AN INTEGRATED APPROACH TO ROBOTIC NAVIGATION UNDER UNCERTAINTY

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Bin Wu
May 2011

This dissertation is online at: http://purl.stanford.edu/dq739bm0780

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Tze Lai, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Thomas Cover, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Peter Glynn

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Autonomous robot navigation has been gaining popularity in the field of robotics research due to its important and broad applications. Sequential Monte Carlo methods, also known as particle filters, are a class of sophisticated Bayesian filters for nonlinear/non-Gaussian model estimation, and have been used for the simultaneous localization and mapping (SLAM) problem in robot navigation in lieu of extended Kalman filters. However, the current particle filters, and their derivatives such as the particle-based SLAM filters for robotic navigation, still need further improvement to have better trade-off between performance and complexity in order to be used for online applications. Also, the current robot navigation approaches often focus on one aspect of the problem, lacking an integrated structure.

In this work, we designed better sampling proposal distributions for particle filters, and demonstrated their superiority in simulation. Then, we applied our new particle filters to design and implement improved particle-based SLAM filters for the application of the SLAM problem in robot navigation, and tested using both simulation and outdoor experimental datasets. Finally, we incorporated the new particle-based SLAM filters in the design of a new framework for solving robotic navigation problems under uncertainty in a continuous environment. The framework balances between exploration and exploitation, and integrates global planning algorithms, local navigation routines, and exploration procedures in order to achieve the global goal, overcoming many common drawbacks of current approaches.

Acknowledgements

This work would not be made possible if it were not for the following people who gave me guidance, offered me collaboration, and provided me with help.

First of all, I would like to express my greatest gratitude to my advisor, Prof. Tze Leung Lai, for his advice and guidance. Since Prof. Lai took me on board three years ago, I have learned a lot from him, and in particular, many insights from Prof. Lai have enlightened my thinking and inspired me with new ideas in solving the hard problems in my research.

Many thanks to Prof. Thomas Cover and Prof. Peter Glynn for their time being on my committee, and also reading and giving comments on my dissertation. I also want to thank Prof. Papanicolaou for being the Chair of my PhD oral defense.

I am thankful to Prof. Yuguo Chen from UIUC for his collaboration on some of the material included in this work. Without him, some of my work would not have been publishable.

Also, I'd like to express my gratitude to all my friends for their support over the years. Among them, I want to particularly thank Su Chen, Ling Chen, Shaojie Deng, and Kevin Sun, who are amazing statisticians, for their suggestions and comments on my work. Also, I would like to thank Sukwon Chung for proofreading my dissertation.

Finally, I would like to thank my parents for their love and support over the years!

Contents

Abstract **iv**

Acknowledgements **v**

1 Introduction **1**

1.1 Background 2

1.2 Bayesian Filtering 4

1.3 Robotic Navigation 5

1.3.1 POMDP Framework 6

1.3.2 SLAM Framework 7

1.4 Dissertation Organization 7

2 Sequential Control in Partially Observable Domain **9**

2.1 Introduction 9

2.2 POMDP Framework 11

2.2.1 Markov Decision Process 11

2.2.2 The POMDP Model 13

2.2.3 Value Functions 15

2.2.4 Value Function α-Representation 16

2.3 Curse of Dimensionality 17

2.3.1 Approximate Value Iteration Methods 17

2.3.2 Heuristic Search Methods 17

2.3.3 Belief Space Compression Methods 18

2.4 Online Suboptimal Control 18

2.4.1 Receding Horizon Control 19
2.4.2 Belief-Based Myopic Control 21
2.5 Comparative Studies 22
2.5.1 Results and Discussion 22

3 Particle-Based SLAM Filters for Nonlinear Filtering 24
3.1 Introduction 24
3.2 Nonlinear Filtering via Non-Particle Filters and Sequential Monte Carlo 27
3.2.1 Non-Particle Bayesian Filtering Methods 29
3.2.2 Sequential Monte Carlo Methods 31
3.2.3 A Comparative of Different Nonlinear Filters 36
3.3 Simultaneous Localization and Mapping 37
3.3.1 Definition of SLAM 37
3.3.2 FastSLAM 40
3.4 Spherical Simplex Unscented Particle Filters 42
3.4.1 The Spherical Simplex Unscented Transformation 43
3.4.2 Spherical Simplex Unscented Kalman Filters 46
3.4.3 SSUKF-Based Proposal Distribution 48
3.5 Spherical Simplex Unscented Particle-Based SLAM Filters 50
3.5.1 System Dynamics 51
3.5.2 State Estimation 52
3.5.3 Feature Estimation 55
3.6 Experimental Results 57
3.6.1 Experiment 1: Simulated Environment 57
3.6.2 Experiment 2: Car Park Dataset 59

4 Integrated Robotic Navigation under Uncertainty 66
4.1 Introduction 66
4.2 Global Planning 67
4.2.1 Classical Path Planning 68
4.2.2 Path Planning under Uncertainty 70
4.3 Local Navigation 70

4.3.1 Obstacle Avoidance 71
4.4 Robotic Exploration 75
4.4.1 Coastal Navigation 75
4.4.2 Frontier-Based Exploration 76
4.4.3 Information-Based Exploration 76
4.5 Integrated Navigation under Uncertainty in an Unknown Environment 77
4.5.1 Limitations of Current Approaches 77
4.5.2 New Framework 79
4.5.3 Problem Statement 80
4.5.4 System Control 81
4.5.5 Global Navigation 83
4.5.6 Obstacle Avoidance 84
4.5.7 Point-Based Exploration 88
4.5.8 The Integrated Navigation Algorithm 92
4.6 Experimental Results 94
4.6.1 Experimental Setup 94
4.6.2 Result and Analysis 94

5 Conclusion and Future Work **100**

Bibliography **102**

List of Tables

2.1 Experimental results for the two Hallway domains. The BBMC achieves similar level of performance as other complicated strategies, but are significantly faster. 23

3.1 The mean square errors of various particle filters using different proposal distributions. The result of the commonly used extended Kalman filter (EKF), a non-particle approach, is quoted as a reference here. . 37

3.2 The performance comparison of different particle-based SLAM filters. The criteria used are the mean and standard deviation of the mean square error (MSE) with the unit being *meters*. 59

4.1 Results of Integrated Navigation. 95

List of Figures

2.1 Convex majorant of linear reward functions of different actions. . . . 16

3.1 The diagram shows the comparison between the states estimated by various algorithms. 38
3.2 The diagram shows the performance comparison among various algorithms. The number of simulated particles is chosen to be 20, 40, 80, 200, and 1000 for each set respectively. 39
3.3 The simulated environment where the SLAM algorithms are tested for experiment 1. 58
3.4 The MSE for the robot pose estimation with noise of various level for experiment 1. 60
3.5 The MSE for the map feature estimationwith noise of various level for experiment 1. 61
3.6 The Car Park dataset. The FastSLAM 2.0 is used as the SLAM filter to carry out the estimation for the robot pose and landmark locations, which are compared with the actual locations. 63
3.7 The Car Park dataset. The U-PBSLAM is used as the SLAM filter to carry out the estimation for the robot pose and landmark locations, which are compared with the actual locations. 64
3.8 The Car Park dataset. The SSU-PBSLAM is used as the SLAM filter to carry out the estimation for the robot pose and landmark locations, which are compared with the actual locations. 65

4.1 The diagram shows the relation among the three subfields: global planning, local navigation, and exploration. The goal is to solve problems within the area covered by all three subfields as indicated by VII. . . 80
4.2 An illustration of the control block to realize global path planning under uncertainty constraints while ensuring robust local obstacle avoidance navigation. 82
4.3 The obstacle avoidance diagram illustrates situations the robot will encounter during local navigation and the corresponding actions to take during the course. 85
4.4 (a) High Safety (HS) situation. The robot maximizes its speed and keep its angle towards the global goal. (b) Low Safety Far Angle (LSFA) Situation. The robot slows down but keeps its original orientation. (c) Low Safety Close Angle (LSCA) Situation. The robot slows down and adjust its direction to at least bypass the landmark at the critical angle. (d) Dangerous Region (DR) Situation. The robot slows down to its minimal speed and turn as sharply as it can to steer away from the landmark. 86
4.5 In all plots, the green stars indicate the landmarks. The red circle around the landmarks indicate those landmarks that are close in range. The red dash straight line from the robot front end points to the steering direction. The pink dots around the robots are the spatial testing points, whose density can be adjusted. (a) Robot explores and tries to circumvent the landmarks. (b) Robot further moves and passes along the landmarks. (c) Robot meets new cluster of landmarks, but see a gap that can go towards the goal. (d) Robot turns and goes through the gap. 96
4.6 The final path of the robot. The landmarks located in the middle part of the map are so dense that there is no chance to find a safe path across them. The robot finds a safe path and goes through the big gap shown in the lower half of the plot. 97

4.7 The final path of the robot. Because the opening of the landmarks in the middle is not big enough, the robot considers going through the middle gap to be unsafe due to its own physical dimensions, so it turns and finds another safer path, i.e., going through the bigger gap located in the lower half of the plot. 98
4.8 The final path of the robot. As the gap in the middle part widens, the robot sees an oppotunity and find a closer safe path to reach the goal location, instead of going the detour. 99

Chapter 1

Introduction

With technologies advancing rapidly over the past decades, building intelligent robots that can accomplish tasks without human aid has fascinated many engineering researchers, especially those in the Artificial Intelligence (AI) community. Among various applications of robots, ranging from gigantic industrial robots that build cars and equipments to human-sized nursing robots that provide personal care at home, one area that has drawn particular interest and large amount of resources is *autonomous robot navigation*, due to its important and broad applications ranging from unmanned detection in military operations and planetary exploration in space missions, to automatic automobile driving in urban transportation. The Defense Advanced Research Projects Agency Grand Challenge [119], which requires technologies that can create fully *autonomous ground vehicles (AGV)* capable of unmanned driving to complete a predefined course, together with the recent media coverage of Google successfully testing unmanned vehicles driving on a freeway, are examples of the research effort that has been dedicated to push this area forward.

One of the key techniques in autonomous robot navigation is the Bayesian estimation, which has wide applications in various research fields. The main usage of Bayesian filters is to estimate the state of the system, such as the location and orientation of the robot and also the information of the surroundings, so that the robot can make control decision on the next move. With the increasingly stringent requirements from practical applications, bayesian filters for nonlinear applications

are of high demand. Sequential Monte Carlo methods, also known as particle filters, are a class of sophisticated statistical techniques for nonlinear/non-Gaussian model estimation, gaining tremendous attention and popularity in the past decade due to its superior performance over other methods such as the widely used extended Kalman filter.

This dissertation concerns the design of good particle-based Bayesian filters that can be applied to the robot navigation, and building a new framework that incorporates the filters for solving robotic navigation problems under uncertainty in a continuous environment.

1.1 Background

Autonomous robot navigation has a wide range of application in the real world. In fact, with the computer technology ever advancing, one of the dreams humans have is to build autonomous robots that can replace human beings to accomplish a lot of tasks that are either mission impossible for humans or can bring convenience and efficiency. Autonomous navigation is one of the goals that autonomous robots can accomplish, and is the main topic in this dissertation.

It will be intriguing to discuss a little about the applications of autonomous robot navigation. On the ground, the most talked about civil application is an autonomous vehicle that can drive automatically from the starting location to the destination. Imagine in the morning, you get into your car, say the name of your workplace whose address has been stored in the memory, and just relax and read newspapers while the vehicle automatically drive you the workplace safely. In the air, the autonomous airplanes fly flawlessly along the predetermined path and send passengers to their destinations. Also, the unmanned airplanes can also be used for scientific research when information about the atmosphere at heights unreachable by humans. For planetary exploration, the rovers, which include those to be sent to planets in the solar system and those already sent to Mars, need the ability to navigate autonomously as they may lose contact with earth for a certain amount of time as the interplanetary communication at such vast distance is itself a big challenge. There are many

other potential applications, which make the research of the autonomous navigation promising.

The robotic navigation problem in the real world has a lot of issues to overcome. We can roughly group those issues into a few categories. The first type of issues are caused by unpredictability, also called uncertainty. For example, after the robot executes the control commands, which normally consist of velocity control and steering angle control, the actual velocity may deviate from the desired values, due to causes such as bumpy road surface, and the steering angle may be different too, which might be caused by, say, the mechanical system errors. Besides control errors, the sensors the robot uses to detect the distance to the surrounding objects are imperfect, due to causes such as its intrinsic system errors and the manufacturing defection, resulting the erroneous detection data that will potentially mislead the robot in navigation decision-making. All there errors are often modeled as noise components, and are unpredictable in advance to the process.

The second type of issues arise from the unobservability. This happens because the robot doesn't know exact its own location. In other words, the robot has no idea of its own state, including its location and orientation, except for some corrupted measurement data taken with respect to the landmarks. The measurements only provide the relative distance and angle of the landmarks with respect to the robot. As the robot doesn't know the location of the landmarks, the measurement data doesn't provide direct information about the state of the robot. In other words, the robot cannot observe directly its own location and orientation. The fact that the indirect information available are also corrupted by noise makes the task of determining its own state even more challenging.

The third type of issues are related to dimensionality. The existing methods in the literature often solve problems with small scale so that the environment can be discretized. This limits the application of the algorithms to real-world problems which are often of large scale and continuous. As the dimension of the problem increases, the computational complexity often increases dramatically. Therefore, coping with the dimensionality is one of the key challenges in the autonomous robot navigation.

1.2 Bayesian Filtering

With all the challenging issues to be solved, a robust mathematical framework that can model the dynamics of the system and capture the uncertainty is needed. Naturally, Bayesian framework became the popular choice due to its simple mathematical structure and tremendous modeling power. The foundation of Bayesian framework is the Bayes' theorem, which has important application in statistical inference. The system process is often modeled as a *state-space model (SSM)*, which means the desired variables are often denoted as the *state* and its evolution is set to be connected in a *Markov chain* with some control inputs exerted on the states. The state at given a time is not directly observable, but its twisted version, often corrupted by noise, is observable, which is called a *measurement*. There are two probabilities defined for the model, a *transition probability*, which describes the statistical evolution of the Markov chain, and a *observation probability*.

The main idea of SSM is to model a set of desired parameters to be a state vector, and evaluate them using certain mathematical tools based on the statistical properties of the system. The probabilistic nature of the system comes from various sources of noise, including the system noise, which is usually inevitable and generated during the state evolving process, and measurement noise, which is often caused by the inefficiency or limitation of the measurement acquiring devices and the random noise from the environment. Therefore, in most of the cases, two models are needed to facilitate the state estimation: First, a process model, or system model, which gives the state for the next time step given the current state and control, and a measurement model, which gives the measurement based on the current state. These two equations forms a rigorous framework for general dynamic state estimation for time-varying system. Then, the estimation of the state is carried out by mainly two interweaved steps, a *prediction* step, where the state is predicted given all the past observations up to the last time step, and an *update step*, which updates the predicted state by incorporating the latest measurement at current time step. These two steps are carried out recursively over time, giving a *sequential* estimation of the state variable.

For special cases, where both the process and measurement models are linear and Gaussian, an optimal solution to the estimation problem is given by the *Kalman filter*. For nonlinear cases, a linearization technique based on Taylor expansion is employed, which results in the *extended Kalman filter*. It linearizes the process and measurement equations using Jacobian matrices, and then apply the Kalman filter. There are also other techniques to cope with the nonlinearity. For example, the *Gaussian sum filters* use a bunch of Gaussians to approximate the posterior distribution of the state; the *unscented filters* choose deterministically a set of so-called *sigma points* based on the statistical properties of the state, and map those sigma points through the nonlinear equations. The transformed sigma points can be used to calculate the statistical parameters of the system, which originally have to be transformed directly through the nonlinear equations. In other words, the unscented filter picks a set of points that are easier to apply the nonlinear equations than the probability density of the state.

Besides those filters just introduced, *sequential Monte Carlo methods*, also called *particle filters*, approach the Bayesian estimation problem in a different way. They use a set of particles to approximately represent the probability distribution, thus obtained the name. The advantage of particle filters over other non-particle methods is that they can solve nonlinear and non-Gaussian problems, due to the fact that they are sampling-based filters. On the other hand, the sampling-based nature imposes more computational burden, as the number of particles increases. As the iterative sampling process carries on, it often happens that most of the particles will have negligible weights, which is called *particle degeneracy*. This can be alleviated by *resampling*, a step that reselects and shuffles the particles according to certain probability distribution determined by the weights.

1.3 Robotic Navigation

Autonomous robot navigation is one of the most promising fields in the research of artificial intelligence (AI). With the rapidly rising demands from the applications, ranging from unmanned urban vehicles and unmanned aircrafts to space rovers landing on Mars, the requirements for the technology for autonomous robot navigation

are becoming more and more challenging. For example, the autonomous vehicle on the ground needs to be capable of driving to the destination while avoiding the traffic and collision with other cars or obstacles. The rover sent to Mars sometimes needs to navigate though tough terrains with little assistance as it takes a long time for the communication signals to transmit between Mars and Earth.

From the technical point of view, autonomous robot navigation mainly concerns the design of robots with abilities in three key aspects: the ability to generate optimal global paths to maneuver itself to a target location in a real-time environment, the ability to reactively correct its local course by circumventing obstacles to avoid collision, and the ability to explore unknown territories. Extensive research effort has been devoted to all three aspects, which were initially treated separately and whose boundaries became blurred later due to the development of novel techniques that can handle more complex problems in real-world applications. In addition, the robot should also be able to cope with uncertainty while still accomplishing desired tasks. Two frameworks are developed to solve navigation problems under uncertainty, as shown below.

1.3.1 POMDP Framework

In recent years, there has been increasing interest in modeling the planning problem under uncertainty by using the framework of *Partially Observable Markov Decision Process (POMDP)*, a generalization of a *Markov Decision Process (MDP)*. It originated in the operations research community, and later gained recognition and popularity in the other communities such as Artificial Intelligence and Automated Planning. The main idea of the framework is to represent the process using a hidden Markov Model and interpret the tasks in terms of certain reward function, thus the planning problem becomes how to find the optimal policy during the execution process to maximize or minimize the reward by employing techniques from dynamic programming and reinforcement learning including value iteration, policy iteration, Q-learning and so on. So far, a vast amount of research has been dedicated to this field, and significant progress has been made. The limitations of POMDP include the

system often being discrete and the dimension not being too high.

1.3.2 SLAM Framework

The *simultaneous localization and mapping (SLAM)* is a challenging research field in robotic navigation. It concerns the design of algorithms that enable the robot to build a map for the surroundings while localizing itself within the same environment. The navigation process is often continuous, during which the robot does not know exactly its own state, including its location and orientation, or the location of the landmarks. The only information the robot has are the control signals, including the velocity and steering angle, and the measurements, which are the relative locations of the landmarks with respect to the robot and are often corrupted by noise. Therefore, how to use the limited information to estimate the robot pose and landmark locations forms the SLAM problem, the solution to which has been a cornerstone in the roadmap of building autonomous robots.

1.4 Dissertation Organization

In this work, we designed new particle-based SLAM filters for robotic navigation problems. Then, we built an integrated framework based on the newly designed SLAM filters for robotic navigation under uncertainty in a continuous environment.

In Chapter 2, we first introduce the POMDP framework, which is widely used in robotic navigation problem. We then describe our online control strategy that was designed to achieve same level of performance but with much less computational complexity, compared to other complicated approaches currently existing in the literature. This is verified by the simulation study carried out at the end of the chapter.

In Chapter 3, we begin by introducing the Bayesian framework for state estimation and some of the widely used non-particle algorithms, such as Kalman filters. Then we introduce the sequential Monte Carlo methods (a.k.a. particle filters), which are followed by the new approach designed to improve the performance of particle filters and verified by simulation studies. The SLAM framework is introduced, followed by

the particle-based SLAM filter, the FastSLAM filters. Then, we extend the improved particle filters and apply to the SLAM problem in order to build the particle-based SLAM filters. The design and implementation will be discussed in details. The new algorithms are tested on both the simulation data and the outdoor experimental dataset.

In Chapter 4, we first introduce the literature of robotic navigation, including the work on global planning, local navigation, and exploration. Then, we discuss the limitations of the current approaches, and propose a new framework for robotic navigation under uncertainty. In particular, it considers the trade-offs between exploration and exploitation, and provides a new approach to integrate global planning, location navigation, and exploration. The particle-based SLAM filters proposed in Chapter 3 are used to cope with uncertainty.

In Chapter 5, we summarize the work in this dissertation, and sketch some ideas of future work, including our continuing effort to design better SLAM filters and build hybrid systems that can approach robotic navigation from a more comprehensive perspective.

Chapter 2

Sequential Control in Partially Observable Domain

2.1 Introduction

Sequential planning has been a popular topic for a wide range of applications such as artificial intelligence, where a robot is required to accomplish assigned task partially with or completely without human assistance. A good example will be robotic navigation where a robot seeks to reach a certain goal location by planning its traveling path sequentially. In most of the real world applications, the environment that the robot interacts with is imperfect, mixed by all kinds of uncertainties arising from both systematic errors and random errors. The systematic errors include an incomplete map, a misrepresented location of a landmark, or a functional deficiency in the robot's range detector, while the random errors are usually caused by noise, which subject to certain probability distribution we can or cannot track. The goal is to overcome these uncertainties by taking advantage of their recognizable patterns, including their probabilistic characteristics, and accomplish the task by making sequential plans. In other words, the question becomes how to do the control given uncertainties.

The classic planning is generally carried out by agents in fully observable environments with conditions usually assumed to be deterministic and discrete. In order to take into account the uncertainty, the traditional planning framework no

longer applies. The *Partially Observable Markov Decision Process (POMDP)* has since emerged as a powerful mathematical framework that accounts for the uncertainty from various sources during the planning process. Since POMDP framework tries to solve the planning problem by providing a general action policy, it requires large computational effort thus becomes more difficult to carry out as the dimension of the problem scales up.

Over the years, a lot of research effort has been devoted to 1) finding approximate solutions in order to increase the dimension of the problems solvable by POMDP, and 2) seeking efficient online planning algorithms instead of offline ones [97]. As the POMDP literature growing rapidly, various approaches have been proposed to solve the POMDP problem with increasing dimension and complexity. Meanwhile, many benchmark problems have also been designed to compare the performance of different methods. In most of these problems, the task is to reach a fixed location while maximizing rewards. In terms of their discrete and continuous nature in the state space and action space, they can be categorized as follows:

Discrete Navigation Environments This means that the state space of the problem is discrete and usually discretization will be required before applying the POMDP methods. The configuration domains are usually of grid type with tens to hundreds of states in total. The action space can be discrete, with a few action choices for the robot to move between adjacent grids, such as *stay, move forward, turn back, turn left*, and *turn right* [16,54,65,82,86,94,95,107,108,111,130], or continuous, as in the bicycle simulator with the actions represented by continuous orientation angles [78,92].

Continuous Navigation Environments This means the state space is continuous. The action and observation spaces can be discrete or continuous [18,85,96].

In this chapter, we will first introduce the concept of *Markov Decision Process*, and extend its definition to *Partially Observable Markov Decision Process*. After talking about the properties of POMDP, we will give a quick overview of existing approaches to solve the POMDP problem. Then, the new suboptimal approach will be introduced and its performance will be compared with existing approaches. Though the experimental examples we choose are not of large dimensions, the purpose is to

show that the suboptimal solution performs similarly to other complicated strategies, while incurring significantly less computational time, thus is more suitable for online applications. This is to lay the foundation for the more interesting problem we will investigate in Chapter 4, where we consider a challenging real-world robotic navigation problem in a continuous domain.

2.2 POMDP Framework

A Markov decision process (MDP) is used to model the interaction between an agent and a stochastic environment [112]. It assumes that the agent has full access to knowing the exact state at each time step. In many applications, the environment is only partially observable to the agent, which stems from causes such as imperfect sensor readings, thus the agent only has a noised reading of its state. In other words, the underlying dynamics follows a MDP process, but the agent cannot directly observe the underlying state. For these cases, the Partially Observable Markov Decision Process models uncertainty resulting from the imperfect interaction between the agent and the environment [112]. In stead of obtaining the true states of the system as in MDP, a POMDP model computes and maintain the so-called *belief* states, which is the probability distribution over the set of state and implies how confident the agent is in determining its true state.

In this section, we will first introduce the Markov Decision Process and then extend it to the POMDP framework.

2.2.1 Markov Decision Process

The Basic Definition

A Markov Decision Process is a 4-tuple system: (S,A,T,R). They are

• *States* S: The state of the system is denoted as s, and the finite set of all possible values are defined as the state set S.

• *Actions* A: Each action of the agent takes to progress in the environment is denoted as a. The finite set of all available actions is defined as the action set A.

• *Transition Probability* $T(s^{'}, a, s) : S \times A \times S \rightarrow [0, 1]$. Given an action $a_t = a$ and the current state $s_t = s$, the probability of transitioning into state $s^{'}$ at time $t+1$ is defined by the conditional probability

$$T(s^{'}, a, s) = Pr(s_{t+1} = s^{'} | s_t = s, a_t = a) \tag{2.1}$$

As it's a probability, it suggests that $\sum_{s' \in S} T(s', a, s) = 1, \forall (s, a)$.

• *Reward Function* $R(s, a) : S \times A \rightarrow \Re$. It assigns a real value in order to quantify the utility or benefit of performing action a at state s. The goal of the planning is to maximize the total rewards over the entire course of the actions.

Value Iteration

Due to stochastic nature of the problem, we can only calculate the expected rewards of the future performance. Mathematically, this can be defined as the summation of all rewards in the future discounted at a given rate γ, $0 \leq \gamma < 1$:

$$E[\sum_{t=0}^{T} \gamma^t R_t] \tag{2.2}$$

At each state, we define *policy* π as the function mapping from the state s to the action a, i.e., $S \rightarrow A$. The goal of the MDP problem is to find the optimal policy π^*, which maximizes the total rewards. One way to achieve this goal is to use *value function* V^{π}, which is defined as the total discounted rewards starting from current sate s and given action a according to policy π. Since the action is also a function of the current state, the value function is a mapping from states to rewards: $V^{\pi} : S \rightarrow \Re$. Using the definition of expected rewards from equation 2.2, the value function can be expressed as

$$V^{\pi}(s) = R(s, \pi(s)) + E[\sum_{t=1}^{\infty} \gamma^t R(s_t, \pi(s_t))] \tag{2.3}$$

If we define the optimal value function to be $V^*(s) = V^{\pi^*}(s)$ and its associated

optimal policy as π^*, the optimal value function satisfies the *Bellman equation*:

$$V^*(s) = \max_{a \in A}[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)V^*(s')] \tag{2.4}$$

which is due to the *Principle of Optimality* [9].

In order to efficiently calculate the optimal value function and its associated optimal policies, the *value iteration* is one of the most popular algorithms adapted from dynamic programming:

$$V_n(s) = \max_{a \in A}[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)V_{n-1}(s')] \tag{2.5}$$

This iteration rule is repeated for all states s until the right-hand side value function converges to be the same as the left-hand side, which is then the optimal value function [12].

