parking garage of MIT's Stata Center. The textured structures indicate the parking garage pillars and stairwell (top right). To create the sensor uncertainty field, Equation (4.2) is evaluated at each location (x, y) in $\mathcal{C}_{free}$ for a fixed height and attitude, and we have assumed the use of a laser sensor with $4m$ range. The pixel intensity corresponds to the information gain, where darker pixels indicate locations which are expected to produce greater information gain.

For instance, in Figure 4-1, the region in the center region of the map, which is bounded by four pillars and stairwell, is a region of very high information gain because the sensor measurements are very different for each sample, and thus the entropy of the posterior distribution is very small. Similarly, locations where the associated sensor measurement detects more than one obstacle in the map tend to have higher information gain compared to those that just encounter one obstacle.

To create Figure 4-1, the sensor uncertainty field for every (x, y) coordinate in the map had to be computed. For most purposes, however, computing the field for realistic domains is impractical, and this diagram is shown here only to illustrate the concept. Instead, we observe that the computation of the information gain associated with each location is independent of the information gain of other locations in the map. We can therefore draw samples randomly from $\mathcal{C}_{free}$ and accept them based on their respective information gains.

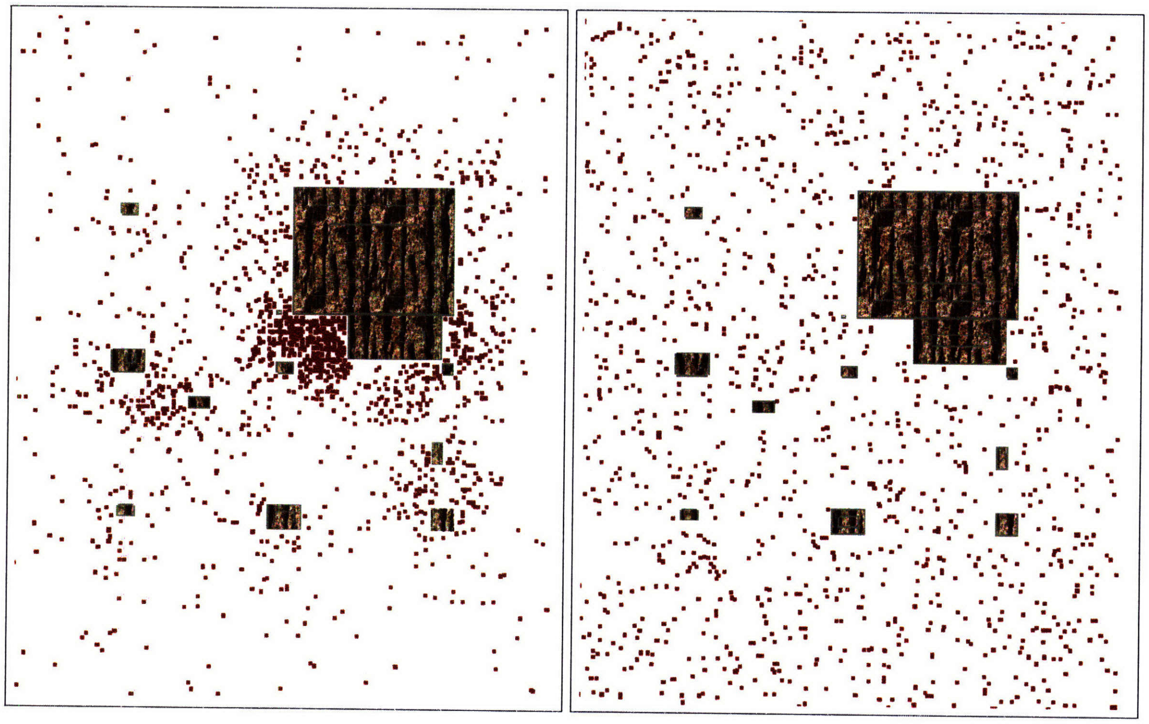
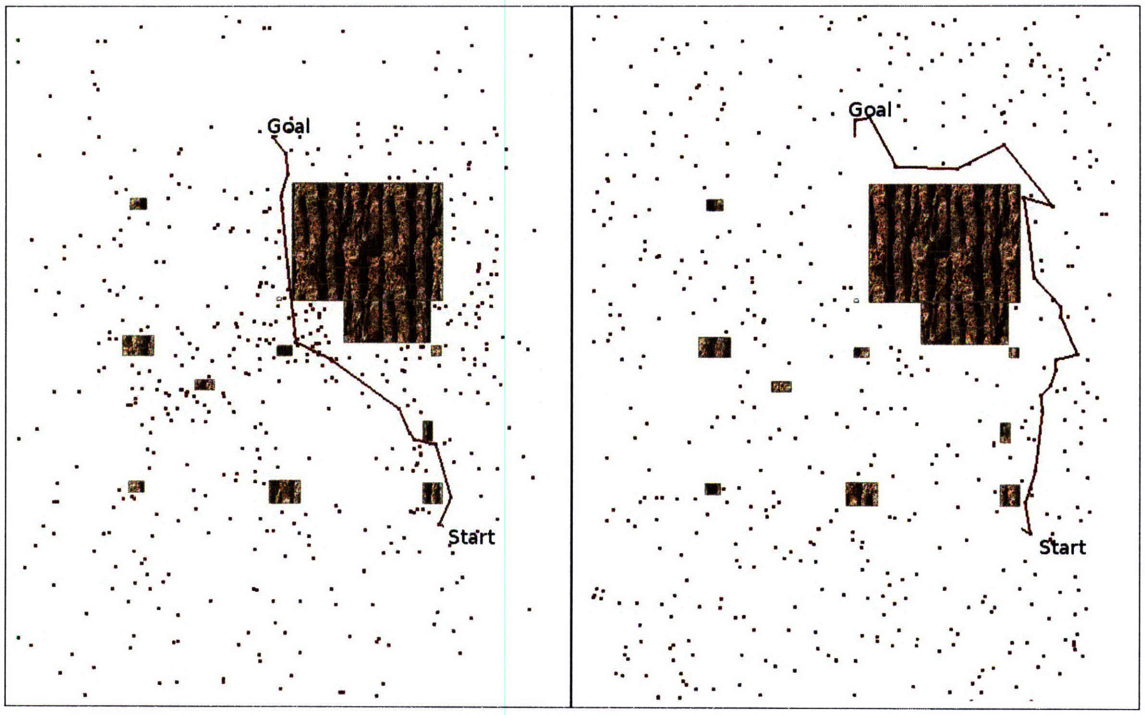

(a) Sensor Uncertainty Sampling (b) Uniform Sampling

Figure 4-2: Distribution of samples using different sampling strategies. 1500 samples were used

Figure 4-2(a) shows the samples drawn according to the sensor uncertainty. Observe that the sampling density is highest in the same region as the dark region in Figure 4-1 (center of map), and is lowest when far from any environmental structure, which consequently provides little or no localization information. For comparison, 4-2(b) shows the samples drawn according to a uniform sampling strategy.



(a) Sensor Uncertainty Sampling (b) Uniform Sampling

Figure 4-3: Paths generated by different sampling strategies. 500 samples were used

The difference in sampling strategies can result in different paths and correspondingly different resultant uncertainties. Figure 4-3 shows a comparison of the paths generated by the sensor uncertainty sampling in comparison to uniform sampling, for the same start and goal locations. The paths created by sensor uncertainty sampling tend to stay in regions with high information gain, since the samples were probabilistically chosen based on the amount of information gain each was expected to provide. In contrast, while the BRM-uniform sampling strategy also attempts to find such a low-uncertainty path, the lack of samples in the regions with high information gain results in a path with higher uncertainty for the uniform sampling strategy.

4.4 Comparison with other sampling techniques

In order to evaluate the effectiveness of the Sensor Uncertainty sampling strategy, we compared it with other sampling strategies that have gained popularity in the literature. Although these algorithms have been proposed to improve the performance of the PRM algorithm, they can nevertheless be used to test the performance of the sensor uncertainty strategy in the BRM context. In this section, we first describe four sampling strategies (Uniform, Gaussian, Bridge, Medial Axis), before reporting the results of the BRM path-planning when using each of these strategies.

Algorithm 7 Uniform Sampling Algorithm

```
Require: Map $\mathcal{C}$, Number of samples $N$,
1: while size of graph < N do
2:     Sample a pose, $c_1$, from $\mathcal{C}$, with equal probability
3:     if $c_1 \in \mathcal{C}_{free}$ then
4:         add $c_1$ to the graph
5:     end if
6: end while
7: return graph
```

4.4.1 Uniform Sampling

As mentioned, uniform sampling is the most basic sampling strategy used by the majority of the sampling-based techniques, including the original PRM and BRM algorithms. Algorithm 7 formally describes the uniform sampling algorithm. This algorithm does not leverage any unique features of the map, and instead merely samples the $\mathcal{C}$space with equal probability to determine if it falls in $\mathcal{C}_{free}$, so that it can be traversed by the agent. Samples that are in the $\mathcal{C}_{free}$ of the map are then added to the graph. By employing a simple collision-check function, the uniform sampling strategy is a very efficient means of obtaining the general connectivity of a given map. Figure 4-2(b) shows an example of the samples generated using this sampling method, while 4-3(b) illustrates a minimum cost path generated when the BRM algorithm is applied.

