UNIVERSITY OF GENOVA

PHD PROGRAM IN BIOENGINEERING AND ROBOTICS

Multimodal Planning under Uncertainty: Task-Motion Planning and Collision Avoidance

by

Antony Thomas

Thesis submitted for the degree of *Doctor of Philosophy* (33° cycle)

July 2021

Professor Marco Baglietto — Supervisors
Professor Fulvio Mastrogiovanni

Professor Giorgio Cannata — Head of the PhD program

Dibris

Department of Informatics, Bioengineering, Robotics and Systems Engineering

To

Papa who believed in me and pushed me towards realizing my dreams.

Amma who has always been a silent supporter.

John for being there.

Maria for all the love and understanding.

Joanna for all the happiness.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation contains fewer than 85,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Antony Thomas
July 2021

Acknowledgements

Seven years before if someone had told me that I would be writing this document in the future, I would have laughed at them. But life is supposed to take its turns and here I am writing this document. Below, I would like to acknowledge some people who have helped me reach where I am today.

Everything that I am began with my parents, Thomas and Asha. They taught me about aspiring, striving and working hard. Their constant support kept me going and I am ever grateful for their upbringing. I owe this document to them. My brother John has always stood by me and I apologize to him if I was not immediately available as I was away for my education since his high school.

One of the earliest seeds of academic interest was sown in my school Bhavans Varuna Vidyalaya. The teachers there taught sincerely and earnestly wished that all students grew up as good human beings. Early influence on research came from my thesis advisor Prof. Nandan Kumar Sinha during the final years of my undergraduate study at IIT Madras. I am grateful to him for his support and wise words that proved decisive many a times. I am also thankful to my friends who stood by me and from whom I have learned a lot: Jackson, Akhil, Nair, Ranjith, Arjun, John, Kamat, Shezhi, Kulls, Achuth, Chetan, SNS, Evera, Ameya, Aditya.

At this point, I would like to express my sincere gratitude to my Master's thesis advisor at the Technion, Prof. Vadim Indelman. He has always been a constant source of motivation. Passion for research is something that I have learned from Vadim and I have also drawn a lot of inspiration from his work ethic. I am also thankful to my collaborator Shashank Pathak, who was always ready to share ideas and discuss them at length and would patiently answer all my questions. My friends at my time at the Technion, especially Sundar, Shruti, Jane, Anita, Himanshu, Yadu, Sandeep, Subhamay and the *shabbat group* have also helped me greatly and made my stay at the Technion memorable.

Now comes the difficult part, my stay at the University of Genova. I think it is right that I first begin with the people who made this document possible, my advisors Marco Baglietto and Fulvio Mastrogiovanni. From the beginning they gave me the freedom to

pursue my research interests and I will not be able to thank them enough for their patience and help during these years. Marco has the remarkable ability to simplify any complex theory using pen and paper. It comes from his immense knowledge and experience, and this was something that I very much needed. Fulvio has taught me a lot about research and writing; perspectives that I didn't even know it existed. I hope I have inherited at least an epsilon of their knowledge and down-to-earth approach.

I would like to thank my reviewers, Prof. Giorgio Battistelli and Dr. Mauro Vallati for agreeing to review my thesis, patiently reading it and providing me with useful feedback and suggestions for future work.

Most professors that I have come in contact at Genova have helped me in many ways. I would particularly like to thank Prof. Giorgio Cannata for his support and timely solutions to certain administrative technicalities.

I would like to thank the administrative people from Villa Bonino, especially Valentina Scanarotti and Roberta Fraguglia who ensured that everything went smoothly, be it the refunding of conference expenses or any other administrative technicalities.

There are many friends and colleagues who became an integral part of life at Genova. Vinil deserves a special mention. While I was still in my early days of research, his research experience, interest in my research area and above all his personality, made for a unique relationship and I learned quite a lot from him. A similar relationship was also fostered with Hassan who shared a common research interest and view of life. Interactions with him always produced something fruitful, something which I look forward to doing again.

Yusha, my apartment-mate, we became and remain good friends. We shared a lot of interests and had innumerable discussions on a variety of topics ranging from general AI, ambient intelligence, meaning of life, and the existence of God. Although his research interests slightly differed from mine, his expertise has often helped. I think his presence was vital during my stay at Genova and I cannot thank him enough. During the later years, Hossein also became a part of our group and it also provided me with an opportunity to collaborate with him.

Other colleagues and friends who influenced/inspired me and was always ready to help include: Praveen, Divya, Cris, Nikesh, Sherona, Keerthi, Aravind, Aman, Luca, Prajval, Suman, Clin, Klelja, Alex, Alessandro Carfi, Alessandro Albini. I also had the opportunity to co-supervise three EMARO masters students: Sunny, Luigi and Suman. It was a pleasing and productive experience.

I would also like to thank my in-law family, mother-in law Annie John, father-in law K. A. Baby and sister-in law Teena for their support.

Finally, I would like to thank my wife Maria and our daughter Joanna. Maria and I got married midway through my PhD and I couldn't have finished this thesis without her constant support and understanding. Joanna has been a constant source of joy and inspiration.

Abstract

In this thesis we investigate the problem of motion planning under environment uncertainty. Specifically, we focus on *Task-Motion Planning* (TMP) and probabilistic collision avoidance which are presented as two parts in this thesis. Though the two parts are largely self-contained, collision avoidance is an integral part of TMP or any robot motion planning problem in general. The problem of TMP which is the subject of Part I is by itself challenging and hence in Part I, collision computation is not the main focus and is addressed with a deterministic approach. Moreover, motion planning is performed offline since we assume static obstacles in the environment. Online TMP, incorporating dynamic obstacles or other environment changes is rather difficult due to the computational challenges associated with updating the changing task domain. As such, we devote Part II entirely to the field of online probabilistic collision avoidance motion planning.

Of late, TMP for manipulation has attracted significant interest resulting in a proliferation of different approaches. In contrast, TMP for navigation has received considerably less attention. Autonomous robots operating in real-world complex scenarios require planning in the discrete (task) space and the continuous (motion) space. In knowledge-intensive domains, on the one hand, a robot has to reason at the highest-level, for example, the objects to procure, the regions to navigate to in order to acquire them; on the other hand, the feasibility of the respective navigation tasks have to be checked at the execution level. This presents a need for motion-planning-aware task planners. In Part I of this thesis, we discuss a probabilistically complete approach that leverages this task-motion interaction for navigating in large knowledge-intensive domains, returning a plan that is optimal at the task-level. The framework is intended for motion planning under motion and sensing uncertainty, which is formally known as *Belief Space Planning* (BSP). The underlying methodology is validated in simulation, in an office environment and its scalability is tested in the larger Willow Garage world. A reasonable comparison with a work that is closest to our approach is also provided. We also demonstrate the adaptability of our method by considering a building floor navigation domain. Finally, we also discuss the limitations of our approach and put forward suggestions for improvements and future work.

In Part II of this thesis, we present a BSP framework that accounts for the landmark uncertainties during robot localization. We further extend the state-of-the-art by computing an exact expression for the collision probability under Gaussian motion and perception uncertainties. Existing BSP approaches assume that the landmark locations are well known or are known with little uncertainty. However, this might not be true in practice. Noisy sensors and imperfect motions compound to the errors originating from the estimate of environment features. Moreover, possible occlusions and dynamic objects in the environment render imperfect landmark estimation. Consequently, not considering this uncertainty can result in wrongly localizing the robot, leading to inefficient plans. Our approach incorporates the landmark uncertainty within the Bayes filter framework. We also analyze the effect of considering this uncertainty and delineate the conditions under which it can be ignored. Furthermore, we also investigate the problem of safe motion planning under Gaussian motion and sensing uncertainties. Existing approaches approximate the collision probability using upper-bounds that can lead to overly conservative estimate and thereby suboptimal plans. We formulate the collision probability process as a quadratic form in random variables. Under Gaussian distribution assumptions, an exact expression for collision probability is thus obtained which is computable in real-time. Further, we compute a tight upper bound for fast online computation of collision probability and also derive a collision avoidance constraint to be used in an optimization setting. We demonstrate and evaluate our approach using a theoretical example and simulations in single and multi-robot settings using mobile and aerial robots. A comparison of our approach to different state-of-the-art methods are also provided.

List of Abbreviations

TMP . . . . . Task-Motion Planning

BSP . . . . . Belief Space Planning

POMDP . . . . . Partially Observable Markov Decision Process

SLAM . . . . . Simultaneous Localization and Mapping

EKF . . . . . Extended Kalman Filter

MPC . . . . . Model Predictive Control

MPTP . . . . . Motion-Planning-aware Task Planning

PDDL . . . . . Planning Domain Definition Language

TRPG . . . . . Temporal Relaxed Plan Graph

PRM . . . . . Probabilistic Roadmap

BRM . . . . . Belief Roadmap

MGP . . . . . Multi-Goal Planning

TSP . . . . . Traveling Salesman Problem

MCI . . . . . Monte Carlo Integration

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Inference, perception and planning form the key components for autonomous navigation. Inference is concerned with estimating the state of the robot or other variables of interest given the information thus far, that is, sensor measurements and controls. Perception deals with what the robot observers through its various sensors and planning involves choosing the next best actions to realize a given task. In this work we are concerned mainly with *planning*. In the fields of Artificial Intelligence and Robotics, planning in the broad sense cover a wide spectrum of approaches. In this thesis, we focus on two facets of planning– *Task-Motion Planning* and collision avoidance planning.

Autonomous robots operating in real-world complex scenarios require planning in the discrete (task) space and the continuous (motion) space. In knowledge-intensive domains, on the one hand, a robot has to reason at the highest level, for example the regions to navigate to; on the other hand, the feasibility of the respective navigation tasks have to be checked at the execution level. This entails the need for TMP– an approach that integrates planning from Artificial Intelligence and motion planning from Robotics. TMP essentially involves combining discrete and continuous decision-making to facilitate an efficient interaction between the two domains. Starting from an initial state, TMP synthesizes a plan to a goal state by a concurrent or interleaved set of discrete actions and continuous collision-free motions. Over the past few years, TMP for manipulation has received considerable interest among the research community, resulting in a proliferation of approaches [99, 28, 52, 13]. In comparison, TMP for robot navigation introduces different challenges and has not received much attention, albeit being pervasive in most real-world scenarios. TMP for navigation essentially involves (a) selecting discrete/high-level actions to navigate to different regions, objects or locations of interest in the environment and (b) deciding the order of these visits. Selecting the best set of discrete actions for a given objective requires computing the

navigation costs for each of these actions. Hence motion planning should be interleaved with task planning to compute the motion costs for each of the discrete actions.

However, current TMP approaches for navigation [77, 112, 70] do not take into account the uncertainties arising out of real-world conditions. Real-world scenarios often induce uncertainties that transpire due to insufficient knowledge about the environment, inexact robot motion or imperfect sensing. In such scenarios, the robot poses or other variables of interest can only be dealt with in terms of probabilities. Planning is therefore done in the *belief space*, which corresponds to the set of all probability distributions over possible robot states. The corresponding problem, BSP is an instantiation of a *Partially Observable Markov Decision Processes* (POMDPs) [50]. We develop a probabilistically complete TMP framework for mobile robot navigation in partially-observable state-spaces with motion and sensing uncertainty. This is the topic of Part I of this thesis.

Robots are becoming more pervasive and are increasingly used in close proximity to humans and other objects in factories, living spaces, elderly care, and robotic surgery, planning for collision free trajectories in real-time is imperative for safe and efficient operation. Further, the nature of uncertain environments are such that they often preclude the existence of collision free trajectories [3]. In the presence of noisy sensors, both the robot and world state cannot be estimated precisely. Moreover, in the case of dynamic obstacles, their future states have to be predicted and they are not known exactly due to the lack of perfect knowledge of their motions. As such, for safe navigation, both the robot state uncertainty and the uncertainty in obstacle estimates need to be considered while computing collision probabilities.

Localization is also a key aspect for safe and efficient navigation as it is a precursor to solving the problems “where to navigate to" and “how to reach there". However, most approaches assume that landmarks are known with high certainty. For example, given the map of the environment, while planning for future actions the standard Markov localization does not take into account the map uncertainty (that is, landmark locations are assumed to be perfect). However, this might not be true in practice. For example, let us consider the map of an environment obtained from a *Simultaneous Localization and Mapping* (SLAM) session. Due to the dynamic nature of the environment the objects of interests could be occluded when viewed from the set of viewpoints which would have otherwise produced a full observation. Moreover, an erroneous localization could lead to wrongly estimated object poses. This object pose uncertainty directly translates to the fact that the viewpoints from which the object can be observed are uncertain. Therefore, one can only reason in terms of the probability of observing the object from the considered pose or the viewpoint. This results

in a probability distribution function for the viewpoints. Consequently, not considering this uncertainty can wrongly localize the robot, leading to inefficient plans causing catastrophes.

In this work, we incorporate landmark or object uncertainties in BSP and derive the resulting Bayes filter in terms of the *Extended Kalman Filter* (EKF) update equations. We then contribute to the literature in collision avoidance planning by deriving from first principles an exact expression for collision probability, given robot and obstacle state uncertainties. We further derive fast and approximate upper bounds for collision probability such that the probability of collision is guaranteed to be less than a specified bound while navigating to the goal. This is the topic of Part II of this thesis.

Though the two parts are largely self-contained, the methods discussed in Part II may be incorporated into Part I to provide a more efficient TMP approach. However, this is not the main focus in Part I and we resort to a deterministic collision avoidance motion planning in Part I.

1.1 Thesis Structure

The thesis is organized into 2 parts with 13 chapters as follows.

Part I

Chapter 2 introduces the concept of TMP, in the context both manipulation and robot navigation. The related scientific literature is also covered.

Chapter 3 provides a brief overview of task planning and motion planning and then formally define the TMP problem. We then define the TMP approach for navigation that we consider in this thesis and state out the assumptions made while formulating the problem.

Chapter 4 discusses our TMP approach, the main contribution of this part of the thesis. We also derive the notion of task-level optimality and then prove the probabilistic completeness of our approach.

Chapter 5 extends our TMP approach to multi-robot planning and incorporate multi-robot constraints within the joint state estimation of the robots.

Chapter 6 presents the results for single robot scenarios. Validation is performed on three different navigation domains and the scalability to increasing task level complexity is also assessed. We later present the results for the multi-robot settings.

Chapter 7 discusses the limitations of our approach and suggest directions for future work. We also provide a discussion on the relationship to Multi-Goal Planning problem. This chapter thus concludes Part I of the thesis.

Part II

Chapter 8 introduces the concept of *object uncertainty* and discuss related scientific literature in the area of collision avoidance motion planning. We also formally define the problem that we intend to tackle.

Chapter 9 provides the derivation of the Bayes filter for robot localization incorporating object uncertainty. We also discuss the conditions under which object uncertainty can be ignored.

Chapter 10 explores the computation of exact collision probability between a robot and an obstacle by formulating the collision constraint as a quadratic form in random variables. The expression for collision probability is obtained as an infinite series and we prove its convergence. An upper bound for the truncation error is also derived. A comparison to other existing approaches is also presented and we also derive the collision constraint for non circular geometries. Further, the approach is validated in single and multi-robot setting.

Chapter 11 relaxes the spherical geometry assumption in Chapter 10 to formulate the collision constraint as the distance between ellipsoids; ellipsoid representation forms a much better approximation as compared to spheres. A tight upper bound for collision probability is derived and the approach is validated in simulation.

Chapter 12 develops an accurate constraint for collision avoidance during motion planning which is much faster when compared to the approaches discussed in Chapter 10 and Chapter 11. This constraint can then be used in an online optimization setting and is validated in simulation using mobile and aerial robots.

Chapter 13 concludes Part II and we discuss the limitations of the approaches developed and suggest directions for future work.

1.2 Thesis Contribution

The main contributions of this thesis are:

Part I

C1 We develop a *Motion-Planning-aware Task Planning* (MPTP) approach providing an interface between task planning and motion planning for navigating in large knowledge-intensive domains. The developed framework is intended for mobile robot navigation under motion and sensing uncertainty.

C2 We show that MPTP is probabilistically complete and also derive the notion of task-level optimality, that is, the plan synthesized is optimal at the task-level since the overall action cost is less than that of other task plans generated.

C3 We also extend MPTP to incorporate multi-robot collaborative planning.

C4 We show that both single and multi-robot MPTP is scalable to increasing task-level complexity.

Part II

C5 We incorporate object uncertainties in belief space planning and derive the resulting Bayes filter in terms of the prediction and measurement updates of the EKF. We also analyze the effect of incorporating object uncertainty while computing the posterior robot belief state.

C6 We derive an expression (infinite series) for collision probability under robot and obstacle state uncertainty; robot and obstacles assumed to be spheres. Unlike previous approaches that compute an upper bound or derive conservative estimates for the probability of collision, we derive an exact expression for computing it.

C7 We prove the convergence of the infinite series and an upper bound for the truncation error is also derived.

C8 We derive the collision constraint for convex shaped polygonal objects using their 2D convex footprints.

C9 We do away with the spherical assumption for robot and obstacle shapes and derive the collision probability for robot and obstacles approximated by their minimum volume enclosing ellipsoids. We also derive a tight upper bound for fast approximation of the collision probability for 3D motion planning.

C10 We further derive an accurate constraint for collision avoidance which is fast and can be used for online *Model Predictive Control* (MPC) based motion planning.

The following works were published/under review as part of this thesis:

- A. Thomas, S. Amatya, F. Mastrogiovanni, M. Baglietto. "Task-Motion Planning in Belief Space," in *RSS Workshop on Exhibition and Benchmarking of Task and Motion Planners*, Carnegie Mellon University, Pittsburgh, PA, USA, June 2018.
- A. Thomas, S. Amatya, F. Mastrogiovanni, M. Baglietto. "Towards Perception Aware Task-Motion Planning," in *Reasoning and Learning in Real-World Systems for Long-Term Autonomy, AAAI Fall Symposium*, Arlington, VA, USA, October 2018.
- A. Thomas, S. Amatya, F. Mastrogiovanni, M. Baglietto. "Task-assisted Motion Planning in Partially Observable Domains," *Technical report, arXiv:1908.10227*, August 2019.
- A. Thomas, F. Mastrogiovanni, M. Baglietto. "Task-Motion Planning for Navigation in Belief Space," in *The International Symposium on Robotics Research (ISRR)*, Hanoi, Vietnam, October 2019.
- A. Thomas, S. Amatya, F. Mastrogiovanni, M. Baglietto. "Task-assisted Motion Planning in Belief Space," in *Italian Conference on Robotics and Intelligent Machines (I-RIM)*, Rome, Italy, October 2019.
- A. Thomas, F. Mastrogiovanni, M. Baglietto. "Towards Multi-Robot Task-Motion Planning for Navigation in Belief Space," in *European Starting AI Researchers' Symposium (STAIRS)*, Santiago de Compostela, Spain, August 2020.
- A. Thomas, F. Mastrogiovanni, M. Baglietto. "Motion Planning with Environment Uncertainty," in *Italian Conference on Robotics and Intelligent Machines (I-RIM)*, Rome, Italy, December 2020.
- A. Thomas, F. Mastrogiovanni, M. Baglietto. "An Integrated Localisation, Motion Planning and Obstacle Avoidance Algorithm in Belief Space," in *Intelligent Service Robotics*, 14(2):235-250, 2021.
- A. Thomas, F. Mastrogiovanni, M. Baglietto. "MPTP: Motion-Planning-aware Task Planning for Navigation in Belief Space," in *Robotics and Autonomous Systems*, 141:103786, 2021.
- A. Thomas, F. Mastrogiovanni, M. Baglietto. "Probabilistic Collision Constraint for Motion Planning in Dynamic Environments," in *International Conference on Intelligent Autonomous Systems (IAS)*, Singapore, June 2021.

- A. Thomas, F. Mastrogiovanni, M. Baglietto. “Bounded Collision Probability for 3D Motion Planning,”, in *IEEE Robotics and Automation Letters*, **under review**.
- A. Thomas, F. Mastrogiovanni, M. Baglietto. “Safe Motion Planning with Environment Uncertainty,”, in *Robotics and Autonomous Systems*, **under review**.

Part I

Task-Motion Planning for Navigation

We investigate the problem of integrating task planning and motion planning for mobile robot navigation in large knowledge-intensive domains. In such domains the robot has to reason about different discrete tasks that need to be achieved. Executing each of these discrete tasks require appropriate robot motions. To this end, we assume that the environment the robot operates upon is known a priori and that it does not change during its execution. We therefore pre-sample robot poses that do not collide with the environment which are then used for motion planning. In that sense, we perform a deterministic collision checking during motion planning since the sampled poses are collision free. It is to be noted that we are mainly concerned with the interaction between the task planning domain and the motion planning domain which is achieved by means of an external library. This library is basically a motion planner and is called by the task planner whenever the motion feasibility of a task needs to be checked. As such, collision avoidance is not the main concern and any such approach can be incorporated within the motion planner. Planning is currently performed offline, however to be robust to various uncertainties arising during execution, we incorporate robot motion and sensing uncertainties during planning.

Chapter 2

Task and Motion Planning

2.1 Introduction

Autonomous robots operating in complex real world scenarios require different levels of planning to execute the assigned tasks. High-level (task) planning helps break down a given set of tasks into a sequence of sub-tasks. Actual execution of each of these sub-tasks would require low-level control actions to generate appropriate robot motions. In fact, the dependency between logical and geometrical aspects is pervasive in both task planning and execution. Hence, planning should be performed in the task-motion or the discrete-continuous space [63].

In recent years, combining high-level task planning with low-level motion planning has been a subject of great interest among the Robotics and *Artificial Intelligence* (AI) communities. Traditionally, task planning and motion planning have evolved as two independent fields. AI planning frameworks such as the *Planning Domain Definition Language* (PDDL) [75] mainly focus on high-level task planning supposing that the geometric preconditions (e.g., grasping poses for a pick-up task [99]) for the robot motion to carry out these tasks are achievable. In reality, such an assumption can be catastrophic as an action or sequence of actions generated by the task planner might turn out to be unfeasible at the controller execution level.