2.2.2 The POMDP Model

The POMDP model is a generalization of the MDP model described in section 2.2.1. On one hand, they are similar, in the sense that they both deal with the system with dynamics evolving according to a Markov process. On the other hand, they differ in the perception of the state: in MDP, the state of the system is fully observable, i.e., its value is completely known; whereas for POMDP, only an observation or measurement of the state usually corrected by noise is available, so the distribution of the possible state values, sometimes called the *belief* [118], is maintained instead of the state. This is the key difference between MDP and POMDP, and is the main reason why a POMDP problem is much more complex and difficult to solve than a MDP problem.

The Basic Definition

A partially observable Markov decision process is a 6-tuple system (S,A,T,R,Z,O), where:

- *States* S: The state of the system is denoted as s, and the finite set of all

possible values is defined as the state set S.

• *Actions* A: Each action of the agent takes to progress in the environment is denoted as a. The finite set of all available actions is defined as the action set A.

• *Transition Probability* $T(s', a, s) : S \times A \times S \to [0, 1]$. Given an action $a_t = a$ and the current state $s_t = s$, the probability of transitioning into state $s^{'}$ at time $t+1$ is defined by the conditional probability

$$T(s^{'}, a, s) = Pr(s_{t+1} = s^{'} | s_t = s, a_t = a) \tag{2.6}$$

As it's a probability, it suggests that $\sum_{s' \in S} T(s', a, s) = 1, \forall(s, a)$.

• *Reward Function* $R(s, a) : S \times A \to \Re$. It assigns a real value in order to quantify the utility or benefit of performing action a at state s. The goal of the planning is to maximize the total rewards over the entire course of the actions.

• *Observations* Z: The observation of the agent state is denoted as z. The finite set of all possible observations is defined as the observation set Z.

• *Observation function* $O(s, a, z) : S \times A \times Z \to [0, 1]$. Given an action $a_t = a$ and the current state $s_t = s$, the probability of observation z for the state s at time $t + 1$ is defined by the following conditional probability

$$O(s, a, z) = Pr(z_{t+1} = z | s_t = s, a_t = a) \tag{2.7}$$

Note that $\sum_{z \in Z} O(s, a, z) = 1, \forall(s, a)$.

Objective Functions

Given a POMDP model, the goal is to find the optimal *control policy* that maximizes the total rewards over a certain time horizon. The objective function can be defined according to the following two cases [45]:

• Finite-Horizon Model:

$$\max E \left(\sum_{t=0}^{T} r_t \right) \tag{2.8}$$

- Infinite-Horizon Model with a discount factor $\gamma, 0 < \gamma < 1$:

$$\max E\left(\sum_{t=0}^{T} \gamma^t r_t\right) \tag{2.9}$$

2.2.3 Value Functions

For POMDP model, the *belief state*, which is a *sufficient statistic* to preserve the necessary information [45], is used in order to capture the statistical information of the system while keeping the operational simplicity. Similar to MDP, we define the value function for POMDP in term of belief state as

$$V^{\pi}(b) = E_{\pi}\left[\sum_{t=0}^{T} \gamma^t r(b_t, \pi(b_t)) \mid b_0 = b\right] \tag{2.10}$$

where r is a one-step reward given belief state b and action $a_t = \pi(b_t)$.

Bellman's Equation

For the case of a discrete state space, the optimal value function when $T = \infty$ can be expressed in terms of the Bellman optimality equation $V^* = HV^*$ in which H is the Bellman backup operator [45,112]:

$$V^*(b) = \max_{a \in A}\{\sum_{s \in S} r(s,a)b(s) + \gamma \sum_{z \in Z} \sum_{s \in S} P(o|s,a)b(s)V^*(b')\}, \tag{2.11}$$

where b' is the updated belief given by

$$b'(s) = \frac{P(z|s,a)}{P(z|b,a)} \sum_{s' \in S} P(s|a,s')b(s'). \tag{2.12}$$

Therefore, the optimal policy π^* can be expressed as

$$\pi^*(b) = arg\max_{a \in A}\{\sum_{s \in S} r(s,a)b(s) + \gamma \sum_{z \in Z} \sum_{s \in S} P(z|s,a)b(s)V^*(b')\}. \tag{2.13}$$

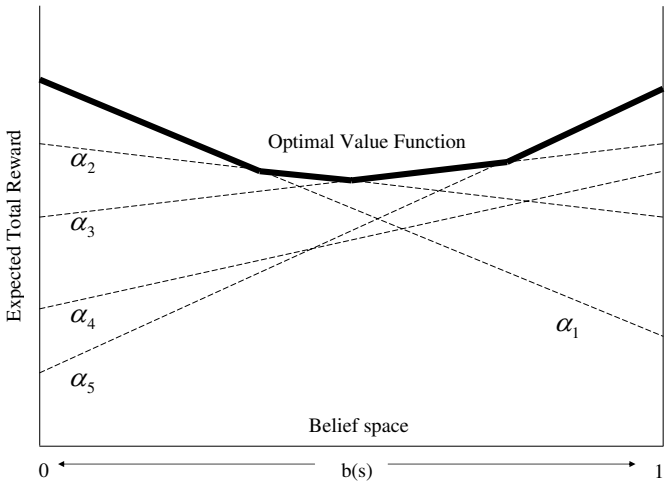


Figure 2.1: Convex majorant of linear reward functions of different actions.

2.2.4 Value Function α-Representation

For the finite-horizon case $T < \infty$, the value function can be represented by a set of multi-dimensional hyperplanes formed by *α-vectors*, each of which is associated with a specific action. For any given belief point, the largest value of the α-vectors at that point is the optimal value and the associated action is the optimal action. Therefore, the optimal value function is piecewise linear and convex [109], as shown by the thick line in Figure 2.1. Computational algorithms use this property of the value function together with the belief points and the α-vectors to make the value iteration process efficient. The *enumeration algorithm* [73] is the most direct way of computing the value function by calculating all the possibilities. The *one-pass algorithm* [109] selects arbitrary belief points, constructs the associated α-vector and finds its dominating regions; the process is carried out iteratively until the entire belief space is covered. The *witness algorithm* starts with a *witness point* and a suboptimal α-vector, and keeps generating new vectors until the vector is optimal over that witness point.

2.3 Curse of Dimensionality

The preceding approach becomes increasingly difficult with increasing dimension of the α-vectors. When the state space is continuous and high-dimensional, discretization becomes prohibitively difficult and approximations are needed [54,82]. Various approximations have been used to tackle the so-called *curse of dimensionality*, and have led to various methods that can be grouped into the following classes:

2.3.1 Approximate Value Iteration Methods

These methods approximate the value function by using only a selected set of belief points, instead of all possible ones, to obtain approximate solutions of the dynamic programming problem [45]. The *point-based value iteration* method [82] is a representative method of this kind. At each step, it uses exploratory stochastic trajectories to sample belief points, with at most one new belief point for each previous belief in order to control the size, and then updates both the value and gradient at those points during each value update. The randomized point-based value iteration method called *Perseus* [110,111] does the value update at each step by randomly selecting a small set of points that is sufficient to improve the values of all belief points. There are also other approximating methods such as the *MDP approximation*, which considers the most likely state as the true state and determines the action of the MDP, *grid-based approximation* which approximates the continuous belief space by a finite set of grid points and uses an interpolation-extrapolation scheme to estimate the value at any given belief point, and Monte Carlo POMDP [117] which approximates the value function by samples generated by Monte Carlo simulations.

2.3.2 Heuristic Search Methods

These methods solve the POMDP problem by heuristic search. The *heuristic search value iteration* method [107,108] stores both the upper and lower bounds on the optimal value function and chooses the set of belief points for local update by tree search based on heuristics. The policy-iteration method introduced in [44] represents

a policy as a finite-state controller, and uses the belief tree to iteratively search the belief space to improve the controller.

2.3.3 Belief Space Compression Methods

Belief compression is a class of techniques that project the high-dimensional belief space to a low-dimensional subspace. The object is to find a good approximation to the optimal value function by restricting to the subspace, instead of computing the value function on the original belief space. This simplifies the calculation at a cost of information loss due to compression. In [87], the value-directed compression technique was introduced. Characterizing lossless compression and linear compression with a *dynamic Bayesian network* and a *Krylov subspace*, they derived an upper bound for lossy compression, and designed an optimization routine that minimizes the error bound for linear lossy compression and an associated structured POMDP model for efficient implementation. Another approach based on *exponential family principal components analysis (E-PCA)* was introduced in [98]. A key component of an E-PCA model representing the reconstructed data, besides a low-dimensional weight vector and a basis matrix, is the *link function* associated with the exponential family. The parameters of an E-PCA model are obtained by minimizing a loss function called generalized *Bregman* divergence between the low-dimensional and high-dimensional representations, which, by choosing an exponential link function, is equivalent to minimizing the unnormalized *Kullback-Leibler* divergence. By planning in the projected low-dimensional space, approximation to the optimal policy can be found even when the POMDP models have significantly large dimension.

2.4 Online Suboptimal Control

The existing approaches are limited by so-called *curse of dimensionality* [54] and *curse of history* [82]. As the dimension of the problem keeps increasing, the optimal solution to the POMDP problem becomes computationally expensive and practically

intractable. So, on the contrary to the existing approaches, we try to seek an alternative way of solving the POMDP problem by providing a fast and suboptimal solution. While for small-dimension problems, the performance of the suboptimal solution underperforms the optimal solution, its advantage will show up as the dimension of the problem becomes greater due to its low computational complexity. For any real-world problem, where the system is continuous instead of discrete, the existing approaches will become useless while the suboptimal solution can still solve the problem and give decent performance. In this section, we will design our online suboptimal control strategy based on the POMDP framework and the receding horizon control.

2.4.1 Receding Horizon Control

Receding horizon control (RHC) is a local planning strategy, in the sense that it looks into the near future while ignoring the long-term global effects. A lot of research findings, e.g. Leung et al. (2006a, 2006b), have shown that RHC works well in path planning involving local obstacle avoidance given physical constraints of the robot and provides informative navigation control decisions based on sensor observation. The basic idea behind RHC is to obtain an optimal control sequence up to the horizon T by minimizing an objective function while obeying certain constraints, and then to execute the first command that will lead the system one step ahead to a new state. In other words, the optimization process is carried out within a window size T that keeps shifting towards the future, with only the first of the resulting control sequence executed at each time. The planning process is recursively computed at every time step, so any new information including environmental change can be incorporated as feedback into the future planning.

Mathematical Definition

Mathematically, the control problem amounts to finding a series of actions at times $(t_0, t_0+1, ..., t_0+T-1)$ such that

$$minimize: \quad \sum_{t_0}^{t_0+T-1} r(x_t, a_t) \tag{2.14}$$

$$subject\ to: \quad a_t \in A, \tag{2.15}$$

$$x_{t+1} = g(x_t, a_t), \tag{2.16}$$

$$x_{t_0} = x_0 \tag{2.17}$$

where $r(\cdot)$ is the objective function at each time step and $g(\cdot)$ is the transition function that governs the state evolution given the past state and action. Although a series of actions are obtained by solving this optimization problem, only the first control action a_{t_0} will be executed at time step t_0. This process is repeated for each time step in the future.

The most important advantage of RHC is that it is like a feedback control such that the system is capable of adapting to new information and uncertainty and process online as the system evolves. On the other hand, the performance of the RHC controller may suffer from a short-sighted outlook, which results in its inability to plan beyond the given time horizon and may lose the potential future benefits. Therefore, a method incorporating both the global knowledge and the advantage of RHC in local planning is desirable in improving the performance of robot navigation.

Related Work

Refinements of RHC [20] have been developed to use a shrinking window, with one side fixed at the future ending time point and another side moving one step a time resulting in a decreasing window size, instead of a moving window of constant time horizon as in RHC. A Lyapunov function-based nonlinear RHC has been combined with navigation functions and randomized optimization algorithms to solve the robot navigation problem with state and input constraints [83,116]. *Mixed-integer linear*

programming has been embedded in the RHC framework to solve the navigation problem in an *a priori* unknown environment [29].

2.4.2 Belief-Based Myopic Control

For real-world applications, online algorithms generally execute a lookahead search in order to find the optimal action to take at each time step [97]. When the number of step is one, it is called the *myopic policy*. As a special case of RHC, it emphasizes on the short-term rewards. As the number of steps is only one, it is computationally more efficient and faster than multi-step RHC, which generally involves constructing a tree-type search process thus takes more computational resources [97].

Besides the benefit of computationally efficient, myopic policy also is proved to be a good and suboptimal policy as base policy [5], and the optimal policy is equal to the myopic rule added by some disturbance [43]. For fully observable cases, i.e., the Markov decision problems, the myopic policy, or the 1-step RHC policy, can be expressed mathematically as:

$$\pi^{myopic}(s_t^*) = \arg\max_{a_t \in A} r(s_t, a_t) \tag{2.18}$$

When the states are only partially observable, the myopic policy above should be modified by incorporate the belief state. At each time step, the belief state represents the probability distribution of the actual state over the possible locations, we can obtain the *most likely state (MLS)*. Mathematically, this can be expressed as

$$s_t^* = \arg\max_{s_t \in S} b_t(s_t) \tag{2.19}$$

where s_t^* is the most likely state at time t, s_t is the true state, $b_t(s_t)$ is the belief state at time t as a function of the true state. If there are multiple states that achieves the same maximum probability, we are randomly pick one with equal probability. In other words, assuming there are n states, $s_t^1, ..., s_t^n$, with the same maximum probability,

we do the following:

$$u \sim \mathbb{U}[0,1] \tag{2.20}$$

$$s_t^* = s_t^i, \quad \text{where} \quad (i-1)/n < u \leq i/n \tag{2.21}$$

Then, applying the myopic strategy based on the MLS, we have

$$\pi^{BBMC}(s_t^*) = arg\max_{a_t \in A} r(s_t^*, a_t) \tag{2.22}$$

where because of the nature of this method, we call it *belief-based myopic control*. We will demonstrate the performance of this control policy in the next section. The application of this control rule will be extended to continuous case in Chapter 4.

2.5 Comparative Studies

In this experiment, we will test our control strategy using two domains introduced in [65], the Hallway and Hallway2. The details of the experiments set up can be found in [65]. We choose these two domains because the number of states for each domain is large enough to demonstrate our point, but not so large that the implementation will be too complex. We will show that our newly proposed *Belief-Based Myopic Control (BBMC)* can reach suboptimal performance, comparable to those complicated scheme, but with much faster speed of several order of magnitude and simpler implementation complexity. This provides possibility for on-line planning in robotic navigation, particularly the problem we will address in Chapter 4.

2.5.1 Results and Discussion

The simulation platform uses a Mac OS X Version 10.6.5 operating system with 2.66 GHz Intel Core 2 Duo CPU and 4GB 1067MHz DDR3 RAM. The discounting factor is $\gamma = 0.95$. The reward is computed by averaging over 1000 simulation runs. Each simulation will terminate after 251 steps, and the success rate is computed by dividing the total number of successful simulations with the total number of simulations.

Problem (state/action/observations)	Goal%	Reward	Time(s)
Hallway(57s 5a 20o)			
QMDP [Littman et al., 1995]	47.4	0.14	0.012
PBUA[Poon, 2001]	100	0.53	450
PBVI[Pineau et al. 2003]	96	0.53	288
BPI [Poupart et al., 2003]	n.a.	0.51	185
Perseus [Spaan et al., 2004]	n.a.	0.51	35
HSVI1 [Smith et al., 2004]	100	0.52	10836
HSVI2 [Smith et al., 2005]	n.a.	0.52	2.4
BBMC	100	0.41	0.0027
Hallway2(89s 5a 17o)			
QMDP [Littman et al, 1995]	25.9	0.052	0.02
PBUA[Poon, 2001]	100	0.35	27898
PBVI[Pineau et al 2003]	98	0.34	360
BPI [Poupart et al, 2003]	n.a.	0.32	790
Perseus [Spaan et al, 2004]	n.a.	0.35	56
HSVI1 [Smith et al, 2004]	100	0.35	10010
HSVI2 [Smith et al, 2005]	n.a.	0.35	1.5
BBMC	100	0.17	0.023

Table 2.1: Experimental results for the two Hallway domains. The BBMC achieves similar level of performance as other complicated strategies, but are significantly faster.

The simulation results are shown in Table 2.1. We see that the value-iteration-based algorithms, such as PBVI [82], HSVI1 [107], and HSVI2 [108], achieve good performance but are generally very slow, thus they are not suitable for online applications.

From the experiments, we see that BBMC is extremely fast thus good for online planning, with a little tradeoff of suboptimal performance in terms of rewards. Moreover, we see that the success rate of BBMC is 100%, which means it is very robust, in addition to its extremely fast speed. This indicates that its extension to continuous problems in real world is possible. We will use this myopic control strategy for the continuous navigation problem in Chapter 4.

Chapter 3

Particle-Based SLAM Filters for Nonlinear Filtering

3.1 Introduction

The problem of state estimation for a stochastic dynamic system based on noisy measurement data has drawn significant amount of research interest and effort, and has applications in many fields such as engineering, applied statistics, econometrics and so on. A popular and effective method to solve this type of problems is the Bayesian approach. In the Bayesian framework, the dynamic system can be represented as a *Bayesian model*, where the hidden states are connected as a Markov chain often in a continuous manner, contrary to a hidden Markov model where the hidden states are discrete. In the general cases, the so-called *state-space model (SSM)* has been widely adopted as an effective mathematical tool to describe a physical system evolving over time. The model utilizes a discrete-time formulation of the problems, which simplifies the model complexity and provides modeling convenience.

With the system representation in place, the central idea of the Bayesian approach is to represent the statistical properties of the system using the *posterior probability density function* conditioned on all available measurement information. Also, in many cases, the estimation is carried out recursively on-line, i.e., the state is estimated right after new measurement is available, which provides benefits such as real-time

update for the time-varying system and reduction in data storage complexity. But the challenging part is that it is generally hard to evaluate the distribution in an analytical form. With a few simplified assumptions such as Gaussian noise and linear system, the Bayesian model has its simplified form called the linear dynamic system, for which the Kalman filter provides the optimal and the most efficient solution. We will also introduce a few variations of Kalman filters, which are all non-particle filters. We will then introduce the *sequential Monte Carlo methods*, a.k.a. particle filters, which are the foundation for SLAM algorithms.

With the expanding application of Bayesian estimation, many practical problems often demand algorithms that can be applied in non-linear and non-Gaussian situations in order to model accurately the underlying state dynamics of a physical system. Moreover, it has become increasingly crucial for the algorithms to have the capability of processing data on-line efficiently. A series of algorithms, such as *extended Kalman filters (EKF)* and *Gaussian sum filters*, have been proposed in literature, but they suffer from various limitations [31,59]. For example, they can only be evaluated after analytical approximation of the true distribution. In recent years, with the tremendous advancement in computing power, a series of simulation-based Monte Carlo methods have been gaining popularity in solving Bayesian estimation problems. They are titled *Sequential Monte Carlo Methods*, or *Particle Filters*. These methods use a set of so-called *particles* to approximate the posterior distribution of the hidden states. The process is carried out sequentially (on-line), thus the name.

One of the research fields where statistical filters play an important role is robotic navigation. Over the past decade, the cornerstone for building fully autonomous robots and one of the key successes that have been achieved in the robotics community is the development of solutions to the *Simultaneous Localization and Mapping (SLAM)* problem. It solves the problem of building a map of the environment where a robot is located and localizing itself concurrently. The history of SLAM dates back to 1986 when it was first introduced at the IEEE Robotics and Automation Conference [33]. Since then, many researchers have devoted a lot of time and effort into this field, seeking solutions with low computational complexity and good convergence behavior. SLAM is a framework rather than a single algorithm and can be

implemented in many ways. The most common representation of the SLAM problem is to use the state-space model, which defines both the pose of the robot and the positions of the map features as one state vector. This naturally leads to the use of the extended Kalman filter (EKF) to solve the SLAM problem. In recent years, with the development of particle filters, or sequential Monte Carlo methods, an efficient SLAM procedure, called FastSLAM, has been developed by using the Rao-Blackwellized particle filter [74]. It utilizes an important property that the map landmarks become independent when conditioned on the trajectory, so that the full distribution of the robot pose and landmarks can be decomposed into independent components and evaluated separately, with the particle filter covering the robot pose and the EKF estimating the landmark locations.

The POMDP problems discussed in Sections 2.3 to 2.5 are relatively simple because they involve finite-state Markov chains, for which the posterior distribution of the states have explicit formulas. Dynamic programming (DP) can be applied to solve the POMDP problem by defining the posterior as the state, i.e., the belief state. The particle (sequential Monte Carlo) filters considered in this chapter deal with much more general state spaces and use simulated samples to represent the posterior distribution of the states. As the dimension of the problem scales up and the domain becomes continuous, as are most of the problems in the real world, *approximate dynamic programming (ADP)*, will be needed to address the challenge [10]. Thrun et al. [118] use reinforcement learning, a form of ADP, to learn value functions in the belief space, with the belief representation and propagation accomplished by Monte Carlo simulation (a.k.a., the particle filter). Bartroff and Lai (2010) [5] use rollout with the myopic policy as the base policy to tackle the stochastic control problem in cancer clinical trial designs. In Section 2.5, we have demonstrated that the myopic rule works well in a class of robotic control problems (POMDP) with much less computational complexity than DP that can be applied to relatively simple finite-state Markov chains. For the robotic applications to be considered in Chapter 4, we use myopic (or receding horizon control) policies built around an efficient particle-based SLAM filter to tackle the much more difficult filtering problem for the unobserved states.

In this Chapter, the filtering solutions to the SLAM problem, which we will call *SLAM filters* for abbreviation, will be introduced. In Section 3.2.2, we will give background introduction to the non-particle filters and also particle filters. In Section 3.3, the SLAM problem will be introduced, and the FastSLAM algorithms [74,75] will be briefly mentioned. In Section 3.4, we will adopt the spherical simplex transformation to form the spherical simplex unscented particle filters, in order to design and implement a new class of particle-based SLAM filters in Section 3.5. The new SLAM filters will serve as the foundation and one of the key tools to be used for the design of a new framework to solve a challenging problem to be discussed in Chapter 4: the integrated robotic navigation under uncertainty in continuous domain.

3.2 Nonlinear Filtering via Non-Particle Filters and Sequential Monte Carlo

In the Bayesian framework, for a set of states $(x_0, x_1, ..., x_k)$ and associated observations $(z_0, z_1, ..., z_k)$, the key is to estimate $p(x_0, x_1, ..., x_k|z_0, z_1, ..., z_k)$, the posterior distribution, from which all relevant information about the system states given observation can be obtained. In many practical applications, it requires that such posterior be estimated online. In other words, instead of directly estimating the full posterior distribution, the so-called *filtering distribution* $p(x_k|z_0, z1, ..., z_k)$ is computed recursively over time. Therefore, the Bayesian framework for state estimation concerns estimating the filtering distribution recursively. It is also known as *recursive Baysian estimation* or a *Bayes filter*.

State-Space Model

For a generic nonlinear dynamic system, its sequence of states evolution over discrete time k can be denoted as $x_k : k \in N$. Given probability distribution for the initial state $p(x_0)$, the following models govern the statistical evolution of the system state:

- System Model:

$$x_k = f(x_{k-1}, u_k, w_k) \tag{3.1}$$

- Measurement Model:

$$z_k = h(x_k) + v_k \tag{3.2}$$

where $f(\cdot)$ and $g(\cdot)$ are transition and observation functions respectively, which are possibly nonlinear, u_k is the input control at time k, $w_k \sim N(0, Q_k)$ is the process noise, and $v_k \sim N(0, R_k)$ is the measurement noise. Both noise processes are multivariate normal, and are assumed to be mutually independent.

Bayesian Estimation

As the state vector x_k being a random variable, the most complete description for the statistical properties of x_k is its *a posteriori* distribution $p(x_k|z_{1:k})$. The Bayesian estimation, based on the *Bayes' Rule*, provides a rigorous framework for dynamic state posterior estimation in a recursive fashion.

Mathematically, at time k, the current state prior conditioned on different past observations can be categorized into three types [4,21,31] :

- Prediction

$$p(x_k|z_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|z_{1:k-1})dx_{k-1} \tag{3.3}$$

- Update

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{\int p(z_k|x_k)p(x_k|z_{1:k-1})dx_k} \tag{3.4}$$

- Smoothing

$$p(x_k|x_{k+1}, z_{0:T}) = \frac{p(x_k|z_{0:k})p(x_{k+1}|x_k)}{\int p(x_k|z_{0:k})p(x_{k+1}|x_k)dx_k} \tag{3.5}$$

The recursive relation shown by Equations (3.3)-(3.4) forms the foundation of Bayesian estimation. The prediction equation (3.3) is sometimes called *Chapman-Kolmogorov* equation [4]. Oftentimes, the integrals in the above equations are hard to evaluate analytically, therefore, in Section 3.2.1, we will introduce some approximating methods for Bayesian estimation

3.2.1 Non-Particle Bayesian Filtering Methods

Before we introduce particle filters, we will first review some of the widely used non-particle filtering methods. These methods often require certain conditions, such as Gaussian noise or linearity for the state equations, in order to approximate the integration in Equations (3.3)-(3.4) to achieve optimal or suboptimal performance. They have the advantage of being faster than particle-based methods, but cannot handle complicated situations involving nonlinearity or non-Gaussian noise.

Extended Kalman Filters

The *Kalman filter (KF)* [55] is a mathematical method used to estimate the system state given measurements corrupted by noise and other inaccuracies. Its basic assumption is that the posterior density function at every time step is Gaussian, therefore the KF only utilizes the first two moments of the state, i.e. the mean and the covariance, to capture the state propagation. Also, KF requires that the system and measurement equations are both linear. Then, the integrals in the recursive relation 3.3 and 3.4 can be solved analytically. They ensure that the KF provides the optimal solution to the state estimation problem in the Gaussian-linear environment. As Kalman filter is a Bayesian filter, it carries out the state estimation recursively, in contrast to the batch estimation techniques which calculates the full posterior distribution. For each time step, the estimation consists of two steps: *prediction* and *update*. The prediction step generates estimates of the current state (mean and variance) based on the state estimation from the last step and current control signal. It equivalently produces the state values with no noise input. The update step incorporates the noisy measurement and updated the previous state estimation from the prediction step. Essentially, the observations calibrate the state estimates and shift them towards their true values.

Under circumstances where the system model function $f(x_k, u_k, w_k)$ and the measurement model equation $h(x_{k-1})$ are nonlinear, the Kalman filter will have to be modified to handle the nonlinearity, which results in the popular extended Kalman

filter (EKF). The basic idea of EKF is to linearize the system equation and measurement equation using Jacobian matrices so that the traditional Kalman filter can be applied. It is an approximating method, and inherits the advantages of Kalman filter. On the other hand, it also suffers from the linearization step as it is sometimes difficult to calculate and often introduces errors.