4.4.2 Gaussian Sampling

Unfortunately, a significant limitation of the uniform sampling strategy is that it often fails to represent important regions in the C_{free} space due to its naive approach towards sampling the given environment. For instance, difficult regions such as narrow corridors and areas around obstacles may not be sampled unless a large number of samples are used, which then incurs a large computation cost. This is especially problematic because often the optimal path does traverse through these particular regions. The configuration space should therefore be sampled in a non-uniform manner.

In this direction, Boor et al. [31] present a Gaussian sampling strategy that attempts to give a better coverage of the difficult parts of the free configuration space, especially those areas that that are close to obstacles. The underlying principle is that clutted areas should get many more samples than open area, not only because the chance that samples in those areas lie in C_{free} is smaller, but also that the motion planning problem through these areas is more difficult.

Algorithm 8 Gaussian Sampling Algorithm

```
Require: Map $C$, Number of samples $N$, Gaussian width $\sigma$
 1: while size of graph < N do
 2:     Sample a pose, $c_1$, from $C$
 3:     Choose a distance, $d$, according to a normal distribution with parameters $p, \sigma$
 4:     Calculate pose, $c_2$, at distance $d$ from $c_1$
 5:     if $c_1 \in C_{free}$ and $c_2 \notin C_{free}$ then
 6:         add $c_1$ to the graph
 7:     else if $c_2 \in C_{free}$ and $c_1 \notin C_{free}$ then
 8:         add $c_2$ to the graph
 9:     else
10:         discard both
11:     end if
12: end while
13: return graph
```

The Gaussian sampling strategy is inspired from the concept of Gaussian blurring, a technique widely used in image processing. This idea is to add a sample that is in C_{free} to the graph only if an obstacle is close to it, according to a normal distribution. Algorithm 8 formally presents the Gaussian sampling algorithm. The algorithm first uniformly samples the C space to obtain a sample, c_1, regardless of whether it is in C_{free} or C_{obs}. A distance value, d, is then chosen according to a normal distribution, and a second sample, c_2, is found by finding the location d distance away from c_1, in the direction of $c_{1,\theta}$. The two samples are then tested to determine if they belong to C_{free} or C_{obs}; if the samples are in separate subspaces of the C space, the sample that is in C_{free} is then added to the graph. For illustration purposes, Figure 4-4 shows the samples generated by the Gaussian sampling strategy.

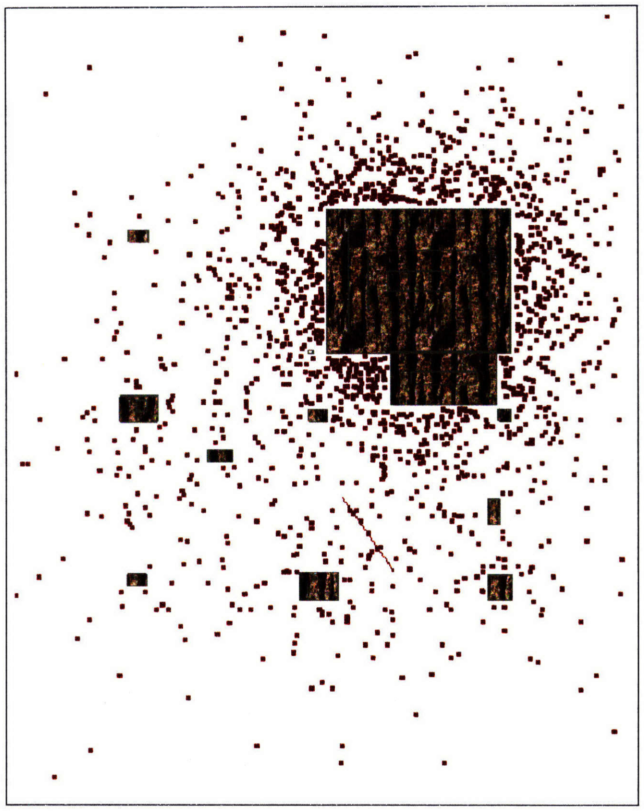


Figure 4-4: Samples generated from Gaussian Sampling

4.4.3 Bridge Sampling

A different algorithm is proposed to address the problem encountered by the uniform sampling strategy of not being able to represent narrow passages of a given environment. Because of the narrow width of the passages, not only do most of the samples in this region fall in C_{obs} and are therefore not represented in the graph, but these passages are often on the optimal path in the path-planning problem, especially in the deterministic case. It is

Algorithm 9 Bridge Sampling Algorithm

```
Require: Map $\mathcal{C}$, Number of samples $N$, Gaussian width $\sigma$
1: while size of graph < N do
2:    Sample a pose, $c_1$, from $\mathcal{C}$
3:    if $c_1 \notin \mathcal{C}_{free}$ then
4:       Choose a point, $c_2$, according to a normal distribution with mean $c_1$ and standard
         deviation $\sigma$
5:       if $c_2 \notin \mathcal{C}_{free}$ then
6:          Choose the point, $p$, that is the midpoint of the segment $\vec{c_1 c_2}$
7:          if $p \in \mathcal{C}_{free}$ then
8:             Insert p into the graph
9:          end if
10:      end if
11:   end if
12: end while
13: return graph
```

therefore necessary to develop strategies that are biased towards finding paths in narrow passages.

To address this problem, the bridge test for sampling narrow passages was developed by Hsu et al. [32]. The key idea is to only add a sample to the graph when it is found to be between two obstacles. Algorithm 9 presents the bridge sampling algorithm. Two samples, c_1 and c_2, are first sampled from the map environment, with c_2 being drawn from a normal distribution with mean c_1 and a given variance σ. If both the samples are found to be in $\mathcal{C}_{obs}$, the midpoint between the two samples is then generated and tested for collisions. This midpoint is then added into the graph if it is in $\mathcal{C}_{free}$. Figure 4-5 shows the samples generated using this strategy.

4.4.4 Medial Axis Sampling

Finally, we present the medial axis sampling strategy, proposed by Wilmarth et al. [33]. This sampling technique seeks to address the same problem as the bridge test, to aid the planner to find paths that will pass through narrow passages. Here, the medial axis is the set of lines in $\mathcal{C}_{free}$ of the map that is equidistant to its two nearest obstacles, and is a one-dimensional graph-like structure that can also be used as a roadmap. However, the medial axis is difficult to compute, and hence the key contribution of this technique is the ability to

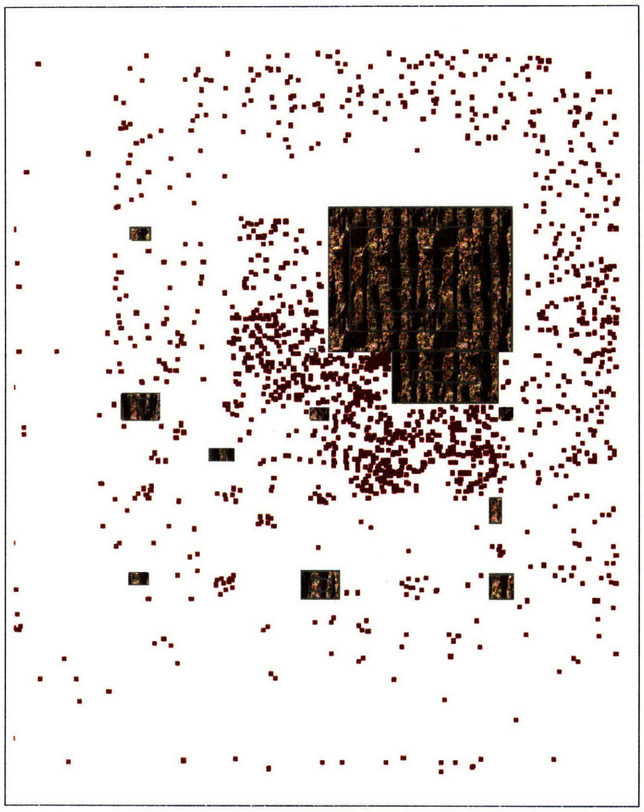


Figure 4-5: Samples generated by Bridge sampling

represent the medial axis of the free space without actually explicitly calculating the medial axis of a given environment. This is done by retracting any point that is sampled in $\mathcal{C}$ in the direction of its medial axis.