Over the past few years, TMP for manipulation has received considerable interest among the research community [99, 28, 52, 13, 29]. Robot-based manipulation domain calls for discrete and continuous reasoning to execute the required action reliably. For example, a simple table top domain requires the robot to reason at the discrete level to decide the objects to be picked up and also the order of these high-level actions. The execution of these discrete actions require continuous reasoning in the configuration space of the robot to

generate appropriate motions. Yet, a discrete action might turn out to be unfeasible due to the end-effector's reachability workspace. This might be due to the availability of a partial map leading to unmodeled objects or occlusions leading to unobserved objects or simply because the robot is too close the target object, rendering a grasp action impossible. This presents the need for a tight coupling between task planning and motion planning, enabling an interface for efficient interaction between the symbolic and geometric layers. TMP for navigation presents different challenges in comparison to TMP for manipulation. As such, TMP for navigation has not yet received much attention and therefore lacks sufficient literature. TMP for navigation essentially involves reasoning about different objects and their properties, deciding which objects to procure, selecting high-level actions that satisfy the low-level continuous motion constraints to navigate to the objects or other locations of interest, and finally procuring the objects and delivering it to the respective goal locations subject to task and motion constraints. For example, consider a robot in an office environment where it needs to deliver documents for evaluation to the respective project managers. At the task level, it is required that the robot first identifies the project in order to navigate to the respective sections, collect the documents and then deliver them to the project manager. A task planner computes a plan in terms of these symbolic actions, subject to minimizing a certain metric. This metric, for example, might correspond to different types of action costs or the number of actions. Since we are concerned with navigation, in this thesis we associate the symbolic actions to their associated motion costs. Certain symbolic actions may not require robot motions. For example, for collecting a document, the robot may have to stay at a particular location for a given amount of time waiting for a human to place the document. Such actions are assigned a fixed cost. Selecting the best set of discrete actions for a given objective requires computing the navigation costs (and other fixed costs) for each of these actions. Hence motion planning should be interleaved with task planning to compute the motion costs for each of the respective discrete actions. Though it can be argued that the motion costs can be approximated a priori and fed to the task planner, in large knowledge-intensive domains such an assumption can be harder to justify, especially in the presence of localization and map uncertainty. Moreover, real-world scenarios often induce uncertainties. Such uncertainties arise due to insufficient knowledge about the environment, inexact robot motion or imperfect sensing. In such scenarios, the robot poses or other variables of interest can only be dealt with, in terms of probabilities. Planning is therefore done in the *belief* space, which corresponds to the probability distributions over possible robot states. Consequently, for efficient planning and decision making, it is required to reason about future belief distributions due to candidate actions and the corresponding expected

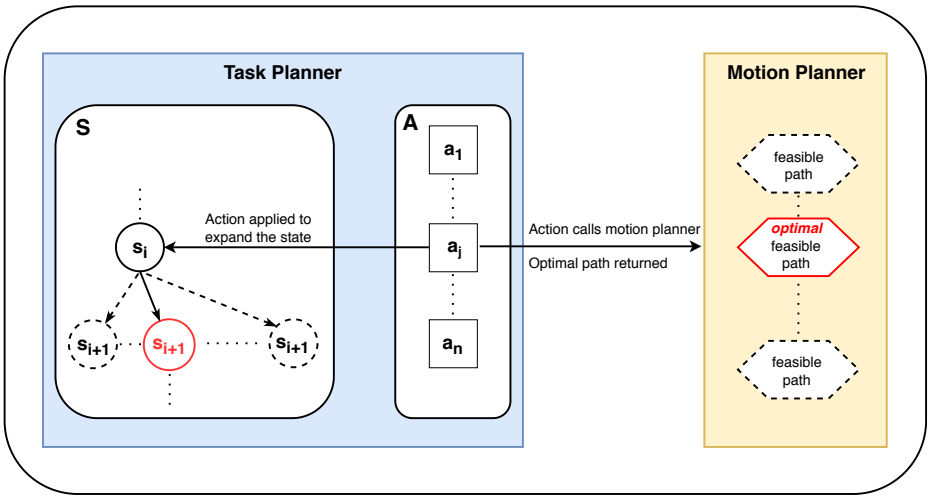


Figure 2.1 The discrete actions available to the planner are denoted by $A = \{a_1, a_2, a_3, \ldots, a_n\}$. Different motion plans are generated for the action that requires appropriate robot motion via an external module. This module is essentially a motion planner. The optimal path among the feasible motion plans is then selected, returning the optimal cost to the task planner. The corresponding action and the optimal path is the task-motion plan for changing the task state of the robot from s_i to s_{i+1}.

observations. Such a problem falls under the category of POMDPs [50]. Our motion planner is therefore equipped to perform planning in partially-observable state-spaces with motion and sensing uncertainty.

Specifically, this thesis contributes to the literature with a *Motion-Planning-aware Task Planning* approach providing an interface between task and motion planning for navigating in large knowledge-intensive domains. Such domains require a robot to reason about different objects and locations to navigate to, subject to minimizing (or maximizing) the navigation cost (objective function). Our task-motion interface layer facilitates this reasoning by communicating the motion feasibility and the corresponding planned motion costs to the task planner, synthesizing an optimal plan. To this end, we develop a probabilistically complete TMP framework for mobile robot navigation under partial-observability, embedding a motion planner within a task planner through an interface layer. We would like to stress the fact that our implementation is independent of any particular form of cost function. In this thesis, we use a standard cost function (see Chapter 4) as the MPTP cost and compare it with different cost functions in Chapter 6.

An overview of our MPTP approach is shown in Fig. 2.1. We define $A = \{a_1, ..., a_n\}$ as the finite set of symbolic/discrete actions available to the task planner. For example, let us again consider an office setting where a robot is tasked with collecting and delivering documents. In such a setting, some of the actions include, `collect_document`– which might

correspond to a human placing the document on the robot and therefore the robot waiting at a specific location for a certain duration, `deliver_document`– similar to `collect_document` action but a human picks up the document, `goto_region`– corresponds to navigating through the environment. Once an action that require appropriate robot motions to be generated is expanded by the task planner, a call to an external library is triggered. The symbolic parameters are then converted to their corresponding geometric instantiations. For example, for an action that takes the robot to a particular cubicle/region, the instantiations would be the different sampled poses in that cubicle. Once the map of the environment is obtained, the geometric instantiations can be pre-sampled. The instantiations give rise to different motion plans and the best among them is chosen according to a certain metric. The cost of the selected motion plan cost is then returned to the task planner as the cost of the corresponding action. The task-motion plan for changing the task state of the robot from the state s_i to s_{i+1} is the ordered tuple of the action a_i and the corresponding optimal path. For instance, in the office setting where a robot navigates from one cubicle (s_i) to another (s_{i+1}), the tuple is $\{\texttt{goto\_region}, \tau_i\}$. Here, `goto_region` is the task-level action a_i and τ_i is the planned trajectory for achieving this high-level action. This tuple is appended for all the task-level actions to generate the complete task-motion plan. While our approach is applicable to any domain that require task-motion interaction, we establish the key ideas in Chapter 4 through two different navigation domains and further validate our approach in Chapter 6 using the same.

2.2 Related Work

TMP has emerged as an active research area in the recent past, with particular focus on robot-based manipulation. Manipulation tasks are often rendered infeasible due to the end-effector's reachability workspace. This calls for an integrated TMP approach to ensure geometric feasibility of high-level tasks.

The genesis of TMP can be credited to Fikes and Nilsson for their work on STRIPS [24] which further led to the Shakey project [78]. Initial works on TMP performed task planning first, synthesizing a sequence of actions to be executed later by a motion planner. Shakey's planner performed a logical search first, assuming that the resulting robot motion plans can be formulated. This assumption limits the capability of the robot as the high-level actions may turn out to be non executable due to geometric limitations of the environment or the robot or both. [17] interleaves task and motion planning by checking individual high-level action feasibility using *semantic attachments*. [10] perform a combined search in the

logical and geometric spaces using a state composed of both the symbolic and geometric paths. The aSyMov planner described in [10] adopts a combination of Metric-FF [37] and a sampling-based motion planner. In contrast, we use a temporal task planner, POPF-TIF [87] with roadmap-based sampling, incorporating robot state uncertainty. Srivastava *et al.* [99] implicitly incorporate geometric variables, performing symbolic-geometric mapping using a planner-independent interface layer. Erdem *et al.*[21] leverage a boolean satisfiability (SAT) solver, computing a task-level plan and then refining it until a feasible motion plan is found.

Kaelbling and Lozano-Péres [51] propose a hierarchical approach that tightly integrates logical and geometric planning. The complexities arising out of long-horizon[1] planning are tackled to the extent that planning is done at different levels of abstraction, thereby reducing the long-horizons to a number of feasible sub-plans of shorter horizon. This regression[2]-based planner assumes that the actions are reversible while backtracking. This work is extended in [52] to consider the current state uncertainty, modeling the planning problem in the belief space. The hierarchical approach is also employed in [80, 14] to compute discrete actions with unbounded continuous variables. A geometric backtrack search is used to instantiate the symbolic actions in [64]. They also prune certain geometric instantiations, reducing the complexity. FFRob [28] performs task planning by performing search over a sampled finite set of poses, grasps and configurations. The authors of [28] extend the FF heuristics, incorporating geometric and kinematic planning constraints that provide a tight estimate of the distance to the goal. Our approach is similar to FFRob in the sense that we also pre-sample robot configurations and then plans with them, incorporating motion constraints.

Toussaint [106] performs optimization over an objective function based on the final geometric configuration (and the cost thereby), finding approximately locally optimal solutions by minimizing the objective function. The planning problem is modeled as a constraint satisfaction problem with symbolic states used to define the constraints in the optimization. This *logic-geometric programming* is applied to a four manipulator setting in [107]. Lozano-Péres and Kaelbling [72] model the motion planning as a constraint satisfaction problem over a subset of the configuration space. Iteratively Deepened Task and Motion Planning (IDTMP) is a constraint-based task planning approach that incorporates geometric information to account for the motion feasibility at the task planning level [13]. In our architecture, the motion costs are returned to the task planner, similar to the motion planner information that guides the IDTMP task planner. IDTMP performs task-motion interaction using abstraction and refinement functions whereas we use *semantic attachments* [16].

[1] Large environments require a robot to perform many actions to reach the goal, resulting in a long planning horizon[61].

[2] Goal regression is the process of planning backwards from the goal [30].

Though the approaches discussed above fall under the category of TMP for manipulation, the scope of TMP is not limited to manipulation problems alone. TMP for navigation is pervasive in most real world scenarios. For example, a mobile office robot may be tasked with collecting documents and delivering them across multiple floors. Yet, TMP for robot navigation has received less attention in the past. Real-world planning problems in large scale environments often require solving several sub-problems. For example, while navigating to a goal, the robot might have to visit other places of interests. Visiting these places of interest are high-level tasks that can be addressed using traditional task planners. Yet, these symbolic planners cannot compute the exact motion costs for these tasks, let alone perform navigation and path planning. This calls for task plans that are motion planning aware, in terms of motion costs and its feasibility.

Task planning for robot *Navigation Among Movable Obstacles* (NAMO) is introduced in [100], where each object is displace at most once throughout the plan. Van Den Berg *et al.* [110] provide a probabilistically complete algorithm for the NAMO class of problems. However, the robot state is assumed to be known perfectly. In contrast, we plan in the belief space, computing an estimate of the robot state at each instant. Hauser and Latombe [34, 35] consider *multi-model motion planning* for manipulation and legged locomotion, wherein the space of feasible configurations consists of intersecting spaces of different dimensions. In [57] a TMP approach is presented in the context of *Human-Robot Interaction* (HRI). They integrate probabilistic reasoning with symbolic reasoning by implementing a spoken dialog system, enabling the robots to ask intelligent queries. Their task planner is based on *Answer Set Programming* (ASP) [68]. Jiang *et al.* [47] focus exclusively on task planning in robotics, assuming that a feasible motion plan exists for the synthesized task plan. They provide a comparison between ASP-based and PDDL-based task planners using different benchmark domains and conclude that PDDL-based planners perform better on tasks with long solutions, and ASP-based planners tend to perform better on shorter tasks. In this thesis, we employ a PDDL-based task planner. UP2TA [77] develops a unified path planning and task planning framework for mobile robot navigation. In this approach, the robot is required to perform a series of tasks at different locations before returning back to the initial location. An interesting feature of UP2TA is its task planner heuristic, which is a combination of the FF heuristic [37] and the Euclidean distance between the waypoints associated with locations. The path planning layer computes the optimal path between each waypoint with the help of a *Digital Terrain Model* (DTM). Wong *et al.* [112] develop a task planning approach

that takes into account the optimal traversal costs[3] to synthesize a plan. Similar to UP2TA, they define tasks that are to be performed at different waypoints. However, the path planner pre-computes an optimal path for all pairs of waypoints, which are then passed to the task planner to find the optimal sequence of tasks. In contrast, we consider a general approach where the robot has to reason at a high-level about different objects or locations or regions to navigate to. The objects/locations/regions are instantiated to their geometric counterpart, by considering a set of sampled poses. For example, if a robot has to reach a location close to a chair, the geometric instantiations of this symbolic goal would correspond to a set of poses around the chair.

Jiang *et al.* [46] introduced a framework that integrates TMP with reinforcement learning that is robust to changes in the environment. The inner loop of their dual layer architecture is a TMP planner that generates task-motion plans to be sent to the outer loop. The outer loop executes the generated plans to learn from rewards. In contrast MPTP is a purely planning approach. Lo *et al.* [70] introduced PETLON, a purely planning approach for navigation that is task-level optimal and is the work closest to our approach. The inner loop in [46] uses a TMP planner that is similar to PETLON. However, in PETLON, the action costs returned by the motion planner is the trajectory length and complete observability is assumed. In contrast, our framework is more general, since we additionally consider the cost due to motion and sensing uncertainty and the distance to the goal. It is to be noted that our approach is not limited to any particular cost function and can be easily adapted to support any general cost formulation. In Chapter 6, we benchmark the scalability of our approach and provide a comparison with PETLON by considering a motion planner that evaluates the geometric-level cost of navigation. In this way we compare MPTP to PETLON by adapting our cost function to incorporate only the geometric-level cost of traversing from one location to another. Further, PETLON first compute a task plan using an admissible heuristic which is then sent to the motion planner for cost evaluation. This updates the heuristic and a refinement process repeats until the optimal plan is found. In contrast, MPTP evaluates the motion cost as each action is expanded by the task planner and hence the plan returned is optimal and needs no refinement.

[3]The costs are defined in terms of mechanical work and the objective is to find the path with optimal mechanical work. For more details, refer to [112].

Chapter 3

Preliminaries, Notations and Problem Definition

We begin by formally defining the notions of task and motion planning introduced in Chapter 2. Then, we state the TMP problem that we discuss in this thesis. The notations and formalism correspond to that of a state-transition system [30].

3.1 Task Planning

Task planning or classical planning can be defined as the process of finding a discrete sequence of actions from the current state to a desired goal state [30].

Definition 1. *A task domain* Ω *can be represented as a state transition system and is a tuple* $\Omega = (S, A, \gamma, s_0, S_g)$ *where:*

- S is a finite set of states, each state is a conjunction of propositions[1];
- A is a finite set of actions;
- $\gamma : S \times A \rightarrow S$ is the state transition function such that $s' = \gamma(s, a)$;
- $s_0 \in S$ is the start state;
- $S_g \subseteq S$ is the set of goal states.

[1] A proposition is represented by a tuple of elements, which may be constants or variables, and can be negated [9].

Definition 2. *The task plan for a task domain Ω is the sequence of actions $a_0,...,a_n$ such that $s_{i+1} = \gamma(s_i,a_i)$, for $i = 0,...,n$ and s_{n+1} satisfies S_g.*

PDDL [75] being the de facto standard syntax for task planning, we resort to the same for modeling our task domain. PDDL is an action-centred language, where each action a_i is described as a tuple $a_i = (pre_{a_i}, eff_{a_i})$, where pre_{a_i} (a set of preconditions for a_i) is a conjunction of propositions with either positive or negative terms that must hold for action execution and eff_{a_i} (the set of effects of a_i) is a conjunction of positive ($eff^+_{a_i}$) and negative ($eff^-_{a_i}$) propositions that are added or deleted upon action application, thereby changing the system state. The set of positive effects $eff^+_{a_i}$ contains propositions that become true upon the execution of action a_i and the set of negative effects $eff^-_{a_i}$ contains propositions that evaluates to false upon action execution. An action a_i is said to be applicable to a state s_i if each proposition of the preconditions holds in s_i, that is, $pre_a \subseteq s_i$. If an action a_i is applicable in state s_i, the corresponding successor state s_{i+1} is obtained as, $s_{i+1} = \gamma(s_i,a_i)$, where $s_{i+1} = (s_i \setminus eff^-_{a_i}) \cup eff^+_{a_i}$. A valid plan is a sequence of actions that when executed from s_0 results in S_g.

A planning problem with PDDL is created by providing a domain description that describes the predicates and action schemas with free variables, and a problem description that specifies the objects, initial state and the goal condition. The objects are used to instantiate the predicates and action schemas, through a process called grounding. Grounding is the process by which every combination of objects is used to replace the free variables in predicates and action schemas to obtain propositions and ground actions respectively. In this thesis, we use an extension of PDDL [25] that supports durative actions and numeric-valued fluents. Temporal planning introduces the possibility of computing concurrent plans. A temporal task domain can be defined by extending the task domain in Definition 1 as follows

Definition 3. *A temporal task domain Ω can be represented as state transition system and is a tuple $\Omega = (S,A,\gamma,s_0,S_g)$ where:*

- S is a finite set of states;
- V is a set of real valued variables;
- A is a finite set of actions;
- $\gamma : S \times A \rightarrow S$ is the state transition function such that $s' = \gamma(s,a)$;
- $s_0 \in S \cup V$ is the start state;

- $S_g \subseteq S \cup V$ is the set of goal states.

A durative action is a tuple $a_i = (pre_{a_i}, eff_{a_i}, dur_{a_i})$, where pre_{a_i} and eff_{a_i} are temporally annotated by specifying conditions/effects that holds at the *start*, *end* or during the *entire* action interval and are expressed using the constructs *at start*, *at end* and *over all* respectively. Note that these constructs are specific to PDDL formalism. dur_{a_i} corresponds to the duration of action a_i.

3.2 Motion Planning

Motion planning finds a sequence of collision free poses from a given initial/start pose (position and orientation) to a desired goal pose [66].

Definition 4. *A motion planning problem is a tuple $M = (C, f, q_0, G)$ where:*

- C is the configuration space or the space of possible robot poses;
- $f = \{0, 1\}$ determines if a configuration/pose is in collision ($f = 0$) or not (C_{free} with $f = 1$). C_{free} denotes the set of all poses that are not in collision;
- q_0 is the initial configuration;
- G is the set of goal configurations.

Definition 5. *A motion plan for M finds a valid trajectory in C from q_0 to $q_n \in G$ such that f evaluates to true for $q_0, ..., q_n$.*

In addition to the sequential form of the definition above, a motion plan can also be defined by a continuous trajectory

Definition 6. *A motion plan for M is a function of the form $\tau : [0, 1] \rightarrow C_{free}$ such that $\tau(0) = q_0$ and $\tau(1) \in G$.*

We will use a combination of the two to define the TMP problem and use roadmap based motion planner to generate collision free configurations.

3.3 Task-Motion Planning

TMP essentially involves combining discrete and continuous decision-making to facilitate efficient interaction between the two domains. Starting from an initial state, TMP synthesizes a plan to a goal state by a concurrent or interleaved set of discrete actions and continuous collision-free motions. Below we define the TMP problem formally.

Definition 7. *A task-motion planning is a tuple* $\Psi = (C, \Omega, \phi, \xi, q_0)$ *where:*

- $\phi : S \to 2^C$, is a function mapping states to the configuration space. For example, if s represents the task state— the robot is in a corridor, then $\phi(s)$ corresponds to all configurations such that the robot is in the corridor;
- $\xi : A \to 2^C$, is a function mapping actions to motion plans. We recall here that motion planning is essentially computing collision free poses in C.

Definition 8. *The TMP problem for the TMP domain* Ψ *is to find a sequence of actions* $a_0, ..., a_n$ *such that* $s_{i+1} = \gamma(s_i, a_i)$, $s_{n+1} \in S_g$ *and to find a sequence of motion plans* $\tau_0, ..., \tau_n$ *such that for* $i = 0, ..., n$, *it holds that*

$$\tau_i(0) \in \phi(s_i) \text{ and } \tau_i(1) \in \phi(s_{i+1}) \tag{3.1}$$

$$\tau_{i+1}(0) = \tau_i(1) \tag{3.2}$$

$$\tau_i \in \xi(a_i) \tag{3.3}$$

3.4 Problem Definition

In this thesis, we consider the TMP problem for a mobile robot (or robots) operating in a partially-observable environment. The map of the environment is either known a priori or is built using a standard SLAM algorithm[2]. At any time k, we denote the robot pose (or configuration q_k) by $x_k \doteq (x, y, \theta)$, the acquired measurement is denoted by z_k and the applied control action is denoted as u_k. We consider a standard motion model with Gaussian noise

$$x_{k+1} = f(x_k, u_k) + w_k \ , \ w_k \sim \mathcal{N}(0, W_k) \tag{3.4}$$

where w_k is the random unobservable noise, modeled as a zero mean Gaussian. To process the landmarks in the environment we measure the range and the bearing of each landmark

[2] http://wiki.ros.org/slam_gmapping/

relative to the robot's local coordinate frame. In general, we consider the observation model with Gaussian noise

$$z_k = h(x_k) + v_k \ , \ v_k \sim \mathcal{N}(0, Q_k) \tag{3.5}$$

It is to be noted that we assume data association as solved and hence given a measurement we know the corresponding landmark that generated it. This is not a limitation and our approach can be extended to incorporate reasoning regarding data association, as shown recently in [83]. The motion (8.1) and observation (8.2) models can be written probabilistically as $p(x_{k+1}|x_k, u_k)$ and $p(z_k|x_k)$ respectively. Given an initial distribution $p(x_0)$, and the motion and observation models, the posterior probability distribution at time k can be written as

$$p(X_{0:k}|Z_{0:k}, U_{0:k-1}) = p(x_0)\prod_{i=1}^{k} p(x_k|x_{k-1}, u_{k-1})p(z_k|x_k) \tag{3.6}$$

where $X_{0:k} \doteq \{x_0, ..., x_k\}$, $Z_{0:k} \doteq \{z_0, ..., z_k\}$ and $U_{0:k-1} \doteq \{u_0, ..., u_{k-1}\}$. This posterior probability distribution is the *belief* at time k, denoted by $b[X_k] \sim \mathcal{N}(\mu_k, \Sigma_k)$. Similarly, given an action u_k, the propagated belief can be written as

$$b[\bar{X_{k+1}}] = p(X_{0:k}|Z_{0:k}, U_{0:k-1})p(x_{k+1}|x_k, u_k) \tag{3.7}$$

Given the current belief $b[X_k]$ and the control u_k, the propagated belief parameters can be computed using the standard EKF [53] prediction as

$$\begin{aligned} \bar{\mu}_{k+1} &= f(\mu_k, u_k) \\ \bar{\Sigma}_{k+1} &= F_k \Sigma_k F_k^T + V_k W_k V_k^T \end{aligned} \tag{3.8}$$

where F_k is the Jacobian of $f(\cdot)$ with respect to x_k and V_k is the Jacobian of $f(\cdot)$ with respect to u_k. For brevity, the linearized process noise will be denoted as $R_k = V_k W_k V_k^T$. Upon receiving a measurement z_k, the posterior belief $b[X_{k+1}]$ is computed using the EKF update equations

$$\begin{aligned} K_k &= \bar{\Sigma}_{k+1} H_k^T (H_k \bar{\Sigma}_{k+1} H_k^T + Q_k)^{-1} \\ \mu_{k+1} &= \bar{\mu}_{k+1} + K_k(z_{k+1} - h(\bar{\mu}_{k+1})) \\ \Sigma_{k+1} &= (I - K_k H_k)\bar{\Sigma}_{k+1} \end{aligned} \tag{3.9}$$

where H_k is the Jacobian of $h(\cdot)$ with respect to x_k, K_k is the Kalman gain and $I \in \mathbb{R}^{3\times 3}$ is the identity matrix.