Gaussian Sum Filters

The Gaussian sum filters can date back to the 1970's [1,2]. The recent work for the application of particle filters, which are yet to be introduced, includes [61]. The rough idea of Gaussian sum filters is to use a bank of Gaussians to approximate the original density functions. The Bayesian relations introduced in Section 3.2 can be therefore modified for the Gaussian sum form. With the basic idea in mind, the technical details are straightforward and can be found in the above mentioned papers.

Unscented Kalman Filters

Julier and Uhlmann first introduced a general method for nonlinear transformation of probability distributions called *unscented transformation (UT)* [50]. It was founded based on the intuition that "it is easier to approximate a probability distribution than it is to approximate an arbitrary nonlinear function or transformation" [49]. The goal of the unscented transformation is to develop a statistical solution as close to the exact solution, which only exists analytically for system with special structures such as linearity and Gaussian processes. It avoids the error-prone partial derivative calculation, which is often needed for calculating higher order terms in matrix Taylor expansion, e.g., in EKF. Instead, the UT constructs a set of points, called *sigma points*, deterministically chosen according to the mean and covariance, and then applies the nonlinear transformation to each point to compute desired statistics. This means that those sigma points carry over the statistical information of the system through the nonlinear transformation, which is hard to calculate directly on the probability distributions. Essentially, the UT selects a set of sigma points that can statistically represent the whole distribution to some extent, and then apply the

nonlinear transformation to these points, which is easier to compute, rather than to the distribution. By applying the unscented transformation to the nonlinear filtering, Julier and Uhlmann developed the *unscented Kalman filter (UKF)* [53]. It captures the mean and covariance of the propagating states and avoids the computation of Jacobians, thus is more computationally accurate and stable than EKF. The details can be found in the original work.

3.2.2 Sequential Monte Carlo Methods

Sequential Monte Carlo Methods, also known as *Particle filters*, are a class of simulation-based statistical estimation techniques widely used for state estimation in nonlinear state-space models. The theoretical foundation of particle filters is Bayesian estimation. The key idea of particle filters is to use a set of randomly generated samples with associated weights to represent the desired posterior distribution of the state variable conditioned on all observations. The advantage of particle filters is their ability to deal with nonlinearity and non-Gaussianity, with the cost of more computational complexity. With the seminal paper by Gordon et al. [42], the research in particle filters has progressed significantly, as shown in [4,21,31] and the most recent work of [32]. In the literature, the sequential Monte Carlo methods are also known as bootstrap filtering, the condensation algorithm, particle filtering, interacting particle approximations, and survival of the fittest, according to [4] and its associated references. The generic particle filter has been proven to have superior performance [31], and its variations, such as *auxiliary Particle filter* [84], have been developed and applied in various applications.

Sequential Importance Sampling (SIS)

The sequential importance sampling (SIS) is a Monte Carlo method based on importance sampling, and forms the basis for the generic particle filter, which employs a resampling step after SIS. The key idea is to represent a desired distribution in terms of a weighted average of samples generated from a *importance density*, or *proposal distribution*. The weights are associated with the particles and will be updated at each

time step. The initial form of the algorithm was mentioned as early as in 1989 [39], and then recaptured and developed in [31] and reviewed in [4,21]. Here, we revisit this algorithm and give some of the details referring to [4,21,31].

Using n Monte Carlo samples, the posterior distribution conditioned on past observations over time can be represented by the following equation [4]:

$$p(x_{0:k}|z_{1:k}) = \sum_{i=1}^{n} w_k^i \delta(x_{0:k} - x_{0:k}^i). \tag{3.6}$$

where the weights are computed by the principle of importance sampling [4,21]:

$$w_k^i = \frac{p(x_{0:k}^i|z_{1:k})}{q(x_{0:k}^i|z_{1:k})} \tag{3.7}$$

where $q(\cdot)$ is the proposal distribution, or *importance density*. After carrying out a series of mathematical manipulations, we can obtain

$$w_k^i = w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k|x_{0:k-1}^i, z_{0:k})} \tag{3.8}$$

In order to distinguish the weights before and after normalization, we use $\tilde{w}_k^i$ to denote the weight of particle i at time step k before normalization, and w_k^i to denote the weight after normalization.

In a lot of applications, we often simply assume the importance density is only dependent on the past state estimation x_{k-1} and current observation z_k. Therefore, we have

$$q(x_k|x_{0:k-1}^i, z_{0:k}) = q(x_k|x_{0:k-1}^i, z_k), \tag{3.9}$$

and thus the weight recursive relation becomes

$$w_k^i = w_{k-1}^i \frac{p(z_k|x_k^i)p(x_k^i|x_{k-1}^i)}{q(x_k|x_{k-1}^i, z_k)} \tag{3.10}$$

Degeneracy Problem

After a few iterations of the SIS algorithm, most of the particles will have negligible weights. In the literature of particle filter, this is called *degeneracy problem*. A measure of degeneracy called *effective sample size* N_{eff} can be calculated as [4,31]:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2} \tag{3.11}$$

where w_k^i is the weight after normalization.

There are two basic methods to alleviate the degeneracy: 1. Choosing a good importance density, and 2. Resampling [4,31]. For the first method, there are two frequently used importance densities: one is the *optimal importance density*, which minimizes the variance of the weights, and another one is to use the prior distribution, which can greatly simplify the calculation. The details can be found in [4]. The second method will be discussed in the next part.

Resampling

When the effective sample size N_{eff} falls under certain threshold value, which means the degeneracy problem has become significant, we need to carry out *resampling* to mitigate the degeneracy. The resampling is to eliminate the particles with small weights and sample with replacement to redistribute the particles. In the seminal paper by Gordon et al. [42], a *Bootstrap* resampling scheme was proposed. The basic idea is to generate N new samples from the current N samples according to a distribution whose probability mass function is proportional to the weights associated with the current N samples.

The three most popular resampling algorithms in literature are systematic resampling, residual resampling, and multinomial resampling. We will take the *systematic sampling*, originally proposed in [59], as an example. It constructs a CDF according to the weights of the N samples, and then sample from the distribution uniformly.

The detailed algorithm can be found in [4]. For details about all the three algorithms, please refer to [32]. All the schemes mentioned above serve the same purpose of reducing degeneracy, but they also have other drawbacks. As mentioned in [4], resampling limits the opportunity to parallelize, introduces *sample impoverishment*, and causes smoothed estimates based on the particles' paths degenerate. So it should be employed with caution.

Sequential Importance Resampling

The *sequential importance resampling (SIR)*, or *particle filtering*, incorporates both the SIS step and resampling step. The resampling is carried out adaptively, depending on the value of the effective size N_{eff}. The detailed algorithm can be found in [4,31].

Rao-Blackwellized Filter

Despite of the powerful performance of particle filters, there are two limiting factors that prohibit the further application of particle filters in more complicated situations in practice [102]: 1. The number of particles needed to asymptotically approach the performance of optimal filter tends to be infinity, which results in high computational cost. 2. The number of particles increases with the state dimension in order to achieve decent state estimation. Therefore, a popular marginalization technique called *Rao-Blackwellization (RB)* is employed to reduce the computational cost for the particle filters.

The Rao-Blackwellization method [22] is one of the variance reduction methods applied to improve the performance of the Monte Carlo simulation. For particle filtering, Rao-Blackwellization implies a class of "hybrid filters where a part of the calculations is realised analytically and the other part using MC methods" [31]. More specifically, the idea is to partition the state vector into two parts:

$$x_k = \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix} \tag{3.12}$$

and then the posterior distribution can be decomposed into two parts:

$$p(x_k^1, x_k^2|z_{0:k}) = p(x_k^1|z_{0:k})p(x_k^2|x_k^1, z_{0:k}) \tag{3.13}$$

Then, the first part of the posterior $p(x_k^1|z_{0:k})$ can be obtained by particle filtering, and the second part $p(x_k^2|x_k^1, z_{0:k})$ conditioned on x_k^1 can be analytically marginalized. In special cases such as the dynamics of the second part conditioned on the first part is linearly Gaussian, then the Kalman filter can be used to evaluate the second part. The details of Rao-Blackwellization and the *Rao-Blackwellizated particle filter (RBPF)* can be found in [31,56].

Improving the Proposal Distribution

The performance of particle filters critically depends on the choice of proposal distributions [4]. The most frequently used method is to apply the prior distribution as the proposal. But it doesn't incorporate the measurement, thus may not be able to cover the highly probable area of the true posterior distribution. The optimal density, which minimizes variance, is sometimes hard to evaluate analytically [4].

Some of the techniques proposed in literature include using an EKF to approximate the optimal proposal distribution. To be more specific, for each particle, the proposal distribution is approximated by a Gaussian and is propagated by its first two moments, the mean and covariance, estimated by the EKF. At each time step, the system takes a sample from the Gaussian with the estimated mean and covariance. Merwe et al. replaced the EKF with a unscented Kalman filter [90], which result in improved performance with the advantages of the unscented filters over the extended Kalman filters as discussed in Section 3.2.1. In Section 3.4, we will introduce the adoption of the spherical simplex unscented Kalman filter as the proposal distribution, which has lower computational cost than the unscented one. The simulation study in the following section will compare the performance of each algorithm.

3.2.3 A Comparative of Different Nonlinear Filters

We consider the following nonlinear system

$$x_k = f_k(x_{k-1}, k) + w_k \tag{3.14}$$

$$z_k = \frac{x_k^2}{20} + v_k \tag{3.15}$$

where

$$f_k(x_{k-1}, k) = \frac{x_{k-1}^2}{2} + 25\frac{x_{k-1}}{1 + x_{k-1}^2} + 8\cos(1.2k) \tag{3.16}$$

and $w_k \sim N(0, Q_k)$ is the process noise, and $v_k \sim N(0, R - k)$ is the measurement noise, with $Q_k = 10$ and $R_k = 1$. Both noise processes are normal, and are assumed to be mutually independent.

We performed $S = 100$ independent simulations with $T = 60$ time steps. For particle filters, we employed $N = 200$ particles. For the unscented filters used for proposal density, we used the following scaling factors $\alpha = 0.1$, $\beta = 2$, and $\kappa = 0$.

We compare the estimation and prediction performance of the filters based on the mean square errors between the predicted states and the actual realized states. Mathematically, for S independent simulations with each being T time steps, the metric can be defined as:

$$MSE_x = \frac{1}{S}\sum_{s=1}^{S}\left(\frac{1}{T}\sum_{t=1}^{T}(x_t - \hat{x}_t^s)^2\right) \tag{3.17}$$

Essentially, the part in the parentheses is the mean-square-error (MSE) between the estimated state and the real state for the sth simulation. When we average it out over the S simulation runs, we obtain the mean of the MSE, which is the MSE_x in the above equation.

Table 3.1 summarizes the performance of different filters. The extended Kalman filter (EKF), a non-particle based filter, is used as a reference. The generic particle filter using the prior as the proposal distribution is used as a benchmark. Figure 3.1 shows one of the realizations of the actual states and the estimated states by various filtering methods. We see that the most commonly used particle filter using the

prior as the proposal distribution cannot capture the state evolution correctly during roughly the second half of the process, due to its lack of the ability to incorporate observations during the sampling process. The particle filter using EKF as the proposal distribution performs worse than the other three, because it only achieves a first order approximation. The PF-UKF and PF-SSUKF have similar performance, but PF-SSUKF has the advantage of having less computational complexity. Figure 3.2 shows the MSE comparison in terms of a bar plot for scenarios with different number of particles. We see that the performance of PF-SSUKF is consistent for various number of particles.

Algorithms	MSE mean	MSE var
EKF (The extended Kalman filter)	19.63	9.32
PF-Prior (Prior distribution as proposal)	9.88	2.42
PF-EKF (EKF as proposal)	5.65	2.38
PF-UKF (UKF as proposal)	4.97	1.72
PF-SSUKF (SSUKF as proposal)	4.94	1.46

Table 3.1: The mean square errors of various particle filters using different proposal distributions. The result of the commonly used extended Kalman filter (EKF), a non-particle approach, is quoted as a reference here.

3.3 Simultaneous Localization and Mapping

3.3.1 Definition of SLAM

SLAM is a robot navigation framework that addresses the problem of building a map for an environment where the robot is located and localizing itself within the map concurrently. In a territory with landmarks, the robot measures its relative locations to the landmarks through laser sensors. Based on the measurements and its history of controls over time, together with its known initial location, the robot estimates its path and the surrounding landmarks within the global reference frame. In our

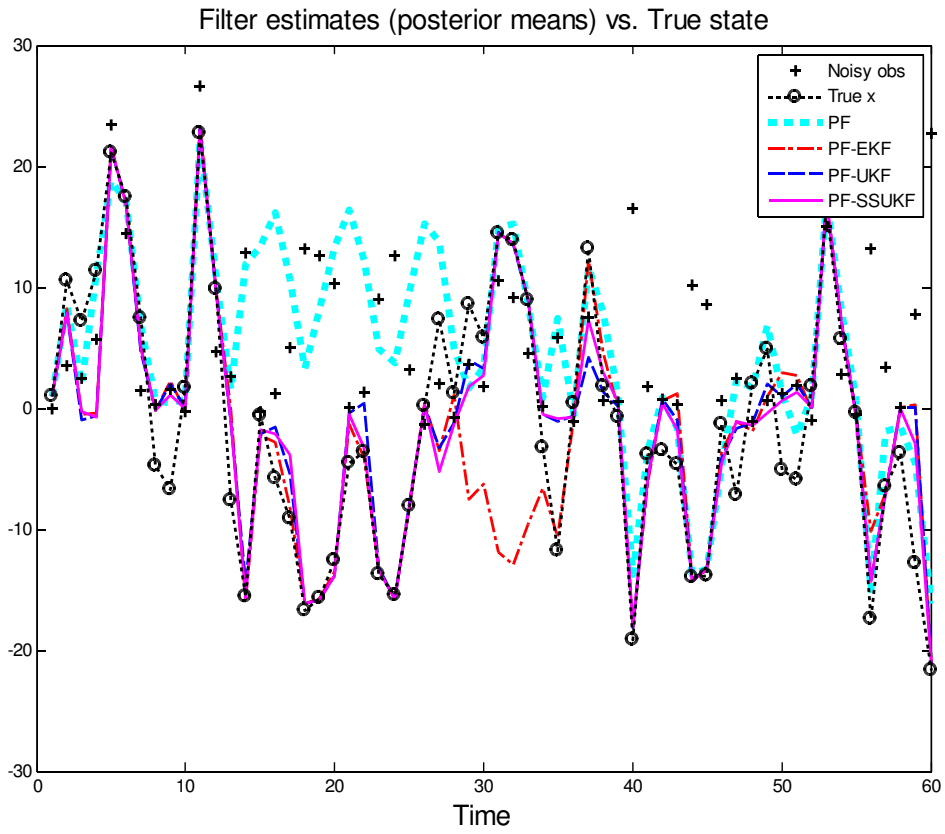


Figure 3.1: The diagram shows the comparison between the states estimated by various algorithms.

discussion, we sometimes mention "features" in a map, which are just synonymous to landmarks.

Mathematically, SLAM estimates the full posterior distribution over the robot pose and landmark locations in a recursive fashion over time, given sensor readings corrupted by noise and systematic errors such as sensor inaccuracy. That is, it estimates the density function:

$$p(\mathbf{x}_t, \mathbf{m} | \mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0) \tag{3.18}$$

where $\mathbf{x}_t$ is the state vector describing the location and orientation of the robot at time t and equal to (x, y, θ), $\mathbf{m}$ contains the time-invariant locations of all the landmarks,

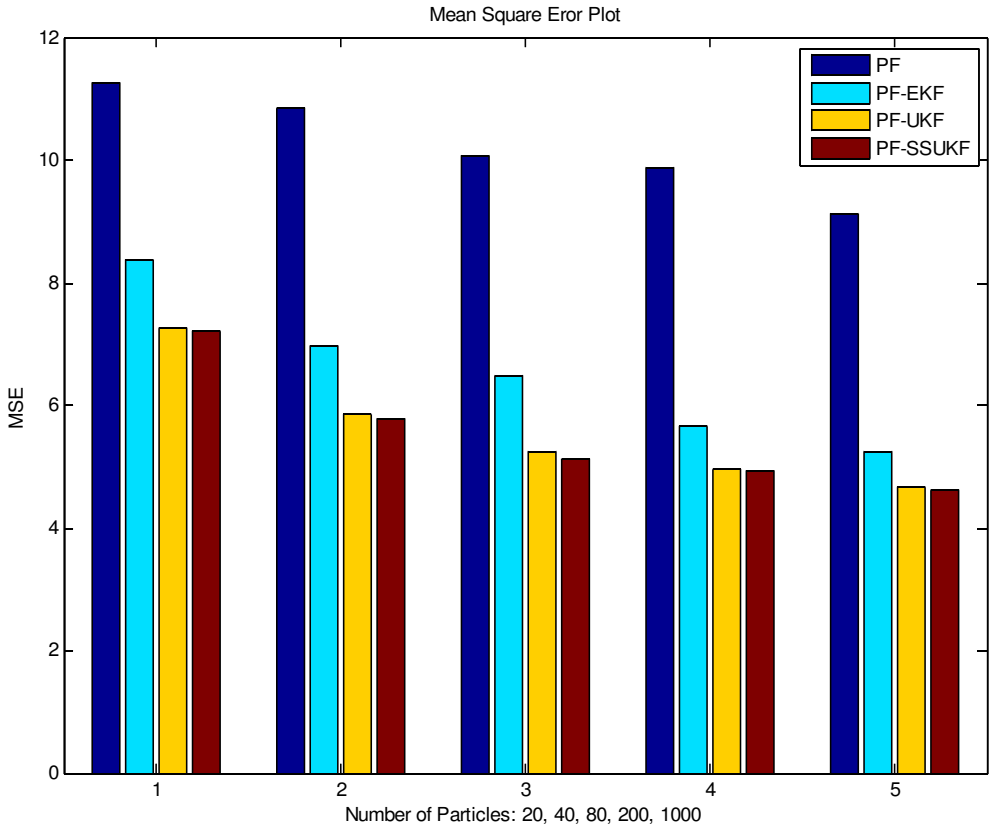


Figure 3.2: The diagram shows the performance comparison among various algorithms. The number of simulated particles is chosen to be 20, 40, 80, 200, and 1000 for each set respectively.

$\mathbf{z}_{0:t} = (z_0, ..., z_t)$ are the collection of observed locations of the landmarks from the robot up to time t, $\mathbf{u}_{0:t} = (u_0, ..., u_t)$ are the set of control actions carried out over that period, and $\mathbf{x}_0$ is the initial state. Although the estimated states of the robot, including both the robot pose and landmark locations, differ from the true values, the robot uses accumulated information it gathered over time to calibrate the estimate. It is imperative to have both a good *motion model* and a good *observation model*. To be more specific, a *motion model* describes the probabilistic relation between the

current robot state $\mathbf{x}_t$ and the past state $\mathbf{x}_{t-1}$ and the control action $\mathbf{u}_t$:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \tag{3.19}$$

An *observation model* describes the sensor measurement of the surrounding environment from the robot governed by some probabilistic law:

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}). \tag{3.20}$$

The measurement task can be accomplished by range finders, which measure the range to the nearby objects along a beam (laser range finders) or within a cone (ultrasonic sensors). They are among the most popular sensors in robotics, as introduced in [118]. Therefore, SLAM is a nonlinear filter in this hidden Markov model.

Note that for all the discussion in this chapter, we assume known data association for the observed landmarks. The case of unknown data association is a research topic different from the focus of this work. For related materials, please refer to [74] and the SLAM literature.

3.3.2 FastSLAM

FastSLAM is a recursive sampling-based algorithm that estimates the full posterior distribution over robot pose and landmark locations using a set of particles, and is based on an exact factorization of the posterior into a product of a distribution over robot paths and the landmark distributions conditioned on the robot path, with logarithmic complexity with respect to the number of landmarks in the map [75]. The first version of FastSLAM samples the robot location from the process model, or the prior distribution of the robot, without taking into account the measurements [75]. This leads to inferior performance for the obvious reason. So in an improved version of FastSLAM, the measurements are incorporated, but it complicates the problem dramatically [74]. The extended Kalman filter is used in order to approximate the sampling distribution, i.e., the proposal distribution, and the performance is improved significantly over the previous version.

FastSLAM 1.0

The above effort in solving the SLAM problem uses the EKF by assuming the linear Gaussian assumption [30]. FastSLAM [75] is a particle-based approach, which assumes that each particle sample is of the following form:

$$\mathbf{X}_t^k = \langle \mathbf{x}_t^k, \mu_{1,t}^k, \Sigma_{1,t}^k, ..., \mu_{M,t}^k, \Sigma_{M,t}^k \rangle \tag{3.21}$$

where $\mu_{i,t}^k$ and $\Sigma_{i,t}^k$ are the mean and covariance of the ith landmark estimate at time t associated with the kth particle. As the EKF approach directly estimates the joint full posterior, FastSLAM, on the other hand, carries out an exact factorization of the full posterior into a product of a distribution over robot paths represented by a set of particles, and conditional landmark distributions estimated by the EKF. Its key theoretical foundation is the factorization

$$p(\mathbf{x}_{0:t}, \mathbf{m}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0) = p(\mathbf{x}_{0:t}|\mathbf{z}_{0:t}, \mathbf{u}_{0:t}, \mathbf{x}_0) \prod_{i=1}^{M} p(\mathbf{m}_i|\mathbf{x}_{0:t}^i, \mathbf{z}_{0:t}) \tag{3.22}$$

where M is the number of landmarks observed so far and $\mathbf{m}_i$ is the location of the ith landmark for $i = 1, 2, ..., M$. This is a consequence of the following lemma: when the map estimate is based on the whole trajectory $\mathbf{x}_{0:t}$, the map landmarks become independent and therefore can be estimated separately, which gives linear complexity in estimation, instead of a quadratic one [33]. For (3.22), the Rao-Blackwellized particle filter can be used to represent the robot pose distribution (i.e., the first term in the equation) by weighted particle samples, whereas the M landmark distributions are independent Gaussian and therefore can be estimated by the EKF. Note that the FastSLAM algorithm extends the path posterior by drawing samples from the following proposal distribution

$$\mathbf{x}_t^k \ \sim \ p(\mathbf{x}_t|\mathbf{x}_{t-1}^k, \mathbf{u}_t), \tag{3.23}$$

where $\mathbf{x}_{t-1}^k$ is the posterior estimate of the robot pose at time $t-1$, $\mathbf{u}_t$ is the action taken at time t, and k indicates the k-th particle. Details can be found in [75].

The associated importance weight can be approximated by a Gaussian given the observation [118].

For feature update, the mean $\mu_{i,t}^k$ and covariance $\Sigma_{i,t}^k$ of the posterior over the ith observed feature, $p(\mathbf{m}_i|\mathbf{x}_{0:t}^i, \mathbf{z}_{0:t})$, where $i = 1, ..., M$, can be estimated based on the mean and covariance at time $t-1$ using the standard extended Kalman filter by treating the map locations as the state and incorporating the new observation [118].

FastSLAM 2.0

Montemerlo et al. (2003) proposed to draw the robot pose at time t from the following improved proposal distribution

$$\mathbf{x}_t^k \sim p(\mathbf{x}_t|\mathbf{x}_{t-1}^k, \mathbf{u}_t, \mathbf{z}_t) \tag{3.24}$$

where $\mathbf{z}_t$ is the observation at time t. The introduction of the extra parameter complicates the problem significantly. Using the EKF-style linearization, the sampling distribution can be approximated by a Gaussian through complicated mathematical derivation. Then, the importance weight, again, can be approximated by a Gaussian, but with more complication form. The feature update, similar to the previous FastSLAM, is estimated using EKF. For details, please see [76,118]. This algorithm is known as FastSLAM 2.0, and the FastSLAM described in the preceding paragraph is sometimes called FastSLAM 1.0.

3.4 Spherical Simplex Unscented Particle Filters

The unscented filtering discussed in Section 3.2.1 has been applied and generalized to a family of Kalman filters, called *sigma-point Kalman filters (SPKF)* [124]. Here, we adopt a new sigma point selection strategy developed in [48], which gives the same level of performance but with much less computational cost.

3.4.1 The Spherical Simplex Unscented Transformation

The new sigma point selection strategy we will adopt and apply to improve particle filters is called the *spherical simplex unscented transformation (SSUT)* proposed in [48], which defines a simplex of $n+2$ points that lie on a hypersphere, with the radius that bounds the points proportional to $\sqrt{n}$ and the weight applied to each point proportional to $\frac{1}{n}$, where n is the dimension of the system. As the computational cost is directly proportional to the number of sigma points selected, the new strategy imposes much less computational complexity.

The scaled unscented transformation In the original unscented transformation proposed in [49], the covariance matrix will potentially be non-positive under some circumstances. Julier addressed this problem and came up with a solution called the *sigma point scaling method*, which leads to the *scaled unscented transformation* that "guarantees the second order accuracy in mean and covariance, giving the same performance as a second order truncated filter but without the need to calculate any Jacobians or Hessians" [51]. The basic idea is to scale the distance between any sigma point X_i and the zeroth sigma point by a factor, which will be carefully chosen in order to improve the performance. This technique can also be applied to the SSUT, so below, we will briefly introduce this technique and quote the material from the original work. For more background on unscented transformation, please refer to [50,52,53,124]

The sigma point scaling method calculates the transformation of a scaled set of sigma points X_i's by introducing α, a positive *scaling factor*, as proposed in [51] :

$$X_i^{'} = X_0 + \alpha(X_i - X_0) \tag{3.25}$$

$$W_i^{'} = \begin{cases} W_0/\alpha^2 + (1 - 1/\alpha^2) & i = 0 \\ W_i/\alpha^2 & i \neq 0 \end{cases} \tag{3.26}$$

Note that the higher order effects can be significantly reduced by decreasing the value of α [51]. In other words, the scaling parameter α can be used to tune the system performance based on the specific situation. With the introduction of α, we define a

new parameter λ to replace the original κ in the following way:

$$\lambda = \alpha^2(L + \kappa) - L. \tag{3.27}$$

so that Equation 3.25 holds. To avoid confusion with the newly introduced scaling parameter α, λ can be called the *secondary scaling factor*. Then the selection scheme of the sigma-point set is given by

$$X_0 = \bar{x} \qquad i = 0 \tag{3.28}$$

$$X_i = \bar{x} + \left(\sqrt{(L+\lambda)P_x}\right)_i \qquad i = 1, ..., L \tag{3.29}$$

$$X_i = \bar{x} - \left(\sqrt{(L+\lambda)P_x}\right)_i \qquad i = L+1, ..., 2L \tag{3.30}$$

and the new weights are

$$w_0^m = w_0 \tag{3.31}$$

$$w_0^c = w_0 + (1 - \alpha^2 + \beta) \tag{3.32}$$

$$w_i^m = w_i^c = w_i, \quad i \neq 0 \tag{3.33}$$

where $w_0 = \frac{\lambda}{L+\lambda}$, $w_i = \frac{1}{2(L+\lambda)}$ for $i \neq 0$, m and c indicate the corresponding weights associated with the mean and covariance calculation respectively. Note that the third parameter β is introduced here to correct the weight of the zeroth order covariance matrix, thus we can call it a *covariance correction factor*. For the case of Gaussian distribution, the optimal choice is $\beta = 2$ [51]. For κ, its value isn't critical, and a good default choice is 0 [124].