Algorithm 10 formally presents the medial axis sampling strategy. In each step of the algorithm, a sample c_1, is first uniformly generated, and the location of the nearest point to c_1 that is on the edge of an obstacle, c_2, is found. The vector connecting the two samples is then stored, and the original sample c_1 is tested for collision. Whether c_1 is in $\mathcal{C}_{obs}$ or $\mathcal{C}_{free}$ determines if we should move towards or away from c_2 towards the medial axis. By using bisection, a sample on the medial axis can thus be easily obtained. Figure 4-6 shows the samples generated from the medial axis sampling strategy.

Algorithm 10 Medial Axis Sampling Algorithm

```
Require: Map $C$, Number of samples $N$
1: while size of graph < N do
2:     Sample a pose, $c_1$, from $C$
3:     Find nearest point, $c_2$, which lies on an obstacle boundary
4:     if $c_1 \in C_{free}$ then
5:         Calculate the vector $c_2c_1$, and choose $c_1$ as the start point, $s$
6:     else
7:         Calculate the vector $c_1c_2$, and choose $c_2$ as the start point, $s$
8:     end if
9:     Use bisection to move $s$ in the direction of $v$ until $c_2$ is no longer the unique point
       on an obstacle that is the closest point to $s$
10:    Add $s$ to the graph.
11: end while
12: return graph
```

4.5 Experiments

4.5.1 Experimental setup

To test the different sampling strategies, we first generated a variety of different map environments. In each of these maps, we then chose a variety of different start and goal locations for the algorithm to plan a path, with the start and goal nodes have a fixed minimum distance. For each set of given conditions, we then ran each of the sampling strategies separately, before running the BRM graph build and graph search processes on each set of samples that were returned by the sampling strategies. A range of sample sizes was used to analyze how the performance of each sampling strategy varied as the map environment became more densely represented, and each of these runs was repeated 10 times before averaging the results. We then evaluated the performance of each strategy by comparing the estimated cost of the optimal path, the ability for the algorithm to find a feasible path, and the time taken for the search.

4.5.2 Results and Evaluation

We ran the different sampling strategies on 3 separate map environments and 10 sets of initial conditions in total. In this section, we present the results from a set of initial conditions,

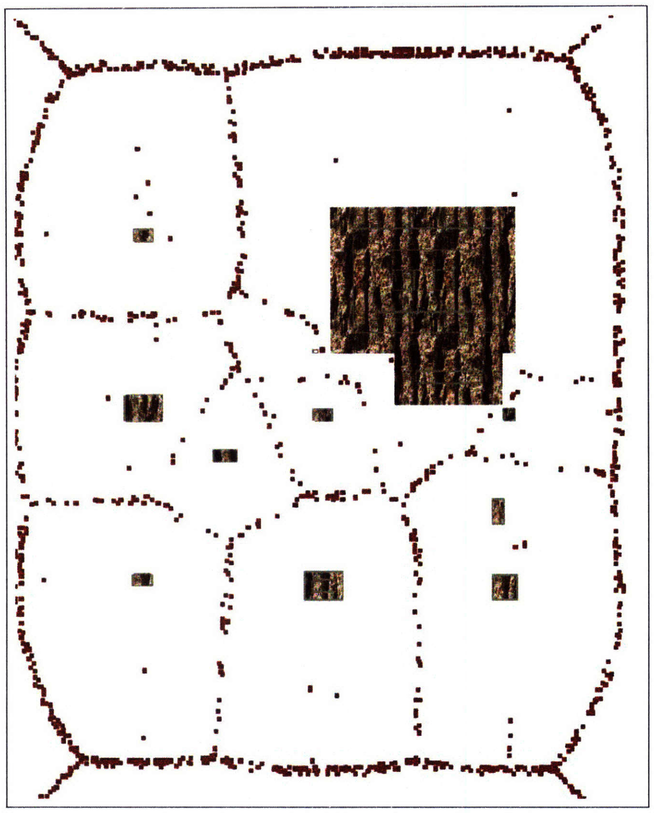


Figure 4-6: Samples generated by Medial sampling algorithm

which we have observed to produce results that are representative of the results in many of the other trials that we conducted.

Figure 4-7 shows the average expected trace of the vehicle's covariance matrix at the goal location for the different sampling strategies, and over a range of sample sizes. In general, the Bridge and Medial Axis sampling strategies tend to fare poorly in finding a good path for the agent to traverse, and the agent tends to reach the goal with large uncertainty in its position in the environment. On the other hand, the remaining three sampling strategies — Sensor Uncertainty sampling, Uniform sampling and Gaussian sampling — all appear to have reasonably similar performance, at least in terms of their average traces of the covariance matrices at the goal location.

To compare the three sampling strategies in greater detail, we zoomed in on the smaller region that focuses only on the three strategies, and plotted the associated error bars indicating one standard deviation from the mean. As Figure 4-8 suggests, the Uniform sampling strategy tends to have a slightly greater resultant uncertainty than the other two sampling

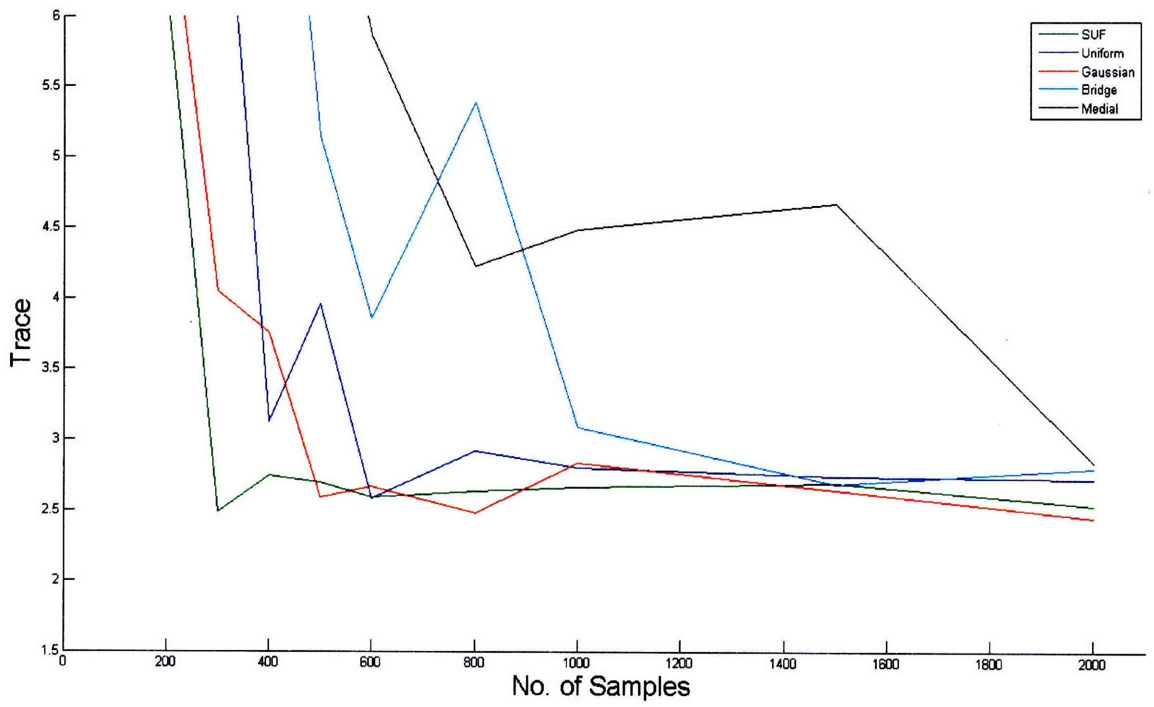


Figure 4-7: Average covariance trace for different sampling strategies over range of sample sizes

strategies, whose difference appears to be statistically insignificant. Here, we note that there is still significant noise in this set of results due to the randomized nature of any sampling strategy. While we could perform a larger number of trials to make a more conclusive statement, we also observe that similar trends have been obtained in the other trials when different initial conditions have been used.