Chapter 4

Single-robot MPTP

In this chapter we develop the MPTP approach and prove its probabilistic completeness and task-level optimality. This chapter thus details our contributions **C1** and **C2.**

4.1 Task Planning

TMP for navigation requires that the task planner takes into account the motion feasibility and the corresponding motion costs while synthesizing a plan. As opposed to the manipulation domain, where the motion feasibility is corroborated with the end-effector's reachability workspace, in navigation domains this is often validated against the cost constraints, for example, a robot navigating in a corridor with a bound on the pose covariance to avoid collisions. As such, any task planner customized to enable the task-motion interface can be employed for our approach. In our tests, PDDL is used to define the task domain.

However, PDDL-based planning frameworks are limited, as they are incapable of handling rigorous numerical calculations[1]. Most approaches perform such calculations via external modules or *semantic attachments*, e.g. [16]. The term semantic attachment was coined by Weyhrauch [111] to describe the association of algorithms to function and predicate symbols via external procedures. However, the effects returned by these semantic attachments are not exploited in identifying *helpful actions* during search and hence do not provide any heuristic guidance, deeming the task unsolvable most often [6]. An action is considered *helpful* if it achieves at least one of the lowest level goals in the relaxed plan to the state at hand [37]. Recently, Bernardini *et al.* [6] developed a PDDL-based temporal planner to implicitly trigger such external calls via a specialized semantic attachments called *external advisors*.

[1]PDDL+ [26], an extension of PDDL supports mixed discrete and continuous non-linear changes.

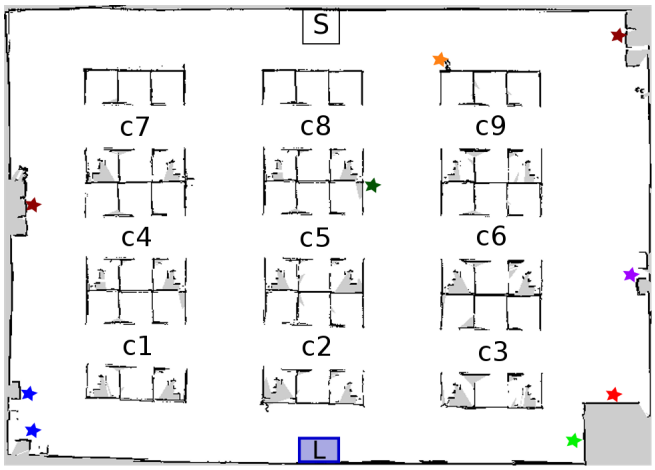


Figure 4.1 Map of the office environment obtained after a SLAM session.

They classify variables into direct (V^{dir}), indirect (V^{ind}) and free (V^{free}). V^{dir} and V^{free} variables are the normal PDDL function variables whose values are changed in the action effects, in accordance with PDDL semantics. V^{ind} variables are affected by the changes in the V^{dir} variables. A change in a V^{dir} variable invokes the external advisor which in turn computes the V^{ind} variables. The *Temporal Relaxed Plan Graph* (TRPG) [12] construction stage of the planner incorporates the indirect variable values for heuristic calculation, thereby synthesizing an efficient goal-directed search. We employ this semantic attachment based approach for the task-motion interface. The overall procedure and the interface layer are discussed in detail in the remainder of this chapter.

Below, we elucidate the PDDL formalism for two different navigation domains that we have considered. It is to be noted that the semantic attachment procedure is domain independent and remains the same in both the domains. But the PDDL domain and problem description differ, as the two domains are different in nature. In the first domain, the underlying roadmap for motion planning does not change during plan computation. However, in the second domain, the roadmap is updated during plan computation. Description of the two domains are detailed below.

4.1.1 Office Domain

We consider a robot navigating in an office environment to collect and deliver documents. The map of the environment following a SLAM session is shown in Fig 4.1 (snapshot of the environment can be seen in Fig. 6.1). The regions $c_1, \ldots, c_9$ are cubicles and L denotes a lift. The robot, starting from region S has to visit certain cubicles to receive documents.

```
(:durative-action goto_region
 :parameters (?v - robot ?from ?to - region)
 :duration (= ?duration 100)
 :condition (at start (robot_in ?v ?from))
 :effect (and (at start (not (robot_in ?v ?from)))
 (at start (increase (triggered ?from ?to) 1))
 (at end (robot_in ?v ?to)) (at end (assign (triggered ?from ?to) 0))
 (at end (increase (act-cost) (external)))
 (at end (increase (goal-trace) (bound))))

(:durative-action collect_document
 :parameters (?v - robot ?r - region)
 :duration (= ?duration 20)
 :condition (and (at start (robot_in ?v ?r)) (at start (> (get ?r) 0))
 (over all (robot_in ?v ?r)))
 :effect (and (at end (collected ?r))(at end (increase (act-cost) 4))))

(:durative-action goto_lift
 :parameters (?v - robot ?from ?to - region)
 :duration (= ?duration 100)
 :condition(at start (robot_in ?v ?from))
 :effect (and (at start (not (robot_in ?v ?from)))
 (at start (increase (triggered ?from ?to) 1))
 (at end (reached ?to)) (at end (assign (triggered ?from ?to) 0))
 (at end (increase (act-cost) (external))))
```

Figure 4.2 A fragment of the PDDL office domain.

Navigating to cubicles/regions is encoded using a single high-level action `goto_region`. Once a robot reaches a cubicle from which a document is to be collected, we assume that a human places the requisite document. Thus, the robot needs to wait at the specific location for a fixed duration of time in which the human places the required document on the robot. This is encoded using a high-level action `collect_document`. These documents then have to be delivered to another floor, which implies using the lift L. Navigating to the lift is modeled using a different high-level action `goto_lift`. This is because, unlike the action `goto_region`, `goto_lift` is to be performed only if the robot has collected all the necessary documents to be delivered. The stars with different colors represent certain unique features assumed to be known and modeled like, printer, trash can, lounge, that aids the robot in better localization. Hence, once the robot knows the regions to visit, then it suffices to perform `goto_region` actions and collect the documents from these regions. However, to synthesize an optimal plan it is necessary to sequence these actions in an order that minimizes the cost function. It is therefore inevitable to obtain the motion costs of these `goto_region` actions, so as to accurately synthesize the optimal plan.

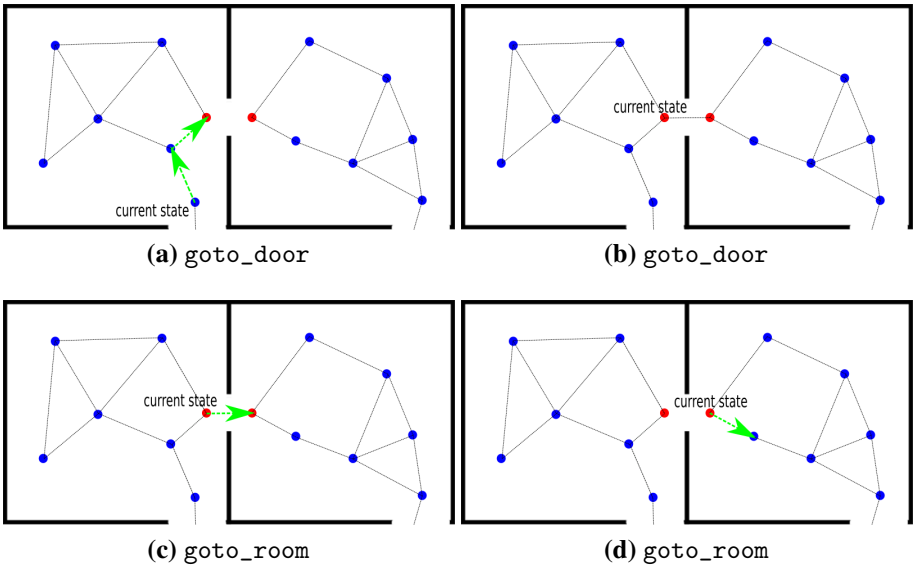
A fragment of the PDDL domain is shown in Fig. 4.2. The PDDL domain dynamics is specified through a set of durative actions (`:durative-action`). We use the following parameters to model these actions: `?v` is the name of the robot, `?from` is the cubicle the robot is currently at and `?to` is the cubicle to which the robot needs to move, `?r` corresponds to the different regions or cubicles in the environment. Each action is described using `:condition` and `:effect`, as defined in Chapter 3, and defines the conditions and effects that holds at the start (`at start`), end (`at end`) or during the entire action interval (`overall`), respectively. The predicate `robot_in` checks if the robot is in a particular region. The function `triggered` encodes the fact that the robot is moving from one cubicle (`from`) to another (`to`). The functions `get` and `collected` model the cubicles from which the document is collected and whether it has been collected. Finally, `act-cost` stores the cost associated with the actions and `goal-trace` keeps the robot state uncertainty bounded. The actions `goto_region` and `goto_lift` invoke the external module call once the facts `(increase (act-cost) (external))` and `(increase (goal-trace) (bound))` are encountered. Here, `act-cost`, `goal-trace` are the direct variables in V^{dir} and `external`, `bound` are the indirect variables V^{ind}. The function `(triggered ?from ?to)` is assigned the numerical value 1 each time the actions are expanded and re-initialized to 0 once the action duration is completed. In this way, the grounded variables `from` (start) and `to` (goal) are communicated to the motion planner. The variables `external` and `bound` returns the motion cost and the goal covariance trace respectively, which are computed by the external module. The action

`collect_document` does not invoke the motion planner. In the problem description, the function `(get ?r)`, where `r` is a free variable denoting cubicles, is initialized to 1 for the cubicles from which the documents are to be collected and to 0 for the remaining.

4.1.2 Corridor Domain

We consider a navigation domain, similar to the one in [47], wherein a robot navigates through a building floor that consists of several rooms connected to one another through a corridor. These rooms have doors, which can either be closed or open, connecting them to the corridor. In addition, some of the rooms are also accessible from each other, through doors in between them. The robot can navigate through the entire building by opening these doors. We assume that once the robot is near to a closed door that directly connects a room to the corridor, a *human opens* the door to allow the robot to pass through. Navigating to rooms can hence be encoded using a single high-level action `goto_room`. However, the doors between any two rooms are automatic, that opens only when the robot is directly in front of the door. This requires the robot to navigate to the door and is encoded using the high-level action `goto_door`. Upon reaching the goal, since the robot is uncertain about its pose, the robot can be anywhere within its current belief distribution. Taking this into account, on reaching the door it is open only if the trace of the pose covariance is within a certain bound η. If the trace is within the bound, an edge is added to the *Probabilistic Roadmap* (PRM) [56] graph between the current node and the nearest node in the next room to which the robot can navigate via the door. Once the robot traverses the door to reach the next room, the newly added edge is removed from the roadmap. This process is illustrated in Fig. 4.3. The addition and deletion of edges is performed by the external module.

A fragment of the corridor PDDL domain is shown in Fig. 4.4. Similar to the office domain, we use the following parameters: `?from` is the room the robot is currently at and `?to` is the room which the robot needs to visit, `?d` is any door. The predicate `visited_in` checks if the robot has visited a room, `hasdoor` checks if the room has a door that opens to another room, and `expanded` model the change in the roadmap. Similar to the previous domain, the actions `goto_room` and `goto_door` invoke the external module call once the fact `(increase (act-cost) (external))` is encountered. Here, `act-cost` is the direct variable in V^{dir} variable and `external` is the indirect variable in V^{ind}. The function `(triggered ?from ?to)` and `(expanded ?r ?d)` are assigned the value of 1 each time the actions are expanded and re-initialized to 0 once the action duration is completed. This is performed so that the grounded variables `from` (start) and `to` (goal) as well as `r` (start) and `d` (goal) are



(a) `goto_door` **(b)** `goto_door`

(c) `goto_room` **(d)** `goto_room`

Figure 4.3 The addition and deletion of an edge to the PRM graph. The red nodes are the ones that are close to the door. (a) Shows a possible path in green, when the `goto_door` action is expanded. Note that there is no edge between the two red colored nodes. (b) Upon satisfying the trace constraint, an edge added between the two nodes close to the door. (c) The `goto_room` action takes the robot to the next room. (d) As the robot navigates towards the first node (red colored node) in the new room, the edge connecting it to the room from which the robot traversed is removed from the roadmap.

communicated to the motion planner. The variables `from`, `to` and `r` are used to denote the rooms and the variable `d` represents the doors available. This can be seen in the `parameters` definition of the actions. The variable `external` returns the motion cost computed by the external module.

4.2 Motion Planning

Independently of the domain, we use a sampling based PRM to instantiate robot poses for the task actions. To begin with, the initial mean and covariance of the robot pose is assumed to be known. This means that the initial state s_0 corresponds to a single pose instantiation q_0. The regions to be navigated to are also instantiated into poses, by sampling from the pose space within each region. Once an action a_i is expanded by the task planner, the corresponding start and goal states, that is s_i and s_{i+1} are communicated to the motion planner. This is facilitated by the functions `triggered` and `expanded`, as detailed in the previous section. For example, the task state s_i might specify that the robot is in cubicle c_2 and the goal state s_{i+1} can be for the robot to reach cubicle c_4. In this scenario $\phi(s_i)$ and $\phi(s_{i+1})$, that is, the mapping from

```
(:durative-action goto_room
 :parameters (?from ?to - room)
 :duration (= ?duration 100)
 :condition (and (at start (robot_in ?from)) (at start
(connected ?from ?to)))
 :effect (and (at start (not (robot_in ?from)))
 (at start (increase (triggered ?from ?to) 1))
 (at end (robot_in ?to)) (at end (assign (triggered ?from ?to) 0))
 (at end (increase (act-cost) (external))) (at end (visited ?to))))


(:durative-action goto_door
 :parameters (?r - room ?d - door)
 :duration (= ?duration 40)
 :condition(and (at start (robot_in ?r)) (at start (hasdoor ?r ?d))
 (over all (robot_in ?r)))
 :effect (and (at start (increase (expanded ?r ?d) 1))
 (at end (assign (expanded ?r ?d) 0))
 (at end (increase (act-cost) (external)))))
```

Figure 4.4 A fragment of the PDDL corridor domain.

states to configurations, correspond to all possible poses such that the robot is in cubicles c_2 and c_4 respectively. Since the set of possible poses is infinite, we randomly sample a set of poses corresponding to each task state s_i. It is to be noted that this sampling is an independent problem and this set is incorporated while building the entire roadmap. For each region s_i, the number of pose instantiations will be denoted by s_i^n and a particular instantiation by $s_i^{n_k}$. With the pose instantiation of s_i as the start node, for each pose instantiation of s_{i+1}, we simulate a sequence of controls along each edge starting from $s_i^{n_k}$ and ending in $s_{i+1}^{n_j}$, estimating the beliefs at the each of these nodes using (3.8)- (9.4). The $s_{i+1}^{n_j}$ that corresponds to the minimum cost is then selected as the goal pose to reach, for the state s_{i+1}. Thereafter, this instantiation becomes the start node when an expansion is attempted from state s_{i+1}. It is true that PRM is in the configuration space and not in the belief space, but the basic problem remains the same since we are essentially finding a sequence of actions that minimizes the objective function which is a function of the resulting beliefs. Our PRM approach is similar to the *Belief Roadmap* (BRM) [89] approach and differs in the way one-step belief updates are performed. Moreover, BRM assume maximum likelihood observations but we do not.

Since we plan in the belief space of the robot state, given the mean and covariance of the starting node we propagate the belief along the edges of the PRM as the roadmap is expanded during the search. Belief update is performed upon reaching a node if a landmark is successfully detected by the robot's perception system. Since we are in the planning phase and yet to obtain observations, we simulate future observations z_{k+1} given the propagated belief $b[\bar{X_{k+1}}]$, the set of landmarks $L_{\mathbb{N}} = l_1, \ldots, l_n$ and the measurement model (8.2). In this work, we model landmarks using AprilTags [79] which are placed on the objects of interest. Given a pose $x \in b[\bar{X_{k+1}}]$, the nominal observation $\hat{z} = h(x, l_i)$ is corrupted with noise to obtain z_{k+1}, which is then used to compute the posterior belief.

4.3 Task-Motion Planning for Navigation

In our approach, the interface between task and motion planning occurs through semantic attachments. Formally, semantic attachment can be defined as

Definition 9. *Semantic attachments is a functional mapping from the set of direct variables to the set of indirect variables, that is,* $\chi : V^{dir} \rightarrow V^{ind}$.

We recall here that for the *office domain* $V^{dir} = \{\texttt{act-cost}, \texttt{goal-trace}\}$ and $V^{ind} = \{\texttt{external}, \texttt{bound}\}$. For the *corridor domain*, we have $V^{dir} = \{\texttt{act-cost}\}$ and $V^{ind} = \{\texttt{external}\}$. The planner receives as input- the PDDL domain, problem description, the

Algorithm 1 TMP for Navigation in Belief Space

Input: $\Psi = (C, \Omega, \phi, \xi, q_0)$: Task-Motion domain, η: Uncertainty budget

while true **do**
  $a_i \leftarrow$ task planning(Ω)
  $\triangleright$ a_i = an action selected to expand the next state
  $\pi^* \leftarrow \emptyset$ // Task-Motion Plan
  if $a_i \in A_s$ **then**
    External module $\leftarrow V^{dir}$
    $\triangleright$ $V^{dir} = \{act - cost, goal - trace\}$
    current task state $\leftarrow s_i$, next task state $\leftarrow s_{i+1}$
    $c \leftarrow \emptyset$, T $\leftarrow \emptyset$
    current task state $\leftarrow \phi(s_i)$, next task state $\leftarrow \phi(s_{i+1})$
    for each $s_{i+1}^{n_j} \in \phi(s_{i+1})$ **do**
      start node $\leftarrow s_i^{n_k}$, goal node $s_{i+1}^{n_j}$
      Belief space search from start node to goal node.
      $c \leftarrow c^j$, $T \leftarrow \tau_i^j$
    end for
    j^* = **arg min** c
    $\tau_i \leftarrow \tau_i^{j^*}$
    $\triangleright$ τ_i is the selected motion plan to arrive at the task state s_{i+1}.
    $V^{ind} \leftarrow$ External module
    $\pi^* \leftarrow$ append(π^*, (a_i, τ_i))
  end if
end while
return π^*

Algorithm 2 Belief space search

Input: Roadmap (sampled poses and edges), start node n with belief (μ_n, Σ_n) corresponding to start state s_i, goal node $(\phi(s_{i+1}))$

1: $\tau_i \leftarrow n$
2: **while** $\phi(s_{i+1})$ not reached **do**
3: **for each** edge from n to n' **do**
4: Propagate the belief (3.8)
5: **if** Landmark within sensing range **then**
6: Compute posterior belief (9.4).
7: **end if**
8: Select n' with minimum cost.
9: $c \leftarrow$ minimum cost, $\tau_i \leftarrow$ append(τ_i,n')
10: $n' = n$
11: **end for**
12: **end while**
13: **return** c, τ_i

shared library and other input parameters. The input parameter specifies the regions/rooms and the corresponding pose instantiations. For the office domain, these pose instantiations are the poses that lie inside the cubicles and for the corridor domain they are the poses that lie inside the rooms. These poses are sampled once the map of the environment is available as described in the previous section.