For n-dimensional systems, the sigma points can be calculated according to the SSUT Algorithm, which is quoted from the original work and shown below. As pointed out in [52]: "This algorithm has two notable features. First, the weight on each sigma point (apart from the zeroth point) is the same and is proportional to $\frac{1-W_0}{n+1}$. Second, all of the sigma points (apart from the zeroth point) lie on the hypersphere of radius $N_x/(1 - W_0)$."

The SSUT Algorithm

1. Choose value $0 \leq W_0 \leq 1$.
2. The sequence of weights are chosen as follows:

$$W_i = \frac{1}{n+1}(1 - W_0) \tag{3.34}$$

3. The scaled version of the weights for both mean and covariance can be obtained according to Equation 3.26:

$$w_i = \begin{cases} W_0/\alpha^2 + (1 - 1/\alpha^2) & i = 0 \\ W_i/\alpha^2 & i \neq 0 \end{cases} \tag{3.35}$$

4. Initialize the spherical simplex matrix by the following:

$$Y_0^1 = 0, \quad Y_1^1 = -\frac{1}{\sqrt{2W_1}}, \quad Y_2^1 = \frac{1}{\sqrt{2W_1}} \tag{3.36}$$

5. The rest of the entries of the matrix can be constructed by:

$$Y_i^j = \begin{cases} \begin{bmatrix} Y_0^{j-1} \\ 0 \end{bmatrix} & i = 0 \\ \begin{bmatrix} Y_i^{j-1} \\ -\frac{1}{j(j+1)W_1} \end{bmatrix} & i = 1, ..., j \\ \begin{bmatrix} 0_{j-1} \\ \frac{j}{j(j+1)W_1} \end{bmatrix} & i = j+1 \end{cases} \tag{3.37}$$

6. Compute the sigma points X_i for $i = 1, ..., n$.

$$X_i = \bar{x} + \sqrt{P_x} Y_i \tag{3.38}$$

where $\sqrt{P_x}$ is a matrix square root of P_x, the covariance matrix of x, and Y_i is the ith column from the spherical simplex matrix computed above.

7. A further modified version of the weights in order to partially capture even

higher order can be obtained by introducing β [52]:

$$\begin{aligned} w_0^m &= w_0 \\ w_0^c &= w_0 + (1 - \alpha^2 + \beta) \\ w_i^m &= w_i^c = w_i, \quad i \neq 0 \end{aligned} \tag{3.39}$$

where $w_i's$ are defined by Equation 3.35, and m and c indicate the corresponding weights are associated with the mean and covariance calculation respectively. Essentially, this only changes the zeroth weight for the covariance matrix. Note that β is called the *covariance correction factor*, and has the optimal value of 2, as it has been introduced in Section 3.4.1

3.4.2 Spherical Simplex Unscented Kalman Filters

The unscented Kalman filter proposed in [53] can be modified to form the *spherical simplex unscented Kalman filter (SSUKF)*. Note that we augment the state with control noise while treating the measurement noise as an additive term in the covariance calculation in Equation 3.49. The complete spherical simplex unscented Kalman filter algorithm is shown in the *SSUKF Algorithm* below.

The SSUKF Algorithm

• Initialization

$$\bar{x}_0 = E[x_0], \quad P_0 = E[(x_0 - \bar{x}_0)(x_0 - \bar{x}_0)^T] \tag{3.40}$$

$$\bar{x}_0^a = [\bar{x}_0, 0]^T, \quad P_0^a = \begin{bmatrix} P_0 & 0 \\ 0 & Q_0 \end{bmatrix} \tag{3.41}$$

• For $k = 1, 2, ...,$

1. State Augmentation:

$$\bar{x}_k^a = \begin{bmatrix} \bar{x}_k \\ \bar{w}_k^{noise} \end{bmatrix}^T = \begin{bmatrix} \bar{x}_k \\ 0 \end{bmatrix}^T, \quad P_k^a = \begin{bmatrix} P_k & 0 \\ 0 & Q_k \end{bmatrix} \tag{3.42}$$

2. Sigma-Points and Weights Calculation:

$$\text{The SSUT Algorithm} \Rightarrow \; X_k^a, \; w_i^c, \; \text{and} \; w_i^m. \tag{3.43}$$

3. Prediction Step

$$X_{k|k-1}^x = f(X_{k-1}^x, u_k, X_{k-1}^w) \tag{3.44}$$

$$\bar{x}_{k|k-1} = \sum_{i=0}^{L_a+1} w_i^m X_{i,k|k-1}^x \tag{3.45}$$

$$P_{k|k-1}^x = \sum_{i=0}^{L_a+1} w_i (X_{i,k|k-1}^x - \bar{x}_{k|k-1})(X_{i,k|k-1}^x - \bar{x}_{k|k-1})^T \tag{3.46}$$

4. Measurement-update Step

$$Z_{k|k-1} = h(X_{k|k-1}^x) \tag{3.47}$$

$$\bar{z}_{k|k-1} = \sum_{i=0}^{L_a+1} w_i^m Z_{i,k|k-1} \tag{3.48}$$

$$P_{k|k-1}^z = \sum_{i=0}^{L_a+1} w_i^c (Z_{i,k|k-1} - \bar{z}_{k|k-1})(Z_{i,k|k-1} - \bar{z}_{k|k-1})^T + R_k \tag{3.49}$$

$$P_{k|k-1}^{xz} = \sum_{i=0}^{L_a+1} w_i^c (X_{i,k|k-1}^x - \bar{x}_{k|k-1})(Z_{i,k|k-1} - \bar{z}_{k|k-1})^T \tag{3.50}$$

$$K_k = P_{k|k-1}^{xz} (P_{k|k-1}^z)^{-1} \tag{3.51}$$

$$\bar{x}_k = \bar{x}_{k|k-1} + K_k (z_k - \bar{z}_{k|k-1}) \tag{3.52}$$

$$P_k = P_{k|k-1}^x - K_k P_{k|k-1}^z K_k^T \tag{3.53}$$

Note that w^{noise} is the noise term, w^m and w^c are the weights for mean and covariance respectively. The original state is denoted as x, the augmented state is x^a, and the sigma points are denoted as X. The state component and noise component of vector X are denoted as X^x and X^w respectively. We now explain in more details the reasoning behind the algorithm. Applying the SSUT, we can create $L+2$ sigma points and their associated weights, $\{X_k^a, w_k\}$. The next step, i.e., the time update step in the Kalman filter, is to generate a new set of sigma points for the next time step $k+1$ conditioned on current time step k through the process model $f(\cdot)$, as shown in Equation 3.44. Then, the predicted mean and covariance can be computed using Equations 3.45 - 3.46. Note that only the x-component of the augmented state X_k^a, denoted as X_k^x, is needed for the calculation.

3.4.3 SSUKF-Based Proposal Distribution

The proposal distribution of a particle filter can be approximated by a Gaussian distribution with mean x and covariance P. Thus the SSUKF can be applied to estimate the mean and covariance recursively for each particle. The detailed implementation for the SSUKF-based particle filter, i.e., the *spherical simplex unscented particle filter (SSUPF)*, is shown below.

The SSUPF Algorithm

- Initialization
  - For each particle $i = 1, ..., N$,
    - Sample particle from the initial prior distribution

$$x_0^i \sim p(x_0) \tag{3.54}$$

- For $k = 1, 2, ...,$
  - Importance Sampling
    - Sampling for each particle $i = 1, ..., N$

1. SSUKF Filtering Step for Proposal distribution

$$(\bar{x}_k^i, P_k^i) = SSUKF(x_{k-1}^i, P_{k-1}^i, u_{k-1}, z_k) \tag{3.55}$$

2. Sampling from proposal distribution

$$x_k^i \sim q(x_k|x_{0:k-1}, z_{0:k}) \sim N(\bar{x}_k^i,\ P_k^i) \tag{3.56}$$

3. Calculate associated important weight:

$$\tilde{w}_k^i \sim w_{k-1}^i \frac{p(y_k|x_k^i)P(x_k^i|x_{k-1}^i)}{q(x_k|x_{k-1}^i, y_k)} \tag{3.57}$$

⋆ Normalize Importance Weights

$$w_k^i = \frac{\tilde{w}_k^i}{\sum_{j=1}^{N} \tilde{w}_k^j} \tag{3.58}$$

∘ Resampling

⋆ If $N_{eff} > N_{thresh}$, do not resample:

$$x_k^i = \tilde{x}_k^i, \quad i = 1:N \tag{3.59}$$

⋆ Otherwise, resample:

1. For $i = 1:N$,

$$\text{sample new index } j(i) \text{ for each } i \text{ (see Section 3.2.2).} \tag{3.60}$$

2. For $i = 1:N$,

$$x_k^i = \tilde{x}_k^{j(i)}, \quad i = 1:N \tag{3.61}$$

3.5 Spherical Simplex Unscented Particle-Based SLAM Filters

In Section 3.4, we introduced the spherical simplex unscented transformation proposed in [48], and applied it to improve the performance of particle filters, which are named the *spherical simplex unscented particle filters*. As we know, the spherical simplex unscented transformation (SSUT) can achieve the similar performance as the unscented transformation (UT), while using much less computational resources due to the fact that the SSUT uses much less sigma-points. In this section, we will apply the same technique to the particle-based SLAM filters to improve the performance. The new class of algorithms are entitled *spherical simplex unscented particle-based SLAM filters*, or *SSU-PBSLAM* filters.

In the SLAM literature, there is some independent work that applies the unscented transformation (UT) to the SLAM problem. For example, unscented Kalman filter was implemented to solve the SLAM problem [3,67], much similar to the EKF-SLAM. But it is not a particle-based approach. There was an effort to incorporate the UT to particle-based SLAM [58,126]. Besides the difference of implementing more computationally efficient SSUT rather than UT, our approach constructs in an additive form to incorporate the measurement noise in the covariance update step for both robot pose and feature estimation, instead of using the augmented form, which requires larger state dimension thus more computational complexity. In addition to that, our approach implements the feature update step by properly constructing a new state in order to transform both the pose and feature location so as to obtain the transformed observation. This is because the original transformation transforms one variable, but for an observation, it implicitly represents two variables, the robot pose and landmark location, so both need to be transformed in order to obtain the correct transformed observation. Transforming only the feature location will cause inferior estimation of the landmark estimation, which will propagate the estimation error to the robot pose estimation and then iterate back to the feature estimation thus amplify the errors accumulatively. We will address how to correctly design this important step in the following sections.

3.5.1 System Dynamics

Before we dig into the design of the SSU-PBSLAM, we first introduce the system dynamic models to form the basis for later discussions, which include the process model, a.k.a vehicle motion model for SLAM problem, and measurement model.

At any time t, the vehicle pose $\mathbf{x}_t$ consists of its location (x_t, y_t) and its orientation θ, namely,

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} \tag{3.62}$$

We will first define the vehicle motion dynamics. At any given time t, the control signal of the vehicle consists of two components, the speed of the vehicle v_t^0 and its steering angle η_t^0, where the superscript 0 indicates the original control signals at time t before corrupted by Gaussian noise with zero mean and covariance $\mathbf{w}_t$. Then, the corrupted version of the control signals carried out are denoted as v_t and η_t. Note that the steering angle is the angle the vehicle tries to turn in the next time step, which is different from its orientation θ_t. Both angles are defined within the range $(-\pi, \pi]$. For each time step, the vehicle motion model is defined as,

$$\mathbf{x}_{t+\Delta t} = f(\mathbf{x}_{t-1}, \mathbf{u}_t, \mathbf{w}_t) = \begin{bmatrix} x_t + v_t \cdot \Delta t \cdot cos(\eta_t + \theta_t) \\ y_t + v_t \cdot \Delta t \cdot sin(\eta_t + \theta_t) \\ \theta_t + v_t \cdot \Delta t \cdot sin(\eta_t)/L_B \end{bmatrix} \tag{3.63}$$

where L_B is the length of the vehicle base.

Given the estimated map $\mathbf{m}$ up to time t and the estimated state $\mathbf{x}_t$, the vehicle measurement model is defined as

$$\mathbf{z}_{t+\Delta t} = h(\mathbf{x}_t) = \begin{bmatrix} \sqrt{(x_t - x_f)^2 + (y_t - y_f)^2} \\ \arctan\left(\dfrac{y_t - y_f}{x_t - x_f}\right) - \theta_t \end{bmatrix} + \mathbf{v}_t \tag{3.64}$$

where (x_f, y_f) is the coordinate of a given feature (i.e. landmark), and $\mathbf{v}_t$ is the additive observation noise.

The model introduced here are the simplified forms of the vehicle models. These models will be used for the simulation study to be carried out in Section 3.6.1. For applications in the real world, such as the outdoor experiment to be introduced in Section 3.6.2, both the motion model and measurement models of the vehicle are much more complicated, and will be introduced in that section.

3.5.2 State Estimation

In order to apply spherical simplex unscented transformation, we need to augment the state, due to the nonlinear nature of the system components. To be more specific, since the process equation, a.k.a, the transition function, is nonlinear, the process noise introduced mainly by the control signals is thus not additive. This means that we have to incorporate the control signals into the state. On the other hand, the measurement noise is often additive, therefore we do not include the measurement noise in the state. Instead, we incorporate the covariance of the measurement in an additive form in one of the covariance matrix calculation equations.

State Prediction

1. State Augmentation:

The implementation of the unscented transformation to the Kalman filter incorporates the process noise w_k and measurement noise v_k. Here, for the SLAM application, we incorporate the control signal to both the state and its associated covariance matrix:

$$\mathbf{x}_t^a = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ v_t \\ \eta_t \end{bmatrix}, \quad P_t^a = \begin{bmatrix} P_t & 0 \\ 0 & Q_t \end{bmatrix} = \begin{bmatrix} P_t^x & 0 & 0 & 0 & 0 \\ 0 & P_t^y & 0 & 0 & 0 \\ 0 & 0 & P_t^\theta & 0 & 0 \\ 0 & 0 & 0 & Q_t^v & 0 \\ 0 & 0 & 0 & 0 & Q_t^\eta \end{bmatrix} \tag{3.65}$$

where v_t and η_t are the speed control and steering control respectively, P_t and Q_t are the covariance matrices for the original state and the control signals respectively.

Note that we assume that the process noise comes from the control noise, thus the noise covariance is the same as that of the control signal. In most of the cases, we assume the components of the augmented states are uncorrelated with each other, which results a diagonal covariance matrix. Note that the total dimension of the augmented state vector x_t^a is $L_a = L_x + L_u = 3 + 2 = 5$, where $L_x = 3$ is the original state dimension and $L_u = 2$ is the control dimension.

2. Generate sigma points and weights.

After having the augmented state, we generate a series of sigma points based on the spherical simplex unscented transformation introduced in Section 3.4.1. To be more specific, we generate the sigma points according to

$$X_i^a = \bar{\mathbf{x}}_t^a + \sqrt{P_t^a} Y_i \tag{3.66}$$

where Y_i is the ith column from the spherical simplex matrix computed using the SSUT algorithm. The associated weights w_i^c and w_i^m for $i = 1, ..., L_a + 2$ can be calculated using Equation 3.39 of the SSUT algorithm.

3. Prediction Step

The predicted vehicle location for the next time step is computed using the following equations:

$$X_{t+\Delta t|t}^x = f(X_t^x, X_t^u) \tag{3.67}$$

$$\bar{x}_{t+\Delta t|t} = \sum_{i=0}^{L_a+1} w_i^m X_{i,t+\Delta t|t}^x \tag{3.68}$$

$$P_{t+\Delta t|t}^x = \sum_{i=0}^{L_a+1} w_i (X_{i,t+\Delta t|t}^x - \bar{x}_{t+\Delta t|t})(X_{i,t+\Delta t|t}^x - \bar{x}_{t+\Delta t|t})^T \tag{3.69}$$

where $f(\cdot)$ is the vehicle motion function.

State Update

1. State Augmentation:

For the state update step, we further incorporate the estimated feature positions

up to time t to the already augmented state and its associated covariance matrix:

$$\mathbf{x}_t^{a,up} = \begin{bmatrix} \mathbf{x}_t \\ \mathbf{u}_t \\ \mathbf{x}_t^{f,o} \end{bmatrix}, \quad P_t^{a,up} = \begin{bmatrix} P_t & 0 & 0 \\ 0 & Q_t & 0 \\ 0 & 0 & P_t^{f,o} \end{bmatrix} \tag{3.70}$$

where *up* indicates it is for the update step only, $\mathbf{x}_t^{f,o}$ is a $L_f \times 1$ column vector denoting the position of the feature observed at time t. $P_t^{f,o}$ is the estimated covariance matrix for the observed features at time t. The total dimension of the augmented state vector $x_t^{a,up}$ is $L_{a,up} = L_a + L_{f,o} = 5 + 2 = 7$.

2. Generate sigma points and weights.

Again, we generate a series of sigma points and the associated weights based on the spherical simplex unscented transformation introduced in Section 3.4.1. Please note that as the dimension of the augmented state in this case is increased, the total number of sigma points will also be different from the prediction step.

3. Measurement-update Step

Given the measurement, the system updates the vehicle location from the predicted values obtained during the prediction step using the following equations:

$$Z_{t+\Delta t|t} = h(X^x_{t+\Delta t|t}, X^{x^f}_{t+\Delta t|t}) \tag{3.71}$$

$$\bar{z}_{t+\Delta t|t} = \sum_{i=0}^{L_a+1} w_i^m Z_{i,t+\Delta t|t} \tag{3.72}$$

$$P^z_{t+\Delta t|t} = \sum_{i=0}^{L_a+1} w_i^c (Z_{i,t+\Delta t|t} - \bar{z}_{t+\Delta t|t})(Z_{i,t+\Delta t|t} - \bar{z}_{t+\Delta t|t})^T + R_{t+\Delta t} \tag{3.73}$$

$$P^{xz}_{t+\Delta t|t} = \sum_{i=0}^{L_a+1} w_i^c (X^x_{i,t+\Delta t|t} - \bar{x}_{t+\Delta t|t})(Z_{i,t+\Delta t|t} - \bar{z}_{t+\Delta t|t})^T \tag{3.74}$$

$$K_{t+\Delta t} = P^{xz}_{t+\Delta t|t}(P^z_{t+\Delta t|t})^{-1} \tag{3.75}$$

$$\bar{x}_{t+\Delta t} = \bar{x}_{t+\Delta t|t} + K_{t+\Delta t}(z_{t+\Delta t} - \bar{z}_{t+\Delta t|t}) \tag{3.76}$$

$$P_{t+\Delta t} = P^x_{t+\Delta t|t} - K_{t+\Delta t} P^z_{t+\Delta t|t} K^T_{t+\Delta t} \tag{3.77}$$

When there are multiple observations obtained at the same time, we apply the above update procedures to each observation, generating a new set of sigma points and weights based on the current observation and updating the mean and covariance matrix iteratively. In other words, the mean and covariance will be updated from the mean and covariance updated from the previous observation. After incorporating all the observations, the output will provide more accurate estimated mean and covariance for the sampling procedure.

3.5.3 Feature Estimation

When there is a new feature observed, we will first judge whether or not it has been observed before. If not, it is a newly observed feature. Then the feature will be added to the map, and it's initial mean and covariance are estimated using the basic method from the FastSLAM 2.0 algorithm [74]. If it has already observed and is the ith feature in the map, we then need to update its location. During this process, we treat the ith feature location $\mathbf{x}_{i,t}^f$ as the 'state', as opposed to the robot pose. Then, the process model for the feature state is just

$$\mathbf{x}_{i,t}^f = \mathbf{x}_{i,t-\Delta t}^f, \tag{3.78}$$

simply because the features don't move. Then, the measurement model for the feature state is

$$\mathbf{z}_{i,t}^f = h(\mathbf{x}_t, \mathbf{x}_{i,t}^f) + \mathbf{v}_t \tag{3.79}$$

We see that the process noise is zero, while the measurement noise still exists. Therefore, we do not need to include the control signals in the augmented state. The measurement noise covariance is still included as an additive term in the covariance calculation step.

1. State Augmentation:

   We further incorporate the robot pose $\mathbf{x}$ into the state and also augment the

associated covariance matrix:

$$\mathbf{x}_t^{a,f} = \begin{bmatrix} \mathbf{x}_{i,t}^f \\ \mathbf{x}_t \end{bmatrix}, \quad P_t^{a,f} = \begin{bmatrix} P_{i,t}^f & 0 \\ 0 & P_t \end{bmatrix} \tag{3.80}$$

where f indicates it's for the feature update step only, $\mathbf{x}_{i,t}^f$ is a $L_f \times 1$ column vector denoting the position of the ith feature observed at time t. $P_{i,t}^f$ is the estimated covariance matrix for the observed features at time t. The total dimension of the augmented state vector $x_t^{a,f}$ is $L_{a,f} = L_x + L_{f,o} = 3 + 2 = 5$.

2. Generate sigma points and weights.

Again, we generate a series of sigma points X_t^f and the associated weights w^f based on the spherical simplex unscented transformation introduced in Section 3.4.1.

3. Measurement-update Step

We denote the mean and covariance matrix of the feature observed at time t as $\bar{x}_t^f$ and P_t^f respectively. Given the measurement, the system updates the feature location from the previous values using the following equations:

$$Z_{t+\Delta t|t} = h(X_{t+\Delta t|t}^{f,x}, X_{t+\Delta}^{f,x^f}) \tag{3.81}$$

$$\bar{z}_{t+\Delta t|t} = \sum_{i=0}^{L_a+1} w_i^m Z_{i,t+\Delta t|t} \tag{3.82}$$

$$P_{t+\Delta t|t}^z = \sum_{i=0}^{L_a+1} w_i^c (Z_{i,t+\Delta t|t} - \bar{z}_{t+\Delta t|t})(Z_{i,t+\Delta t|t} - \bar{z}_{t+\Delta t|t})^T + R_{t+\Delta t} \tag{3.83}$$

$$P_{t+\Delta t|t}^{zx^f z} = \sum_{i=0}^{L_a+1} w_i^c (X_{i,t+\Delta t|t}^{f,x^f} - \bar{x}_t^f)(Z_{i,t+\Delta t|t} - \bar{z}_{t+\Delta t|t})^T \tag{3.84}$$

$$K_{t+\Delta t} = P_{t+\Delta t|t}^{x^f z} (P_{t+\Delta t|t}^z)^{-1} \tag{3.85}$$

$$\bar{x}_{t+\Delta t}^f = \bar{x}_t^f + K_{t+\Delta t}(z_{t+\Delta t} - \bar{z}_{t+\Delta t|t}) \tag{3.86}$$

$$P_{t+\Delta t}^f = P_{t+\Delta t|t}^{f,x^f} - K_{t+\Delta t} P_{t+\Delta t|t}^z K_{t+\Delta t}^T \tag{3.87}$$

Note that there is no prediction step, because the features are static. In other words, Equation 3.78 implies that $\bar{x}_{t+\Delta t|t}^f = \bar{x}_t^f$. The updated feature location is just the first

two dimensions of the augmented feature state. If multiple features are observed at the same time, they will be updated independently.

3.6 Experimental Results

3.6.1 Experiment 1: Simulated Environment

We first test our algorithms for a robot vehicle in a simulated environment as shown in Figure 3.3. The red circles are the way points, which guide the vehicle to reach the goal location. To be more specific, at any time t, the vehicle will point to a way point, after reaching a given range of that way point, say, $1m$, the vehicle will point to the next way point. This process continues until it reaches the goal. Note that the way point essentially act as the control mechanism for the vehicle in this case. It starts at the center of the map, and finish at the point in the lower part of the map. The blue line is the true robot path, and the red line is the estimated path. The green dots indicate the static landmarks, and the red dots are the estimated locations of the landmarks.

The dimension of the vehicle is $2m \times 4m$. The maximum range of the laser is $30m$. The measurement is carried out every 8 time steps, while the control signals are executed every step. Between the observations, only state prediction is carried out when there is no observation. The standard deviation for the velocity control signal is $0.3m/s$, and that for steering angle is $3\pi/180rad$. The maximum velocity of the vehicle is set to be $10m/s$.

In Table 3.2, we show the performance of the FastSLAM 2.0, U-PBSLAM, and SSU-PBSLAM. FastSLAM 2.0 has been introduced in Section 3.3.2. For the unscented particle-based SLAM filter (U-PBSLAM), we implemented the algorithm by ourselves using the unscented particle filter, which is similar to earlier work such as [58,126] but differs in implementation details. The simulations were carried out for 10 times in order to obtain the mean and standard deviation of the mean-square-error

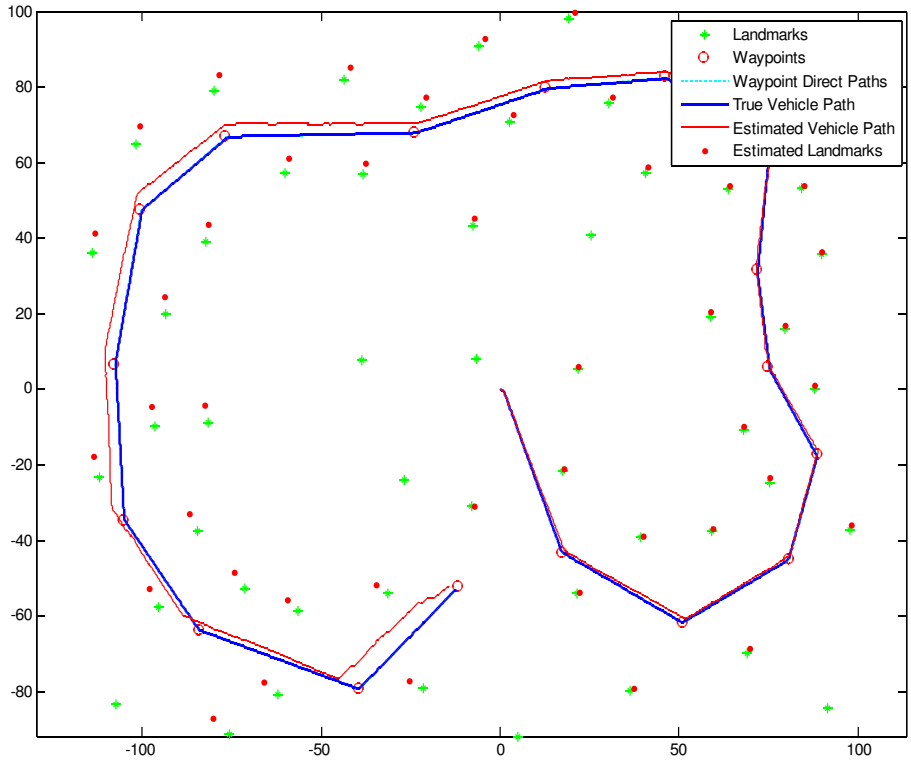


Figure 3.3: The simulated environment where the SLAM algorithms are tested for experiment 1.