The data suggests that the Gaussian and SU sampling strategies do equally well in generating paths that result in low uncertainty at the goal, and that both take approximately the same number of samples to find paths with low covariances. This can be explained by the fact that the Gaussian sampling strategy biases the samples to regions that are close to obstacles in the environment, and because these obstacles are the very same features that provide the sensor with information of the agent's location in the environment, the samples of both strategies tend to be biased towards the same regions. Furthermore, the nature of our sensor in question, the Hokuyo laser rangefinder, reduces the distinction between the Sensor Uncertainty and Gaussian sampling strategies due to its large field-of-view. Because

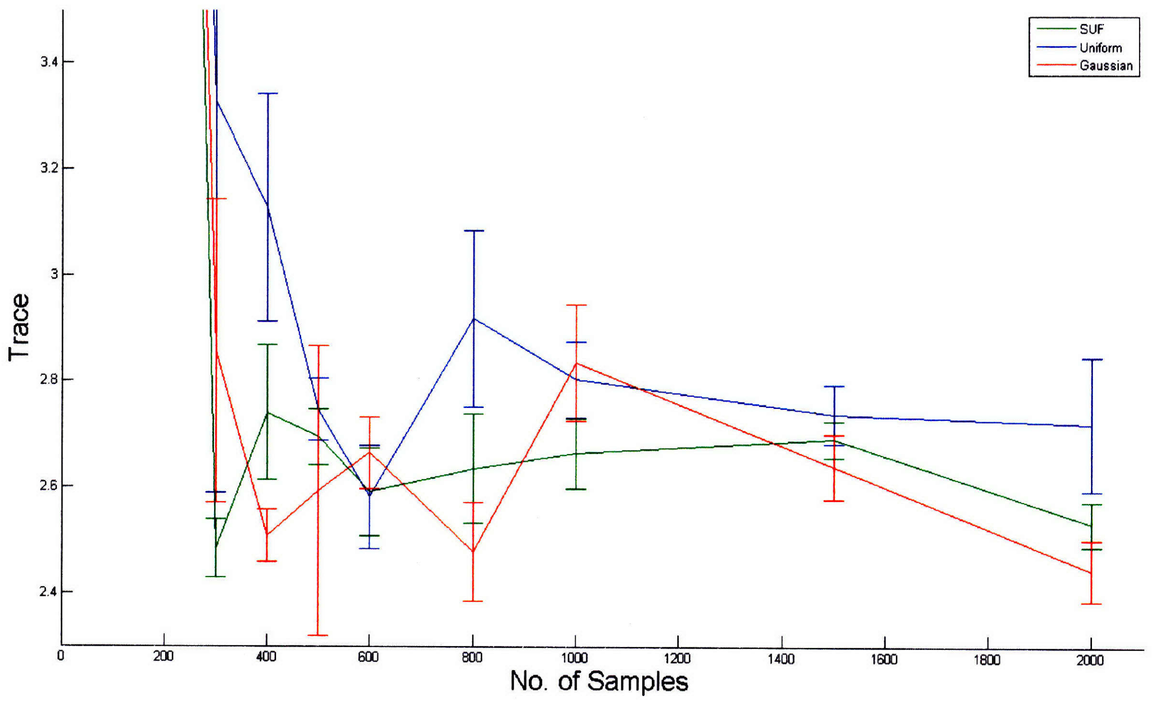


Figure 4-8: Comparison of SU sampling, Uniform sampling and Gaussian sampling strategies: Average trace with error bars

we have assumed that the Hokuyo laser rangefinder has a large 240° field-of-view, the fact that the Gaussian sampling strategy ignored the expected information gain when sampling in the yaw angle becomes less significant. We believe that if we instead extend the BRM to other sensor models such as cameras, which have a significantly reduced field-of-view, or if we include sampling in the roll and pitch axes, we will then witness a noticeable difference between the two sampling strategies.

However, while the resultant covariances of the SU and Gaussian sampling strategies may be similar if a path is found, the data suggests that the SU sampling strategy tends to find a path more often when only a small number of samples are used. Figure 4-9 shows the number of feasible paths that were found by each sampling strategy out of 10 trials, over a range of different sample sizes. The figure does suggest that the SU sampling strategy tends to perform slightly better in generating feasible paths at low sample sizes, though more experiments will need to be conducted to ascertain if the difference is statistically significant.

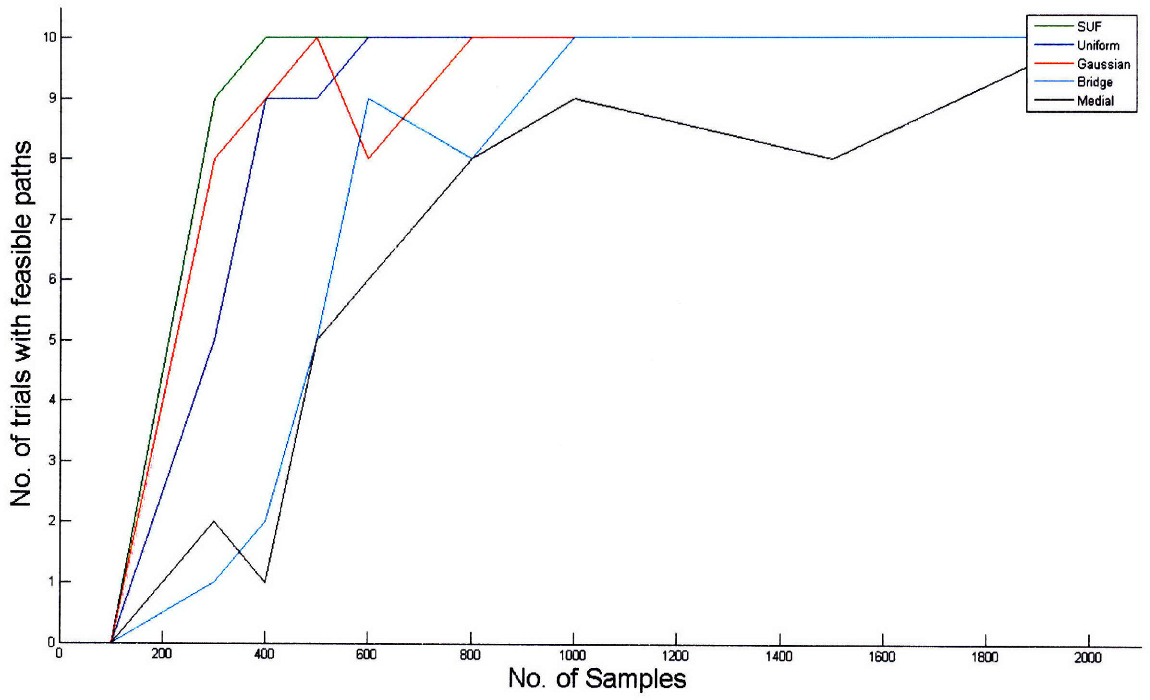


Figure 4-9: Comparison of sampling strategies: Number of feasible paths found out of 10 trials

Finally, we compared the time complexity of each of the algorithms by calculating the amount of time taken by each of the sampling strategies to generate the requisite number of samples. Figure 4-10 shows the distribution of time taken over a range of different sample size, and unsurprisingly, the amount of time taken to sample for all the sampling strategies grows linearly with the number of samples. Furthermore, the graph suggests that while the SU sampling strategy takes a longer time than the uniform and Gaussian sampling strategies, it is not the most computationally intensive sampling strategy. In addition, the total amount of time to perform the sampling (less than 1 second for 2000 samples), suggests that the additional calculation of the information gain in the SU sampling strategy is not a significant constraint on computational power.