An overview of our TMP approach is presented in Algorithm 1. The external module computes the V^{ind} values and is invoked only when a change occurs in V^{dir} variables due to the action effects. The PDDL keyword `increase` is overloaded to refer to an encapsulated object [87] and the external module is called if the PDDL action to be expanded has an *effect* of the form `(increase` (v_i^{dir}) (v_j^{ind})`)`, where $v_i^{dir} \in V^{dir}$ and $v_j^{ind} \in V^{ind}$. We denote the set of such actions by A_s. It is to be noted that the elements of this set can vary depending on the requirements of a particular domain. However, the process for achieving the semantic attachments of the external module remains the same. In this thesis, the set $A_s = \{$`goto_region`, `goto_lift`, `goto_room`, `goto_door`$\}$. Every time a v_i^{dir} is changed due to the direct effects of an action $a_i \in A_s$, the values of the respective v_j^{ind} is calculated by the external module, attaching the computed value to the indirect variable v_j^{ind}, thereby updating the state. Once an action a_i is expanded by the task planner, the corresponding start (s_i) and goal (s_{i+1}) task states are communicated to the motion planner through the the function `(triggered ?from ?to)` (line 6). For the task state s_i, the robot pose $\tau_i(0) = \phi(s_i)$ is known since it is the mean of the current belief distribution. For the task state s_{i+1}, each

pose instantiation $s_{i+1}^{n_j} \in \phi(s_{i+1})$ is considered as a goal node (line 9). With $\tau_i(0)$ as the start node, a motion plan is attempted to each of the goal node $s_{i+1}^{n_j}$. The set of feasible motion plans is obtained by performing a search over the roadmap. Along each edge of the roadmap, the belief at s_i is propagated to $s_{i+1}^{n_j}$ by simulating the sequence of controls and observations. We use EKF to compute the appropriate matrices for belief computation as shown in 3.8. The posterior belief is computed at each node if a landmark is detected by the robot's sensor. This belief search process is shown in Algorithm 2. The motion costs and the corresponding feasible motion plans are populated to the sets c and T respectively (line 12). The motion plan that corresponds to minimum cost is then computed as $\tau_i^{j^*}$ (lines 14-15). The computed values by the external module is then passed to the respective indirect variables V^{ind} (line 16), achieving semantic attachments. The corresponding motion plan τ_i and the goal node $s_{i+1}^{n_{j^*}}$ are stored and this goal node subsequently becomes the start node for the roadmap search from s_{i+1}. Consequently, the belief estimates returned by the semantic attachments guide the TRPG in identifying the *helpful actions*, besides providing an efficient heuristic evaluation for the task plan.

For the *office domain*, the feasibility of the motion plan $\tau_i^{j^*}$ is checked by accounting for the trace of the covariance matrix upon reaching a cubicle associated with s_{i+1}, that is, $trace(\Sigma_{s_{i+1}^{j^*}})$. Since the cubicle doors are of specific length, we bound the trace by a constant η. However, the failure of an action a_i to find a feasible motion plan during the current expansion does not mean that it has to be discarded. Feasibility also depends on the sequence of actions performed earlier. A different action sequence prior to a_i can render a_i feasible. Hence infeasible actions are not discarded and are set aside for reattempting later. Consequently the feasibility check is performed for the returned optimal plan π^*. The plan is feasible if for each $a_i \in \pi^*$, the $trace(\Sigma_{s_{i+1}^{j^*}}) < \eta$; else there is no is feasible plan.

4.3.1 Cost Function

So far we have been agnostic about the cost function used while selecting the nodes for expansion. Though our formulation can be adapted to any generic cost functions we use a standard cost function [43]

$$c \doteq M_u c_u + M_G c_G + M_\Sigma c_\Sigma \tag{4.1}$$

where c_u is the control usage, c_G is the distance to goal and c_Σ is the cost due to uncertainty, defined as $trace(\Sigma)$, where Σ is the state covariance associated with the robot belief. M_u, M_G and M_Σ are user-defined weights. For the current node n that is considered for expansion, the cost c is computed for each of the nodes that shares an edge with n. The node with the

minimum c is selected as the next node n^* for expansion. As such, this can be extended to non-myopic planning in a trivial manner, but it is not the current focus of this thesis. It is to be noted that n^* is considered only if it is not already in the expanded path with the n being the last node added to the path. So if n^* leads to a cycle, the next best node n^{**} is selected.

As mentioned in the previous section, in case of the *office domain* we add the condition $c_{\Sigma_g} < \eta$, where $_{\Sigma_g}$ is the trace of the goal state covariance and η is a constant. The cubicle doors have a width of $2m$ and considering maximum uncertainty along the door width we fix $\eta = 3m^2$ as the maximum upper limit and discard the motion plans with $c_{\Sigma_g} > 3$ (see lines 19-24, Algorithm 1). For the *corridor domain*, since the automatic doors are of $1m$ in length, we set an upper bound of $\eta = 0.75m^2$, which corresponds to an uncertainty budget of $0.5m$ in each of the pose component. This check is performed when the robot is at a node directly in front of the door as a result of executing the action `goto_door`. If the estimated covariance is within the uncertainty budget an edge is added between the current node and the nearest node in the next room to which the robot can navigate via the door. Once the robot traverses the door to reach the next room by executing the action `goto_room`, the newly added edge is removed from the roadmap. The process of addition and deletion of an edge occur within the external module as a consequence of the `goto_door` and `goto_room` actions.

4.3.2 Optimality

For a given roadmap, the plan synthesized by our approach is optimal at the task-level. This means that the task plan cost returned by our approach (c^*) is lower than any of the other possible task plan costs (c). Let us denote the optimal plan corresponding to c^* as π^*. Suppose that there exists a plan π with associated cost c such that $c < c^*$. If π and π^* have the same sequence of actions, this is not possible since the action costs are evaluated by the motion planner and for a given roadmap, the motion cost returned is the optimal for each action, giving $c^* \leq c$. If π and π^* have a different sequence of actions, the task planner ensures that the returned plan is optimal, giving $c^* \leq c$. Therefore, in both the case, we have $c^* \leq c$.

4.3.3 Completeness

We provide a sufficient condition under which the probability of our approach returning a plan approaches one exponentially with the number of samples used in the construction of the roadmap. A task planning problem, $\Omega = (S, A, \gamma, s_0, S_g)$ is complete if it does contain any dead-ends [38], that is there are no states from which goal states cannot be reached.

The PRM motion planner is probabilistically complete [55], that is the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap. Therefore, if the motion planner terminates each time it is invoked then probability of finding a plan, if it exists, approaches one.

On the one hand our approach is probabilistically complete; on the other hand, it is also resolution complete since the motion plan feasibility depends on the parameter η. Nevertheless, given a fixed value of η, the probability that the planner fails to return a solution, if one exists, tends to zero as the number of samples approaches infinity. In this sense the best that we can guarantee is probabilistic completeness.

Chapter 5

Multi-robot MPTP

We extend the concepts discussed in Chapter 4 to facilitate multi-robot TMP which corresponds to our contribution **C3**. To this end, we consider a distributed multi-robot TMP framework where robots operating in a known environment can observe each other, thereby facilitating collaborative multi-robot localization. When one robot detects another, the resulting localization uncertainty for both the robots is less than when there is no such mutual observations [92]. This stems from the integration of multi-robot constraints into the joint robot beliefs. In this work we only consider robots mutually observing themselves at the same time. However, it should be noted that multi-robot constraints can also be formulated for different robots observing the same environment at different time instances [42]. As before, the map of the environment is either known *a priori* or is built using a standard SLAM algorithm. At any time k, we denote the robot pose (or configuration q_k) by $x_k \doteq (x, y, \theta)$, the acquired measurement is denoted by z_k and the applied control action is denoted as u_k. We consider a standard motion (8.1) and observation model (8.2) with Gaussian distributed noises as introduced in Chapter 3, that is, $x_{k+1} = f(x_k, u_k) + w_k$ and $z_k = h(x_k) + v_k$, respectively. It is to be noted that we assume data association as solved and hence given a measurement we know the corresponding landmark that generated it. This is not a limitation and our approach can be extended to incorporate reasoning regarding data association, as shown recently in [83].

We now derive the joint multi-robot belief and show how mutual observation influence the inference. A formal definition for the belief of a robot was already given in Chapter 3, however for the convenience of the reader we re-state them below. The motion (8.1) and observation (8.2) models can be written probabilistically as $p(x_{k+1}|x_k, u_k)$ and $p(z_k|x_k)$, respectively. Given an initial distribution $p(x_0)$, the motion and observation models, the

posterior probability distribution at time k can be written as

$$p(x_k|Z_{0:k}, U_{0:k-1}) = \eta\, p(z_k|x_k) \int p(x_k|x_{k-1}, u_{k-1}) b[x_{k-1}] \tag{5.1}$$

where $Z_{0:k} \doteq \{z_0, ..., z_k\}$, $U_{0:k-1} \doteq \{u_0, ..., u_{k-1}\}$ and $b[x_{k-1}] \sim \mathcal{N}(\mu_{k-1}, \Sigma_{k-1})$ is the posterior probability distribution or *belief* at time $k-1$. Similarly, given an action u_k, the propagated belief can be written as

$$b[\bar{x_{k+1}}] = \int p(x_{k+1}|x_k, u_k) b[x_k] \tag{5.2}$$

Given the current belief $b[x_k]$ and the control u_k, the propagated belief parameters can be computed using the standard EKF prediction as

$$\begin{aligned} \bar{\mu}_{k+1} &= f(\mu_k, u_k) \\ \bar{\Sigma}_{k+1} &= F_k \Sigma_k F_k^T + V_k W_k V_k^T \end{aligned} \tag{5.3}$$

where F_k is the Jacobian of $f(\cdot)$ with respect to x_k and V_k is the Jacobian of $f(\cdot)$ with respect to u_k. For brevity, the linearized process noise will be denoted as $R_k = V_k W_k V_k^T$. Upon receiving a measurement z_k, the posterior belief $b[x_{k+1}]$ is computed using the EKF update equations

$$\begin{aligned} K_k &= \bar{\Sigma}_{k+1} H_k^T \left(H_k \bar{\Sigma}_{k+1} H_k^T + Q_k \right)^{-1} \\ \mu_{k+1} &= \bar{\mu}_{k+1} + K_k \left(z_{k+1} - h(\bar{\mu}_{k+1}) \right) \\ \Sigma_{k+1} &= (I - K_k H_k) \bar{\Sigma}_{k+1} \end{aligned} \tag{5.4}$$

where H_k is the Jacobian of $h(\cdot)$ with respect to x_k, K_k is the Kalman gain and $I \in \mathbb{R}^{3\times 3}$ is the identity matrix. In the following we formulate the multi-robot localization problem. For simplicity we consider only two robots r and r', but the formulation can be trivially expanded to incorporate R robots. At any time k, we denote the pose of robot r by x_k^r, the acquired measurement is denoted by z_k^r and the applied control action is denoted as u_k^r. We first consider the case in which there are no mutual observations between the robots. For two

robots r and r', the joint belief at time k is given by

$$\begin{aligned} p(x_k^r, x_k^{r'} | Z_{0:k}^r, Z_{0:k}^{r'}, U_{0:k-1}^r, U_{0:k-1}^{r'}) \\ &= p(x_k^r | Z_{0:k}^r, U_{0:k-1}^r) p(x_k^{r'} | Z_{0:k}^{r'}, U_{0:k-1}^{r'}) \\ &= \eta\, p(z_k^r | x_k^r) \int p(x_k^r | x_{k-1}^r, u_{k-1}^r) b[x_{k-1}^r] \cdot \\ &\qquad p(z_k^{r'} | x_k^{r'}) \int p(x_k^{r'} | x_{k-1}^{r'}, u_{k-1}^{r'}) b[x_{k-1}^{r'}] \end{aligned} \tag{5.5}$$

As seen above the joint belief is factorized into individual beliefs of robot r and r'. Let $x_k = [x_k^r, x_k^{r'}]$ be the joint state, then the EKF prediction can be written as

$$\begin{aligned} \bar{\mu}_k &= [f(\mu_{k-1}^r, u_{k-1}^r), f(\mu_{k-1}^{r'}, u_{k-1}^{r'})] \\ \bar{\Sigma}_k &= F_{k-1} \Sigma_{k-1} F_{k-1}^T + R_{k-1} \end{aligned} \tag{5.6}$$

where F_{k-1}, Σ_{k-1} and R_{k-1} are diagonal matrices. This renders the predicted covariance matrix $\bar{\mu}_k$ diagonal. Since we do not consider mutual observations, the Kalman gain is also a diagonal matrix

$$\begin{aligned} F_{k-1} = \begin{bmatrix} F_{k-1}^r & 0 \\ 0 & F_{k-1}^{r'} \end{bmatrix}, \quad \Sigma_{k-1} = \begin{bmatrix} \Sigma_{k-1}^r & 0 \\ 0 & \Sigma_{k-1}^{r'} \end{bmatrix} \\ R_{k-1} = \begin{bmatrix} R_{k-1}^r & 0 \\ 0 & R_{k-1}^{r'} \end{bmatrix}, \quad K_{k-1} = \begin{bmatrix} K_{k-1}^r & 0 \\ 0 & K_{k-1}^{r'} \end{bmatrix} \end{aligned} \tag{5.7}$$

giving a diagonal covariance matrix Σ_k. As such this corresponds to performing the belief propagation and updates for each robot individually [92].

Now let us consider the case when robots can mutually observe each other. When robot r observes robot r' at time k, the measurement constraint will be denoted by $\zeta_k^{r,r'}$. It is assumed that a common reference frame is established so that the robots can communicate relevant

information with each other. The joint belief at time k is given by

$$\begin{aligned}
&p(x_k^r, x_k^{r'} | Z_{0:k}^r, Z_{0:k}^{r'}, \zeta_k^{r,r'}, U_{0:k-1}^r, U_{0:k-1}^{r'}) \\
&\qquad = p(x_k^r | x_k^{r'}, Z_{0:k}^r, \zeta_k^{r,r'}, U_{0:k-1}^r) p(x_k^{r'} | Z_{0:k}^{r'}, U_{0:k-1}^{r'}) \\
&\qquad = \eta\, p(z_k^r | x_k^r) p(x_k^r | x_k^{r'}, Z_{0:k-1}^r, \zeta_k^{r,r'}, U_{0:k-1}^r) \cdot \\
&\qquad\qquad p(z_k^{r'} | x_k^{r'}) p(x_k^{r'} | Z_{0:k-1}^{r'}, U_{0:k-1}^{r'}) \\
&\qquad = \eta\, p(z_k^r | x_k^r) p(\zeta_k^{r,r'} | x_k^r, x_k^{r'}) \int p(x_k^r | x_{k-1}^r, u_{k-1}^r) b[x_{k-1}^r] \cdot \\
&\qquad\qquad p(z_k^{r'} | x_k^{r'}) \int p(x_k^{r'} | x_{k-1}^{r'}, u_{k-1}^{r'}) b[x_{k-1}^{r'}] \qquad (5.8)
\end{aligned}$$

The measurement likelihood term $p\left(\zeta_k^{r,r'} | x_k^r, x_k^{r'}\right)$ introduces cross correlations in Σ_k. This is because the measurement Jacobian is computed with respect to x_{k-1}^r and $x_{k-1}^{r'}$ [74] unlike the previous scenario where the measurement Jacobian was computed separately for each r using its corresponding x_{k-1}^r. We assume that robot r measures the range and bearing of r', that is, $\zeta_k^{r,r'} = [d_k^{r,r'}, \phi_k^{r,r'}]^T$ where

$$\begin{aligned}
d_k^{r,r'} &= \sqrt{(x_k^{r'}(1) - x_k^r(1))^2 + (x_k^{r'}(2) - x_k^r(2))^2} \\
\phi_k^{r,r'} &= \arctan\left(\frac{x_k^{r'}(2) - x_k^r(2)}{x_k^{r'}(1) - x_k^r(1)}\right) - x_k^r(3) \qquad (5.9)
\end{aligned}$$

Thus the Jacobian H_{k-1}, which is the partial derivative of the measurement function with respect to the joint state is

$$\begin{aligned}
H_{k-1} &= \begin{bmatrix}
\frac{\partial d_k^{r,r'}}{\partial x_k^r(1)} & \frac{\partial d_k^{r,r'}}{\partial x_k^r(2)} & \frac{\partial d_k^{r,r'}}{\partial x_k^r(3)} & \frac{\partial d_k^{r,r'}}{\partial x_k^{r'}(1)} & \frac{\partial d_k^{r,r'}}{\partial x_k^{r'}(2)} & \frac{\partial d_k^{r,r'}}{\partial x_k^{r'}(3)} \\
\frac{\partial \phi_k^{r,r'}}{\partial x_k^r(1)} & \frac{\partial \phi_k^{r,r'}}{\partial x_k^r(2)} & \frac{\partial \phi_k^{r,r'}}{\partial x_k^r(3)} & \frac{\partial \phi_k^{r,r'}}{\partial x_k^{r'}(1)} & \frac{\partial \phi_k^{r,r'}}{\partial x_k^{r'}(2)} & \frac{\partial \phi_k^{r,r'}}{\partial x_k^{r'}(3)}
\end{bmatrix} \\
&= \begin{bmatrix}
-\frac{(x_k^{r'}(1) - x_k^r(1)}{d_k^{r,r'}} & -\frac{(x_k^{r'}(2) - x_k^r(2))}{d_k^{r,r'}} & 0 & \frac{(x_k^{r'}(1) - x_k^r(1)}{d_k^{r,r'}} & \frac{(x_k^{r'}(2) - x_k^r(2))}{d_k^{r,r'}} & 0 \\
\frac{(x_k^{r'}(2) - x_k^r(2))}{(d_k^{r,r'})^2} & -\frac{(x_k^{r'}(1) - x_k^r(1))}{(d_k^{r,r'})^2} & -1 & -\frac{(x_k^{r'}(1) - x_k^r(1))}{(d_k^{r,r'})^2} & \frac{(x_k^{r'}(2) - x_k^r(2))}{(d_k^{r,r'})^2} & 0
\end{bmatrix}
\end{aligned} \qquad (5.10)$$

The first time when a mutual observation is incorporated (say at time k), the measurement Jacobian introduces cross correlations in Σ_k and thereafter the matrices are no longer diagonal.

```
(:durative-action goto_room
:parameters(?from1 ?from2 ?to1 ?to2 - room ?r1 ?r2 - robot)
:duration (= ?duration 100)
:condition (and (at start (robot_in ?r1 ?from1)) (at start
 connected ?from1 ?to1))(at start (robot_in ?r2 ?from2))
(at start (connected ?from2 ?to2)))
:effect (and (at start (not (robot_in ?r1 ?from1)))(at start
(not (robot_in ?r2 ?from2)))(at start (increase
(triggered ?r1 ?from1 ?to1 ?r2 ?from2 ?to2) 1))(at end
(robot_in ?r1 ?to1))(at end (robot_in ?r2 ?to2))(at end
(assign (triggered ?r1 ?from1 ?to1 ?r2 ?from2 ?to2) 0))
(at end (increase (act-cost) (external)))
(at end (visited ?to1)) (at end (visited ?to2)))
```

Figure 5.1 A fragment of the PDDL room domain.

5.1 Approach

A fragment of the corresponding PDDL domain is shown in Fig. 5.1. The external module computes the V^{ind} values and is invoked only when a change occurs in the V^{dir} variables due to action effects. The PDDL keyword `increase` is overloaded to refer to an encapsulated object [87] and the external module is called if the PDDL action to be expanded has an *effect* of the form `(increase` (v_i^{dir}) (v_j^{ind})`)`, where $v_i^{dir} \in V^{dir}$ and $v_j^{ind} \in V^{ind}$. Once such an action a_i is expanded by the task planner, the corresponding start and goal states of robot r, that is, s_i^r and s_{i+1}^r are communicated to the motion planner. This is facilitated through the function `(triggered ?r1 ?from1 ?to1 ?r2 ?from2 ?to2) 1)`. This specifies that robot $r1$ is navigating from `from1` to `to1` and that robot $r2$ is navigating from `from2` to `to2`, where `from1`, `from2` and `to1`, `to2` are free variables denoting the start and goal states (corresponds to different rooms) of robots $r1$ and $r2$ respectively. In PDDL, the symbols starting with question marks denote variables and the types they represent (room or robot in our case) can be seen in the action parameters in Fig. 5.1. The function `triggered` is assigned the value of 1 each time the actions are expanded and re-initialized to 0 once the action duration is completed. This is performed so that the grounded variables are communicated to the motion planner. For each region s_i, the number of pose instantiations will be denoted by s_i^n and a particular instantiation by $s_i^{n_k}$. For each robot r, with the pose instantiation of s_i^r as the start node, for each pose instantiation of s_{i+1}^r, we simulate a sequence of controls and observations along each edge of the roadmap starting from s_i^{r,n_k} and ending in s_{i+1}^{r,n_j}, estimating the beliefs at the each of these nodes using (5.8). The s_{i+1}^{r,n_j} that corresponds

to the minimum cost is then selected as the goal pose for robot r for the state s^r_{i+1}. Thereafter, this instantiation becomes the start node when an expansion is attempted from state s^r_{i+1} for robot r. It is true that PRM is in the configuration space and not in the belief space but fundamentally, planning in the belief space is just increasing the state space of the robot (for example by including covariance). The basic problem remains the same since we are essentially finding a sequence of actions that minimizes the objective function which by itself is now a function of beliefs at different time steps. Our PRM approach is similar to BRM [89] and differs in the way one-step belief updates are performed. Moreover, BRM assume maximum likelihood observations but we do not.

Though our formulation can be adapted to any generic cost function we use a standard cost function [43], $c \doteq M_u c_u + M_G c_G + M_\Sigma c_\Sigma$, where c_u is the control usage, c_G is the distance to the goal and c_Σ is the cost due to uncertainty, defined as $trace(\Sigma)$, where Σ is the state covariance associated with the robot belief. M_u, M_G and M_Σ are user-defined weights. The cost of the selected motion plan is then returned to the task planner as the cost of the corresponding action. The variable `external` returns the motion cost computed by the external module and achieves semantic attachment by passing its value to the task-level cost variable `act-cost` (see Fig. 5.1). The task-motion plan for changing the task state of the robot from the state s_i to s_{i+1} is the ordered tuple of the action a_i and the corresponding optimal path. The tuple is appended for all the task-level actions to generate the complete task-motion plan.

5.2 Simulating Future Observations

Since we plan in the belief space of the robot's state, given the mean and covariance of the starting node we propagate the belief along the edges of the PRM as the roadmap is expanded during the search. Belief update is performed upon reaching a node if a landmark is successfully detected by the robot's perception system. Since mutual observation between robots are explicitly considered the update is also performed if robot r observes r'. In our experiments a multi-robot constraint $\zeta^{r,r'}_{k+1}$ is formulated if r' is within 4 m of r (has been set to 4 just for pseudo-realism). Since we are in the planning phase and yet to obtain observations, we simulate future observations z_{k+1} and $\zeta^{r,r'}_{k+1}$ by corrupting the nominal observations with noise.

5.3 Optimality and Completeness

The multi-robot formulation preserves the notions of optimality and completeness discussed in Chapter 4.

Chapter 6

Experimental Results

In this chapter we discuss our contribution **C4** by validatng our MPTP approach for single and multiple robots as discuused in Chapter 4 and Chapter 5, respectively. We use the temporal POPF-TIF [6] as our task planner by customizing it to achieve semantic attachments of an external module. The external module performs a PRM-based planning in the belief space and is implemented as a dynamically loaded shared library that is passed as an input to the planner. The enumeration into direct variables V^{dir} and indirect variables V^{ind} are listed in the external module. The performance are evaluated on an Intel® Core i7-6500U under Ubuntu 16.04 LTS.