(MSE), which is defined in Equation 3.17. Again, as we noted right below that equation, the part in the parentheses is the MSE for the sth simulation, so with 10 runs, we can obtain the mean and standard deviation of the MSE. We show the results for both the vehicle pose and the landmarks. Figure 3.4 and Figure 3.5 visualize the results by showing the mean of MSE for both the vehicle pose and landmark locations using bar plot respectively. Indeed, we see that the SSU-PBSLAM algorithm has at least or superior performance compared with other two algorithms, with its computational advantage over the U-PBSLAM algorithm.

Algorithms	Robot Pose		Landmark Locations	
	mean	std	mean	std
FastSLAM 2.0	5.63	0.90	6.61	1.12
U-PBSLAM	4.26	0.48	4.60	0.68
SSU-PBSLAM	4.25	0.63	4.54	0.88

Table 3.2: The performance comparison of different particle-based SLAM filters. The criteria used are the mean and standard deviation of the mean square error (MSE) with the unit being *meters*.

3.6.2 Experiment 2: Car Park Dataset

In this experiment, we test our algorithms on the dataset of an outdoor environment. The dataset was obtained on the top level of the car park building the University of Sidney, thus was often referred to as the *University Car Park Dataset*. It is available publicly online [77]. In the experiment, the landmarks are artificially made using $60mm$ steel poles covered with reflective tape, which facilitates the feature extraction and provides more accurate observations. The actual positions of the robot vehicle during the process were measured by the Global Positioning System (GPS) for the purpose of comparison.

Before proceeding to processing the data, we first briefly introduce the information and dynamics of the vehicle referring to [77], where the pictures of the vehicle and more detailed information are available. The vehicle used in the experiment is a pick-up truck, and has the following physical parameters: the distance from the front wheels to the back wheels is $L = 2.83m$; the distance from the center of the axle to the wheel is $H = 0.76m$; the laser sensor is located at the front of the vehicle, $b = 0.5m$ on the left from the center; and it is $a = 3.78m$ away from the back axle. The speed v in the data set was measured by an encoder located at the back left wheel of the vehicle, so we need to convert this speed to the speed at the center of the back axle with the following equation:

$$v_c = \frac{v}{1 - \frac{H}{L}\tan(\eta_t)} \tag{3.88}$$

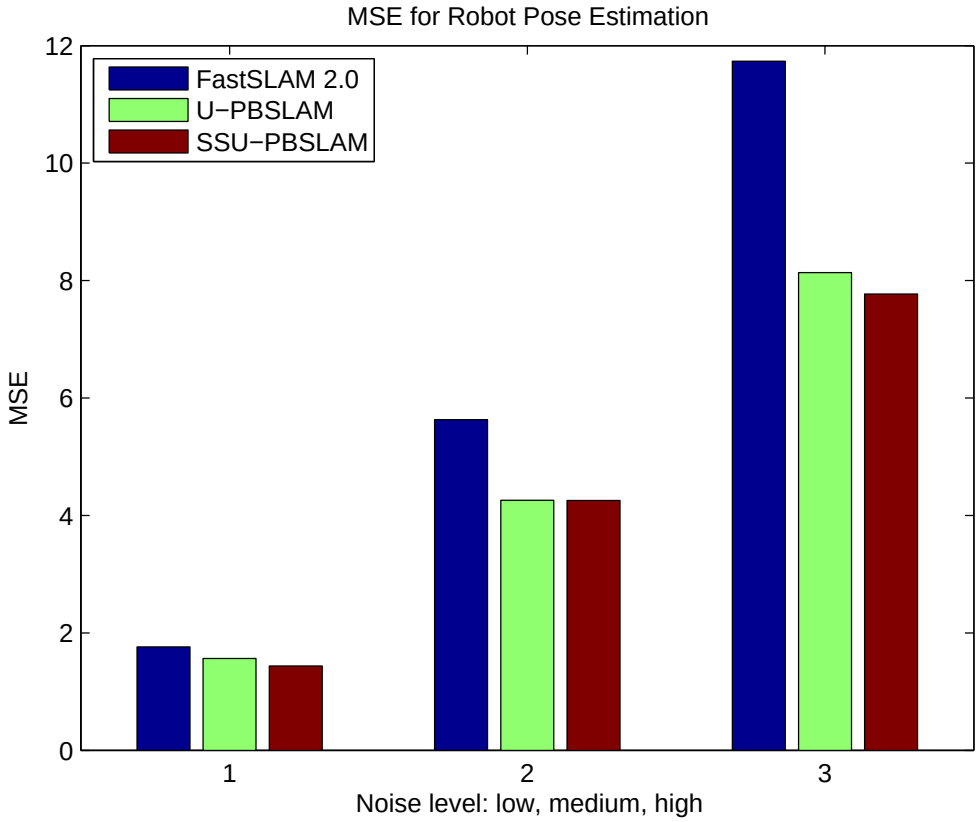


Figure 3.4: The MSE for the robot pose estimation with noise of various level for experiment 1.

where v_c is the speed at the center of the back axle of the vehicle, v is the measured speed in the data set, and η_t is the steering angle which is one of the control signals. If we define the state of the vehicle to be $\mathbf{x}_t = (x_t, y_t, \phi)^T$, where (x, y) and ϕ are the location and orientation of the vehicle respectively, the process model of the vehicle is

$$\mathbf{x}_{t+\Delta t} = \begin{bmatrix} x_t + v_c \Delta t \left(\cos(\phi) - \frac{1}{L} \tan(\eta_t)(a \sin(\phi) + b \cos(\phi)) \right) \\ y_t + v_c \Delta t \left(\sin(\phi) + \frac{1}{L} \tan(\eta_t)(a \cos(\phi) - b \sin(\phi)) \right) \\ \phi + \Delta t \frac{v_c}{L} \tan(\eta_t) \end{bmatrix} \tag{3.89}$$

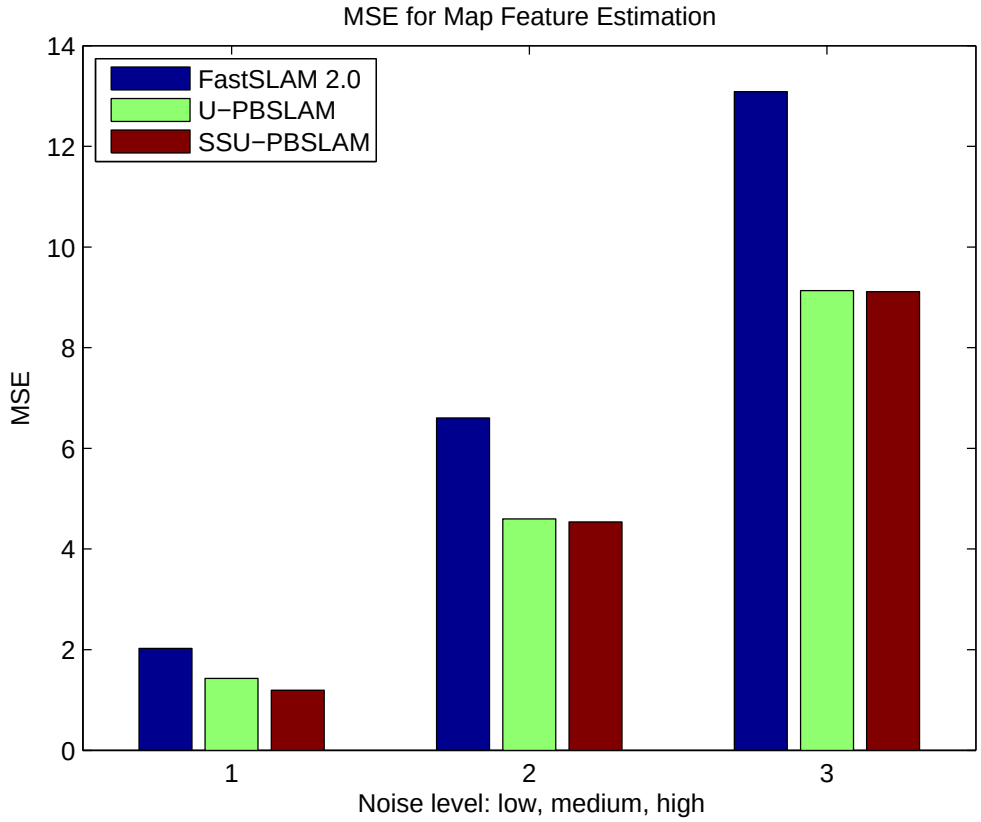


Figure 3.5: The MSE for the map feature estimationwith noise of various level for experiment 1.

And the observation model of the vehicle is given by

$$\begin{bmatrix} z_r \\ z_\beta \end{bmatrix} = h(\mathbf{x}_t) = \begin{bmatrix} \sqrt{(x - x_f)^2 + (y - y_f)^2} \\ \arctan\left(\dfrac{y - y_f}{x - x_f}\right) - \phi + \dfrac{\pi}{2} \end{bmatrix} \tag{3.90}$$

where (x_f, y_f) is the coordinate of a given feature (i.e. landmark).

In order to compare the performance of the various particle-based SLAM algorithms, we try to eliminate the errors caused by unknown data association. To do this, we use a well-tuned EKF filter, which uses nearest neighbor χ^2 test for unknown data association, to run though the same dataset and record the observed landmark

index with its associated observation. Then when we run the particle-based SLAM filters, the observations are all matched with the landmark index. Here is another benefit from this operation: as proposed in [74], each particle carries out the data association procedure independently with the particle with wrong data association more likely to be eliminated during the resampling process, so the total number of landmarks identified in each particle may vary, which will cause trouble in calculating the expected location of the landmarks.

Figure 3.6-3.8 show the performance of the various particle-based SLAM algorithms, using EKF, UKF, and SSUKF as proposal distribution approximation respectively. Note that FastSLAM2.0 uses EKF to approximate its proposal distributions. The SSU-PBSLAM has the best performance (much better than FastSLAM 2.0, and slightly better or at least on the same level compared with the U-PBSLAM but with less computational complexity), as we can see that the estimated vehicle path and landmark locations are closer to the actual positions.

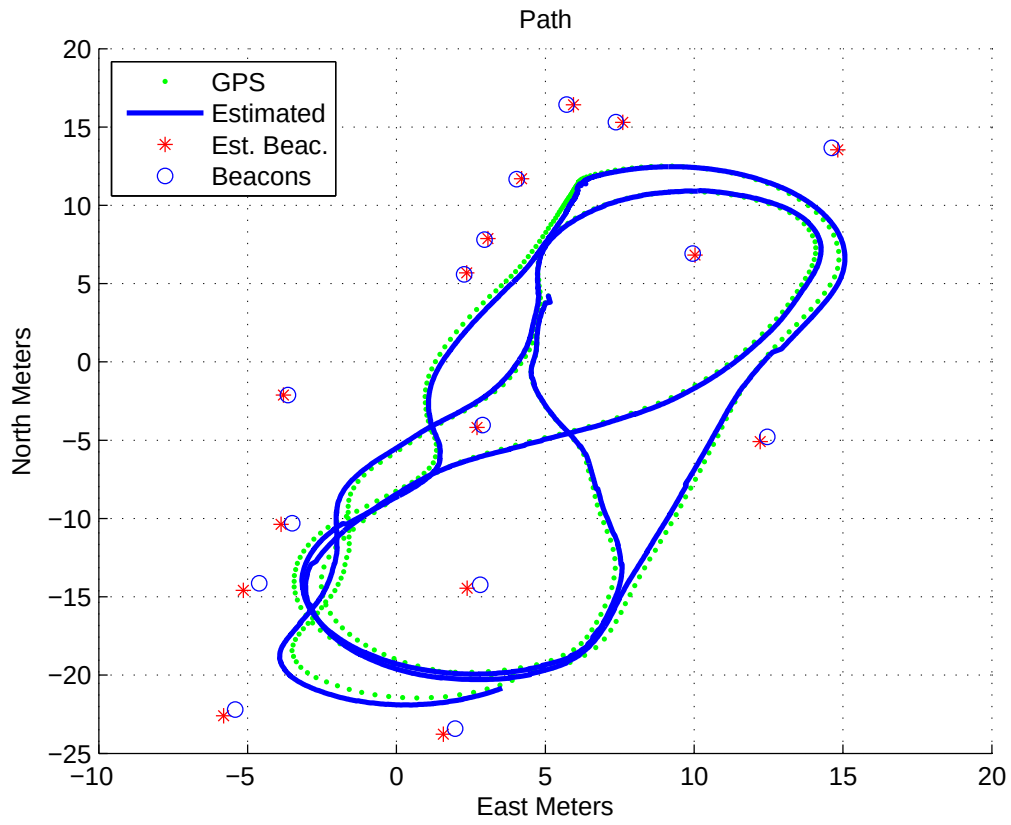


Figure 3.6: The Car Park dataset. The FastSLAM 2.0 is used as the SLAM filter to carry out the estimation for the robot pose and landmark locations, which are compared with the actual locations.

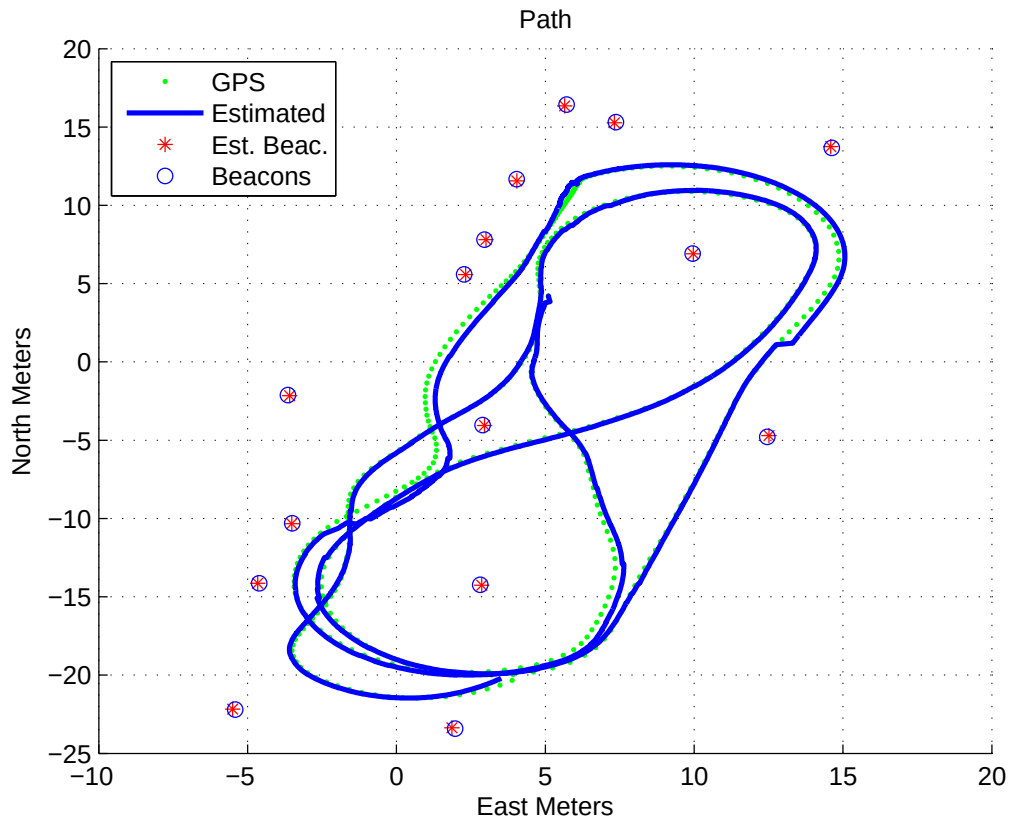


Figure 3.7: The Car Park dataset. The U-PBSLAM is used as the SLAM filter to carry out the estimation for the robot pose and landmark locations, which are compared with the actual locations.

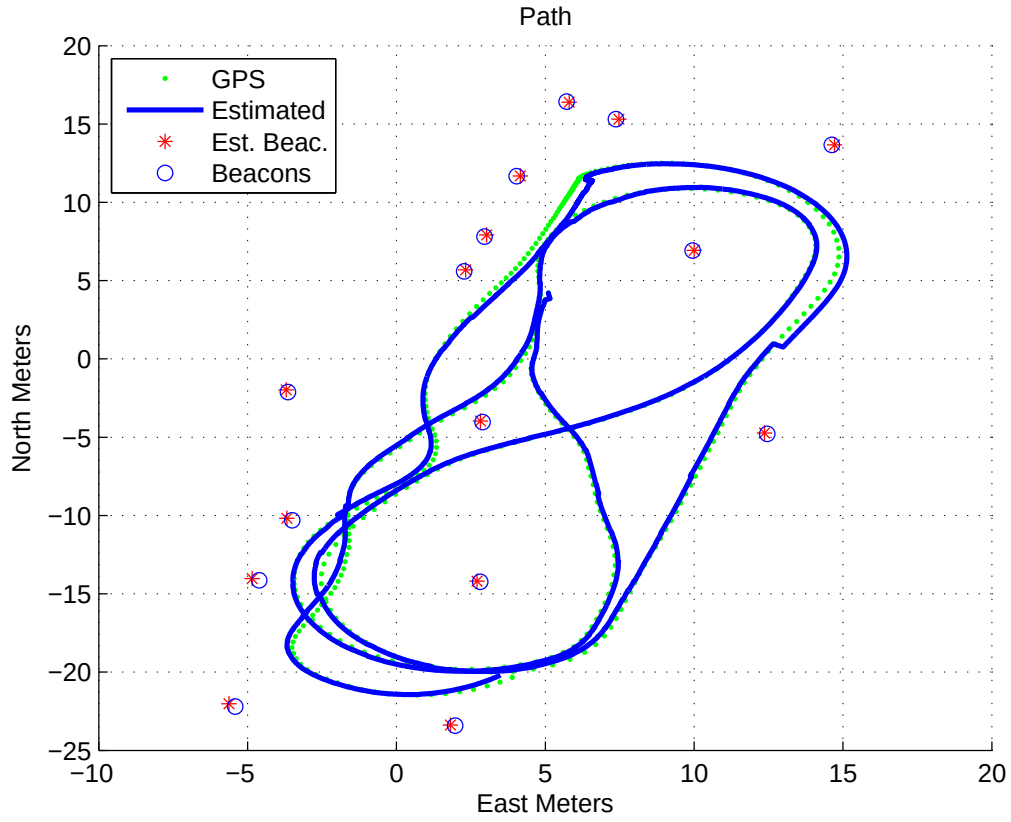


Figure 3.8: The Car Park dataset. The SSU-PBSLAM is used as the SLAM filter to carry out the estimation for the robot pose and landmark locations, which are compared with the actual locations.

Chapter 4

Integrated Robotic Navigation under Uncertainty

4.1 Introduction

Robotic navigation has profound applications in real world problems. As vast amount of research effort has been devoted to this field, the progress is exhilarating. For example, unmanned vehicles such as Mars rover has successfully accomplished a lot of tasks on the tough surface of Mars, sending back huge amount of scientifically valuable data. The DARPA Challenge is another example of successful advancement in the unmanned navigation, which keeps gaining momentum of steady development.

From a technical point of view, robotic navigation can be categorized into three main areas: global planning, local navigation, and exploration. These three areas function differently, but are closely interweaved. With more strenuous requirements from the practical applications, the robot should also possess the ability to handle uncertainty, which is ubiquitous in the real world and arises from various sources such as sensor measurement errors, and domains in a continuous setting, which is how the real world behaves. The existence of uncertainty tremendously complicates the navigation problem, so coping with the uncertainty has been one of the most important problems to tackle in the robotic navigation research. The continuous domain, which the POMDP framework introduced in Chapter 2 practically fails to handle, has been

a hurdle limiting the real applications of a lot of theoretical techniques.

In this chapter, we will provide a new framework to integrate the global planning, local navigation, and exploration, with the aid of a good particle-based SLAM filter, to solve the robotic navigation under uncertainty in a continuous domain. The robot plans to reach a global destination without any *a priori* knowledge of the environment, while avoiding collision with local landmarks. Due to the existence of uncertainties, the SLAM filter will be kept running in the background to keep track of the robot pose and landmark locations. Under certain circumstances, for example, the robot enters a landmark-dense area, then exploration will be carried out to localize the surrounding landmarks with better estimation accuracy in order to find a safe path. So the new framework balances between exploration and exploitation, with an suboptimal control strategy designed to lead the robot to reach goal location on a global setting while also achieving local objectives.

The chapter is organized as follows: Section 4.2 to 4.4 review the literature for global planning, local navigation, and exploration respectively. Section 4.5 discusses the drawbacks of current approaches in literature, formulates a new and more challenging robot navigation problem, and introduces our new framework to solve this problem. Section 4.6 shows the experimental study carried out in a simulated environment to verify our method, as no experimental dataset is available for the challenging problem we formulate.

4.2 Global Planning

Global path planning addresses the task of an autonomous robot, such as an *Unmanned Ground Vehicle* [40,46,114,119], an unmanned aircraft or rotorcraft [28,37], or a Mars Rover [120], acquiring information of the environment and planning a collision-free trajectory to navigate to a destined location under certain physical constraints. It is complementary to *local planning*, which is concerned more with the issue of vehicle stability and safety in response to the presence of local obstacles by generating new paths in order to avoid collision. Early work of global path planning

assumes the robot to have complete knowledge of the environment and its own location. The assumption is invalid when the algorithms are used in the real world, where the environmental information is generally only partially available or completely unavailable in advance. More recent work has focused on how to generate a global path in the presence of sensor noise and map incompleteness. In this section, we mainly review classical path planning algorithms, and will briefly mention planning with uncertainty, because the frameworks, SLAM and POMDP, have been discussed in the previous chapters.

4.2.1 Classical Path Planning

The classical path planner usually represents the world using the so-called *configuration space*, which contains all the possible configurations associated with the position and orientation of the robot. In other words, each possible combination of the robot's position and orientation is mapped to a single point in the configuration space. The key to the classical planner is to choose an optimal representation of the configuration space, with the smallest dimension and lowest complexity, in order to facilitate fast planning. Below, we will give a brief review for the various methods used for classical path planner. Details can be found in more comprehensive surveys such as [40,41].

Cell Decomposition Methods

The *cell decomposition* methods are the most popular approaches with wide applications in robotics. These methods discretize the configuration space into smaller convex polygons, called *cells*, and use path search methods to search through cells and find the optimal path to the goal. Sometimes, a regular grid is laid over the configuration space to break the space into cells, with predefined shape and size which are easier for path search but do not match the exact boundaries of the physical objects. This is called *approximate cell decomposition*, in contrast to *exact cell decomposition*, which generates cells exactly according to the physical boundaries. There is also *adaptive cell decomposition*, which starts with coarse grids and recursively refines the size for cells partially occupied until the resolution is good enough, thus requiring less

memory and processing time.

Roadmap Methods

The *roadmap method* fits the configuration space with *roadmaps*, i.e., graphs that contain *nodes* representing reachable robot poses, and *edges* which are one-dimensional curves, representing the free space between the nodes in accordance to certain topographical properties. A comparison between cell decomposition and roadmap methods is discussed in [125]. The key to the roadmap method is how to select the nodes in the graph so that the least number of nodes can be used to represent most of the characteristics of the configuration space. The graph search algorithms are employed to find the optimal path. Traditional roadmap methods to find the shortest path include *visibility graphs*, which connect the nodes of polygonal obstacles, and *Voronoi roadmap*, which uses Voronoi borders as edges. The *probabilistic roadmap (PRM)*, a recent advance in the roadmap method, generates feasible paths by random sampling from the entire configuration space and discards those associated with obstacle regions. It can solve path planning problems with higher dimensions and greater complexity, but has the drawbacks of slow convergence rate and potential non-optimal performance. PRM was applied to solve path planning under uncertainty in [60]. A variation of PRM, called *rapidly exploring random tree (RRT)*, was originally developed in [63]. The key idea of the method is to explore the configuration space rapidly by expanding search trees incrementally, instead of random sampling. Besides the capability of handling high-dimensional space, it is particularly well-suited for planning problems with differential constraints [64].

Potential Field Methods

The *potential field method* approaches the navigation problem from a completely different perspective. In lieu of graph searching in the configuration space, it assigns a mathematical function called the *potential field* to either the physical space or the configuration space so that the robot can move from higher-value locations to lower-value ones like the movement in a gravitational field. The function consists of

attractive forces exerted by the goal location and repulsive forces generated by the local obstacles. Early algorithms of this type include the *virtual force field* [13] that was designed for local obstacle avoidance. By integrating the methods of certainty grids used for obstacle representation and potential fields devised for navigation and by considering the entire path, one can achieve global planning and avoid being trapped in local minima [6]. A subset of the methods, called the *harmonic potential field method*, uses Laplace's equation to constrain the generation of a navigation potential function and thereby finds the potential path by solving a Laplace boundary value problem [25,69,100,128].

4.2.2 Path Planning under Uncertainty

Classical planning is generally carried out by robots in fully observable environments with conditions usually assumed to be deterministic and discrete. To incorporate uncertainty from various sources, probability-based frameworks with the aid of statistical tools such as the extended Kalman filter (EKF) and the particle filter (or sequential Monte Carlo) have been developed since the 1980's. One framework that solves the autonomous robot navigation problem under uncertainty is *Simultaneous Localization and Mapping*, or simply *SLAM*, which builds a map of the environment where the robot is located and localizing itself concurrently thus obtains the name. It has been introduced and discussed in Chapter 3. In recent years, the *partially observable Markov decision process (POMDP)* has emerged as a powerful framework for path planning, which has been discussed in Chapter 2. Since the POMDP framework tries to solve the planning problem by providing a general policy, it requires large computational effort and becomes increasingly difficult to carry out as the dimension of the problem scales up.

4.3 Local Navigation

A global path planner computes a geometric trajectory that passes around known static obstacles and reaches the goal. In comparison, local navigation follows the

global path and determines the next motion command based on local observations in real time in order to generate a new path by overwriting the original global path in response to regional change in environment such as new obstacles. When the environment becomes more dynamic or unknown in advance to the robot, the traditional path planner fails to cope with the obstacles that are "not in the plan". *Reactive navigation*, a local navigation framework that bridges path planning and sensor-based control and makes control decisions based on newly observed local conditions, is capable of dealing with unknown and dynamic scenarios and becomes imperative for robot navigation. In summary, global planning needs complete information of the environment in advance and plans the path before advancing, while local navigation stays reactive to the local environmental change in realtime and corrects predetermined global path on a local setting to avoid collision.