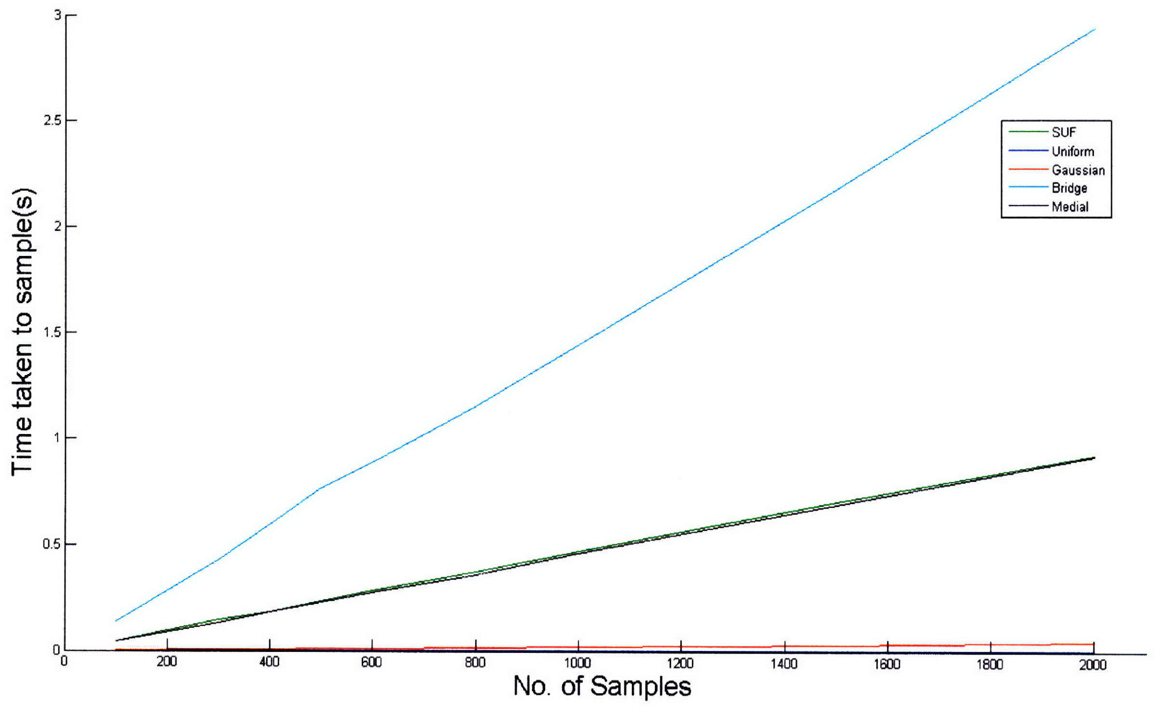


Figure 4-10: Comparison of sampling strategies: Time taken to generate samples

4.6 Conclusion

Sampling-based approaches to path-planning have been popular in the literature because they allow us to represent the feasible space of an environment very efficiently. In general, the sampling strategies that we have discussed in this chapter have all attempted to represent particular aspects of the map, and are suited to the particular applications that they have been designed for. In this regard, the Sensor Uncertainty sampling strategy that we have proposed in this chapter has attempted to leverage off the notion that for the Belief Roadmap Algorithm, we would like to plan paths through regions where we can maximize the amount of information available. We have therefore developed an information-based sampling strategy that enables us to generate minimum-cost paths without having to incur the large computation cost associated with building a graph that contains a large number of samples.

Chapter 5

Helicopter Implementation

5.1 Introduction

Algorithms are meaningless if they cannot actually be used in the real world. In this chapter, we detail the efforts that we have made towards implementing the algorithms described in previous chapters on an actual helicopter robotics platform. In particular, we describe the steps involved in developing a quadrotor helicopter, equipped with a Hokuyo laser rangefinder sensor, to navigate autonomously in an indoor environment. We first describe the key hardware components used in building the system, as well as the software architecture that was developed for achieving autonomous navigation. Finally, we conclude with some results of experiments testing the helicopter's ability to execute the trajectories computed by the different planning algorithms, thereby illustrating the difference in navigation performance that can be achieved when uncertainty is taken into account during planning.

5.2 Hardware

We used numerous pieces of hardware for achieving autonomous navigation in a 3-dimensional environment, including an Ascending Technologies Quadrotor helicopter, a Hokuyo laser range-finder sensor, and an Intel Research Labs iMote2 wireless sensor node platform. We describe each of these in turn.

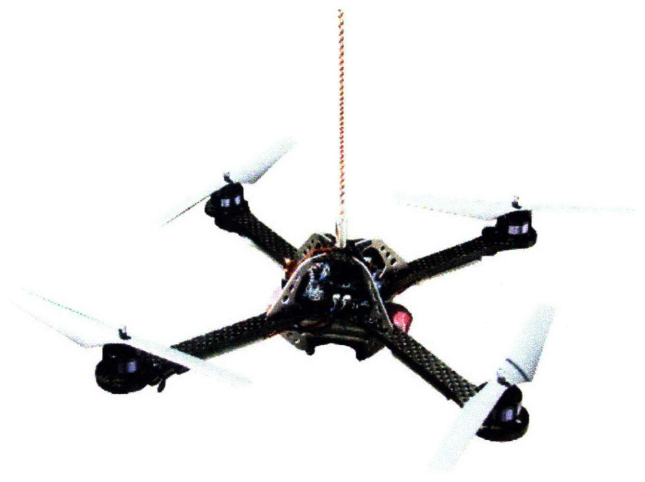


Figure 5-1: AscTec Hummingbird, developed by Ascending Technologies GmbH

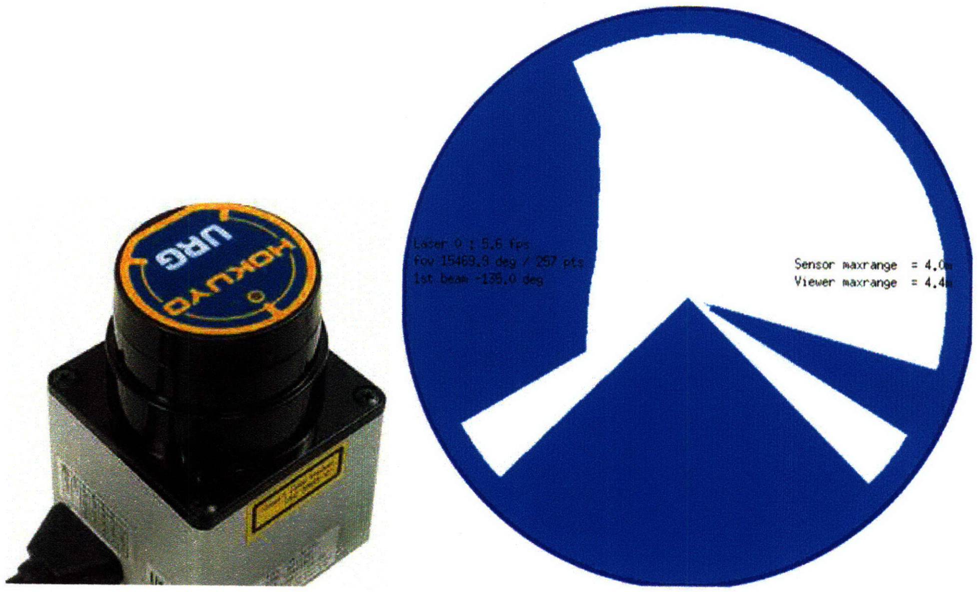
5.2.1 Helicopter

Figure 5-1 shows the helicopter platform that was used for our experiments — the AscTec Hummingbird. Quadrotors have proven to be an increasingly attractive rotary-wing concept, especially at the micro aerial vehicle (MAV) scale. This is not only because they provide significantly more stability than conventional tail-rotor helicopters, but they can also provide large amounts of thrust at small scales to carry non-trivial payloads. For instance, our quadrotor helicopter, designed and manufactured by Ascending Technologies GmbH [2], has a small form factor of less than $53cm$ in any dimension, but is still able to carry payloads of $200g$ for up to 12 minutes. Without any payload, the quadrotor helicopter is able to stay airborne for up to 25 minutes. The vehicle's small size makes it an ideal candidate for operating in an indoor environment, and if we can achieve relatively accurate control of the helicopter, we will be able to make it navigate through doors and other indoor structures.

The quadrotor helicopter is powered by four brushless motors, each of which is attached to a single propeller. By having two propellers each rotating in opposite directions, the quadrotor is able to provide thrust to the vehicle while causing only minimal torque. This helicopter platform by Ascending Technologies is also equipped with onboard attitude stabilization, actively canceling out any rotation in the roll and pitch axes caused by the vehicle dynamics. This stabilization is accomplished by performing sensor and data fusion locally onboard the helicopter at a rate of $1Khz$, thereby providing a stable platform upon which we can perform autonomous path-planning and test the effectiveness of our algorithms. In addition, the onboard attitude stabilization enables us to use a simple PD controller to stabilize the dynamics of our helicopter and enable it to hover in a fixed position. With four different control signals (yaw, pitch and roll angles, and thrust), we are able to control the three translational axes and heading of the vehicle relatively easily once the pitch and roll angles are constrained to level flight. This means that we have full control authority of the vehicle and can therefore ignore most of the vehicle's dynamics. So long as we are not planning to perform fancy acrobatics onboard the helicopter, the quadrotor platform is a robust system that can be approximated as a holonomic robot, allowing us to treat the problem as a kinematic motion planning problem.

Despite the stability of the platform, however, it should still be emphasized that the control problem for a quadrotor helicopter is substantially more challenging than that of a ground robot. A helicopter requires constant control input just to achieve basic hovering capability, in contrast to a ground robot that can stay in a fixed position to try and relocalize itself if it is lost. In the context of our research problem, there is therefore a greater need for accurate, real-time and robust estimates of the helicopter's state for effective localization and control. This underscores the need for us to plan trajectories that minimizes the expected uncertainty of the vehicle.

Traditionally, the control inputs for the helicopter are sent via a $72Mhz$ RC radio transmitter that is operated by a human pilot. However, because we are seeking a fully autonomous helicopter, we modified the communication protocol slightly for the helicopter to receive control inputs from the base station via a 2.4Ghz Zigbee datalink. This datalink is also used by the helicopter to send sensor information, such as acceleration and inertial measurement unit (IMU) information, back to the base station for state estimation and feedback.



```
Laser 0 : 5.6 fps
fov 15469.3 deg / 257 pts
1st beam -135.0 deg
```

```
Sensor maxrange  = 4.0m
Viewer maxrange  = 4.4m
```

Figure 5-2: (a) Hokuyo Laser rangefinder and (b) Sample laser range scan

5.2.2 Laser Range-finder

The quadrotor's 200g payload capability enables us to mount a Hokuyo laser rangefinder sensor on the frame of the helicopter, giving local sensor information of the helicopter's immediate surroundings. The Hokuyo laser rangefinder sensor has a planar field-of-view of 240° and can sense obstacles in its environment up to a range of $4m$. Figure 5-2 shows the Hokuyo laser rangefinder sensor and a sample range scan return.