6.1 Single-robot Scenarios

We validate our approach in two different robot navigation domains, namely *office domain* and *corridor domain* as described in Chapter 4. First, we present the motion and sensor models used in our experiments[1]. Then, we discuss the metrics devised to evaluate the usefulness and validity of our approach. Finally, we present the evaluation of our approach in the two navigation domains using the devised metrics.

[1] To simplify the notation, most variables are presented without time indexes.

6.1.1 Motion and Sensor Model

The robot dynamics is modeled using the following non-linear model [105]

$$\begin{aligned} x_{k+1}(1) &= x_k(1) + \delta_{trans} \cdot \cos(x_k(3) + \delta_{rot1}) \\ x_{k+1}(2) &= x_k(2) + \delta_{trans} \cdot \sin(x_k(3) + \delta_{rot1}) \\ x_{k+1}(3) &= x_k(3) + \delta_{rot1} + \delta_{rot2} \end{aligned} \tag{6.1}$$

where $x_k \doteq (x, y, \theta)$, is the robot pose at time k with $x_k(1) = x, x_k(2) = y$ and $x_k(3) = \theta$ and $u_k \doteq (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$ is the applied control. The model in (10.56) assumes that the robot ideally implements the following commands in order: rotation by an angle of δ_{rot1}, translation of δ_{trans} and a final rotation of δ_{rot2} orienting the robot in the required direction[2]. It is to be noted that the robot accrue translational and rotational errors while executing u_k.

In the EKF, the Jacobian of the state transition model with respect to the state x_k denoted by F_k (see (3.8) and (9.4)) is obtained by linearizing the state transition function about the mean state at x_k and is given by

$$F_k = \begin{bmatrix} \frac{\partial f}{\partial x_k(1)} & \frac{\partial f}{\partial x_k(3)} & \frac{\partial f}{\partial x_k(3)} \\ \frac{\partial f}{\partial x_k(1)} & \frac{\partial f}{\partial x_k(3)} & \frac{\partial f}{\partial x_k(3)} \\ \frac{\partial f}{\partial x_k(1)} & \frac{\partial f}{\partial x_k(3)} & \frac{\partial f}{\partial x_k(3)} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\delta_{trans} \cdot \sin(x_k(3) + \delta_{rot1}) \\ 0 & 1 & \delta_{trans} \cdot \cos(x_k(3) + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

Similarly, the linearized process noise, $R_k = V_k W_k V_k^T$, is obtained by computing the Jacobian of V_k

$$V_k = \begin{bmatrix} \frac{\partial f}{\partial \delta_{rot1}} & \frac{\partial f}{\partial \delta_{trans}} & \frac{\partial f}{\partial \delta_{rot2}} \\ \frac{\partial f}{\partial \delta_{rot1}} & \frac{\partial f}{\partial \delta_{trans}} & \frac{\partial f}{\partial \delta_{rot2}} \\ \frac{\partial f}{\partial \delta_{rot1}} & \frac{\partial f}{\partial \delta_{trans}} & \frac{\partial f}{\partial \delta_{rot2}} \end{bmatrix} = \begin{bmatrix} -\delta_{trans} \cdot \sin(x_k(3) + \delta_{rot1}) & \cos(x_k(3) + \delta_{rot1}) & 0 \\ \delta_{trans} \cdot \cos(x_k(3) + \delta_{rot1}) & \sin(x_k(3) + \delta_{rot1}) & 0 \\ 1 & 0 & 1 \end{bmatrix} \tag{6.3}$$

The noise covariance matrix W_k is formulated as below with α_1 to α_4 being the robot-specific error parameters [105] modeling the accuracy of the robot motion

$$W_k = \begin{bmatrix} \alpha_1 \cdot \delta_{rot1}^2 + \alpha_2 \cdot \delta_{trans}^2 & 0 & 0 \\ 0 & \alpha_3 \cdot \delta_{trans}^2 + \alpha_4 \cdot (\delta_{rot1}^2 + \delta_{rot2}^2) & 0 \\ 0 & 0 & \alpha_2 \cdot \delta_{trans}^2 + \alpha_1 \cdot \delta_{rot2}^2 \end{bmatrix} \tag{6.4}$$

[2]The state transition model form of (10.56) is given in (8.1).

As for the sensor model, we use a landmark-base model

$$z_k = \begin{bmatrix} r = \sqrt{(l_i(1) - x_k(1))^2 + (l_i(2) - x_k(2))^2} \\ \\ \phi = \arctan(\frac{l_i(2) - x_k(2)}{l_i(1) - x_k(1)}) - x_k(3) \end{bmatrix} + v_k \ , \ v_k \sim \mathcal{N}(0, Q_k) \tag{6.5}$$

where r and ϕ are the range and bearing of the i-th landmark l_i relative to the robot frame. The sensor model is linearized to obtain the Jacobian H_k, which is the partial derivative of the measurement function with respect to the robot state[3].

$$H_k = \begin{bmatrix} \frac{\partial r}{\partial x_k(1)} & \frac{\partial r}{\partial x_k(2)} & \frac{\partial r}{\partial x_k(3)} \\ \frac{\partial \phi}{\partial x_k(1)} & \frac{\partial \phi}{\partial x_k(2)} & \frac{\partial \phi}{\partial x_k(3)} \end{bmatrix} = \begin{bmatrix} -\frac{(l_i(1) - x_k(1))}{r} & -\frac{(l_i(2) - x_k(2))}{r} & 0 \\ \frac{(l_i(2) - x_k(2))}{r^2} & -\frac{(l_i(1) - x_k(1))}{r^2} & -1 \end{bmatrix} \tag{6.6}$$

We would like to reiterate the fact that since we are in the planning phase, the nominal observation $\hat{z} = h(x, l_i)$ is corrupted with noise to simulate future observations.

6.1.2 Plan Metrics

To benchmark our approach we consider four different cost formulations that differ in their motion cost computation and thereby the task-level action costs. Though our formulation can be adapted to any general cost function (see Chapter 4), we choose the following four cost functions to demonstrate the efficiency of our approach:

- *Euclidean cost*: The motion planner is never called and the task cost are evaluated computing the Euclidean distance c_{euc} between the geometric instantiations of s_i and s_{i+1}, that is, between $\tau_i^j(0)$ and $\tau_i^j(1)$. Here $c \doteq c_{euc}$.

- $\sigma-$*Euclidean cost*: This configuration evaluates the motion cost as the sum of Euclidean distance between $\tau_i(0)$ and $\tau_i(1)$ and the cost due to uncertainty, defined as $c_\Sigma = trace(\Sigma)$, where Σ is the covariance at each node of τ_i. The general form of this cost function is $c \doteq M_{euc}c_{euc} + M_\Sigma c_\Sigma$.

- *PETLON cost*: In this configuration, the motion planner returns the trajectory length or the geometric-level cost of traversing from s_i to s_{i+1}, that is, from $\tau_i^j(0) \in \phi(s_i)$ to $\tau_i^j(1) \in \phi(s_{i+1})$. The general form of the cost for this configuration is $c \doteq M_u c_u + M_G c_G$, where c_u is the control usage and c_G is the distance to goal. Since we assume

[3]The measurement function form of (10.58) is given in (8.2).

straight line path between two sampled poses, the applied control for translation, that is δ_{trans} represents the trajectory length. We note here that the motion planner in PETLON [70] computes the geometric-level cost of traversing from one state to another and hence this configuration will be used to compare MPTP with PETLON.

- *MPTP cost*: In this configuration, we use the cost function as defined in Chapter 4, that is, $c \doteq M_u c_u + M_G c_G + M_\Sigma c_\Sigma$, where c_u is the control usage, c_G is the distance to goal and c_Σ is the cost due to uncertainty. It is noteworthy that *PETLON cost* is subsumed in *MPTP cost* since *MPTP cost* is fundamentally *PETLON cost* added with the cost due to uncertainty.

6.1.3 Office Domain

This domain is simulated in Gazebo [58] by constructing an office environment of $36m \times 25m$; top view of the simulated environment is shown in Fig. 6.1. We note here that the landmarks considered in this domain are the objects outside the cubicles like printers, trash cans, lounge, vending machines and book-shelves. The robot is required to collect documents from different cubicles, and the documents are then taken to the next floor via the lift *L*.

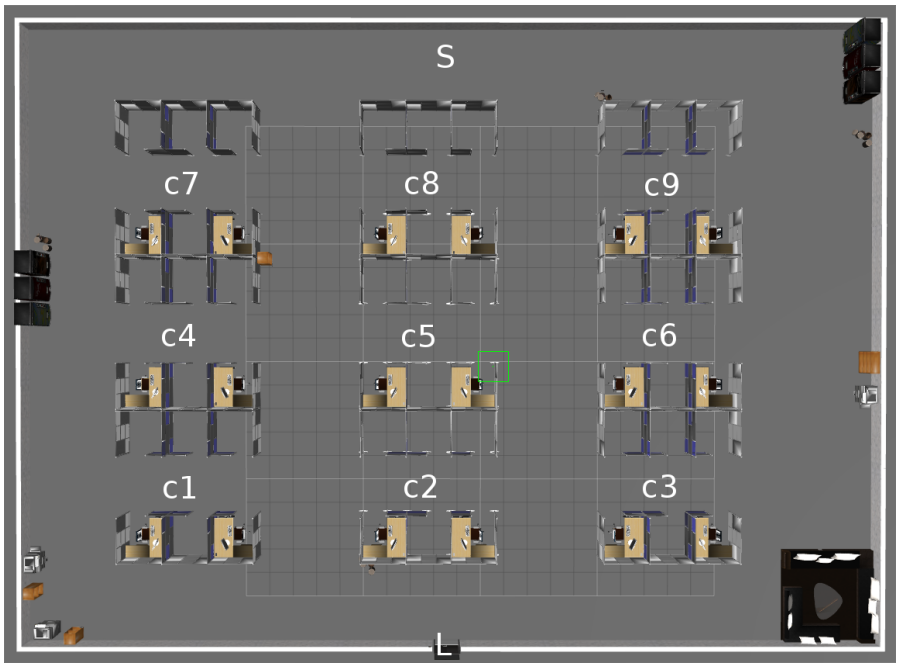


Figure 6.1 Top view of the simulated environment in Gazebo. See *office domain* in Chapter 4 for a detailed description.

Validation

We first demonstrate the need for a combined TMP for navigation. Unless otherwise stated, the panning times presented is an average for 25 different planning sessions. Consider the following scenario in which the robot is required to collect documents from the cubicles $c3$, $c4$, $c6$ and $c9$. We first run the planner with *Euclidean cost* to synthesize the task plan. We remind that in this configuration the motion planner is never called and the action costs are evaluated by considering the Euclidean distance between the start and goal regions. The plan synthesized is $S \rightarrow c3 \rightarrow c4 \rightarrow c6 \rightarrow c9 \rightarrow L$. This plan is then given to the motion planner, to compute the corresponding cost due to uncertainty c_Σ which is the trace of the robot state covariance. The task planning cost and the motion planning cost are added to estimate the overall planning cost, which equated to 298.84. The addition of the two costs is possible because we first compute the task plan which is then passed to the motion planner to compute the cost due to uncertainty. Therefore the overall planning cost is the task planning cost combined with motion planning cost. In the same way, the overall planning time was computed to be 0.94 ($\pm$0.09) seconds by adding the time for task planning and motion planning, respectively. Next, we ran the planner with $\sigma-$*Euclidean cost*, returning the plan $S \rightarrow c4 \rightarrow c9 \rightarrow c6 \rightarrow c3 \rightarrow L$, in 1.28 ($\pm$0.06) seconds with a total cost of 90.89. This configuration evaluates the motion cost as the sum of Euclidean distance and the cost due to uncertainty. It is seen that there is a significant difference in the plan quality as the cost is improved by a factor of 3 for $\sigma-$*Euclidean cost*. This difference in cost is attributed to the different task sequence synthesized. Essentially, *Euclidean cost* corresponds to planners that pre-compute motion costs of all task-level actions or use an admissible heuristic for the same (for example, the approach in [112]). The task plan is then given to the motion planner for execution, assuming that such a motion plan exists. In contrast, $\sigma-$*Euclidean cost* checks for the motion feasibility and estimates the motion costs while expanding each task-level action and thus corresponds to an integrated TMP approach as discussed in this thesis. The difference in plan quality between *Euclidean cost* and $\sigma-$*Euclidean cost* clearly demonstrates the efficiency of a combined TMP approach as opposed to performing task planning and motion planning separately. Though our considered scenario is much less knowledge-intensive than real-world scenarios, the above example conveys the need for a combined task-motion planner.

Next, we run the planner with *PETLON cost* and *MPTP cost* to demonstrate the advantage of planning in belief space, that is using our MPTP approach. We recall here that similar to PETLON [70], with *PETLON cost*, the motion planner evaluates the geometric-level cost of traversing $\tau_i(0)$ to $\tau_i(1)$, whereas with *MPTP cost*, in addition to considering the

	d	Overall time (s)			Cost		
		$c = 2$	$c = 4$	$c = 6$	$c = 2$	$c = 4$	$c = 6$
MPTP cost	1	1.34 ± 0.05	2.24 ± 0.15	-	83.84	90.27	-
	1.5	3.41 ± 0.08	7.16 ± 0.12	14.04 ± 0.09	88.18	101.01	237.59
	2	9.11 ± 1.17	28.48 ± 1.19	46.15 ± 2.23	92.32	126.96	260.092
		$c = 2$	$c = 4$	$c = 6$	$c = 2$	$c = 4$	$c = 6$
PETLON cost	1	0.47 ± 0.02	0.77 ± 0.04	1.77 ± 0.01	47.80	84.88	161.47
	1.5	3.17 ± 0.03	4.91 ± 0.02	7.10 ± 0.10	55.77	95.74	174.90
	2	6.08 ± 0.11	9.86 ± 0.17	15.14 ± 1.09	56.19	95.77	181.06

Table 6.1 Overall planning time and cost returned while running the task-motion planner with *MPTP cost* and *PETLON cost*. The average number of samples per square meter is denoted by d. $c = 2, 4$ and 6 denotes the number of cubicles to be visited, increasing the task-level complexity. '-' denotes the fact that no plan is found as the condition $\eta < 1$ is violated.

geometric-level cost of traversing, the cost due to uncertainty is also incorporated. We consider a scenario in which the robot has to collect a document from cubicle $c3$. The planned trajectories in both the scenarios with the corresponding covariance estimated at each node (only the (x,y) portion is shown) is shown in Fig. 6.2. Clearly, the belief space task-motion planner (*MPTP cost*) returns a route which is rich in sensor information (see Fig. 6.2 in the mid), enabling effective localization. *PETLON cost* returns the shortest path trajectory but with an increased robot state uncertainty. Fig. 6.2 on the right hand side shows the traces of true robot state for 25 different simulations while running on *MPTP cost*—the initial state being sampled from the known initial belief.

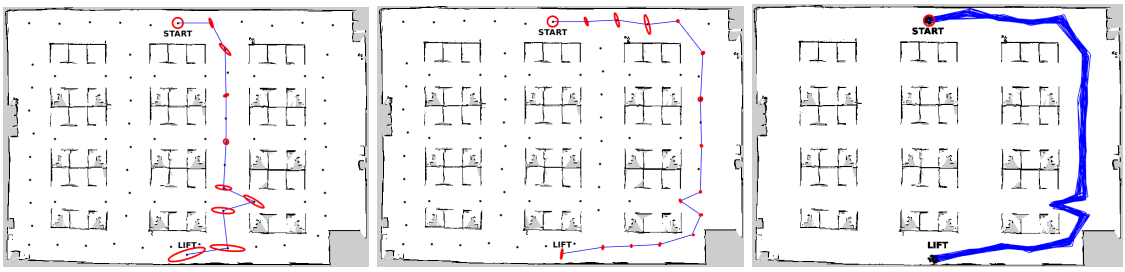


Figure 6.2 (*left* and *center*) The propagated belief distributions along the planned paths for *PETLON cost* and *MPTP cost*. The belief estimates for a single planning instantiation corresponding to a unique set of simulated observations are shown. Black dots represent the sampled poses. (*left*) Shortest path route that corresponds to *PETLON cost*. (*center*) Belief space planning corresponding to *MPTP cost*, returning an information rich route. (*right*) Traces of robot's true state while starting from the initial belief– run with *MPTP cost*.

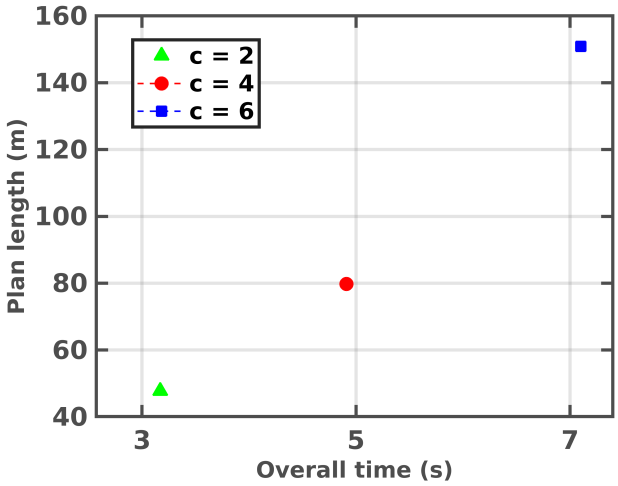


Figure 6.3 Plan length with overall planning time. MPTP is run with *PETLON cost* and a sampling density of $d = 1.5$.

Scalability

We test the scalability of our approach by increasing the task-level complexity. We run our planner on three different scenarios where 2, 4, 6 number of cubicles ($\mathsf{c} = 2,4,6$) are to be visited to collect the corresponding number of documents. This results in evaluating more task-level actions, escalating the task level complexity. We also test these scenarios on varying levels of sample densities. We choose $d = 1, 1.5, 2$, where $d = i$ corresponds to an average of i samples per square meter. The tests are run using *MPTP cost* and *PETLON cost*. The overall planning time and the returned cost can be seen in Table 12.2. While we ran with the MPTP cost, for $d = 1$ and $\mathsf{c} = 6$, no feasible motion plan is found since the condition $\eta < 1$ is violated. However, for higher sample densities, a feasible motion plan is found. The plan quality is increased with increase in d, but at the expense of exponentially increasing computation time. It is clearly seen that for our considered scenario $d = 1.5$ can be chosen, without much loss of plan quality.

In [70], TMP for navigation is performed by evaluating the geometric cost of traversing. They consider a scenario in which nine objects are placed at different locations. Two objects from among them are to be collected and delivered to a person such that the geometric cost of traversing is minimum. They report a total planning time of about 15 seconds with a plan length of 37 m. Though the environment considered in [70] is larger than ours, to provide a comparison with PETLON, we run our task-motion planner with *PETLON cost* and evaluate the planning time with respect to the plan length. In comparison, MPTP with *PETLON cost* fares superiorly with respect to increased task-level complexity. To demonstrate this, we first

(a) **(b)**

Figure 6.4 (a) Willow Garage world with nine objects whose instances are marked as green blobs. The optimal path when two objects are to be collected is shown in blue. The planner is run with *PETLON cost*. (b) Overall planning time with increasing number of objects to be collected for delivering.

consider three scenarios where 2, 4, and, 6 documents are to be collected to be delivered to the next floor. The results can be seen in Table 12.2 under the *PETLON cost* section. We note here that for $d = 1.5$ and collecting 6 documents ($\mathtt{c} = 6$) MPTP with *PETLON cost* took only about 7 (± 0.34) seconds with a plan length of about 150 m (see Fig. 6.3). To provide a better comparison, we also evaluate our approach by considering a much larger environment, the Willow Garage world of $58m \times 45m$ as shown in Fig. 6.4(a). In this example, the robot (at start) needs to collect any two objects from among nine different objects (location of objects marked as green blobs), and deliver it to a person at the goal location (shown in red). We ran our planner with *PETLON cost*, returning an optimal plan of length 53.94 m in 3.69 ($\pm$ 0.09) seconds. We recall here that for the same scenario, PETLON [70] report a planning time of about 15 seconds for a plan length of 37 m. In contrast, MPTP with *PETLON cost* is almost three faster. This clearly elucidates the superiority of our approach. PETLON first computes a task plan using an admissible heuristic which is then sent to the motion planner for actual cost evaluation. This cost refinement process is iterated until the optimal plan is found. MPTP does not require such an iteration since it evaluates the motion cost using semantic attachments as the action is expanded by the task planner. The scalability to increasing task complexity is tested by varying the number of objects to be collected (see Fig. 6.4(b)). The task in which four objects are to be collected was completed in only about 25 ($\pm$ 1.64) seconds. Therefore MPTP reveals to be much faster than PETLON and is robust to the increasing number of objects and map size.

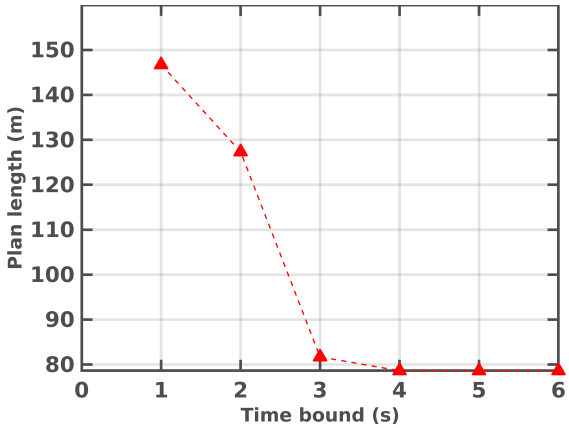


Figure 6.5 Anytime property of MPTP. Valid solutions are returned even when strict bounds are placed on the planning time.

POPF-TIF supports *anytime* planning which means that the planner searches for improved solutions until it has exhausted the search space or is interrupted. Specifically, POPF-TIF is run with a `-n` flag to activate anytime search. A time bound may be specified with the flag `-tx`, where `x` is the time bound in seconds and is used in situations with strict time bounds where optimality is sacrificed. We demonstrate this by considering the Willow Garage world in which the robot needs to collect any three documents from among the nine objects and deliver it to a person. We start with a time bound of 1 second and increment it by a second until an optimal solution is found. The result is plotted in Fig. 6.5. As the time bound is incremented, the plan quality is increased and for a planning time bound of 4 seconds, the optimal plan length of 78.63 m is returned.