Global planning strategies have admittedly high computational cost, and it is not feasible to update or re-calculate the entire path. On the contrary, local techniques have low computational complexity, which is particularly important when the knowledge of the environment is updated frequently based on new sensor information. We describe here a number of obstacle avoidance techniques.

4.3.1 Obstacle Avoidance

Potential Field Method

The *potential field method* is a well known local path planning technique using the concept of *artificial potential fields* [57]. It has been discussed in section 4.2.1 as a global path planning method, but as already mentioned there, it was originally developed for local obstacle avoidance navigation. Basically, in a local regime, the robot's movement follows the gradient of a repulsive force field generated by the obstacles which push the robot away, in contrast to the goal location which attracts the robot with attractive potential force. While the potential field method has the advantage of relatively low computational cost, its disadvantage is the possibility of getting trapped in local minima. The *virtual force field method* [13] mentioned in section 4.2.1 integrates the concepts of both the potential fields and certainty grids,

but for local navigation, it has substantial shortcomings such as "instability and inability to pass through narrow passages like doors (local minima problem)" and the "repulsive forces from the both sides of the doorway" pushing the robot away [8].

Polar Histogram Methods

The *polar histogram methods* proposed by Borenstein and coauthors in the 1990's are purely local obstacle avoidance algorithms and provided an important improvement over previous techniques. The first approach of this type, called *vector field histogram* [14], constructs a one-dimensional polar histogram for the polar obstacle density in all directions calculated by using a certainty grids model, and then selects the moving direction with the lowest polar obstacle density. A subsequent improved version [121] was proposed to take into account the effects arising from the width and trajectory of the mobile robot, using a threshold hysteresis to reduce path oscillation, and executing better direction selection using a cost function. A further improved version [122] is capable of handling situations where purely local algorithms fail, with look-ahead verification performed by a heuristic search method in conjunction with appropriate cost and heuristic functions.

Velocity-Based methods

Another class of methods based on the admissible robot velocities consists of the *steer angle field* approaches [7]. An example is the *curvature-velocity method* [106], which works in the velocity space and assumes that the robot travels along arcs instead of having sharp turns. The curvature-distance to the obstacles are approximated to facilitate maximizing the objective function. The most representative algorithm of this class of methods is the *dynamic window approach (DWA)* [36] for static environment and later extended to multiple moving objects in unknown environment [23]. It searches a *dynamic window* constructed using attainable velocities within the next short time frame constrained by robot kinetics, including possible accelerations for feasible translational and rotational velocities, to maximize an objective function consisting of goal realization progress, forward velocity, and distance to obstacles. The

dynamic window is centered at the current velocity and is rectangle-shaped as a result of the independence of translational and rotational velocities. Like other local techniques, the DWA is also susceptible to local minima, so the *global dynamic window approach* [17] was developed to overcome this drawback by adding a global feature to the DWA. A global and minima-free navigation function based on a wave propagation technique labeling cells in the occupancy grid with distance to the goal is incorporated with reactive local DWA to form a local minima-free potential function. This procedure allows the robot to plan navigation trajectory with no *a priori* knowledge.

Relative Velocity Paradigm

The *relative velocity paradigm* [35], or *velocity obstacle approach*, transforms the absolute velocities of the maneuvering robot and moving obstacles to relative velocities of the obstacles with respect to the robot, and thereby converts the dynamic planning problem to a static one. The feasible velocity is chosen by avoiding the directions within the *absolute collision cone* associated with multiple obstacles. Some improvements [88] were made for applications to design robotic intelligent wheelchairs fully capable of transporting elderly and disabled people. Recently, a new method called *reciprocal velocity obstacle* [123] was proposed for multi-agent navigation. The novelty of this approach lies in the fact that, in addition to the relative velocity concept, it also assumes that all the agents make a similar collision-avoidance reasoning. It was proved to be able to execute safe navigation with collision-free trajectories for hundreds of agents densely populated in an environments with both static and moving obstacles. In addition, a subsequent refinement [8] was developed to navigate the robot at a constant speed towards the goal location as much as it can until an obstacle is close, and then to change the direction of the robot to avoid collision.

Nearness Diagram (ND) Navigation

A reactive navigation method called *nearness diagram (ND) navigation* [71,72] is based on the situated-activity paradigm of behavioral design that identifies situations and apply corresponding actions. More specifically, the paradigm gives guidance on

designing the navigation method on a symbolic level, and then uses the "divide and conquer" strategy to decompose the situations and solves them by determining associated actions. It can robustly achieve navigation in very dense, cluttered, and complex scenarios. Similar to ND navigation, the *fuzzy logic* navigation is also a type of behavior-based local planning methodology. It uses a set of linguistic fuzzy rules to deal with various situations and to imitate human reasoning without generating control commands by complex equations. More specifically, the fuzzy logic algorithm decomposes the problem into simpler tasks (independent behaviors), each of which has a set of fuzzy logic rule statements designed for the goal of achieving a predefined objective, and uses a command fusion to combine the outputs associated with various behaviors. Its robustness of dealing with variability and uncertainty is well suited for mobile robot navigation especially in unknown environments [93,129]. A layered design of fuzzy logic with primitive basic behaviors on one layer and the supervision on another layer has been proved successful in indoor robot obstacle avoidance navigation with the presence of static obstacles [34]. An adaptive fuzzy control system with a learning algorithm using a weighting scheme enables the robot to navigate in an environment with both static and dynamic obstacles without *a priori* map information and was tested on a robotic wheelchair equipped with laser sensor [70]. A dual fuzzy logic controller was designed to reduce the proneness to getting trapped in U-shaped obstacles [79].

Spline-Based Methods

In recent years, a series of spline-based algorithms were introduced for obstacle avoiding path planning [26,27,66]. These methods use splines to traverse the given points in the field and iteratively refine the path by adjusting and bending the spline to avoid obstacles, which leads to a collision-free path in real time in an unstructured environment [11,62,80,81,91,103,115]. The *elastic-band* framework [89] was proposed to model a collision-free path as an elastic band in the presence of artificial forces exerted by the obstacles; the elastic band can deform in shape to respond to newly detected changes in the environment such as unexpected and moving obstacles. A modified

version of this method [101] was introduced for path planning for cars with collision avoidance systems. It was also extended to dynamic path planning and obstacle avoidance for emergency situations in lane change maneuvers [47], vehicle-following system [38], and robot navigation in an intelligent environment with distributed wireless visual sensors [24].

4.4 Robotic Exploration

Several approaches have been proposed in the robotics literature to control the robot's movement along a path that minimizes the mapping and localization uncertainty. In this section, we consider three such approaches, namely, *coastal navigation*, *frontier-based exploration*, and *information-based exploration*.

4.4.1 Coastal Navigation

A POMDP-type method called *coastal navigation* [99] was inspired by traditional navigation of ships which travel along the coastlines to determine its location if direct localization tools are not available. The method aims at reaching the goal location with maximum probability by seeking the balance between traditional planners, which are tractable but lack robustness, and POMDP-type planners, which are reliable but have insurmountable computational complexity. An important idea of coastal navigation is to augment the state by adding to the original state S, which is the pose X consisting of the location (x, y) and direction θ, the uncertainty of the pose represented by the *entropy*

$$H(p_{\mathsf{X}}) = -\int_{\mathsf{X}} p(\mathsf{x}) log(p(\mathsf{x}))d\mathsf{x}. \tag{4.1}$$

Thus the augmented state becomes $S = (x, y, \theta, H(x, y, \theta))$. This essentially models the uncertainty of the robot's position as a state variable. After modifying the transition rule and Bellman's equation accordingly for the particular application, value iteration is employed to search the augmented pose-uncertainty state space in order to generate trajectories. In order to avoid intractability like POMDP, it makes a

crucial assumption that may be too strong in many applications: the conditional probability distribution of the robot's pose is Gaussian with its mean being the current pose, which is equivalent to compressing a multi-dimensional belief space to a one-dimensional space. By incorporating the uncertainty into the state, it enables the robot to find the path that minimizes the positional uncertainty so that it is less likely to get lost during the navigation process. A limitation of the method is that it assumes a known map of the environment.

4.4.2 Frontier-Based Exploration

The frontier-based exploration, proposed in [127], centers around the idea that in order to gain the most information about the partially known world, the robot should move to the successive frontiers, or boundaries between known open space and unknown territory. The strategy is to make the robot navigate to the nearest reachable, unvisited frontier, update the information of the new surroundings, and then navigate to the next nearest reachable and unvisited frontier, and to repeat the process. The planner is a grid-based navigation system using the so-called *evidence grid*, and employs a depth-first graph search to find the closest collision-free path to the destination, using an obstacle avoidance routine to prevent the robot from collision with an obstacle not pre-determined in the grid map. For each destination, it will either be reached then classified as *visited*, or not accessed within a certain time frame and denoted as *inaccessible*.

4.4.3 Information-Based Exploration

As the task of exploration is to gather information of the environment, it is essential to choose a metric to measure the information. The control rule is to navigate the robot in a path that will minimize the chosen information metric. One popular metric is *entropy* [15,104,105,113], as defined in Equation (4.1) with the pose X replaced by the system state $\xi_t = (\mathsf{X}_t, \mathbf{m})$ combining both X_t, the robot pose at time t, and $\mathbf{m}$, the positions of the map features. For SLAM, the probability distribution of the state is conditioned on past observations $z_{0:t}$ and actions $u_{0:t-1}$, so the probability p_ξ in the

entropy definition becomes the posterior distribution $p_{\xi_t|z_{0:t},u_{0:t-1}}$. For the EKF SLAM, the linearity and Gaussian noise assumption leads to Gaussian approximation of the posterior: $p_\xi \sim N(\mu, \Psi)$ [105]. Maximizing the information about a state estimate is equivalent to minimizing the determinant of the state variance $det(\Psi)$. This leads to the *D-optimality* as defined in the optimal design theory. In addition, the determinant of the square covariance matrix is equal to the product of its eigenvalue λ_i's. Thus, the *D-optimality* criterion is equivalent to minimizing $\prod_{i=1}^{n} \lambda_i$. In contrast to *D-optimality* where the product of the eigenvalues of the covariance matrix is used, *A-optimality* uses the sum of the eigenvalues, or equivalently, the trace of the covariance matrix $trace(\Psi)$ to measure the average uncertainty of the model [105].

4.5 Integrated Navigation under Uncertainty in an Unknown Environment

We briefly summarize some common limitations of current approaches, and then introduce a framework to circumvent them in navigation problems and to integrate global planning with local navigation and exploration.

4.5.1 Limitations of Current Approaches

Discrete vs Continuous, and Linear vs Nonlinear

Most of the work done so far in the three aspects of the navigation problems described above requires some discretization in the state, action, or observation space. Not much has been done when all three spaces are continuous. A relatively simple case concerning only global navigation with no obstacle and the mapping and localization of the robot has been considered in [96]. For exploration, the Global-A optimal approach [105] was tested on a simulated environment of a 200×200 grid. As the real world is continuous, how to shift from discrete environments to continuous ones is one of the key challenges yet to be resolved. Due to the daunting mathematical challenge imposed by nonlinear systems, most of the work has been focused on linear systems

or on linearizing nonlinear systems. The particle filter can be used for nonlinear systems, but its full potential has not been realized.

Computational Complexity, and Off-line vs On-line

The POMDP framework is a powerful tool to solve the robotic navigation problem under uncertainty, but has intractable computational complexity as the dimension of the problem scales up. Most solutions are not applicable to scenarios with over 1000 states [82]. Although Du et al. (2010) have been able to handle somewhat large number of states within reasonable time, their method is still not able to really solve real-world problems that have larger state spaces.

The majority of the path planning algorithms are off-line algorithms, i.e., they compute the plan before the robot starts moving. This is one of the biggest drawbacks of the traditional planning algorithms because of its lack of consideration for the dynamics of the environment, which requires the robot to navigate reactively and plan on-line. On the contrary, on-line algorithms, which are the results of more recent research, "generally consist of a look-ahead search to find the best action to execute at each time step in an environment" [97], and are therefore capable of adjusting to the environment adaptively. Much work still needs to be done for on-line algorithms in order to track the changing dynamics of the real environment.

Uncertainty

Another limitation of current global and local navigation algorithms is that uncertainty is often not taken into account. In other words, full observability of the robot's environment by perfect sensing with no uncertainty is usually assumed. In addition to that, the condition of known robot location is imposed. This is the case for most of the classical planners, and also for most of the obstacle avoidance routines. SLAM does address the problem of uncertainty and is capable of both mapping and localization, but it does not generate control signals for navigation and is not concerned with obstacle avoidance. The POMDP framework takes into account the uncertainty too, but it only applies to global planning and does no mapping. How to incorporate the

uncertainty into the broader navigation framework with all three aspects involved is an important and challenging problem in robotic navigation research.

Prior Knowledge

Almost all the papers addressing the path planning problem assume *a priori* knowledge of the map, i.e., the robot knows about the terrain in advance. But this seldom holds in practical applications, such as planetary exploration like Mars Rovers. Therefore a robust path planner applicable to unknown environments and adaptable to environmental changes is yet to be developed for future autonomous robot navigation.

4.5.2 New Framework

As we discussed earlier, there are three aspects of the robotic navigation problem: global planning (reaching a goal), local navigation (obstacle avoidance), and exploration (SLAM and handling uncertainty), as shown in Figure 4.1. These three subfields have their own constraints and solution frameworks, although they are closely related to each other and coexist in real-world problems. Past research effort has been focused on problems concerning one of these three subfields, as indicated by I, II, and III in the figure. Some work has been done to add global thinking to the local navigation [17], covering the area IV in the figure, but discretization of the environment is required. Martinez-Cantin et al. (2007) have tried to solve planning and exploration under uncertainty, but they only considered three landmarks that lead to a six-dimensional optimization problem, and ignored local navigation [68]. Little effort has been devoted to explicitly address the problems involving all the three subfields, i.e. VII in Figure 4.1, which has a lot of real-world applications. For example, ground rescue mission requires accurate mapping of the terrain and precise localization of the rescue unit, when positioning systems are not available, in order to reach the target location within minimum time. Planetary rovers such as the Mars rovers have limited time for each mission, so they need to get as much information about the planet surface as they can within a given time constraint, while trying

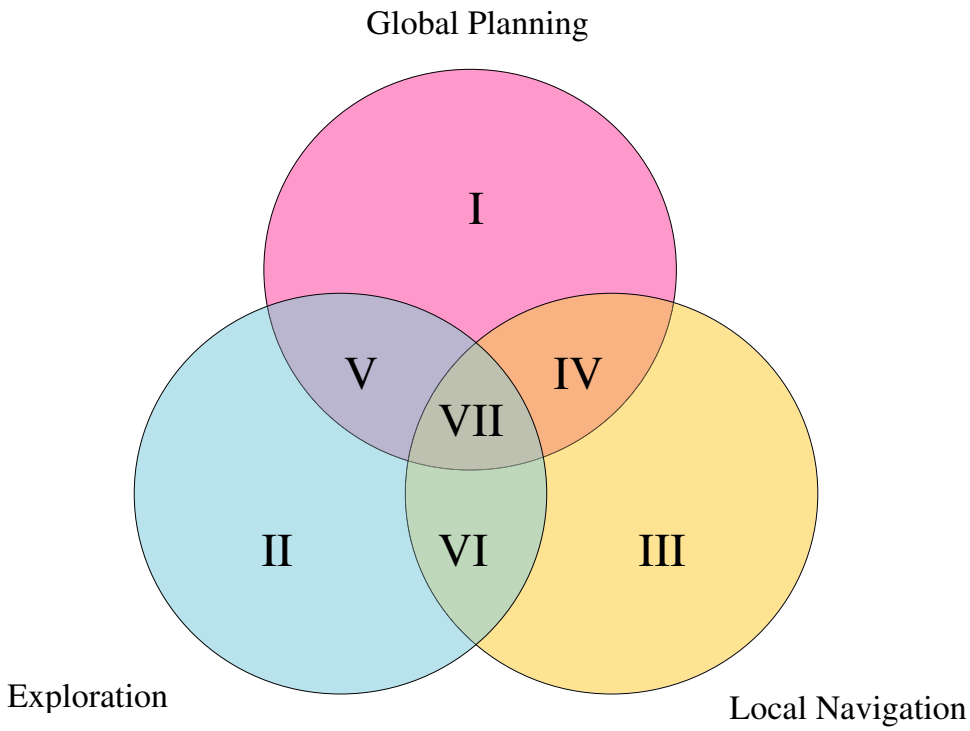


Figure 4.1: The diagram shows the relation among the three subfields: global planning, local navigation, and exploration. The goal is to solve problems within the area covered by all three subfields as indicated by VII.

to reach the predetermined goal location. Here, we provide a framework for solving the sequential planning problem concerning global planning, location navigation and exploration in the presence of uncertainty in a continuous environment.

4.5.3 Problem Statement

We consider the following navigation problem: in a continuous environment with feature-based landmarks, plan a trajectory for a robot with continuous action control and no *a priori* knowledge of the environment, so that it can minimize the traveling time to the goal location, while accomplishing other tasks such as mapping the landmarks seen along the way, localizing itself, selectively exploring local landmark-dense

area, and actively avoiding collision with the landmarks. Formally, we can define this as the following control problem:

$$\begin{aligned} \min \quad & J(\mathbf{x}, \mathbf{m}, \mathbf{u}, t) && (4.2)\\ \text{subject to} \quad & h(\mathbf{x}, \mathbf{m}) > d_c && (4.3)\\ & \dot{\mathbf{x}} = g(\mathbf{x}_{t-1}, \mathbf{u}_t) && (4.4)\\ & \mathbf{x}_{t_0} = \mathbf{x}_0 && (4.5) \end{aligned}$$

where $\mathbf{x}$ is the robot pose, $\mathbf{m}$ is the map estimation, $\mathbf{u}$ is the action, $J(\cdot)$ is the cost function, i.e., the total traveling time in our case, $h(\cdot)$ is the environmental constraint function that monitors the distances between the robot and all the landmarks in the map so that they will not fall below the critical range d_c which will potentially cause collision, $g(\cdot)$ is the physical motion constraint function of the robot, and $\mathbf{x}_0$ is the initial condition.

Compared to problems considered in the literature, the new problem involves continuous spaces (for state, action and observation), and requires exploration and local navigation (obstacle avoidance) on top of the global navigation which is the main goal. Note that throughout this section, the landmarks are just the obstacles, thus the two words are synonymous.

4.5.4 System Control

The diagram of the system control block is shown in Figure 4.2. Basically, the system switches between three modes: the global navigation mode, the obstacle avoidance (OA) mode, and the exploration mode, depending on the specific condition. When a new obstacle is observed within a certain range, the OA routine is launched to achieve a local sub-goal, which is to avoid collision with the obstacle and to pass around it. If the number of close landmarks exceeds a certain threshold, the robot will enter the local exploration mode to gather more information about the surroundings, with the immediate goal set to find a safe path to navigate out of the landmark-populated region. The SLAM filter will be used during exploration to carry out

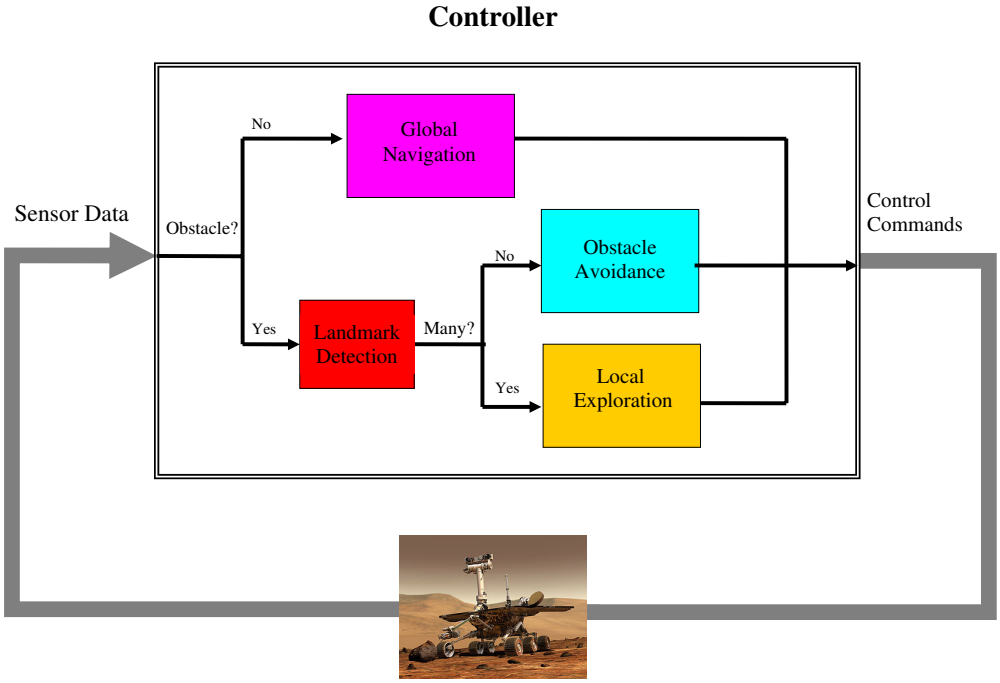


Figure 4.2: An illustration of the control block to realize global path planning under uncertainty constraints while ensuring robust local obstacle avoidance navigation.

better estimation of the robot pose and map locations, and will be kept running in background for other modes. If there is no nearby obstacles, the robot will be set in the global navigation mode to optimize the global planning objective, i.e., traveling towards the goal location as quickly as possible. Since collision avoidance is critical in protecting the robot, there is an emergency routine used to navigate the robot away from any landmark that is too close to the robot, in order to prevent collision. This routine is embedded in both the OA mode and the exploration mode. It has the highest priority once it is activated, and can overwrite the commands from the other modes. The control signal is dependent on robot dynamics including the velocity and orientation of the robot relative to the landmark, and safety constraints, i.e. whether or not the robot is within the *safe region*, where the robot is free to travel at any

speed, the *slow-down region*, where the robot must travel at a lower speed due to its close distance to the landmark, or the *dangerous region*, where the emergency routine will be turned on to lead robot to a safer place. The definition of these regions will be talked about in details in Section 4.5.6.

The myopic control, which is an one-step special case of receding horizon control (RHC), is used for both global and local navigation, because, as discussed in Chapter 2, the myopic policy can perform on the similar level to other complicated control strategies, while incurring significantly less computational cost. When it is in the exploration mode, the robot temporarily forgets about the global goal, and focuses on finding a collision-free path in order to get out of the landmark-dense area, as opposed to the obstacle avoidance mode, in which the robot still remembers the global goal, thus will choose the myopic action that will not only most effectively avoid landmarks, but also lead the robot towards the target location.

These techniques will be discussed in more depths in the following sections. Although described separately, they are all integrated and interweaved together to achieve the global goal, which is to arrive at the target location as quickly as possible with no collision in a continuous environment.

4.5.5 Global Navigation

In Newtonian mechanics, we know that the traveling time is defined by

$$T = \int dt = \int \frac{d\vec{\mathbf{S}}}{\vec{\mathbf{v}}} \tag{4.6}$$

where $\vec{\mathbf{S}}$ is the *displacement vector*, and $\vec{\mathbf{v}}$ is the *velocity vector*.

If we adopt the spherical coordinate centered at the goal location and project the vectors in the radial coordinate, we can have

$$T = \int \frac{dS_r}{v_r} \tag{4.7}$$

This means that if we want to minimize the total traveling time T, the myopic control rule will choose the speed and angle such that the robot's instant velocity component

pointing to the goal location, i.e., the radial component of its velocity, is maximized at that moment.

4.5.6 Obstacle Avoidance

As the robot gets close to a landmark-sparse region, the robot will turn on the reactive obstacle avoidance (OA) routine to pass the landmarks. In this section, we focus on how to control the robot to avoid collision when there is one close landmark, while the multiple-landmark case will be discussed in the section on point-based exploration. For a landmark-dense region, the OA routine will get into trouble, as the robot will not be able to choose the correct control action, simply due to the complexity of the local environment. In this case, the robot needs to explore the local area and get more accurate information about the location of the landmarks in order to find a safe path to get out. This is the exploration mode, which will be discussed in Section 4.5.7.

We adapt the situated-activity paradigm of design method, which was introduced to collision avoidance literature in [71,72], to illustrate our OA routine design. The overall control flow for the obstacle avoidance routine is shown in Figure 4.3. We will explain in details each of the situations that the robot will encounter during the process, and then talk about the associated actions to take to avoid collision. Please note that, in the OA routine, the global goal is still in consideration. In other words, for example, if there are a few control options equally good for obstacle avoidance, we will choose the one that will best achieve the global goal among the available options.

The Situations

For each landmark, we define three regions around it. For an area with distance larger than d_f, where d_f is the *minimum distance for free navigation*, we call it *safety region*. For area with distance smaller than d_c, where d_c is the *critical distance* to avoid collision, we call it *dangerous region*. The area between these two regions are called *slow-down region*, i.e., the robot should be vigilant about the close landmark and not navigate full speed in this region. These regions have been illustrated in Figure 4.4.