The laser is mounted in the X-Y plane of the helicopter, providing information of obstacles in the helicopter's surroundings that are approximately at the same height as the helicopter. In addition, we obtained information of the helicopter's current height by modifying the laser with an optical prism to redirect 20 ° of the helicopter's field-of-view downward, thus providing a small set of range measurements in the (downward) z direction (see Figure 1.2). In reality, we observe that these re-directed downward laser measurements have a maximum range of $1.5m$, because the redirection causes a significant power loss in the range scan. Furthermore, the measurement of the ground plane is also relatively noisy, although sufficient for closed-loop altitude control. With a single laser range scan, the helicopter therefore obtains information of its position, orientation and altitude relative to obstacles and features in the environment.

To transmit the laser data to the base station for processing and state estimation, we make use of the Intel iMote2 sensor node package, kindly donated by Intel Labs Seattle and the University of Washington, as an onboard processor with wireless Bluetooth capability. In our current setup, the iMote2 merely acts as a router to transmit the laser data from a USB datalink into Bluetooth packets, before transmitting these packets to the base station wirelessly for post-processing and state estimation. Nevertheless, the iMote2 does have computational processing power that can be harnessed, and hence in the near future, we hope to extend this research further by centralizing computation onboard the helicopter, thereby creating a completely independent platform for autonomous 3-D path-planning and navigation.

5.3 Software

Having described the necessary hardware used in this research, we next turn to the software architecture that we employed to equip the helicopter with the capability to perform path-planning and navigation autonomously.

As a starting point, we developed most of our software by building upon the CARMEN Robotics Toolkit,[1] an open-source collection of software frequently used in mobile robot control. The open-source package not only provides a convenient code base of basic robotic navigation and planning capabilities that was used for further development, but its modular approach to software coding also makes the entire system relatively robust against isolated hardware failures.

5.3.1 Software Architecture

Figure 5-3 describes the software architecture employed in achieving autonomous control and path-planning of the helicopter. Because we desire to solely rely on local sensors that are onboard the helicopter, data is constantly streamed from the quadrotor to the base station and used for state estimation. This data comes in two forms - Inertial Measurement Unit (IMU) data from the vehicle, as well as the laser range scans from the Hokuyo laser

[1]CARMEN. http://carmen.sourceforge.net/home.html

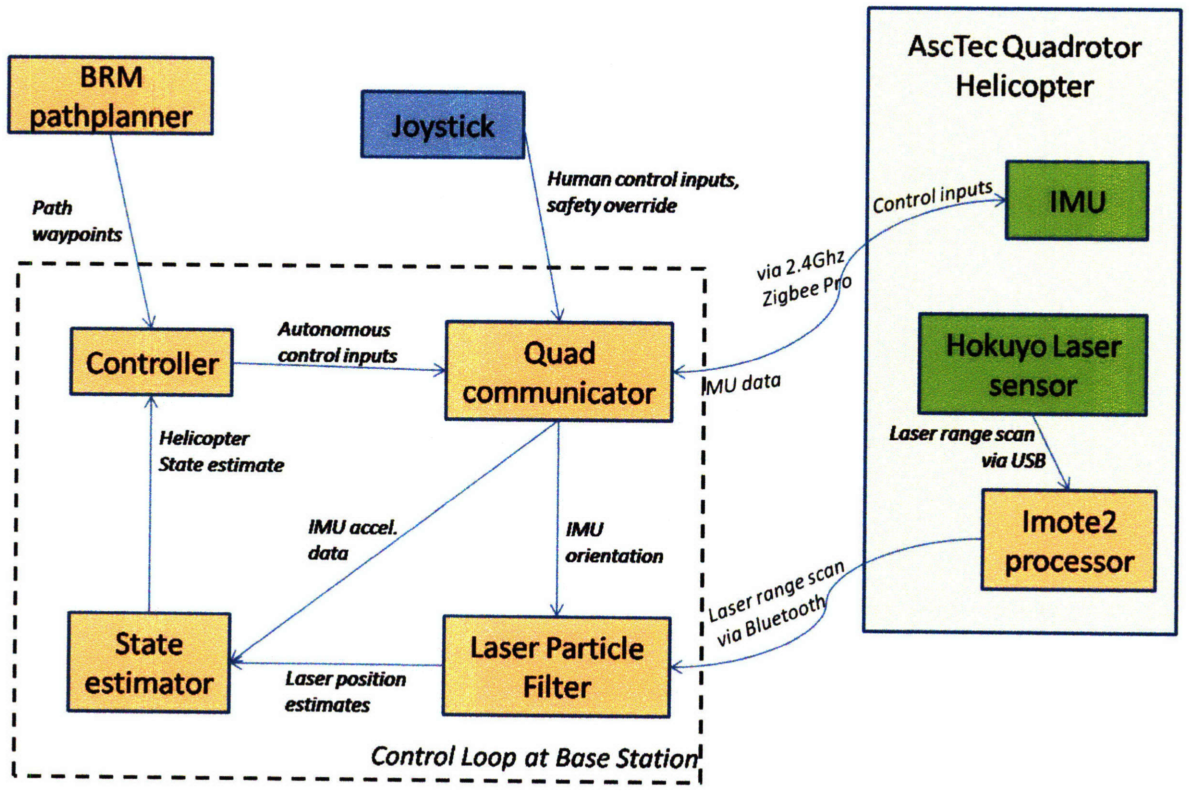


Figure 5-3: Software architecture for autonomous control of quadrotor helicopter

range-finder sensor mounted on top of the helicopter. The IMU data contains both raw and processed measurements, information on the vehicle's translational acceleration, its angular velocity, as well as an estimate of its current pitch and roll rotation angles (which the vehicle uses for onboard attitude stabilization). This data is streamed to the base station via the $2.4Ghz$ Zigbee Pro datalink, and updates at approximately $15hz$. Similarly, the laser range scans from the Hokuyo laser sensor are streamed through the iMote2 and sent to the base station via Bluetooth datalink. Due to the bandwidth limitations of the Bluetooth communication interface, we downsampled the laser by a factor of two to reduce the size of the data packets, whilst still retaining the same field-of-view and the $10hz$ update rate.

5.3.2 State Estimation

The streamed data from the helicopter is first used by a particle filter to estimate the helicopter's position relative to a given map environment. The altitude data from the IMU is used to determine the orientation of the plane within which the laser range scans should be projected in, and by comparing the projected laser range scans with the given map, the likelihood that the helicopter is at a particular pose in the map can be computed. This gives each particle, which represents an x, y, z and yaw estimate of the helicopter's position, an associated weight that is then used for subsequent resampling, as well as for calculating the mean estimate of the helicopter's position at every instant in time.

It is worth recognizing here that by using the particle filter to perform part of the state estimation, we are employing a different localization technique from those assumed in the Belief Roadmap Algorithm, such as the Extended Kalman Filter (Chapter 2) and the Unscented Kalman Filter (Chapter 3). This choice was made because we found that while the UKF is an effective technique for localizing in many environments using a laser rangefinder sensor, there are nevertheless instances when the UKF would diverge. In contrast, the particle filter was found to provide good estimates of the helicopter's position, especially when there were significant non-linearities in the measurement model. Nevertheless, the Kalman filter family of algorithms provide nice parametric representations of the control and measurement updates during the localization process, and hence allows us to estimate the covariance propagations in an efficient manner, as per the BRM. We therefore believe that it is reasonable to implement a different localization technique onboard the helicopter from the one used in the BRM, although we hope to perform more analysis in the near future to fully understand the conditions under which the UKF fails, and why this is so from the theoretical standpoint.

The position estimate of the helicopter is then sent to a state estimator, which is required to estimate both the position and velocity of the helicopter so that PD control can be applied in each of the translational axes, as well as the yaw rotation axis. Here, we found experimentally that while the laser particle filter estimate provides good estimates of the helicopter's position, the inherent noise in the estimates often produces high frequency noise that makes the extraction of accurate velocity from the laser estimates alone difficult. On the other hand, we do have data from accelerometers, which provide translational accelerations that can be integrated to recover the translational velocities. However, using accelerometers to recover velocities encounters a different problem - a low frequency noise often exists in the measurements, which results in a wind-up of the velocity estimates over time.