We stress here the fact that in this work we are mainly concerned with planning and the synthesized plans are given to the robot for execution. Thus, any such execution approach may be employed. In this work, the generated plans are executed with a TurtleBot robot in the simulated Gazebo environment. We use AprilTags [79] to identify objects like printers, trash cans, as landmarks. TurtleBot robot in front of one such landmark is seen in Fig. 6.6. A ROS-based architecture has been developed to implement the approach. Belief estimation is carried out using EKF. We note here that presently we consider static obstacles while planning and therefore the planned trajectories are collision-free. However, to be robust to dynamic obstacles, the plan execution is trivially extended to employ any collision avoidance approach in dynamic environments [82, 113]. Snapshots of dynamic obstacle avoidance during the execution of a plan can be seen in Fig. 6.7. As seen in the figure, dynamic obstacles are simulated using TurtleBot robots (white in figure). We now report here the execution time for the scenario discussed in Section 6.1.3. When 2, 4, 6 number of cubicles are to be

visited to collect the corresponding number of documents, the execution times are 140.21s ($\pm$ 3.11s), 366.40s ($\pm$ 4.99s), and 664.71s ($\pm$ 16.28s), respectively. We note here that the execution time varies with robot and its control limits.

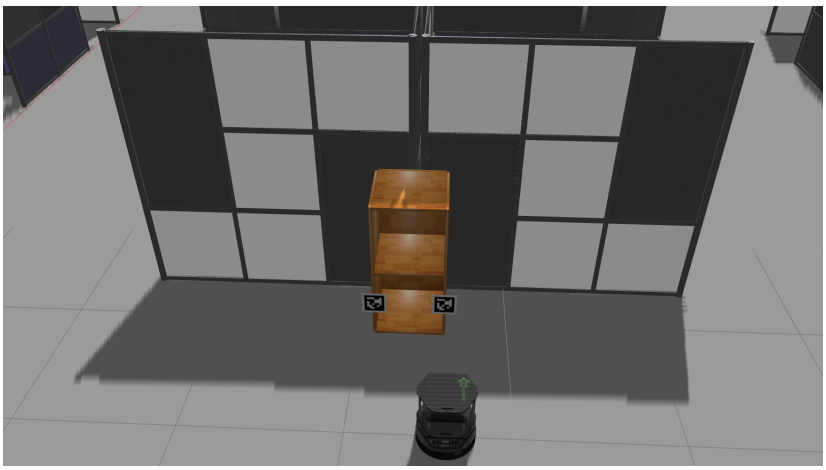


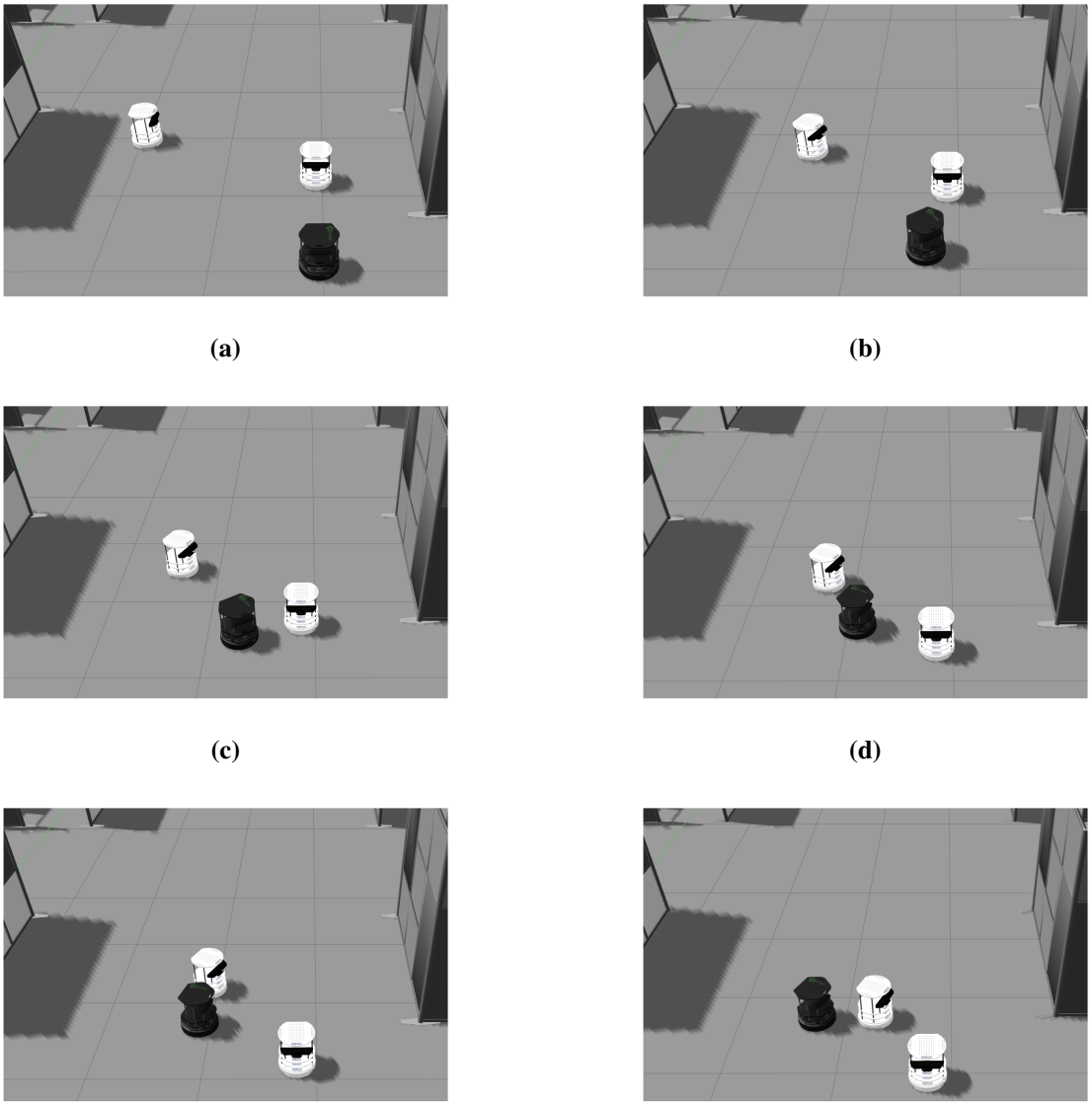
Figure 6.6 A robot in front of AprilTags which provide the transformation between the robot pose and the landmark pose.

6.1.4 Corridor Domain

Our *corridor domain* (see Chapter 4 for a detailed description) is a variant of the robot navigation domain in [47]. However, they treat it as a task planning problem assuming that feasible motion plans exist for the synthesized task plans. In contrast, we perform task-motion planning. In this domain of $12m \times 25m$, a mobile robot, starting from a given room, navigates an office floor to visit a set of rooms that are selected randomly. The office floor has ten rooms and the robot is initially located in room 1. All the rooms are connected to each other through the central corridor. In addition, five rooms are directly connected with each other via doors which need to be opened by the robot. The goal is to visit a set of rooms R that are randomly selected for each run. Since these visits have to be carried out expending as less cost as possible, the robot needs to assess the accessibility between the rooms that are directly connected to each other via a door. This is facilitated through the `goto_door` action as discussed in Chapter 4. The map of the building floor is as shown in Fig. 6.8.

Validation and Scalability

First, we run the planner with *MPTP cost*. For a fixed set cardinality $|R|$ (set elements are the rooms to be visited), 25 trials are performed, where the set elements are selected randomly



(a) **(b)**

(c) **(d)**

(e) **(f)**

Figure 6.7 A robot avoiding a couple of dynamic obstacles (white TurtleBot robots) during execution. Our approach is not restrictive to any particular execution strategy and any approach that employs dynamic obstacle avoidance may be used.

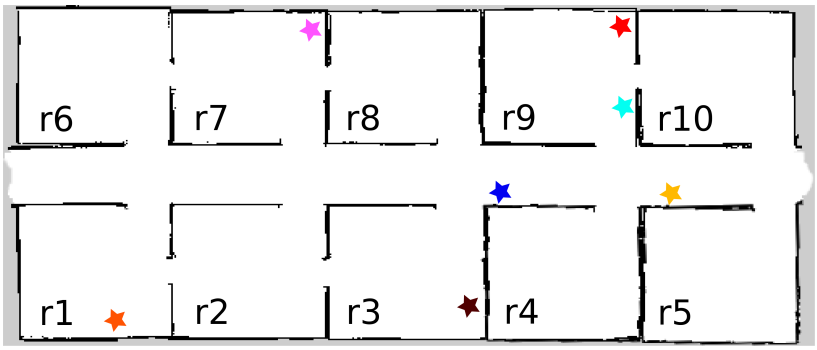


Figure 6.8 Map of the building floor environment with half the rooms connected directly by doors. The stars with different colors represent landmarks that aids the robot in better localization.

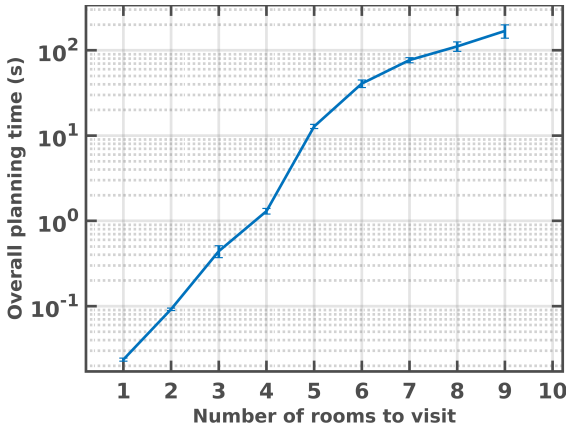


Figure 6.9 Overall task-motion planning time for different number of rooms that need to be visited in log scale. Planning times are the average for 25 different runs.

for each trial. The average planning time for each of them is shown in Fig. 6.9. While the planning time does scale exponentially with $|R|$, the plan for $|R| = 9$ is computed in less than 3 minutes. The work in [47] evaluates the task planning performance on a similar domain randomly selecting the number of rooms to visit in each trial. Since MPTP performs task-motion planning, the overall MPTP planning times with increasing $|R|$ is greater than those reported in [47]. However the graph of $|R|$ with planning time (Fig. 6.9) follows a similar trend to that reported in [47]. It is noteworthy that for a given $|R|$, the difference in MPTP planning time and the planning time reported in [47] is significantly less.

Next, we run the planner with *PETLON cost* and *MPTP cost*. We consider a scenario in which the robot, starting from room $r1$, has to visit rooms $r2$ and $r3$. As seen in Fig. 6.8, rooms $r1$, $r2$ and $r2$, $r3$ are also connected by doors between them. Fig. 6.10 on top-left and top-right shows the planned trajectories in both the scenarios with the corresponding covariance estimated at each node (only the (x,y) portion is shown). Note that the illustrations

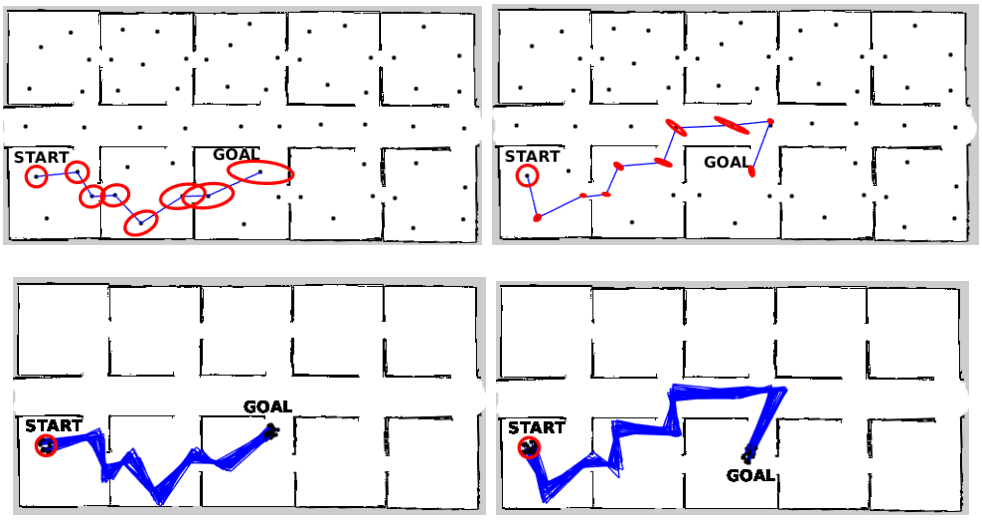


Figure 6.10 (*top-left* and *top-right*) The propagated belief distributions along the planned paths while running MPTP with *PETLON cost* and *MPTP cost*. The belief estimates for a single planning instantiation corresponding to a unique set of simulated observations are shown. The black dots represent the sampled poses. (*top-left*) Shortest path route that corresponds to running the planner with *PETLON cost*. (*top-right*) Belief space planning corresponding to running the planner with *MPTP cost*, returning an information rich route. (*bottom-left*) Traces of robot's true state while starting from the initial belief and run on *PETLON cost*– 80% of the trajectories lead to collision. (*bottom-right*) Traces of robot's true state while starting from the initial belief and run on *PETLON cost*– only 8% of the trajectories lead to collision.

show a single planning instantiation corresponding to a unique set of simulated observations Z. Belief space planning (*MPTP cost*) enables effective localization by returning a route which is rich in sensor information (see Fig. 6.10 on top-right). Fig. 6.10 on the bottom-left, shows the traces of true robot states for 25 different simulations while running on *PETLON cost*. The initial poses are sampled from the known initial belief distribution. Out of the 25 trials, 20 lead to collision on the walls, giving a success rate of only 20%. The traces of true robot pose for 25 different simulations while running on *MPTP cost* is shown in Fig. 6.10 (bottom-right). Only 2 trials lead to collision, giving a success rate of 92%.

Finally, we test the scalability of our approach by running the planner with varying number of rooms that are directly connected by doors between them. We consider a scenario in which seven rooms are to be visited. We consider five different cases of this scenario, each of which has a fixed number of rooms that are directly connected by the doors. For each case, 25 trails are performed and for each trial, the rooms with doors between them are randomly selected. The overall planning time is seen in Fig. 6.11.

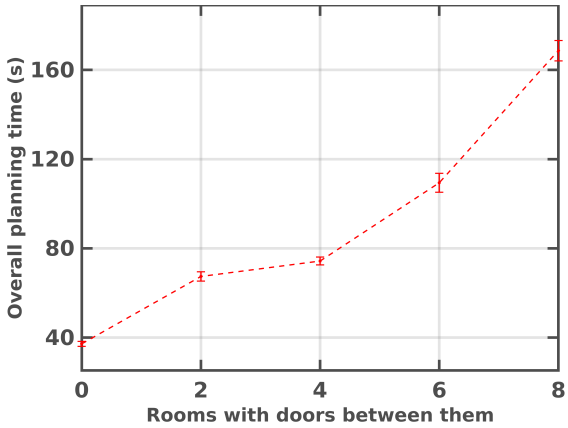
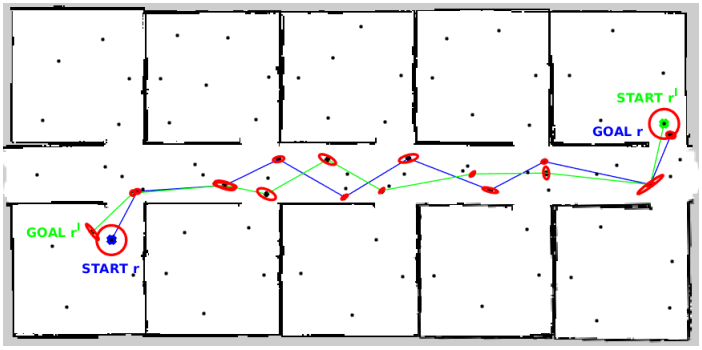
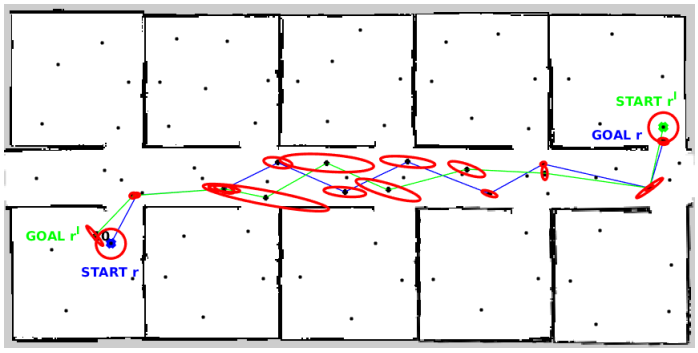


Figure 6.11 Overall planning time for visiting 7 rooms when the number of rooms directly connected by doors are varying. Average time for 25 trails are plotted in each case.

6.2 Multi-robot Scenarios



(a) config 1



(b) config 2

Figure 6.12 Pose covariance evolution for robots r (blue trajectory) and r' (green trajectory). The belief evolution for a single planning instantiation corresponding to a unique set of simulated observations are shown. Black dots represent the sampled poses and the covariance estimates (only (x,y) portion shown) are shown as red ellipses. (a) *config 1* incorporating mutual observations between the robots. (b) No mutual observations considered.

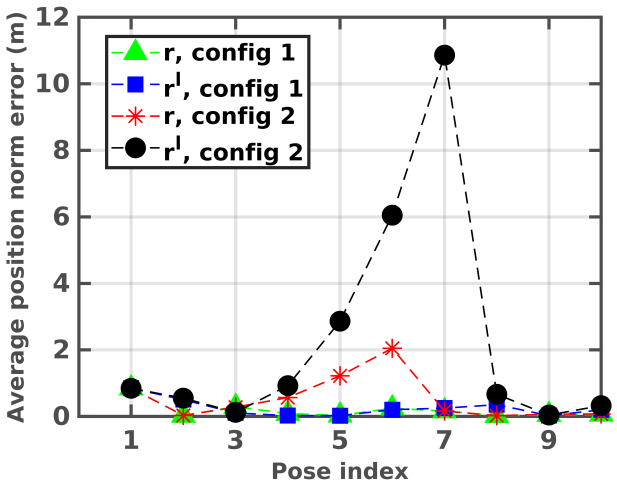


Figure 6.13 Average position estimation errors. This corroborates the single instance of belief evolution as shown in Fig. 6.12.

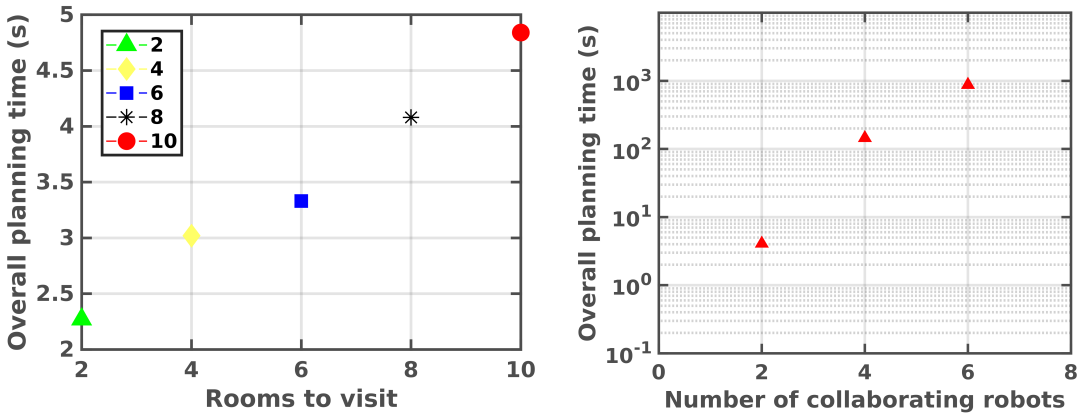


Figure 6.14 (a) Average planning time with increasing number of rooms to visit for 2 robots. The planning time is only about 5 seconds when 10 rooms are to be visited. (b) Average planning time in log-scale for different number of collaborating robots.

We evaluate our approach in a simulated corridor environment whose map is as shown in Fig. 6.8. The robot's can navigate to rooms $\mathbf{L} = L1, \ldots, L10$ that are connected to one another through a corridor. These rooms have doors, which can either be closed or open, connecting them to the corridor. We assume that once the robot is near to a closed door that directly connects a room to the corridor, it is able to open the door– for example using human aid. Navigating to rooms can hence be encoded using a single high-level PDDL action `goto_room` as seen in Chapter 5, Fig. 5.1. The stars with different colors represent

certain unique features assumed to be known and modeled like a printer or trash can that aid the robot's in better localization.

We first validate our approach by considering a scenario in which robot r starting at room $L1$ has to visit room $L10$ and robot r' starting at $L10$ has to visit $L1$. Fig. 6.12 shows the planned trajectories with belief evolution (pose covariances) for robots r and r'. Multi-robot constraints are incorporated in Fig. 6.12(a) and correspond to *config 1* while *config 2* as seen in Fig. 6.12(b) does not consider mutual observations between the robots. Clearly, incorporating mutual observation constraints facilitate improved localization. We ran the same scenario for 25 different planning sessions, each time sampling the initial position of the robots r and r' from the known initial beliefs. The average position errors at each node along the planned trajectories are shown in Fig. 6.13. This performance evaluation shows the improved estimation accuracy for both the robots while incorporating multi-robot constraints. In particular, for robot r' in *config 2*, that is, without multi-robot constraints, it is seen that there is significant pose uncertainty along its path. This is attributed to the lack of landmarks, rendering inaccurate localization. However, incorporating multi-robot constraints significantly improves localization, with the worst case position norm error for r' reducing by about 90%.

Next, we test the scalability for an increasing number of rooms to visit. As the number of rooms to visit increase, the task-level complexity increase as the task completion requires more task-level actions. We ran *config 1* for five different scenarios that correspond to visiting 2, 4, 6, 8 and 10 rooms, respectively. For each scenario, 25 different planning sessions are conducted with the rooms to be visited being selected randomly at each run. The average planning time with two robots are shown in Fig. 6.14(a), the plans being computed in less than 5 seconds in all cases. As seen from the figure, the planning time increase almost linearly as there is only one single high-level action, namely, `goto_room`. Currently, this high-level action models the navigation of both r and r' and therefore the complexity is directly dependent on the number of rooms.