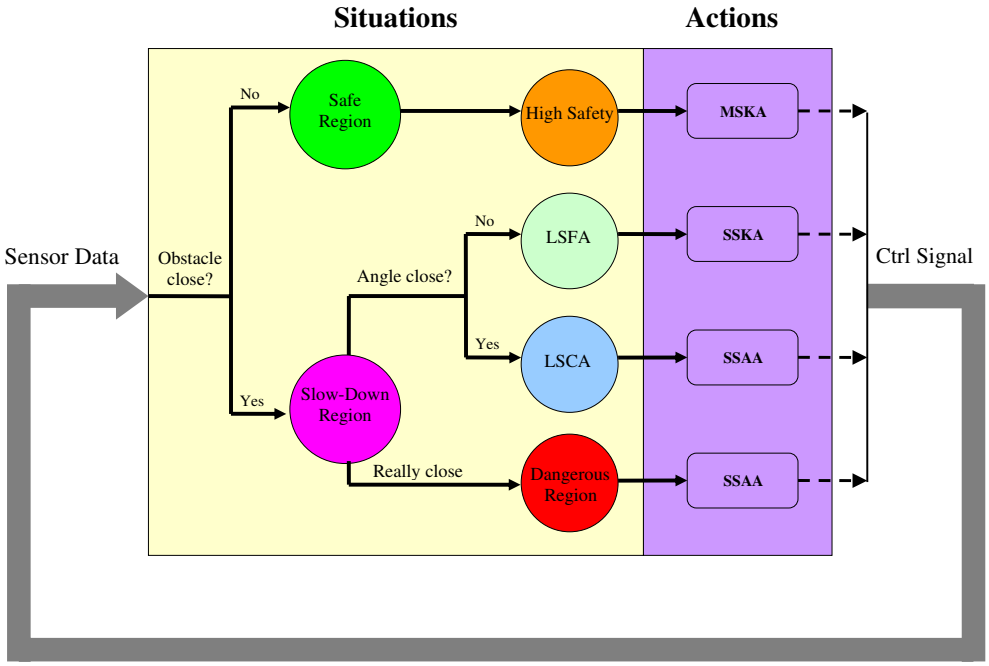
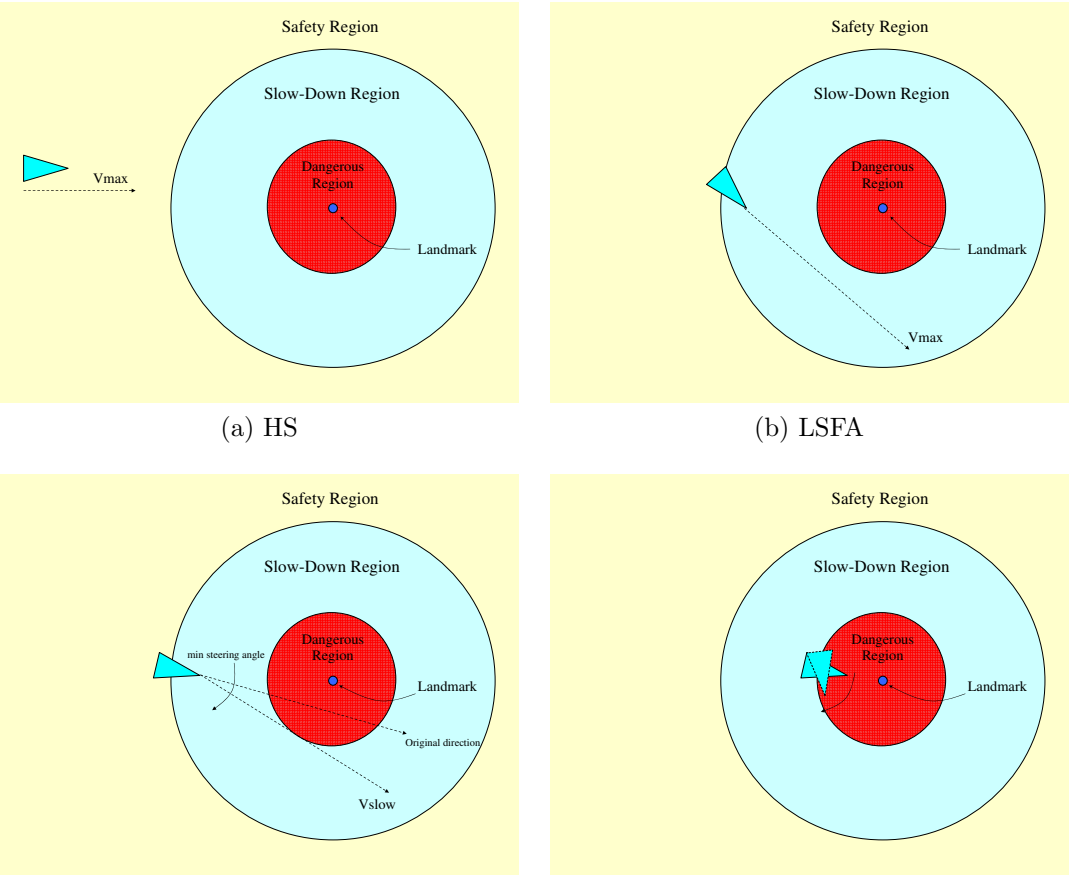


Figure 4.3: The obstacle avoidance diagram illustrates situations the robot will encounter during local navigation and the corresponding actions to take during the course.

As shown in Figure 4.3, there are totally four situations, which will completely cover all the possibilities of the interaction between the robot and the landmark, based on the physical parameters such as distance and angle.

High Safety (HS) Situation When the robot is within the safety region, as shown in Figure 4.4a, the robot is far from the nearest landmark, thus can move at full speed without worrying about collision.

Low Safety Far Angle (LSFA) Situation As the robot moves close to a landmark, it enters the slow-down region. If the forward trajectory of the robot is pointing outside the dangerous region, as shown in Figure 4.4b, the robot does not have the risk of entering the dangerous region of the landmark. We define the *critical*



(a) HS (b) LSFA

(c) LSCA (d) DR

Figure 4.4: (a) High Safety (HS) situation. The robot maximizes its speed and keep its angle towards the global goal. (b) Low Safety Far Angle (LSFA) Situation. The robot slows down but keeps its original orientation. (c) Low Safety Close Angle (LSCA) Situation. The robot slows down and adjust its direction to at least bypass the landmark at the critical angle. (d) Dangerous Region (DR) Situation. The robot slows down to its minimal speed and turn as sharply as it can to steer away from the landmark.

angle, which is tangible to the edge of the dangerous area from the robot's location, as

$$\eta_c = \arcsin\left(\frac{d_c}{d}\right) \tag{4.8}$$

where d is the distance between the robot and the landmark and d_c is the *minimum*

safety distance or *critical distance* already defined earlier. Similarly, for the specific landmark, we define the *angle* of the robot with respect to the landmark η as the angle between its current direction and the connected line between the robot and landmark. When $\eta < \eta_c$, we say the angle η is *close*. Otherwise, we say it is *far*. Note that both η and η_c are landmark-dependent.

Low Safety Close Angle (LSCA) Situation This situation is similar to the LSFA one, but with the difference that the forward trajectory of the robot is pointing inside the dangerous region, as shown in Figure 4.4c. In other words, if the robot keeps moving in the current direction, it will go into the dangerous region of the landmark and may cause potential collision. In this situation, the angle η is smaller than the critical angle η_c, thus the robot is moving alone a path close to the landmark. We call this a *close angle*.

Dangerous Region (DR) Situation When the distance between the robot and the landmark becomes smaller than the critical distance d_c, the robot is within the dangerous region and has high risk of collision. See Figure 4.4d.

The Actions

For each situations, the action associated with the corresponding situations provides the control signals including speed and steering angle to command the robot to avoid collision and move towards the goal location. As myopic rule is adapted as the control policy, the control commands are designed accordingly.

HS Situation - Maximize Speed & **Keep Angle (MSKA)** The robot can move at full speed and keep its direction towards to global goal location, as shown in Figure 4.4a. Note that it doesn't matter whether or not the forward trajectory will go through the dangerous region of the landmark.

LSFA Situation - Slow-Down Speed & **Keep Angle (SSKA)** Since the forward trajectory of the robot is pointing outside the dangerous region, as shown in Figure 4.4b, the robot does not have to change its course of direction. But it does need to slow down to navigate carefully in order to pass the landmark.

LSCA Situation - Slow-Down Speed & **Adjust Angle (SSAA)** Besides slowing down the speed in this situation, the robot should also adjust its angle to avoid entering the dangerous region in the near future. The minimal angle to turn is the difference between the current angle and the critical angle, as shown in Figure 4.4c, in order to let the robot bypass the landmark safely.

DR Situation - Slow-Down Speed & **Adjust Angle (SSAA)** Similar actions as those for the LSCA situation, with the differences that, as the robot just enters the dangerous region, it will slow down to the mimimal speed it can achieve, much slower than the speed taken in the LSCA situation, and turn the maximal angle it can possibly turn, in order to steer away from the landmark as soon as possible. This is the *emergency routine* and has been illustrated in Figure 4.4d.

4.5.7 Point-Based Exploration

In Section 4.5.6, we discussed the obstacle avoidance routine for the case in which the robot encounters a landmark. In this section, we will talk about the case in which the robot enters a landmark-dense region, as opposed to the landmark-sparse region in the previous section.

In landmark-dense local area, the robot needs to slow down and explore the surroundings, in order to determine a safe path free of landmarks and navigate out of the region. During the exploration process, the robot will put identifying local landmark locations as the top priority, instead of the global goal. In other words, the robot will temporarily forget about the global goal to focus on local exploration. This means that the robot may even move in directions that may temporarily lead the robot further away from the goal under extreme situations in order to navigate safely out of the obstacle-occupied region.

Due to existence of measurement uncertainty and also the robot system uncertainty, the FastSLAM algorithms discussed in Chapter 3 are utilized to estimate the landmark locations and the robot pose. Please note that by FastSLAM we mean a class of particle-based SLAM algorithms, including FastSLAM 2.0, Uncented FastSLAM, and Spherical Simplex Unscented FastSLAM.

Entering Exploration Mode

If c landmarks are observed to be close, i.e., within d_f distance from the robot, the angles of the robot η^i with respect to each landmark will be calculated. If no angle is close, i.e., smaller than the critical angle η_c^i associated with the ith observed close landmark, the robot can keep on moving in the original direction at its maximum speed. If there are n close angles, two cases will be considered.

First, if $n \leq n_{th}$, where n_{th} is a predetermined threshold with some small integer value, it means the number of landmarks that are blocking the way is relatively small. In this case, we can calculate the steering angle for each landmark, and then choose the one closest to the direction of the global goal location.

Then, if $n > n_{th}$, it means there are quite many landmarks which are close in both distance and angle to the robot. In this case, the robot will enter the exploration mode, setting the top priority to be exploring the local environment. It will estimate the local landmark locations as accurate as possible, in order to find a safe path to get out of this landmark-dense region.

Spatial Testing Points

Once the robot enters the exploration mode, the SLAM routine will be put on spot and its estimation of the robot pose and landmark locations will be closely monitored. For operational modes other than exploration, the SLAM filter can be kept running in background. As the goal of the exploration is to find a safe path, we define a set of so-called *Spatial Testing Points (STP)*, at which we will calculate the probability of existence of any landmarks. Before showing the formal definition, we will first carry out some mathematical calculations to facilitate the definition.

In the FastSLAM algorithm, we assume N particles, with each particle containing the robot pose and means and covariance matrices of M landmarks, as defined in Equation 3.21. Here we present again the definition of each particle:

$$\mathbf{X}_t^k = \langle \mathbf{x}_t^k, \mu_{1,t}^k, \Sigma_{1,t}^k, ..., \mu_{M,t}^k, \Sigma_{M,t}^k \rangle \tag{4.9}$$

Then the mean of the robot pose at time t will be

$$\bar{\mathbf{x}}_t = \sum_{k=1}^{N} \mathbf{x}_t^k \cdot w_k \tag{4.10}$$

where w_k is the weight for the kth particle. For the ith close landmark, we assume the distance measured is r_i. Then, we give the definition for the STPs:

Definition (*Spatial Testing Points*). Given c close landmarks detected and n out of c are close in terms of both distance and angle, if $n > n_{th}$, a set of *Spatial Testing Points* at time t are defined on c semicircles centered at the estimated robot location $(\bar{x}, \bar{y})$, with the points on the same semicircle separated equally by δ radians and the radius equal to the distance between the robot and the associated landmark r_i. Mathematically, the jth point on the ith semicircle, $\mathbf{x}_{stp}^{ij} = (x_t^{ij}, y_t^{ij})$, is defined as

$$x_t^{ij} = \bar{x}_t + r_i \cdot \cos\left((j-1)\delta - \frac{\pi}{2}\right) \tag{4.11}$$

$$y_t^{ij} = \bar{y}_t + r_i \cdot \sin\left((j-1)\delta - \frac{\pi}{2}\right) \tag{4.12}$$

The c testing points lined up on the same angle $(j-1)\delta - \frac{\pi}{2}$ forms the jth *spatial testing path*.

The spatial testing path provides a safe direction for the robot to move in the next time step. Due to the discrete nature of the definition of such path, we can increase the density of the testing points, i.e., decrease δ, in order to achieve more subtle steering angle selection. But this comes at a cost of larger computational complexity. In the next part, we will talk about how to use these testing points to find a safe path.

Safety Zone Testing

For each testing point, we impose two criteria, the *safety confidence* and *safe distance*. The first criterion takes into account the effect of uncertainty and calculates the probability density of having no landmark (HNL) at a specific testing point from the posterior distribution estimated by SLAM filter. The second criterion evaluates

the distance between the testing point and the landmarks, based on the estimated landmark locations. A path can be classified as safe for time t only when the joint probability density of HNL of each testing point on that path is higher than a given threshold p_{th} and the distances from each testing point on the path to all the landmarks are larger than d_c, the critical range or equivalently the radius of the dangerous zone. Please note that whether a short-term path is safe is time-dependent, because the status will change as new information is gathered. Below, we will elaborate the tests in more details.

Assuming data association relation $j(i)$, we can obtain the index of the ith close landmark has index $j(i)$ in each particle. This index $j(i)$ can also be regarded as the overall index of the landmark among all the landmarks, or the *identifier* of the landmark. Note that data association isn't the main topic of this dissertation, so we will skip the details related to data association.

For the lth testing point on the qth semicircle, which we simply call the (q,l)th testing point, the probability density of having ith close landmark in the kth particle can be calculated using a 2D multivariate Gaussian density function:

$$p^{ql,k}_{j(i),t} = (2\pi)^{-1}|\Sigma^k_{j(i),t}|^{1/2} \exp\left((\mathbf{x}^{ql}_{stp} - \mu^k_{j(i),t})^T(\Sigma^k_{j(i),t})^{-1}(\mathbf{x}^{ql}_{stp} - \mu^k_{j(i),t})\right) \tag{4.13}$$

The weighted average of such probability from the N particles can be computed as

$$p^{ql}_{j(i),t} = \sum_{k=1}^{N} p^{ql,k}_{j(i),t} \cdot w_k \tag{4.14}$$

Because the landmarks are independent given the robot pose, the probability of having no landmark at the (q,l)th point is

$$p^{ql}_t = \prod_{i=1}^{c}\left(1 - p^{ql}_{j(i),t}\right) \tag{4.15}$$

where c is the number of landmarks observed to be close, i.e., within d_f distance from the robot, as defined earlier. Note that we only consider those landmarks close in range, which can save computational time and is also reasonable due to the fact that

those far landmarks are almost impossible to have impact on the local path selection.

After introducing the first criterion, the safety confidence represented by p_t^{ql} for the (q,l)th testing point at time t, we now talk about the second criterion, the safety distance. First, the mean of the ith close landmarks at time t can be calculated by the weighted average of the mean in each particle obtained by Kalman filters (EKF, UKF, and etc):

$$\bar{\mathbf{x}}_{j(i),t}^{f} = \sum_{k=1}^{N} \mu_{j(i),t}^{k} \cdot w_k \tag{4.16}$$

Then the safety distance between the (q,l)th testing point and ith close landmarks can be computed by

$$d_{j(i),t}^{ql} = \sqrt{(\bar{x}_{j(i),t}^{f} - x_t^{ql})^2 + (\bar{y}_{j(i),t}^{f} - y_t^{ql})^2} \tag{4.17}$$

For the (q,l)th testing point at time t, it can be categorized as safe for the time being if

$$p_t^{ql} > p_{th} \quad \& \quad d_{j(i),t}^{ql} > d_c, \quad \text{for all } i \in 1, ..., c \tag{4.18}$$

In other words, if the testing point has high probability density of having no landmark and is far from all the landmarks. Then, the lth spatial testing path will be declared as safe at time t, if all the c testing points on it are safe, i.e., Equation 4.18 holds for all $l \in 1, ..., c$.

4.5.8 The Integrated Navigation Algorithm

We now integrate the previous sections and present a complete algorithm to solve the problem proposed in Section 4.5.3. The integration is based on the control block introduced in Section 4.5.4.

For each landmark, there is a critical angle η_c given the robot's location when the robot is close, i.e., within d_f distance (slow-down region) from the landmark. Note that for all control commands, the steering angle change in one time step is limited by the maximal angular speed and the maximum steering angle the robot can achieve.

The Integrated Navigation Algorithm

- Initialization
- While the goal is not reached yet

  Assume the current steering angle pointing to the goal is $G = G_g$

  If no landmark observed,

  we set $V = V_{max}$, and keep G.

  else (there are landmarks observed)

  if no close landmark within d_f,

  $V = V_{max}$ and keep G.

  else (there are close landmarks)

  Case 1: all close landmarks have far angles (outside critical angles.)

  $V = V_{max}$ and keep G.

  Case 2: some close landmarks also have close angles ($\eta < \eta_c$.)

  slow down: $V = V_{medium}$

  n = number of landmarks within d_f and with $\eta < \eta_c$

  if $n \leq n_{th}$,

  calculate steering angle G_i for each landmark

  choose the one closest to G_g, the goal direction.

  else ($n > n_{th}$, enter local exploration mode!)

  set up *spatial testing points*

  calculate safety confidence for each testing point

  calculate safety distance for each testing point

  find the safety path

  if multiple directions are found, choose the one closest to G_g.

  For any case above, if robot enters the dangerous region of any landmark

  overwrite above control commands and set $V = V_{slow}$ and $G = G_{max}$

  to turn away from the landmark slowly.

4.6 Experimental Results

In this section, we will demonstrate the idea of integrated navigation proposed in the previous sections in a simulated environment, where a robot tries to reach a designated goal location in an unknown and landmark-populated environment under system and measurement uncertainties.

4.6.1 Experimental Setup

The robot is a triangle-shaped vehicle with length of $4m$ and width $2m$. Its speed ranges from $1m/s$ to $10m/s$. The medium and slow speeds in the speed limit cases are $6m/s$ and $3m/s$ respectively. For the demonstration purpose, we set the maximum steering angel to be $80\pi/180$ radians, and the maximal steering angular speed to be 2π/s. The control signal is carried out every $0.025s$, which is set to be the time for one time step. The laser measurement is executed every 8 time steps, and can reach area as far as 30 meters. The radius of the slow-down region for each landmark, d_f, is set to $10m$. The critical distance d_c is set to $3m$. Note that the robot only observes the landmarks within the semicircle range in the forward moving direction. For the SLAM filters, we use 100 particles in the simulation. The minimum number of effective particles ratio to trigger resampling is 0.75. The control signals are executed every 0.025s, and the measurements are carried out every 8 control time steps, i.e., every $0.2s$.

4.6.2 Result and Analysis

Figure 4.5 shows the process of how the robot navigate in the environment especially during the exploration mode. In Figure 4.5a, the robot has entered the exploration mode after observing several landmarks close in terms of both distance and angle, as indicated by the red circles in the figure. The spatial testing points are drawn in the plot in pink dots. After a series of calculations discussed in Section 4.5.7, the robot determines the three discrete directions roughly pointing to the south are safe. It chooses the third one from the left as it's the closest to the goal orientation. The red

dashed line indicates the actual steering angle, as the control signal is corrupted by system noise. Then, the robot turns and navigates along the landmarks as shown in Figure 4.5b. Up to some point, the robot sees a set of new landmarks that are close in terms of both distance and angle, as show in Figure 4.5c. This time, the robot figures there are two sets of discrete directions are safe, one towards the southwest, and one towards the northeast, which is pointing through the gap between the landmarks. Then, the global goal comes into play and leads the robot to decide on steering towards the direction indicated by the red dashed line. Eventually, the robot turns, and navigate through the gap and goes towards the goal location. The final trajectory the robot takes to reach the goal location is shown in Figure 4.6, with the simulation results shown in Table 4.1. Figure 4.7 and Figure 4.8 show two different scenarios from Figure 4.6: the landmark configuration in Figure 4.7 has slightly wider gap in the middle, and that in Figure 4.8 has a much wider gap. The robot in 4.7 still chooses the gaps located in the lower half of the plot, as the middle gap is not wide enough to be considered as a safe path. But in Figure 4.8, the robot went through the middle gap, as it determined that the gap was wide enough to form a safe path, thus it could avoid detours such as going through the gap below.

Quantities to Evaluate	Value
Total Traveling Time (s)	10.28
Total Distance (m)	63.20
MSE of Robot Path Estimation	0.0163
MSE of Landmark Estimation	0.0120

Table 4.1: Results of Integrated Navigation.

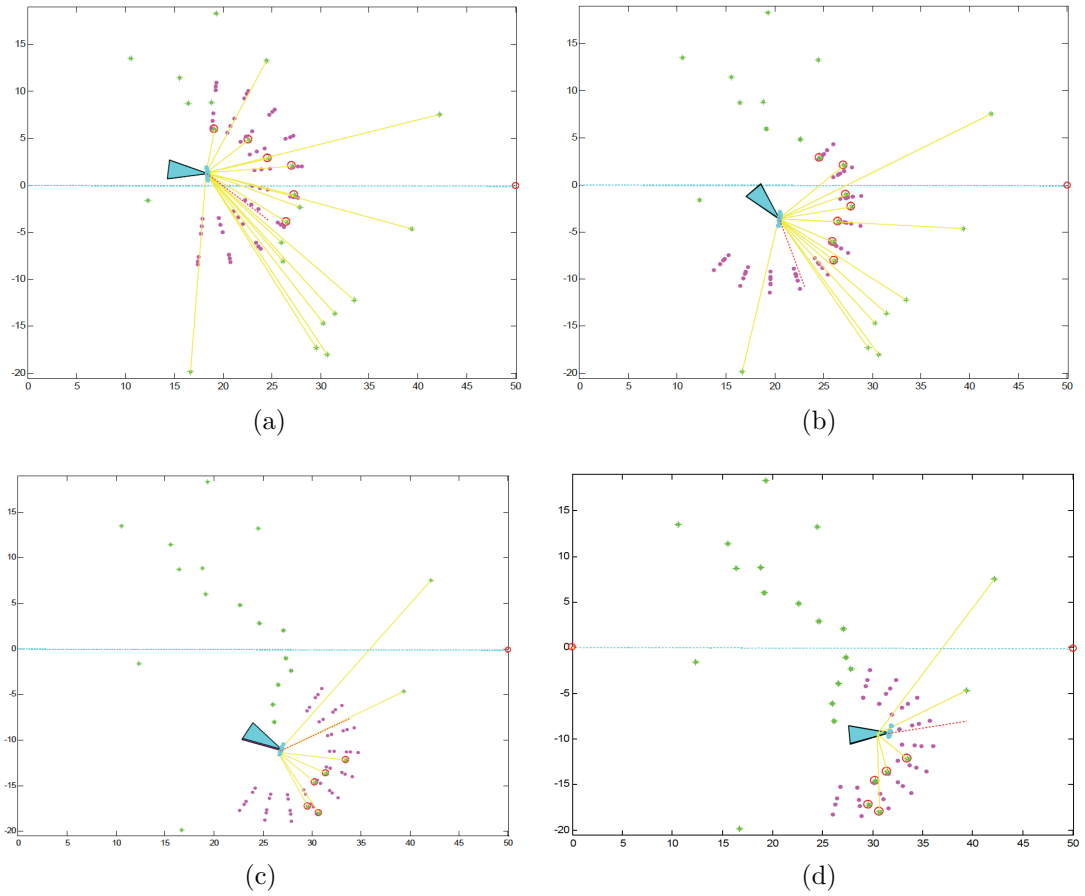


Figure 4.5: In all plots, the green stars indicate the landmarks. The red circle around the landmarks indicate those landmarks that are close in range. The red dash straight line from the robot front end points to the steering direction. The pink dots around the robots are the spatial testing points, whose density can be adjusted. (a) Robot explores and tries to circumvent the landmarks. (b) Robot further moves and passes along the landmarks. (c) Robot meets new cluster of landmarks, but see a gap that can go towards the goal. (d) Robot turns and goes through the gap.

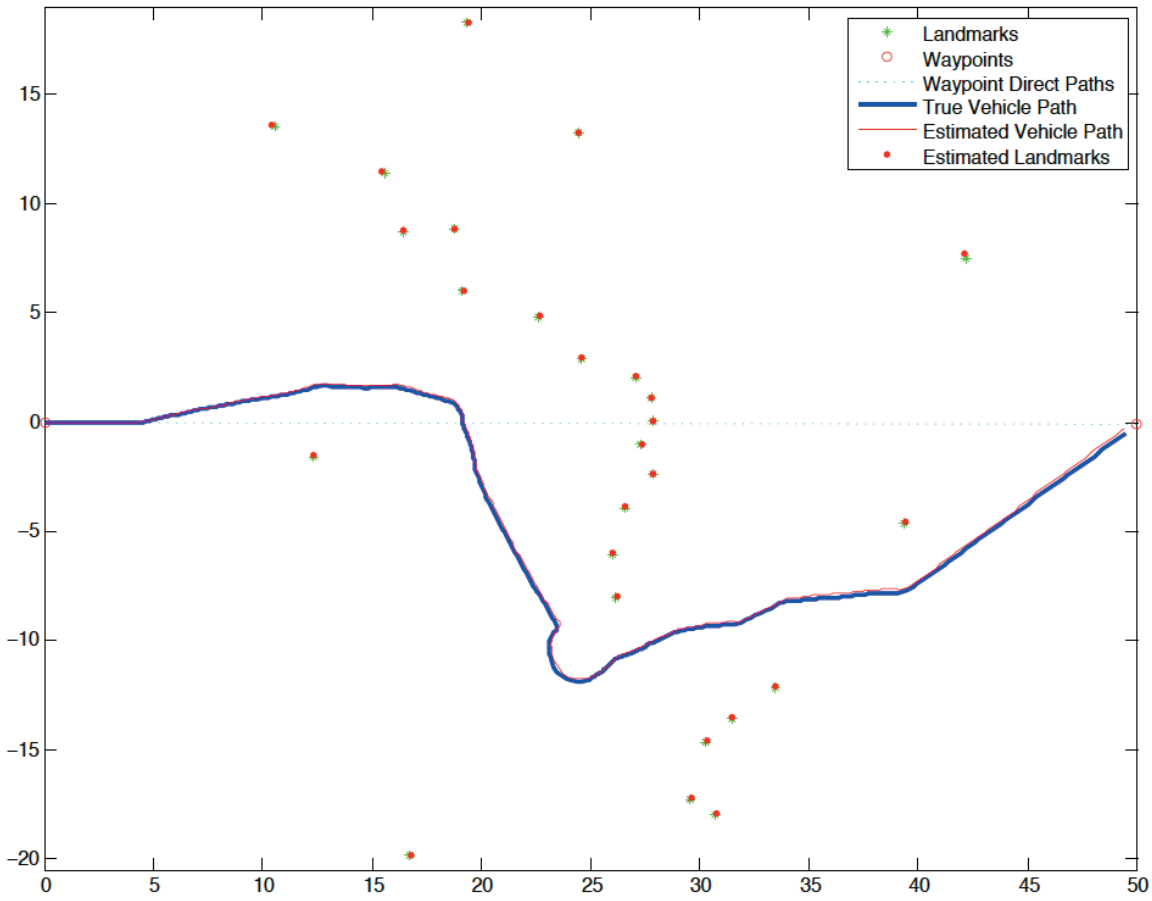


Figure 4.6: The final path of the robot. The landmarks located in the middle part of the map are so dense that there is no chance to find a safe path across them. The robot finds a safe path and goes through the big gap shown in the lower half of the plot.