A widely used technique to avoid this problem has been to leverage on the complementary characteristics of the two measurement sources. By incorporating both types of sensor data in a data-fusion filter such as the complementary filter or the Kalman filter [23], both the low and high frequency noise can be filtered away to obtain a more accurate velocity estimate. This step is performed in the state estimator, and outputs a filtered estimate of the helicopter's position and velocity estimate in the global environment.[2]

5.3.3 Controller

This state estimate is used as a means of feedback for control the helicopter to follow a particular trajectory. Given a map $\mathcal{M}$, a start s_0 and goal s_g position, the lowest final variance path is pre-computed by the BRM pathplanner and fed to the controller. At every time step, the controller therefore takes in a state estimate and uses a PD control, based on a set of finely tuned PD gains, to determine the required control actions that should be transmitted to the helicopter. The controller is therefore effectively trying to hover the helicopter above a point on the trajectory, though this point moves along the trajectory at a certain desired speed.

The final software component that is used to close the measurement-localization-control loop for the helicopter is the communication interface between the base station and the helicopter. This interface transmits and receives the different data packets for wireless control of the vehicle. In addition, it also takes in user input via the joystick, thereby acting as the overarching safety switch in an emergency. The joystick enables the user to take over

[2]We had developed a Kalman Filter to augment the state estimation in the velocity domain, but at the time of this thesis submission, we ended up being able to obtain relatively accurate velocity estimates from the laser data alone, and hence decided upon using the accelerometers data only for future work.

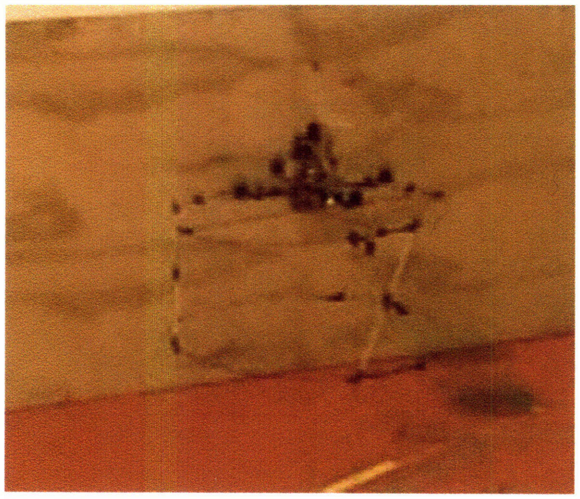


Figure 5-4: Quadrotor helicopter performing autonomous hovering, relying on local sensors for state estimate.

full control of the helicopter at any point in time, and also to control only particular axes if desired. The control messages are then sent to the helicopter via the Zigbee datalink at approximately $15hz$.

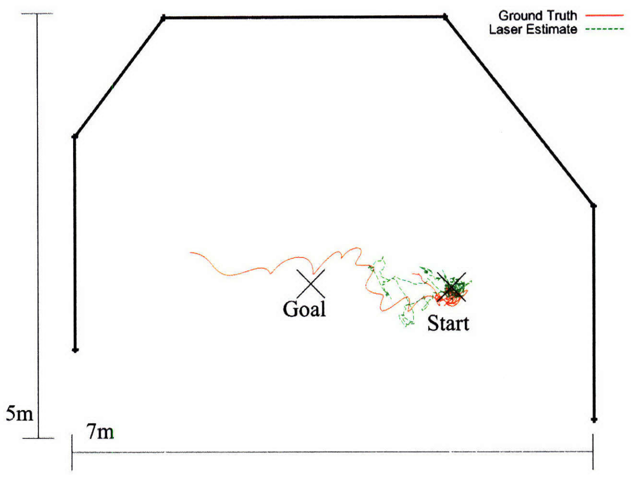
Figure 5-4 shows our autonomous helicopter in action. The vehicle attempts to hover over a particular position, constantly using the state estimates to correct its control actions.

5.4 Experiments

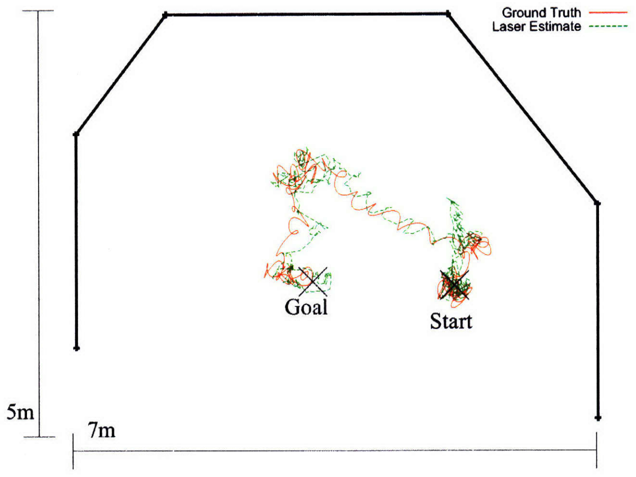
Finally, we present some results of the helicopter's performance when different path-planning algorithms are implemented. Here, we seek to demonstrate that planning algorithms that take into account uncertainty when planning a trajectory will result in measurably better performance when implemented on an actual helicopter platform.

Figure 5-5 shows a bird's eye view of an example indoor environment that was used to compare the algorithms. Here, the BRM algorithm, using the UKF localization formulation and the sensor uncertainty sampling strategy, shown in Figure 5-5(a), is compared with the PRM algorithm that uses the uniform sampling strategy to sample the configuration space, shown in Figure 5-5(b).

The green line shows the helicopter's position estimate from the laser sensor measurements, and is used for localization and control. On the other hand, the red line shows the



(a) PRM trajectory, uniform sampling



(b) BRM trajectory, SU sampling

Figure 5-5: Localization performance of helicopter executing different path-planning algorithms with different sampling strategies.

true path of the helicopter that was obtained using a motion capture system. The helicopter is required to plan a path from the starting position to the end goal, and must localize itself using the laser sensor while executing the trajectory. For the experiments, we first plan a path for the helicopter using each algorithm. We then attempt to fly the helicopter autonomously through the environment using the planned trajectories, and determine if the helicopter is able to successfully reach the end goal by maintaining accurate localization. The performance of each algorithm is determined by comparing the laser-localized state estimate against the ground truth of the vehicle, as captured by a motion capture system.

Figure 5-5(a) shows an example trajectory generated by the traditional PRM planner, which finds a direct path from the start point to the goal location. Because this plan ignores the helicopter's need for sensor information to localize itself, the helicopter gets lost while in flight, incurring a position estimation error of at least $3.6m$ and falsely believing that it is still in the center of the environment when it has already flown to the left. This localization failure eventually causes the helicopter to crash into the left wall, thereby again underscoring the need for accurate and consistent state estimates for effective control of the quadrotor helicopter.

On the other hand, an example BRM trajectory using sensor uncertainty sampling enables the helicopter to stay well-localized, incurring a position estimation error of only $.17m$, as shown in Figure 5-5(b). The helicopter achieves this by detouring from the shortest path toward areas of high sensor information, successfully reaching its desired goal with high certainty. This demonstrates that the BRM trajectory leads to measurably more accurate performance.

5.5 Conclusion

In this chapter, we have shown that not only do we have an algorithm that enables us to plan efficiently in belief space, but that this algorithm can be implemented onboard an actual robotics platform to achieve autonomous path-planning and control, while relying solely on sensors that are local to the helicopter platform. We have also shown experimentally that the BRM algorithm can be used for planning paths for a helicopter to autonomously navigate a given environment. Finally, we believe that in the near future, we will be able to centralize the system entirely onboard the quadrotor helicopter, so that we can develop an autonomous helicopter platform that is truly independent.

Chapter 6

Conclusion

6.1 Summary

We began this thesis with the motivating problem of achieving autonomous path-planning, navigation and control of a quadrotor helicopter in a GPS-denied environment. Due to the lack of external sensors that can provide accurate information of the helicopter's position in a global environment, the helicopter must rely only on local, onboard sensors, such as the Hokuyo laser rangefinder sensor and an IMU for information of its position relative to a given map. Unfortunately, because the laser only has a limited range and field-of-view, it is possible for the helicopter to lose track of its own position in some parts of the environment.

Nevertheless, we have shown in this thesis that the Belief Roadmap Algorithm [4] can be applied to our problem to plan trajectories through the environment by incorporating a predictive model of sensing, thereby allowing the planner to plan paths that will minimize the likelihood of the vehicle getting lost and maximizing the probability of the vehicle being able to reach its goal. We have shown that while the original BRM algorithm utilizes the symplectic form of the Extended Kalman Filter to perform covariance propagation along a path very efficiently, the algorithm can similarly be extended to use the Unscented Kalman Filter algorithm. Furthermore, we have proposed an information-based sampling strategy that attempts to sample a given map in regions where good paths are expected to be found. By choosing an appropriate sampling algorithm, the BRM is therefore able to find better trajectories with fewer samples than if it had used a naive uniform sampling strategy.