Finally, we test the scalability for an increasing number of collaborating robots. In the considered scenario eight rooms are to be visited with 2, 4 and 6 different robots. For each run, the rooms to be visited are randomly selected and the average time for 25 different planning sessions are plotted in log-scale in Fig. 6.14(b). It is seen that planning time scales exponentially with an increasing number of collaborating robots. This is quite intuitive as planning is to be performed for all possible robot pairs.

Chapter 7

Conclusions

In this chapter, we first discuss some limitations of our approach and later comment on the relation to multi-goal planning and travelling salesman problems. Finally we conclude this part of the thesis.

7.1 Discussion

MPTP has few limitations and assumptions and relaxing them would enhance the capability and robustness of our approach in challenging scenarios. First, we sample collision-free poses and therefore considering static obstacles, the planned trajectories are collision-free. In this sense, we employ a deterministic collision avoidance approach and do not compute the probability of collisions while computing a path during planning. It is a reasonable assumption for all practical purposes but is not the case in general while planning in narrow regions or corridors. The execution may be trivially extended to consider collision probabilities, making it robust to both static and dynamic obstacles. Second, we assume straight line path between two sampled poses. This might not fare well in some experimental domains and can lead to larger prediction uncertainties. Presently, as the number of samples vary, the search is performed again. It is our future direction to efficiently utilize the previous search results to reduce the computation time for increased samples. It is also an interesting future direction to extend the framework to an online real-time planning approach.

Multi-Goal Planning (MGP) [94], where a robot visits a sequence of goal configurations is a subset of the general class of TMP problems. Most existing MGP approaches [94, 40, 2, 41] leverage the *Traveling Salesman Problem* (TSP) [4] solvers for task sequencing. A TSP problem finds a minimum cost path traversing a set of points such that every point is visited once. In an MGP problem these points correspond to the set of goal configurations the

robot needs to visit. It can be argued that all MGP problems can be modeled as a TMP problem but not vice versa. For instance, consider the *office domain* presented in Chapter 4. In this scenario the robot not only has to visit regions of interest but execute actions such as collecting the documents, which is to be performed when visiting each cubicle ensuring that the action preconditions are met. Moreover, in certain scenarios cubicles may need to be visited multiple times violating the single visit constraint of traditional TSP solvers. The *corridor domain* (see Chapter 4) presents additional challenges for TSP solvers. If we consider that there are no doors between the rooms, then the problem reduces to just visiting different rooms and can be solved using TSP solvers. However, in the considered scenario there are doors between certain rooms and the accessibility between the rooms that are directly connected to each other via a door needs to be assessed by the robot. This requires different levels of reasoning to verify the action preconditions such as, checking if a door exists, navigating to the door, checking if the trace of the robot pose covariance is within the uncertainty budget and if yes, then updating the roadmap. Moreover, if the robot passes through the door, the accomplishment of the action effect (in this case, closing the door corresponds to updating the roadmap) needs to be established. Thus MPTP is able to solve a larger class of problems than traditional TSP solvers.

7.2 Concluding Remarks

This part of the thesis introduced an approach for task-motion planning under motion and sensing uncertainty in the context of both single and multi-robot settings. Task-motion interaction is facilitated by means of semantic attachments that return motion costs to the task planner. In this way, the action costs of the task plans are evaluated using a motion planner. The plan synthesized is optimal at the task-level since the overall action cost is less than that of other task plans generated for a given roadmap. It is to be noted that the action cost also encompasses the motion cost. The proposed approach is probabilistically complete and we have validated the framework using a simulated office environment in Gazebo and a corridor environment. In the single robot TMP setting, the approach has been evaluated with different configurations that correspond to different motion cost computation, illustrating the need for a combined TMP approach for navigation in belief space. Though we have validated MPTP in two different robot navigation domains, real-world scenarios often require large number of tasks to be performed. Real-world domains are much more knowledge-intensive, significantly increasing the task-level and motion-level complexity. The scalability results

suggest that our approach fares well with respect to increased task-level complexity and plan length.

In the multi-robot TMP setting, our approach scales well with an increasing task-level complexity. Though there is an exponential increase in planning time as the number of cooperative robots that perform the task increases, caching and reusing plans might help alleviate this complexity in some cases. Presently, our approach fares well only when there are an even number of rooms to visit. For odd number of rooms to visit, the generated plan can force additional robot motions since our task-level action is defined for a pair of robots. Let us consider a scenario with 4 robots $r1,\ldots,r4$ and rooms $L1$, $L3$ and $L7$ to be visited. The synthesized plan might be that $r1$ visits $L1$, $r2$ visits $L3$ and $r3$ visits $L7$, $r4$ visits Li (where $i = 1,\ldots,10$). Robot $r4$ visiting Li is an additional room visit, even though it is not specified in the goal condition. This visit helps to obtain mutual observations between $r3$ and $r4$ but in practice we only need $r1$ visiting $L1$, $r2$ visiting $L3$ and $r3$ visiting $L7$. However, it is to be noted that this formulation still preserves the task-level optimality. Decoupling and defining the action for each robot can rescind the additional motions. Yet, the computational challenge associated with the semantic attachment architecture needs to be analyzed and it is an immediate work for future. Another remark that needs to be stressed is that the extension of our formulation in (5.8) to incorporate more than two robots is an approximation of the joint belief. This is so because we only consider pairwise mutual observations. Nevertheless, it is a fairly common practice [42].

Part II

Motion Planning with Environment Uncertainty

This part focuses on computing the probability of collision for a robot while encountering obstacles in the environment. As in the previous part, we assume that the environment is known a priori. However, we relax the assumption of static obstacles and the approaches developed in this part are applicable to dynamic obstacles as well. We develop an approach for computing the exact collision probability and also compute tight upper bounds for the same. As opposed to the previous part, in this part we perform online planning incorporating robot motion, sensing and obstacle state uncertainties.

Chapter 8

Motion Planning with Environment Uncertainty

In this part of the thesis we first introduce the concept of *object uncertainty* (Chapter 9) and then compute exact (Chapter 10) and approximate collision probability bounds (Chapters 11 and 12) for safe motion planning.

8.1 Introduction

Robots have become more pervasive and are being increasingly used in close proximity to humans and other objects (both static and dynamic) in factories, living spaces, elderly care, and robotic surgery. Planning for collision free trajectories in real-time is imperative for robots to operate safely and efficiently in such realistic conditions. However, uncertainties often arise due to insufficient knowledge about the environment, imperfect sensing or inexact robot motions. In these situations, it is indispensable to employ approaches that perform safe motion planning under motion and sensing uncertainties. We therefore resort to BSP based appraoches which are an instantiation of POMDPs. However, at the planning time, future observations are yet to be obtained. Thus, for efficient planning and decision making, it is required to reason about future belief distributions due to possible actions and the corresponding expected future observations.

Uncertain environments are such that they often preclude the existence of collision free trajectories ([3]). In the presence of noisy sensors, both the robot and the environment state cannot be estimated precisely and one can only reason in terms of the corresponding belief states. Moreover, in case of dynamic obstacles, their future states have to be predicted and they are not known exactly due to the lack of perfect knowledge of their motions. As

such, providing safety guarantees is difficult and for safe navigation, both the robot state uncertainty and the uncertainty in obstacle estimates need to be considered while computing collision probabilities.

Most robotic tasks require the knowledge of where the robot is with respect to the environment. Localization is therefore one of the most fundamental problem in robotics and significantly impacts planning and decision making. As such, localization is therefore a key aspect for safe and efficient navigation. However, existing approaches assume that the landmark locations are known precisely or with little uncertainty. For example, given the map of the environment, while planning for future actions the standard Markov localization does not take into account the map uncertainty (that is, landmark locations are assumed to be perfect). This means that given the map and the sensing range, there exists a region from which the landmark can be observed. This however, might not be true in practice. For example, let us consider a SLAM session. Wrong data association or dynamics objects preventing loop closures could lead to wrongly estimated landmark locations and thereby the corresponding map. Thus landmark estimates arising out of such a SLAM session might not be known precisely. It is noteworthy that due to this landmark location uncertainty the regions from which the landmark can be observed are also uncertain. This is visualized in Fig. 8.1. We define the *pose space* as the set of all possible poses the robot can assume. The blue blob denotes the object which when viewed from a pose x produces an observation z. Different observations are produced when the object is viewed from distinct poses such that the object falls within the sensing range and there are no occlusions. The set of all such poses is a subset of the pose space and is defined to be the *viewpoint space* (green region in the figure). We note that the viewpoint space is sensor-dependent and is determined by the sensing range and other aspects such as occluding objects. Intuitively, also the viewpoint space is object depended and this set changes for each object. However, a pose x' that falls outside the viewpoint space does not produce an observation. Consequently, as seen on the left hand side of the figure, when the object location is known precisely, there exists a subset of the pose space from which the object can be observed. On the right hand side of the figure, the light-blue shaded region denotes the uncertainty in object location. Thus in practice the object can be anywhere within the uncertainty region. As a result, given a pose, it cannot be said with certainty that the object can be observed. Subsequently, the cardinality of the subset of the pose space from which the object can be observed is increased, that is, the viewpoint space has increased (green region in the figure). As seen in the figure, depending on the object really is, it can be observed from either x or x' or both the poses. As a result, one can only reason in terms of the probability of observing the object from

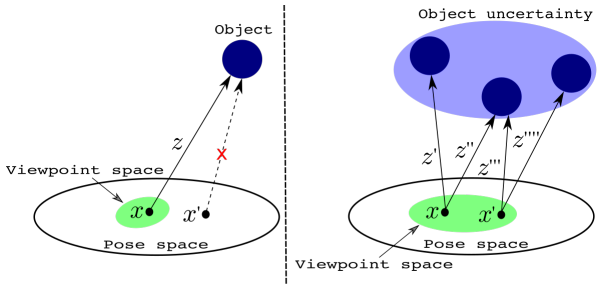


Figure 8.1 The blue blob denotes an object in the environment. The green region is the viewpoint space corresponding to the set of poses from which the object can be observed. Robot pose x produces an observation z however x' does not produce an observation. On the right hand side, the light-blue shaded region denotes the uncertainty in object location. In practice the object can be anywhere within the uncertainty region. As a result, depending on where the object really is, it can be observed from either x or x' or both the poses.

the considered pose or the viewpoint. Therefore, a probability distribution function for the viewpoint space is obtained where the mean viewpoint corresponds to observing the object with highest probability. Not accounting for this uncertainty can cause localization errors, leading to inefficient plans. In this thesis, we will use the term *object uncertainty* to refer to this notion of uncertainty in landmark location.

8.2 Related Work

Much research activities about BSP have been carried out in the past few years, with applications spanning a variety of areas [89, 109, 52, 1, 62, 83, 102, 29, 103]. Yet, most approaches assume that the landmarks are fairly well known or are known with little uncertainty. [52] consider object uncertainty since they are planning in an unknown environment and require several measurements to obtain confidence estimates of object locations. Thus they perform active perception, that is, to look for robot actions that enhances information to reduce the object uncertainty. This context is different from ours since we consider a known environment with object uncertainty and focus on active localization incorporating these uncertainties. In [83], the concept of object uncertainty is commented upon (they call it scene uncertainty); however they do not show how it affects the state estimation. Dynamic environments are considered in [62, 1] however the landmark/beacon locations are assumed to be known perfectly; [102, 103] also consider perfect landmark locations in the context of task and motion planning. Thus most active and passive localization-based approaches focus on robot state uncertainty and assume perfect knowledge about the location of the objects in the environment. However, in practice, the environment is seldom known with high certainty

and hence providing formal guarantees for safe robotic tasks under environment uncertainty is of vital importance.

The starting point of all approaches is to formulate the collision constraint, that is, the conditions under which a collision occurs between a robot and an obstacle. The methods differ in their formulation of collision constraint due to different assumptions regarding the shape of the robot and the obstacles (for example, point, spherical, ellipsoidal, or rectangular shapes for the robot and the obstacle), modeling of uncertainties. Overall, most approaches tend to be overly conservative and provide loose upper bounds. This can lead to sub-optimal plans or in some case render plans infeasible and thus it is desirable to compute accurate collision probabilities to ensure safe and efficient trajectories. Patil *et al.* [86] truncate [49] the estimated *a priori* Gaussian state distributions to consider only the collision free samples. Thus propagating these truncated distributions enable them to compute collision free trajectories. The approach of truncating Gaussian distributions is leveraged to compute risk-aware and asymptotically optimal trajectories by [69]. [8], the future state distributions are predicted and the uncertainties are used to compute bounded collision probabilities. [67] use sigma hulls for robot links and compute the signed distance of these hulls to the obstacles to formulate the collision avoidance constraints.

Exact collision probability can be computed by marginalizing the joint distribution between the robot and obstacle locations [18]. This integration is then performed over the set of robot and obstacle locations that satisfy the collision constraint. Such an appraoch is used to derive the collision constraint in [19] and [82]. However, there is no closed form solution to the integral and numerical integration or *Monte Carlo* (MC) techniques are employed [96] to obtain an approximate value. Assuming that the robot radius is negligible the joint distribution can be approximated as the product of the volume occupied by the robot and the conditional distribution of the obstacle evaluated at the robot location. Furthermore, Park *et al.* [82] compute an upper bound for the collision probability. [65], an approximation is computed using *Monte Carlo Integration* (MCI), which is nonetheless computationally intensive. Another related work that uses a Monte Carlo approach and is real-time compatible is *Monte Carlo Motion Planning* (MCMP) [44]. They first solve a deterministic motion planning problem with an inflated obstacle and later adjust the inflation to compute the desired safe path.

A chance-constrained[1] approach to compute the bounded probability of collision along a trajectory is presented by [7]. This approach is leveraged to compute bounded collision-free

[1] A chance-constrained approach finds the optimal sequence of control inputs subject to the constraint that the collision probability must be below a user-specified threshold. This constraint is known as a chance constraint.

trajectories with dynamic obstacles by [113], wherein the dynamic obstacles follow a constant velocity model with Gaussian noise. In [3], a *Gaussian Process* (GP) based approach is used to learn motion patterns (a mapping from states to trajectory derivatives) to identify possible future obstacles trajectories. Newton's method combined with chance-constraints is used in [101] to obtain an upper bound for collision probability. In [27] the first-exist times for Brownian motions are leveraged to compute collision probabilities. [5] focus exclusively on obstacle uncertainty. They formalize a notion of *shadows*, which are the geometric equivalent of confidence intervals for uncertain obstacles. The shadows fundamentally give rise to loose bounds but the computational complexity of bounding the collision probability is greatly reduced. Uncertain obstacles are modelled as polytopes with Gaussian-distributed faces by [98]. Planning a collision-free path in the presence of *risk zones* is considered by [95] by penalizing the time spent in these zones. Risk contours map, which take into account the risk information (uncertainties in location, size and geometry of obstacles) in uncertain environments are used by [45] to obtain safe paths with bounded risks. A related approach for randomly moving obstacles is presented by [32]. Formal verification methods have also been used to construct safe plans [15, 93].

Most of the approaches discussed above leverage Boole's inequality to compute the collision probability along a path by summing or multiplying the probabilities along different waypoints in the path. However, the additive approach assumes that the probabilities along the waypoints are mutually exclusive and the multiplicative approach treats them as independent. Such approaches tend to be overly conservative and rather than computing bounded collision probabilities along a path, the bound should be checked for each configuration along the path itself. Moreover, in most approaches, the collision probability computed along each waypoint is an approximation of the true value. For example, the MCI approach of [65] approximates the resulting double summation expression for collision probability to a single summation. [113] compute an approximate upper bound for collision probability by linearizing the collision condition. [82] and [19] assume the volume occupied by the robot to be negligible. On the one hand, such approximations can overly penalize paths and could gauge all plans to be infeasible. On the other hand some approximations can be lower[2] than the true collision probability values and can lead to synthesizing unsafe plans.

[2]For example, the approach of [19] computes a lower value than the actual when the robot state covariance is small.

8.3 Notations and Problem Definition

We shall denote vectors by bold lower case letters, that is $\mathbf{x}$ and its components by lower case letters. The transpose of $\mathbf{x}$ will be denoted by $\mathbf{x}^T$ and its Euclidean norm by $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}}$. The the expected value of a random vector[3] $\mathbf{x}$ will be denoted by $\mathbb{E}(\mathbf{x})$. A multivariate Gaussian distribution of $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance Σ will be denoted using the notation $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. Matrices will be denoted by capital letters, that is M. The trace of a square matrix M will be denoted by $tr(M)$ and its determinant by $det(M)$. The identity matrix will be denoted by I or I_n when the dimension needs to be stressed. A diagonal matrix with diagonal elements $\lambda_1, \ldots, \lambda_n$ will be denoted by $diag(\lambda_1, \ldots, \lambda_n)$. Sets will be denoted using mathcal fonts, that is $\mathscr{S}$. Unless otherwise mentioned, subscripts on vectors/matrices will be used to denote time indexes and (whenever necessary) superscripts will be used to indicate the robot or the object that it refers to. For example, $\mathbf{x}_k^i$ represents the state of robot i at time k. We use the notation $\mathscr{E}^n(A, \mathbf{a})$ to denote an ellipsoid with center $\mathbf{a} \in \mathbb{R}^n$ and A being a positive definite $n \times n$ matrix. The superscript n will be avoided when there is no cause for confusion. The notation $P(\cdot)$ will be used to denote the probability of an event and the probability density function (pdf) will be denoted by $p(\cdot)$.

We now formally define the problem that we tackle in this part of the thesis. Let us consider a mobile robot operating in a partially-observable environment. The map of the environment is either known *a priori* or it is built using a standard SLAM algorithm. At any time k, we denote the robot pose (or configuration) by $\mathbf{x}_k \doteq (x_k, y_k, \theta_k)$, the acquired measurement from objects is denoted by $\mathbf{z}_k$ and the applied control action is denoted as $\mathbf{u}_k$. It is noteworthy that by *objects* we refer to both the landmarks and the obstacles in the environment. We also make the following assumptions: (1) the uncertainties are modelled using Gaussian distributions, (2) the robot and obstacles are assumed to be non-deformable objects.

To describe the dynamics of the robot, we consider a standard motion model with Gaussian noise

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) + w_k \ , \ w_k \sim \mathcal{N}(0, R_k) \tag{8.1}$$

where w_k is the random unobservable noise, modeled as a zero mean Gaussian. Objects are detected through the robot's sensors and assuming known data association, the observation model can be written as

$$\mathbf{z}_k = h(\mathbf{x}_k, O_k^i) + v_k \ , \ v_k \sim \mathcal{N}(0, Q_k) \tag{8.2}$$

[3] By a random vector we refer to a vector random variable.

where O_k^i is the $i-$th detected object and v_k is the zero mean Gaussian noise. The function $h(\mathbf{x}_k, O_k^i)$ denotes the fact that at time k, the measurement $\mathbf{z}_k$ is obtained by observing the $i-th$ object O_k^i from viewpoint (robot location) $\mathbf{x}_k$. In the case of a laser-range finder the function h could be defined as the distance between $\mathbf{x}_k$ and the location of the object (or any particular point on the object) O_k^i. If we consider the case of a camera, h may be defined as a pinhole projection operator, projecting the object O_k^i onto the image plane. We note here that the motion (8.1) and observation (8.2) models can be written probabilistically as $p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k)$ and $p(\mathbf{z}_k|\mathbf{x}_k, O_k^i)$, respectively.

Given the models in (8.1) and (8.2), in this thesis we compute *safe* plans, wherein the probability of collision of the robot with any obstacle is guaranteed to be less than a specified bound while navigating to the goal. To this end, we consider the object uncertainties while localizing the robot. Given the robot and obstacle locations, we compute the probability of collision under motion, sensing uncertainty, and the uncertainty in obstacle location.

Chapter 9

Object Uncertainty

As discussed in Chapter 8, most localization approaches assume that the landmark locations are known precisely or with little uncertainty. This however, might not be true in practice due to noisy measurements and/or imperfect sensors. Thus, it is pertinent that landmark uncertainties are considered within the localization and planning framework. Below, we delineate the incorporation of object uncertainty within the Bayes filter which correspond to **C5** of our contribution.

We define the collection of all objects in the environment to be the *object space* $\mathscr{O} = \{O^i | O^i \text{ is an object, and } 1 \leq i \leq |\mathscr{O}|\}$. Let us posit that at time k the robot received a measurement $\mathbf{z}_k$ which was originated by observing the object O^i_k. Given[1] an initial distribution $p(\mathbf{x}_0)$, and the motion and observation models in (8.1) and (8.2), the posterior probability distribution at time k is the *belief* $b[\mathbf{x}_k]$ and can be written as

$$b[\mathbf{x}_k] = p(\mathbf{x}_k | \mathbf{z}_k, O^i_k, \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k-1}) \tag{9.1}$$

where O^i_k is the object observed at time k, $\mathbf{z}_{0:k-1} \doteq \{\mathbf{z}_0, ..., \mathbf{z}_{k-1}\}$ and $\mathbf{u}_{0:k-1} \doteq \{\mathbf{u}_0, ..., \mathbf{u}_{k-1}\}$. We note that this posterior belief is computed after incorporating the measurement at time k, that is, $\mathbf{z}_k$.

Given the belief $b[\mathbf{x}_k]$ and an action $\mathbf{u}_k$, the belief before incorporating a measurement will be called the propagated belief and can be written as

$$b[\bar{\mathbf{x}}_{k+1}] = \int_{\mathbf{x}_k} p(\mathbf{x}_{k+1} | \mathbf{x}_k, \mathbf{u}_k) b[\mathbf{x}_k] \tag{9.2}$$

[1] Robot belief and inference was already introduced in Chapter 3. As we additionally incorporate object uncertainty it is imperative for us to furnish the complete derivation to provide a comprehensive understating for the reader.