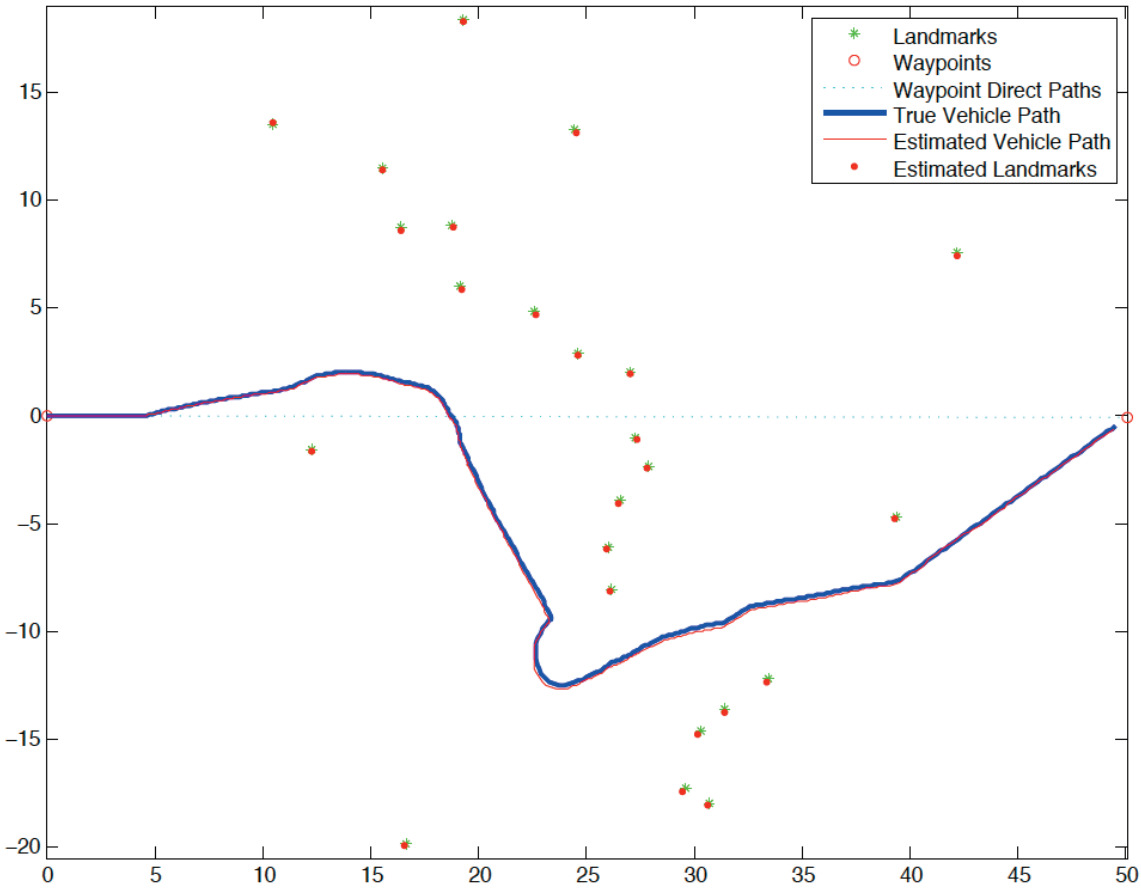


Figure 4.7: The final path of the robot. Because the opening of the landmarks in the middle is not big enough, the robot considers going through the middle gap to be unsafe due to its own physical dimensions, so it turns and finds another safer path, i.e., going through the bigger gap located in the lower half of the plot.

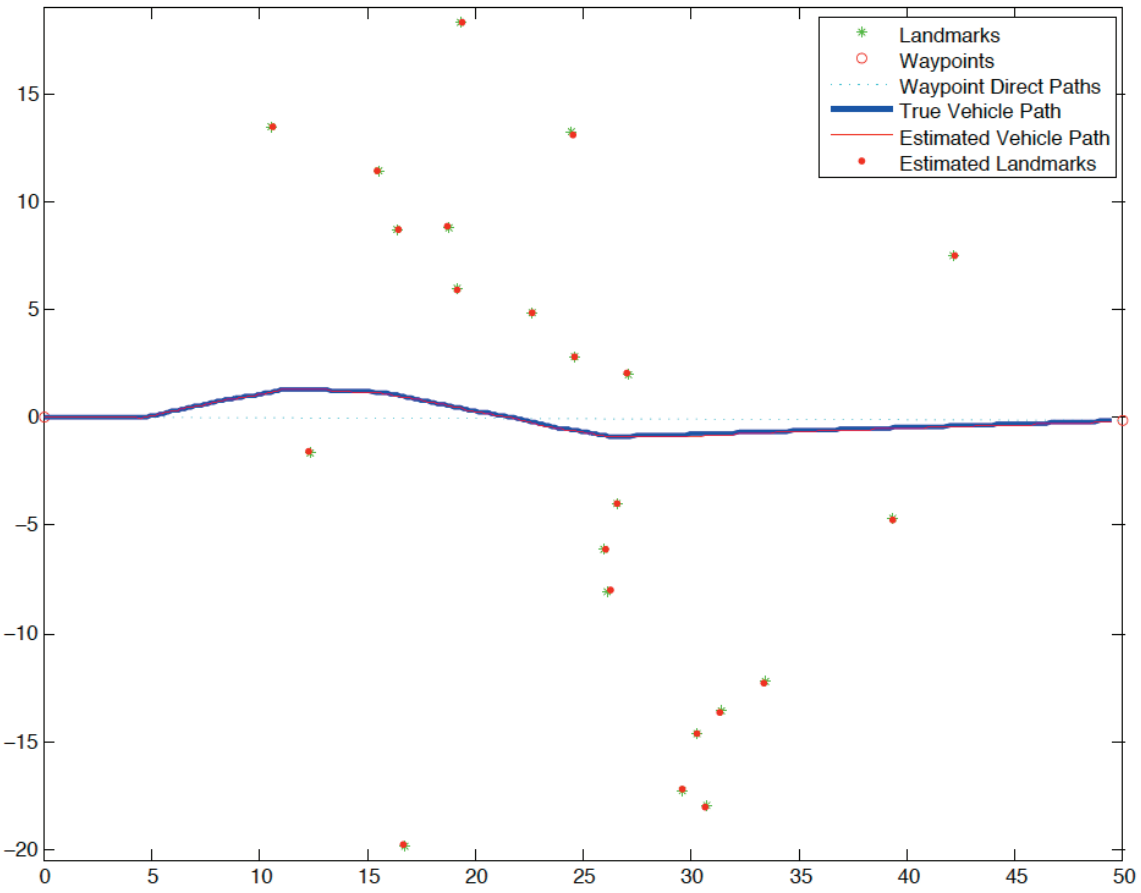


Figure 4.8: The final path of the robot. As the gap in the middle part widens, the robot sees an oppotunity and find a closer safe path to reach the goal location, instead of going the detour.

Chapter 5

Conclusion and Future Work

Bayesian estimation is an important state estimation technique for robotic navigation. The particle filters are capable of dealing with nonlinear and non-Gaussian systems, but there are issues such as degeneracy that need to be improved. We designed new particle-based SLAM filters in Chapter 3. We are continuing our effort to develop better SLAM filters and are exploring the use of Gaussian sum particle filters that have been shown to be quite effective in [61] even though the number of mixands "has to be determined by trial and error" and there is no systematic way to determine the mean and variance of each Gaussian component.

In Chapter 4, we reviewed the existing frameworks and recent advances for solving the three aspects of the robotic navigation problem: global planning (Section 4.2), local navigation (Section 4.3), and exploration (Section 4.4). In global planning, frameworks capable of coping with uncertainty have become increasingly popular, such as the POMDP and SLAM. For local navigation, a number of methods have been proposed for obstacle avoidance, and a special case of receding horizon control, called myopic control, has been proved to be effective and is gaining ground in this area. In Chapter 2, we demonstrated that myopic control has similar level of performance as other complicated strategies but has significantly less computational complexity. To achieve effective exploration, several approaches have been developed, which are able to seek paths that maximize the information of the environment.

Although the research in autonomous robot navigation has made significant progress,

most of the work only focuses on one of the three aspects discussed above. The applications in the real world impose significant challenges and require the capability of solving navigation problems involving all three aspects and providing innovative solutions that are nearly optimal. Therefore, it is essential that global planning algorithms, local navigation routines, and exploration procedures be integrated together in order to achieve the global goal.

Section 4.5 has sketched a new framework for solving the navigation problem in a continuous environment with feature-based landmarks. The goal is to reach a destined location as quickly as possible while actively maintaining a certain level of confidence in the estimated environment and avoiding direct collision with obstacles. The framework leads to an integrated approach that balances between exploration and exploitation and has control blocks for each of the three navigation aspects. It also overcomes many of the common drawbacks of current approaches discussed in Section 4.5.1.

A promising direction of the future work in autonomous robot navigation is to design hybrid systems which are capable of handling all aspects of robotic navigation requirements. This is important as human beings are extending the presence in the universe and a lot of missions impose stringent requirements on the robots. The design of hybrid systems is technically challenging, but its application is promising. Moreover, how to interconnect the various aspects of robotic navigation and coordinate harmoniously the task goals is hard but exciting topics to develop.

In this work, we have presented a general framework for integrated robotic navigation. For future work, we can improve the framework by the following: 1. Achieve better local obstacle avoidance by designing better reactive algorithms that can react faster and take better actions that can help achieve the global goal. 2. Design better exploration algorithms that can achieve the same level of effectiveness in exploring the surroundings but with less time. 3. Design better filters to cope with uncertainties.

It is noteworthy to point out that, as uncertainty is ubiquitous in real-world problems, “probabilistic techniques will continue to be the most robust approach to state estimation problems” and “will serve as a powerful tool for understanding problems and the approximation of their optimal solutions” [19].

Bibliography

[1] D. Alspach and H. Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transaction on Automatic Control*, 17(4):439–448, 1972.

[2] B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Englewood Ciffs, NJ: Prentice-Hall, 1979.

[3] J. Andrade-cetto, T. Vidal-calleja, and A. Sanfeliu. Unscented transformation of vehicle states in slam. In *In Proc. IEEE Int. Conf. Robot. Automat*, pages 324–329, 2005.

[4] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–88, 2002.

[5] J. Bartroff and T. L. Lai. Approximate dynamic programming and its applications to the design of phase i cancer trials. *Statistical Science*, 25(2):245257, 2010.

[6] A. Elnagar; A. Basu. Global path planning using artificial potential fields. In *IEEE International Conference on Robotics and Automation*, pages 316 – 321, 1989.

[7] Rudolf Bauer, Wendelin Feiten, and Gisbert Lawitzky. Steer angle fields: An approach to robust manoeuvring in cluttered, unknown environments. *Robotics and Autonomous Systems*, 12(3-4):209 – 212, 1994.

[8] M. Becker, C. M. Dantas, and W. P. Macedo. Obstacle avoidance procedure for mobile robots. In *ABCM Symposium Series in Mechatronics*, volume 2, pages 250–257, 2006.

[9] R.E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[10] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2nd edition, 2000.

[11] C. Guarino Lo Bianco and A. Piazzi. Optimal trajectory planning with quintic g2-splines. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000*, pages 620–625, Oct 2000.

[12] J. Blythe. *An overview of planning under uncertainty*, pages 85–110. Springer-Verlag, Berlin, Heidelberg, 1999.

[13] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), 1989.

[14] J. Borenstein and Y. Koren. The vector field histogram - Fast obstacle avoidance for mobile robots. *IEEE Journal of Robotics and Automation*, 7:278–288, 1991.

[15] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 540–545, 2002.

[16] R. I. Brafman. A heuristic variable grid solution method for POMDPs. In *Proceedings of the National Conference on Artificial Intelligence*, 1997.

[17] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation*, pages 341–346, 1999.

[18] A. Brooks. *Parametric POMDPs for Planning in Continuous State Spaces*. PhD dissertation, Australian Centre for Field Robotics, University of Sydney, 2007.

[19] W. Burgard. Probabilistic approaches to robot navigation. *IEEE Robotics and Automation Magazine*, 15(2):8–13, 2008.

[20] C.V. Caldwell, E.G. Collins, and S. Palanki. Integrated guidance and control of AUVs using shrinking horizon model predictive control. In *Oceans'06 MTS/IEEE*, pages 1–6, Boston, MA, 2006.

[21] O. Cappé, S. J. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential monte carlo. *IEEE Proceedings*, 95(5):899–924, 2007.

[22] GEORGE CASELLA and CHRISTIAN P. ROBERT. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94, 1996.

[23] Daniel Castro, Urbano Nunes, and Antonio Ruano. Obstacle avoidance in local navigation. In *IEEE Mediterranean Conference on Control and Automation*, 2002.

[24] Y. Cheng, P. Jiang, and Y.F. Hu. A distributed snake algorithm for mobile robots path planning with curvature constraints. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 2056 – 2062, 2008.

[25] C. I. Connolly, J. B. Burns, and R. Weiss. Path planning using laplace's equation. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 2102–2106, 1990.

[26] John Connors and Gabriel Elkaim. Analysis of a spline based, obstacle avoiding path planning algorithm. In *IEEE Vehicle Technology Conference*, pages 2565 – 2569, 2007.

[27] John Connors and Gabriel Elkaim. Manipulating b-spline based paths for obstacle avoidance in autonomous ground vehicles. In *ION National Technical Meeting*, pages 1081–1088, 2007.

[28] Konstantinos Dalamagkidis, Kimon P. Valavanis, and Les A. Piegl. Autonomous autorotation of unmanned rotorcraft using nonlinear model predictive control. *Journal of Intelligent and Robotic Systems*, 57(1-4):351–369, 2010.

[29] M. Deittert, A. Richards, and G. Mathews. Receding horizon control in unknown environments: Experimental results. In *IEEE International Conference on Robotics and Automation*, pages 3008–3013, Anchorage, AK, May 2010.

[30] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[31] A. Doucet, S. Godsill, and C. Andrieu. On sequential monte carlo methods for bayesian filtering. *Statistics and Computing*, 10(3):197–08, 2000.

[32] A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. Technical report, Department of Statistics, University of British Columbia, Vancouver, Dec 2008.

[33] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–110, 2006.

[34] A. Fatmi, A. A. Yahmadi, L. Khriji, and N. Masmoudi. A fuzzy logic based navigation of a mobile robot. *World Academy of Science, Engineering and Technology*, 22, 2006.

[35] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 560 – 565, 1993.

[36] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1):23–33, 1997.

[37] E. W. Frew, Langelaan J, and Joo S. Adaptive receding horizon control for vision-based navigation of small unmanned aircraft. In *Proceedings 2006 American Control Conference*, pages 2160–2165, Minneapolis, MN, June 2006.

[38] S.K. Gehrig and F.J. Stein. Collision avoidance for vehicle-following systems. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):233 – 244, 1993.

[39] J. Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–39, November 1989.

[40] J. Giesbrecht. Global path planning for unmanned ground vehicles. Technical report, Defense RnD Canada - Suffield, December 2004.

[41] C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous uav guidance. *Journal of Intelligent and Robotic Systems*, 57(1-4):65–100, 2010.

[42] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F on Radar and Signal Processing*, 140(2):107–113, April 1993.

[43] J. Han, T. L. Lai, and V. Spivakovksy. Approximate policy optimization and adaptive control in regression models. *Computational Economics*, 27:433–452, 2006.

[44] Eric A. Hansen. Solving POMDPs by searching in policy space. In *Proceedings of the Fourteenth International Conference on Uncertainty In Artificial Intelligence*, pages 211–219, 1998.

[45] M. Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.

[46] M. Hebert, C. Thorpe, and A. Stentz. *Intelligent Unmanned Ground Vehicles: Autonomous Navigation Research at Carnegie Mellon.* KluwerAcademic Publishers, 1997.

[47] J. Hilgert, K. Hirsch, T. Bertram, and M. Hiller. Emergency path planning for autonomous vehicles using elastic band theory. In *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, volume 2, pages 1390 – 1395, 2003.

[48] S. Julier. The spherical simplex unscented transformation. In *The Proceedings of the IEEE American Control Conference*, pages 2430–2434, 2003.

[49] S. Julier, J. Uhlmann, and H. F. Durrant-Whyte. A new method for the nonlinear transformation of means and covariances in filters and estimators. *IEEE Transactions on Automatic Control*, 45(3):477–482, 2000.

[50] S. Julier and J. K. Uhlmann. A general method for approximating nonlinear transformations of probability distributions. Technical report, 1996.

[51] S. J. Julier and I. Industries. The scaled unscented transformation. In *Proceedings of the American Control Conference*, volume 6, pages 4555–4559, 2002.

[52] S. J. Julier and J. K. Uhlmann. Unscented filtering and nonlinear estimation. In *Proceedings of the IEEE*, pages 401–422, 2004.

[53] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, pages 182–193, 1997.

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

[55] R. E. Kalman. A new approach to linear filtering and prediction problems. *T-ASME*, 1960:35–45, March 1960.

[56] Z. Khan, T. Balch, and F. Dellaert. A rao-blackwellized particle filter for eigentracking. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:980–986, 2004.

[57] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5:90–98, April 1986.

[58] C. Kim, R. Sakthivel, and W. K. Chung. Unscented fastslam: A robust and efficient solution to the slam problem. *IEEE Transactions on Robotics*, 24(4):808–820, 2008.

[59] G. Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

[60] M. Kneebone and R. Dearden. Navigation planning in probabilistic roadmaps with uncertainty. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling*, 2009.

[61] Jayesh H. Kotecha and Petar M. Djuric. Gaussian sum particle filtering. *IEEE Transactions on Signal Processing*, 51:2602 – 2612, 2003.

[62] E. Koyuncu and G. Inalhan. A probabilistic B-spline motion planning algorithm for unmanned helicopters flying in dense 3d environments. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 815–821, 2008.

[63] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.

[64] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proceedings Workshop on the Algorithmic Foundations of Robotics*, 2000.

[65] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. Technical report, Department of Computer Science, Brown University, Providence, RI, July 1995.

[66] E. Magid, D. Keren, E. Rivlin, and I. Yavneh. Spline-based robot navigation. In *The IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2296–2301, 2006.

[67] R. Martinez-Cantin and J. A. Castellanos. Unscented slam for large-scale outdoor environments. In *In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 328–333, 2005.

[68] R. Martinez-Cantin, N. de Freitas, A. Doucet, and J. Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[69] A. A. Masoud. A harmonic potential field approach for navigating a rigid, nonholonomic robot in a cluttered environment. In *Proceedings of The IEEE International Conference on Robotics and Automation*, pages 7–13, 2009.

[70] M.A.O. Mendez and J.A.F. Madrigal. Fuzzy logic user adaptive navigation control system for mobile robots in unknown environments. In *IEEE International Symposium on Intelligent Signal Processing*, pages 1–6, 2007.

[71] J. Minguez and L. Montano. Nearness diagram (ND) navigation: Collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20:45 – 59, 2004.

[72] J. Minguez, J. Osuna, and L. Montano. A "divide and conquer" strategy based on situations to achieve reactive collision avoidance in troublesome scenarios. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3855–3862, 2004.

[73] G. E. Monahan. A survey of partially observable Markov decision processes: theory, models and algorithms. *Management Science*, 28(1), 1982.

[74] M. Montemerlo. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem with Unknown Data Association.* PhD dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, July 2003.

[75] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of The AAAI National Conference on Artificial Intelligence*, pages 593–598, Edmonton, Canada, 2002.

[76] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1151–1156. IJCAI, 2003.

[77] E. Nebot. University car park dataset. ACFR - The University of Sidney. `http://www-personal.acfr.usyd.edu.au/nebot/car_park.htm/`.

[78] A. Ng and M. Jordan. Pegasus: A policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

[79] K. Park and N. Zhang. Behavior-based autonomous robot navigation on challenging terrain: A dual fuzzy logic approach. In *Annual Foreign Ownership, Control or Influence Conference*, pages 239–244, 2007.

[80] A. Piazzi and C. Guarino Lo Bianco. Optimal trajectory planning with quintic G2-splines. In *Proceedings of the IEEE intelligent Vehicles Symposium*, pages 198–203, Oct 2000.

[81] A. Piazzi, M. Romano, and C. Guarino Lo Bianco. G3-splines for the path planning of wheeled mobile robots. In *Proceedings of the European Control Conference, Cambridge (UK)*, Sep 2003.

[82] J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: an anytime algorithm for pomdps. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[83] J. L. Piovesan and H. G. Tanner. Randomized model predictive control for robot navigation. In *Proceedings of The IEEE International Conference on Robotics and Automation*, pages 1817–1822, 2009.

[84] M. K. Pitt and N. Shephard. Filtering via simulation: auxiliary particle filter. *Journal of the American Statistical Association*, 94:590–599, 1999.

[85] J. M. Porta, N. Vlassis, M. T. J. Spaan, and P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7:2329–2367, 2006.

[86] P. Poupart. *Exploiting structure to efficiently solve large scale partially observable Markov decision processes.* PhD dissertation, Department of Computer Science, University of Toronto, 2005.

[87] P. Poupart and C. Boutilier. Value-directed compression of POMDPs. In *Annual Conference on Neural Information Processing Systems*, pages 1547–1554, 2002.

[88] E. Prassler, J. Scholz, M. Strobe1, and P. Fiorini. An intelligent (semi-) autonomous passenger transportation system. In *IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, pages 374 – 379, 1999.

[89] S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Proceedings of the International Conference on Robotics and Automation*, pages 802–807, 1993.

[90] R. R. van der Merwe, N. de Freitas, A. Doucet, and E. Wan. The unscented particle filter (cued/f-infeng/tr-380). Technical Report CUED/F-INFENG/TR 380, Cambridge University Engineering Department, Cambridge, England, August 2000.

[91] S. Rajesh, K. Sandeep, and R.K. Mittal. Robot motion planning on rough terrain using multiresolution second generation wavelets and non-uniform B-splines. In *IEEE International Conference on Industrial Technology*, pages 967–972, 2006.

[92] J. Randlov and P. Alstrm. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998.

[93] Patrick Reignier. Fuzzy logic techniques for mobile robot obstacle avoidance. *Robotics and Autonomous Systems*, 12(3-4):143–153, 1994.

[94] S. Ross and B. Chaib-draa. AEMS: an anytime online search algorithm for approximate policy refinement in large POMDPs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 2592–2598, 2007.

[95] S. Ross, B. Chaib-draa, and J. Pineau. Bayes-adaptive POMDPs. In *Advances in Neural Information Processing Systems*, 2007.

[96] S. Ross, B. Chaib-draa, and J. Pineau. Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2845–2851, Pasadena, CA, 2008.

[97] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, 2008.

[98] N. Roy. *Finding Approximate POMDP solutions Through Belief Compression*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2003.

[99] N. Roy and S. Thrun. Coastal navigation with mobile robots. In *Advances in Neural Processing Systems*, pages 1043–1049, 1999.

[100] K. Sato. Collision avoidance in multi-dimensional space using Laplace potential. In *Proc. 15th Conf. Robotics Soc. Jpn.*, pages 155–156, 1987.

[101] T. Sattel and T. Brandt. Ground vehicle guidance along collision-free trajectories using elastic bands. In *Proceedings of the American Control Conference*, pages 4991 – 4996, 2005.

[102] T. Schon, F. Gustafsson, and P. Nordlund. Marginalized particle filters for mixed linear nonlinear state-space models. *IEEE Trans. on Signal Processing*, 53:2279–2289, 2005.

[103] E. Shan, B. Dai, J. Song, and Z. Sun. A dynamic RRT path planning algorithm based on B-spline. In *International Symposium on Computational Intelligence and Design*, pages 25–29, 2009.

[104] R. Sim and N. Roy. Active exploration planning for SLAM using extended information filters. In *Proceedings of Conference on Uncertainty in Artificial Intelligence*, 2004.

[105] R. Sim and N. Roy. Global A-optimal robot exploration in SLAM. In *Proceedings of The IEEE International Conference on Robotics and Automation*, 2005.

[106] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *International Conference on Robotics and Automation*, April 1996.

[107] T. Smith and R. G. Simmons. Heuristic search value iteration for POMDPs. In *Proceeding of International Conference on Uncertainty in Artificial Intelligence*, 2004.

[108] T. Smith and R. G. Simmons. Point-based pomdp algorithms: Improved analysis and implementation. In *Proceeding of International Conference on Uncertainty in Artificial Intelligence*, 2005.

[109] E. J. Sondik. *The optimal control of partially observable Markov decision processes*. PhD dissertation, Stanford University, 1973.

[110] M. Spaan and N. Vlassis. A point-based pomdp algorithm for robot planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, New Orleans, Louisiana, April 2004.

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

[112] M. T.J. Spaan. *Approximate planning under uncertainty in partially observable environments*. PhD dissertation, Universiteit van Amsterdam, Amsterdam, Netherlands, 2006.

[113] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Robotics: Science and Systems Conference*, pages 65–72, 2005.

[114] A. Stentz, A. Kelly, P. Rander, H. Herman, O. Amidi, R. Mandelbaum, G. Salgian, and J. Pedersen. Real-time, multi-perspective perception for unmanned ground vehicles. In *Proceedings of AUVSI's Unmanned Systems Symposium*, July 2003.

[115] F. Suryawan, J. De Dona, and M. Seron. On splines and polynomial tools for constrained motion planning. In *18th Mediterranean Conference on Control and Automation*, June 2010.

[116] H. Tanner and J. Piovesan. Randomized receding horizon navigation. *IEEE Transactions on Automatic Control*, 55(11):2640–2644, August 2010.

[117] S. Thrun. Monte carlo POMDPs. In *Advances in Neural Information Processing Systems*, pages 1064–1070, 2000.

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

[119] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661C692, 2006.

[120] P. Tompkins, A. Stentz, and D. Wettergreen. Global path planning for mars rover exploration. In *Proceedings of The IEEE Aerospace Conference*, March 2004.

[121] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1572–1577, 1998.

[122] I. Ulrich and J. Borenstein. VFH*: local obstacle avoidance with look-ahead verification. In *IEEE International Conference on Robotics and Automation*, volume 3, pages 2505–2511, 2000.

[123] J. van den Berg, M. C. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation*, pages 1928–1935, 2008.

[124] R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University, Portland, OR, USA, April 2004.

[125] Miloš Šeda. Roadmap methods vs. cell decomposition in robot motion planning. In *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, pages 127–132, 2007.

[126] X. Wang and H. Zhang. 2007 ieee international conference on robotics and automation, icra 2007, 10-14 april 2007, roma, italy. In *ICRA*. IEEE, 2007.

[127] B. Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, pages 47–53, New York, NY, USA, 1998.

[128] Y. Yanoshita and S. Tsuda. Space robot path planning for collision avoidance. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, 2009.

[129] P. G. Zavlangas, S. G. Tzafestas, and K. Althoefer. Fuzzy obstacle avoidance and navigation for omnidirectional mobile robots. In *European Symposium on Intelligent Techniques*, 2000.

[130] R. Zhou and E. A. Hansen. An improved grid-based approximation algorithm for POMDPs. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001.