Finally, we have demonstrated that the BRM algorithm can be implemented onboard our quadrotor helicopter, enabling it to execute a trajectory from a given start to goal without getting lost.

6.2 Suggestions for Future Work

The techniques that have been developed in this thesis, as well as the results that have been presented, have revealed many new areas for future research, which we hope for the opportunity to explore further in the near future. Our suggestions for future work can be broadly classified into two categories - further extensions to the Belief Roadmap algorithm, as well as to achieve greater autonomy, robustness and control of our quadrotor helicopter.

6.2.1 Extensions to the Belief Roadmap Algorithm

We believe that the Belief Roadmap algorithm holds significant promise of being an efficient method for planning in *belief space*, which thus far has remained a challenging problem due to the computational issues involved. In particular, we would like to extend the work further in the following ways:

1. **Alternative cost functions:** In the current implementation of the BRM, the cost function seeks to minimize the vehicle's uncertainty at the goal after navigating a particular trajectory. There are, however, other objective functions that could be used depending on the issue of concern. For example, if we are interested in reducing the likelihood of not getting lost along the trajectory, our objective function could instead be to minimize the maximum uncertainty encountered along the path. Alternatively, if we want to avoid the risk of collisions, we could add the additional constraint that the covariance ellipse must not intersect with any of the known obstacles in the environment.

2. **Additional sensor models:** Given the form of the BRM transfer function, the BRM algorithm is dependent on the type of sensors used for localization. To date, the BRM algorithm has been effectively applied to both ultra-wideband and laser rangefinder

sensor models. With the increasing prevalence of other sensors such as miniature cameras and the introduction of new systems such as the SwissRanger range imaging camera, it would be meaningful to test if the BRM can similarly be applied to these sensor models as well.

3. **Search heuristic:** Thus far in the research, we have used the breadth-first search strategy in the BRM algorithm to search through the graph of mean poses. This was done to ensure that we will find the path in the graph that maximizes our objective function. However, it is possible that more efficient search strategies for searching the belief graph exist. For instance, if we can devise an admissible heuristic for the cost function, we could then employ the more efficient A* search strategy.

4. **Extend BRM to planning in uncertain maps:** The problem statement in this thesis assumes the knowledge of an accurate map of the environment. However, it may be possible to extend the BRM to perform path-planning and exploration in environments where the map is not known *a priori*, i.e., to operate in the context of SLAM. Given the ability of the BRM algorithm to perform efficient covariance propagation using the transfer function, we believe that the BRM could be adapted to execute information-based exploration that maximizes both the information obtained about the map and the localization ability of the vehicle.

6.2.2 Greater autonomous control of quadrotor helicopter

In this research, we have achieved autonomous control of our quadrotor helicopter, and have demonstrated its ability to navigate a path planned by the BRM. Nevertheless, there are a number of improvements we would like to make to enhance both the quality and robustness of the helicopter's autonomy capability.

1. **State estimation:** As described in Chapter 5, both the position and velocity state estimates of the helicopter were obtained by matching the orientation data and laser range scans to a given map. Unfortunately, attempting to obtain the velocity data by differentiating the laser position estimates yields noisy velocity estimates, which

then affect the quality of the control. Instead, we observe that velocity estimates can actually be obtained independently from the map by simply comparing between successive laser range scans to obtain the relative change in position over time. Although this adds an additional filtering level to the software architecture, we believe that it may yield better velocity estimates for higher quality control.

2. **Helicopter model:** Throughout this research, we have assumed a naive model of our helicopter, based purely on our intuition obtained from flying the vehicle manually. A natural extension of this research is therefore to create a more representative model of the helicopter using system identification techniques. This will not only give us with a better model to obtain the control update equations for localization and the BRM algorithm, but will also allow us to attempt more complicated control maneuvers and trajectories in future.

3. **Characterizing the UKF:** While carrying out the experiments onboard the helicopter, we noticed that while the Unscented Kalman Filter localizes the helicopter effectively in many environments, it nevertheless diverges in some environments, hence resulting in our use of the particle filter for online localization. We would therefore like to understand the Unscented Kalman Filter further, and be able to characterize exactly under what conditions does the filter fail in localization.

4. **Onboard computation:** Finally, we hope in the near future to leverage the computational capabilities onboard the helicopter, both on the iMote2 and the helicopter's X-3D high-level processor, to centralize all autonomous functionalities onboard the helicopter, thereby closing the loop internally and avoiding the need for a base station. This will allow us to avoid bandwidth issues inherent in our wireless communications.

Bibliography

[1] C. Kemp. Visual Control of a Miniature Quad-Rotor Helicopter.

[2] D. Gurdan, J. Stumpf, M. Achtelik, K.M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Proc. ICRA*, 2007.

[3] Hokuyo Automation, http://www.hokuyo-aut.jp/products/urg/urg.htm. *Scanning laser range finder for robotics.*

[4] S.J. Prentice and N. Roy. The belief roadmap: Efficient planning in linear pomdps by factoring the covariance. In *Proceedings of the International Symposium on Robotics Research*, 2007.

[5] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[6] S.J. Julier, J.K. Uhlmann, and H.F. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of the American Control Conference*, pages 1628–1632, 1995.

[7] R. He, S.J. Prentice, and N. Roy. Planning in Information Space for Quadrotor Helicopter in a GPS-denied Environment. *Robotics and Automation, 2008. Proceedings. ICRA'08. IEEE International Conference on*, 1, 2008.

[8] B. Bonet and H. Geffner. Planning with incomplete information as heuristic search in belief space. *Proc. AIPS*, pages 52–61, 2000.

[9] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, February 1983.

[10] P. Leven and S. A. Hutchinson. Real-time path planning in changing environments. *IEEE Transactions on Robotics & Automation*, 21(12):999–1030, December 2002.

[11] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for robot path planning. In G. Giralt and G. Hirzinger, editors, *Proceedings International Symposium on Robotics Research*, pages 249–264. Springer-Verlag, New York, 1996.

[12] S. Caselli and M. Reggiani. ERPP: An experience-based randomized path planner. In *Proceedings IEEE International Conference on Robotics & Automation*, 2000.

[13] J. Pettré, J.-P. Laumond, and T. Siméon. A 2-stages locomotion planner for digital actors. In *Proceedings Eurographics/SIGGRAPH Symposium on Computer Animation*, pages 258–264, 2003.

[14] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. *Journal of Computational Biology*, 26(2):282–304, 2002.

[15] J. van den Berg and M. Overmars. Roadmap-based motion planning in dynamic environments. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1598–1605, 2004.

[16] Y. Yu and K. Gupta. On sensor-based roadmap: A framework for motion planning for a manipulator arm in unknown environments. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1919–1924, 1998.

[17] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 630–637, 1998.

[18] R. Bohlin and L. Kavraki. Path planning using Lazy PRM. In *Proceedings IEEE International Conference on Robotics & Automation*, 2000.

[19] B. Burns and O. Brock. Sampling-based motion planning using predictive models. In *Proceedings IEEE International Conference on Robotics & Automation*, 2005.

[20] P. Isto. Constructing probabilistic roadmaps with powerful local planning and path optimization. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2323–2328, 2002.

[21] P. Leven and S. A. Hutchinson. Using manipulability to bias sampling during the construction of probabilistic roadmaps. *IEEE Transactions on Robotics & Automation*, 19(6):1020–1026, December 2003.

[22] J. O. Berger. *Statistical Decision Theory*. Springer-Verlag, Berlin, 1980.

[23] R.E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[24] S. Thrun, Y. Liu, D. Koller, A.Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.

[25] D. Vaughan. A nonrecursive algebraic solution for the discrete Riccati equation. *Automatic Control, IEEE Transactions on*, 15(5):597–599, 1970.

[26] S. Thrun, W. Burgard, and D. Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), 2005.

[27] H. Takeda and J.C. Latombe. Sensory uncertainty field for mobile robot navigation. *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*, pages 2465–2472, 1992.

[28] H. W. Kuhn. Extensive games and the problem of information. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games*, pages 196–216. Princeton University Press, Princeton, NJ, 1953.

[29] H. Takeda, C. Facchinetti, and J.C. Latombe. Planning the motions of a mobile robot in a sensory uncertaintyfield. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(10):1002–1017, 1994.

[30] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems 12*, volume 12, 1999.

[31] V. Boor, M.H. Overmars, and A.F. van der Stappen. Gaussian sampling strategy for probabilistic roadmap planners. *PROC IEEE INT CONF ROB AUTOM*, 2:1018–1023, 1999.

[32] D. Hsu, T. Jiang, J. Reif, and Z. Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, 3, 2003.

[33] S.A. Wilmarth, N.M. Amato, and P.F. Stiller. MAPRM: a probabilistic roadmap planner with sampling on the medialaxis of the free space. *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, 2, 1999.