Now let us consider that a measurement $\mathbf{z}_{k+1}$ is obtained that corresponds to observing the object O^i_{k+1}. The posterior distribution $b[\mathbf{x}_{k+1}]$ can then be computed using Bayes rule and the theorem of total probability. This expansion is obtained in terms of the belief at the previous time step since the Bayes filter is recursive. Thus we have

$$p(\mathbf{x}_{k+1}|\mathbf{z}_{k+1}, O^i_{k+1}, \mathbf{z}_{0:k}, \mathbf{u}_{0:k}) = \\ \eta\, p(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}, O^i_{k+1}) p(O^i_{k+1}|\mathbf{x}_{k+1}) \int_{\mathbf{x}_k} p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) b[\mathbf{x}_k] \quad (9.3)$$

where $\eta = 1/p(\mathbf{z}_{k+1}|\mathbf{z}_{0:k}, \mathbf{u}_{0:k})$ is the normalization constant. The term $p(O^i_{k+1}|\mathbf{x}_{k+1})$ denotes the probability of observing the object O^i_{k+1} from the pose $\mathbf{x}_{k+1}$. In other words, this term models the fact, how likely it is to observe O^i_{k+1} from $\mathbf{x}_{k+1}$ and thus models the object uncertainty. The term $p(O^i_{k+1}|\mathbf{x}_{k+1})$ also additionally model aspects such as occlusions due to static obstacles that hinder the observation, occlusions that results due to dynamic obstacles, faulty sensors or other aspects that impedes observations of objects of interest. Thus, given an object one can only reason probabilistically about observing it to obtain the corresponding measurement. However, when the object uncertainty and other additional aspects are ignored an object is observed whenever the robot is within the viewpoint space (see left hand side of Fig. 8.1). Thus, in the case of such an assumption, for poses within the viewpoint space the term is equal to unity, that is, $p(O^i_{k+1}|\mathbf{x}_{k+1}) = 1$. For poses that lie outside the viewpoint space $p(O^i_{k+1}|\mathbf{x}_{k+1}) = 0$, and hence no measurement can be obtained. As such, when the object uncertainty is ignored, the term $p(O^i_{k+1}|\mathbf{x}_{k+1})$ can be removed from (11.27) and the posterior belief parameters can be computed using the standard EKF update equation as

$$\begin{aligned} K_{k+1} &= \bar{\Sigma}_{k+1} H^T_{k+1} \left(H_{k+1} \bar{\Sigma}_{k+1} H^T_{k+1} + Q_{k+1} \right)^{-1} \\ \boldsymbol{\mu}_{k+1} &= \bar{\boldsymbol{\mu}}_{k+1} + K_{k+1} \left(\mathbf{z}_{k+1} - h(\bar{\boldsymbol{\mu}}_{k+1}) \right) \\ \Sigma_{k+1} &= (I - K_{k+1} H_{k+1}) \bar{\Sigma}_{k+1} \end{aligned} \quad (9.4)$$

where H_{k+1} is the Jacobian of $h(\cdot)$ with respect to $\mathbf{x}_{k+1}$, and K_{k+1} is the Kalman gain.

The exposition so far has been agnostic to the actual model of $p(O^i_{k+1}|\mathbf{x}_{k+1})$. In general, this term can be modeled given the environment map, the sensing capabilities and the robot objectives. These aspects should hence be incorporated to obtain the actual object uncertainty model. However, in this work we approximate the object distribution as a

Gaussian distribution:

$$p(O^i_{k+1}|\mathbf{x}_{k+1}) \sim \mathcal{N}(\boldsymbol{\mu}_{O^i_{k+1}}, \Sigma_{O^i_{k+1}}) \tag{9.5}$$

where $\boldsymbol{\mu}_{O^i_{k+1}}$ is the viewpoint/pose that corresponds to the maximum probability of observing O^i_{k+1} and $\Sigma_{O^i_{k+1}}$ is the associated uncertainty in the observation.

We will now consider the object uncertainty term $p(O^i_{k+1}|\mathbf{x}_{k+1})$ and derive the Gaussian belief parameters by expanding (11.27). Expanding the right hand side of (11.27) using the probability density function (pdf) of multivariate Gaussian distributions, we have $b[\mathbf{x}_{k+1}] = \eta' \int \exp(-\mathscr{J}_{k+1})$, where η' contains the non-exponential terms and $\mathscr{J}_{k+1}$ is given by

$$\begin{aligned}\mathscr{J}_{k+1} = \frac{1}{2}\Big(\mathbf{z}_{k+1} - h\left(\bar{\boldsymbol{\mu}}_{k+1}\right) - H_{k+1}\left(\mathbf{x}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right)\Big)^T Q^{-1}_{k+1}\Big(\mathbf{z}_{k+1} - h\left(\bar{\boldsymbol{\mu}}_{k+1}\right) \\ - H_{k+1}\left(\mathbf{x}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right)\Big) + \frac{1}{2}(\boldsymbol{x}_{k+1} - \boldsymbol{\mu}_{O^i_{k+1}})^T \Sigma^{-1}_{O^i_{k+1}}(\mathbf{x}_{k+1} - \boldsymbol{\mu}_{O^i_{k+1}}) \\ + \frac{1}{2}(\mathbf{x}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1})^T \bar{\Sigma}^{-1}_{k+1}(\mathbf{x}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1})\end{aligned} \tag{9.6}$$

where H_{k+1} is the Jacobian of $h(\cdot)$ with respect to $\mathbf{x}_{k+1}$. As shown in [105], the covariance Σ_{k+1} is obtained as the inverse of the second derivative of $\mathscr{J}_{k+1}$ with respect to $\mathbf{x}_{k+1}$. The expression for the second derivative is obtained as

$$\frac{\partial^2 \mathscr{J}_{k+1}}{\partial \mathbf{x}^2_{k+1}} = H^T_{k+1} Q^{-1}_{k+1} H_{k+1} + \Sigma^{-1}_{O^i_{k+1}} + \bar{\Sigma}^{-1}_{k+1} \tag{9.7}$$

Therefore the posterior covariance is obtained as

$$\Sigma^{-1}_{k+1} = H^T_{k+1} Q^{-1}_{k+1} H_{k+1} + \Sigma^{-1}_{O^i_{k+1}} + \bar{\Sigma}^{-1}_{k+1} \tag{9.8}$$

The mean of $b[\mathbf{x}_{k+1}]$ is the value that maximizes $b[\mathbf{x}_{k+1}]$ and hence is obtained by equating the first derivative of $\mathscr{J}_{k+1}$ to zero. The expression for the mean $\boldsymbol{\mu}_{k+1}$ is obtained as (see Appendix A for derivation)

$$\boldsymbol{\mu}_{k+1} = \bar{\boldsymbol{\mu}}_{k+1} + K_{k+1}\Big(\mathbf{z}_{k+1} - h\left(\bar{\boldsymbol{\mu}}_{k+1}\right)\Big) + \Sigma_{k+1}\Sigma^{-1}_{O^i_{k+1}}\Big(\boldsymbol{\mu}_{O^i_{k+1}} - \bar{\boldsymbol{\mu}}_{k+1}\Big) \tag{9.9}$$

where $K_{k+1} = \Sigma_{k+1} H^T_{k+1} Q^{-1}_{k+1}$ is the Kalman gain. We note that when no object uncertainty is considered the update step of the standard EKF gives $\boldsymbol{\mu}_{k+1} = \bar{\boldsymbol{\mu}}_{k+1} + K_{k+1}\Big(\mathbf{z}_{k+1} - h\left(\bar{\boldsymbol{\mu}}_{k+1}\right)\Big)$. The additional term in (9.9) rightly adjusts the mean $\boldsymbol{\mu}_{k+1}$ accounting for the fact that the object location is uncertain.

As in the standard EKF based Bayes filter, the expression for the covariance Σ_{k+l} can also be derived in terms of the Kalman gain K_{k+1} and the predicted covariance $\bar{\Sigma}_{k+1}$. Using the matrix inversion lemma on (9.8), the following expression is obtained (see Appendix B for derivation)

$$\Sigma_{k+1} = (I - K_{k+1}H_{k+1})\,\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} \tag{9.10}$$

where $\tilde{\Sigma}_{k+l} = \left(\bar{\Sigma}_{k+1} + \Sigma_{O^i_{k+1}}\right)^{-1}$.

When object uncertainty is not considered, the update step of the standard EKF gives $\Sigma_{k+1} = (I - K_{k+1}H_{k+1})\,\bar{\Sigma}_{k+1}$. The extra terms in (9.10) account for the object uncertainty and scale the posterior covariance accordingly. We note that when object uncertainty is not considered, $p(O^i_{k+1}|\mathbf{x}_{k+1}) = 1$ and hence the results in (9.9) and (9.10) reduce to that of the standard EKF case in (9.4). The method presented above is easily generalized to multiple objects observed at any time instant. This is done by following the sequential-sensor method ([20]), considering the fact that given the current state estimate, the observations are independent of each other.

Let us now analyse the effect of object uncertainty. As discussed above when object uncertainty is not assumed, $p(O^i_{k+1}|\mathbf{x}_{k+1}) = 1$, and therefore the posterior belief parameters reduce to that of the standard EKF case. However, in practice, one should consider object uncertainty and the posterior belief parameters are as delineated in (9.9) and (9.10). Yet, the impact of considering object uncertainty in localisation depends on the covariance of the estimated object location. When the covariance of the object location is much larger compared to the predicted robot belief state covariance, the impact of considering object uncertainty is greatly reduced.

Lemma 1. *If the uncertainty in the estimated object location is much larger than the predicted robot state uncertainty, that is, when $\Sigma_{O^i_{k+1}} \gg \bar{\Sigma}_{k+1}$, then the respective object uncertainty is no longer significant.*

Proof. In order to prove the above lemma, it suffices to show that when $\Sigma_{O^i_{k+1}} \gg \bar{\Sigma}_{k+1}$, the posterior belief parameters reduce to that of the standard EKF update case as given in (9.4). Let us first consider the expession in (9.8). Using the fact that $\Sigma_{O^i_{k+1}} \gg \bar{\Sigma}_{k+1}$, then $\Sigma^{-1}_{O^i_{k+1}} \ll \bar{\Sigma}^{-1}_{k+1}$ and hence it can be neglected when compared to $\bar{\Sigma}^{-1}_{k+1}$. This gives $\Sigma_{k+1} = \left(H^T_{k+1}Q^{-1}_{k+1}H_{k+1} + \bar{\Sigma}^{-1}_{k+1}\right)^{-1}$, and is the expression for the posterior belief covariance when object uncertainty is not considered. Again, using $\Sigma_{O^i_{k+1}} \gg \bar{\Sigma}_{k+1}$, $\bar{\Sigma}_{k+1}$ can be neglected

from the sum $\left(\bar{\Sigma}_{k+1} + \Sigma_{O^i_{k+1}}\right)$. The expression for Kalman gain thus reduces to

$$K_{k+1} = \bar{\Sigma}_{k+1}\left(\Sigma_{O^i_{k+1}}\right)^{-1}\Sigma_{O^i_{k+1}}H^T_{k+1} \left(H_{k+1}\bar{\Sigma}_{k+1}\left(\Sigma_{O^i_{k+1}}\right)^{-1}\Sigma_{O^i_{k+1}}H^T_{k+1} + Q_{k+1}\right)^{-1} = \bar{\Sigma}_{k+1}H^T_{k+1}\left(H_{k+1}\bar{\Sigma}_{k+1}H^T_{k+1} + Q_{k+1}\right)^{-1} \quad (9.11)$$

Thus, as can be seen in (9.4), the Kalman gain is exactly the gain obtained when object uncertainty is not considered. Similarly, we have $\Sigma_{k+1}\Sigma^{-1}_{O^i_{k+1}} \ll 1$, thus $\boldsymbol{\mu}_{k+1} = \bar{\boldsymbol{\mu}}_{k+1} + K_{k+1}\left(\mathbf{z}_{k+1} - h\left(\bar{\boldsymbol{\mu}}_{k+1}\right)\right)$. Following a similar argument, it is easily seen that $\Sigma_{k+1} = (I - K_{k+1}H_{k+1})\bar{\Sigma}_{k+1}$. This completes the proof of Lemma 1.

Although the above result might seem counter-intuitive at first, we note here that the viewpoint space, when object uncertainty is not considered, is the space centred around the mean of the viewpoint space when the object uncertainty is considered. When the covariance of the object location is very high, then the probability values for viewpoints slightly away from the mean reduces drastically. Consequently considering these viewpoints adds little impact.

Chapter 10

Exact Collision Probability

In this chapter, we approximate both the robot and obstacles using their bounding volume sphere (or a combination of spheres). This is relaxed in Chapter 11 where we assume bounding volume ellipsoids which form a much better approximation than bounding volume spheres. Our contributions **C6**, **C7** and **C8** are discussed in this chapter.

We denote by $\mathscr{R}$ the set of all points occupied by a rigid-body robot at any given time. Similarly, let $\mathscr{S}$ represent the set of all points occupied by a rigid-body obstacle. A collision occurs if there exits a point such that it is in both $\mathscr{R}$ and $\mathscr{S}$. Thus the collision condition is defined as

$$\mathscr{R} \cap \mathscr{S} \neq \{\phi\} \tag{10.1}$$

and we denote the probability of collision as $P\left(\mathscr{R} \cap \mathscr{S} \neq \{\phi\}\right)$. In this work we assume spherical geometries for $\mathscr{R}$ and $\mathscr{S}$ with radii r_1 and s_1, respectively. We assign body-fixed reference frames to both the robot and the obstacle located at $\mathbf{x}_k$ and $\mathbf{s}_k$, respectively in the global frame. By abuse of notation we will use $\mathbf{x}_k$ and $\mathbf{s}_k$ equivalently to $\mathscr{R}$ and $\mathscr{S}$. The collision condition is thus defined in terms of the body-fixed frames as

$$\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k} : \mathscr{R} \cap \mathscr{S} \neq \{\phi\} \tag{10.2}$$

We recall here that the locations of the obstacles are in general uncertain. Let us now consider an obstacle at any given time instant, distributed according to the Gaussian $\mathbf{s}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{s}_k}\right)$, where $\boldsymbol{\mu}_{\mathbf{s}_k}$ represents the mean and $\Sigma_{\mathbf{s}_k}$ the associated covariance. Given the current robot state $\mathbf{x}_k$ and the obstacle state $\mathbf{s}_k$, the probability of collision can be formulated if the joint distribution between the robot and the obstacle state is known. In such a case the collision

probability is given by

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = \int_{\mathbf{x}_k}\int_{\mathbf{s}_k} I_c(\mathbf{x}_k,\mathbf{s}_k)p(\mathbf{x}_k,\mathbf{s}_k) \tag{10.3}$$

where $\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}$ as defined above represents the fact that the robot configuration $\mathbf{x}_k$ and its collision with an obstacle at location $\mathbf{s}_k$ is considered, and I_c is an indicator function defined as

$$I_c(\mathbf{x}_k,\mathbf{s}_k) = \begin{cases} 1 & \text{if } \mathscr{R} \cap \mathscr{S} \neq \{\phi\} \\ 0 & \text{otherwise.} \end{cases} \tag{10.4}$$

and $p(\mathbf{x}_k,\mathbf{s}_k)$ is the joint distribution of the robot and the obstacle. [113] compute an approximate upper bound for the collision probability by linearizing the collision condition. [65] use MCI to compute (11.13). However, the resulting double summation is approximated to a single summation to reduce computational complexity. [19], [82] approximate the integral in (11.13) as $Vp(\mathbf{x}_k,\mathbf{s}_k)$, where V is the volume occupied by the robot. For computing $p(\mathbf{x}_k,\mathbf{s}_k)$, they first assume a distribution centered around the obstacle with the covariance being the sum of the robot and obstacle location uncertainties. Then the density $p(\mathbf{x}_k,\mathbf{s}_k)$ is computed by assuming a constant robot location. Du Toit and Burdick use the robot center, whereas in [82] the maximum of $p(\mathbf{x}_k,\mathbf{s}_k)$ on the surface of the robot is used to obtain an upper bound. However, the approximation is valid only when the robot radius is negligible. To demonstrate, let us re-write the collision condition as

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = \int_{\mathbf{x}_k}\left[\int_{\mathbf{s}_k\in\mathscr{R}} p(\mathbf{s}_k|\mathbf{x}_k)\right]p(\mathbf{x}_k) \tag{10.5}$$

If the robot radius is negligible then it can be assumed that $\mathbf{s}_k = \mathbf{x}_k$, giving

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = \int_{\mathbf{x}_k}\left[p(\mathbf{s}_k = \mathbf{x}_k|\mathbf{x}_k)\int_{\mathbf{s}_k\in\mathscr{R}}\right]p(\mathbf{x}_k) \tag{10.6}$$

Thus assuming a constant value of the obstacle evaluated at the robot location, we have

$$V = \int_{\mathbf{s}_k\in\mathscr{R}} \tag{10.7}$$

where V is the volume occupied by the robot. The approximate collision probability is thus

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) \approx V\int_{\mathbf{x}_k} p(\mathbf{s}_k = \mathbf{x}_k|\mathbf{x}_k)p(\mathbf{x}_k) \tag{10.8}$$

Assuming that the robot and the obstacle locations are independent Gaussian distributions with $\mathbf{s}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{s}_k}\right)$ and $\mathbf{x}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{x}_k}, \Sigma_{\mathbf{x}_k}\right)$, the collision probability can be approximately written as

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) \approx Vp(\mathbf{x}_k = \boldsymbol{\mu}_{\mathbf{x}_k}, \mathbf{s}_k = \boldsymbol{\mu}_{\mathbf{s}_k}) \tag{10.9}$$

where

$$p(\mathbf{x}_k = \boldsymbol{\mu}_{\mathbf{x}_k}, \mathbf{s}_k = \boldsymbol{\mu}_{\mathbf{s}_k}) = det\left(2\pi\left(\Sigma_{\mathbf{s}_k} + \Sigma_{\mathbf{x}_k}\right)\right)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{\mu}_{\mathbf{x}_k} - \boldsymbol{\mu}_{\mathbf{s}_k})^T \Sigma^{-1} (\boldsymbol{\mu}_{\mathbf{x}_k} - \boldsymbol{\mu}_{\mathbf{s}_k})\right) \tag{10.10}$$

Other existing approaches truncate the state distributions or compute approximate upper bounds using chance-constraints. As such, these approaches compute an approximation of the collision probability. In contrast, we formulate the collision constraint as a quadratic form in random variables, allowing us to compute an exact expression for the collision probability. In the remainder of this Section, a rigorous treatment of the same is presented.

Since the robot and the obstacles are assumed to be spherical objects, the collision constraint is written as

$$\|\mathbf{x}_k - \mathbf{s}_k\|^2 \leq (r_1 + s_1)^2 \tag{10.11}$$

where (as before) $\mathbf{x}_k$ and $\mathbf{s}_k$ are the random vectors that denote the robot and obstacle locations, respectively. Let the current estimates of the two random vectors be distributed according to $\mathbf{s}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{s}_k}\right)$ and $\mathbf{x}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{x}_k}, \Sigma_{\mathbf{x}_k}\right)$. Let us denote the difference between the two random variables by $\boldsymbol{w} = \mathbf{x}_k - \mathbf{s}_k$. Using the expression for the difference between two Gaussian distributions, we have $\boldsymbol{w} \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{x}_k} - \boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{x}_k} + \Sigma_{\mathbf{s}_k}\right)$. The collision constraint in (12.3) can now be written in terms of $\boldsymbol{w}$,

$$\boldsymbol{y} = \|\boldsymbol{w}\|^2 = \boldsymbol{w}^T\boldsymbol{w} \leq (r_1 + s_1)^2 \tag{10.12}$$

where $\boldsymbol{y}$ is a random vector distributed according to the squared L_2-norm of $\boldsymbol{w}$. Now, given the probability density function (pdf) of $\boldsymbol{y}$, the collision constraint reduces to solving the integral

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = \int_0^{(r_1+s_1)^2} p(y) \tag{10.13}$$

where $p(y) = P_{\boldsymbol{y}}(\boldsymbol{y} = y)$ is the pdf of $\boldsymbol{y}$. It is noteworthy that the above integral is the cumulative distribution function (cdf) of $\boldsymbol{y}$, that is, $P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = F_{\boldsymbol{y}}(y)$, where $F_{\boldsymbol{y}}(y)$ denotes the cdf. Thus the collision condition reduces to finding the cdf of $\boldsymbol{y}$ such that $\boldsymbol{y} \leq (r_1 + s_1)^2$.

As a consequence, we have

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = P\left(\boldsymbol{y} \leq (r_1+s_1)^2\right) = F_{\boldsymbol{y}}\left((r_1+s_1)^2\right) \tag{10.14}$$

In the following Sections, we will first show that the collision constraint is a quadratic form in random variables and later derive an exact expression for the cdf of the quadratic from.

10.1 Quadratic Form in Random Variables

We define a quadratic form in random variables:

Definition 10. *Let $\boldsymbol{x} = (x_1,\ldots,x_n)^T$ denote a random vector with mean $\boldsymbol{\mu} = (\mu_1,\ldots,\mu_n)^T$ and covariance matrix Σ. Then the quadratic form in the random variables $x_1,\ldots,x_n$ associated with an $n \times n$ symmetric matrix $A = (a_{ij})$, with i and j in $1,\ldots,n$, is*

$$Q(\boldsymbol{x}) = Q(x_1,\ldots,x_n) = \boldsymbol{x}^T A\boldsymbol{x} = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j \tag{10.15}$$

Let us define $\boldsymbol{v} = \Sigma^{-\frac{1}{2}}\boldsymbol{x}$ and define a random vector $\boldsymbol{z} = \left(\boldsymbol{v} - \Sigma^{-\frac{1}{2}}\boldsymbol{\mu}\right)$. The resulting distribution of $\boldsymbol{z}$ is thus zero mean with covariance being the identity matrix. Therefore, the quadratic form becomes

$$Q(\boldsymbol{x}) = \left(\boldsymbol{z} + \Sigma^{-\frac{1}{2}}\boldsymbol{\mu}\right)^T \Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}} \left(\boldsymbol{z} + \Sigma^{-\frac{1}{2}}\boldsymbol{\mu}\right) \tag{10.16}$$

Let us suppose there exists an orthogonal matrix P, that is, $PP^T = I$ which diagonalizes $\Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}}$, then $P^T \Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}} P = \operatorname{diag}(\lambda_1,\ldots,\lambda_n)$, where $\lambda_1,\ldots,\lambda_n$ are the eigenvalues of $\Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}}$. The quadratic form can now be written as

$$\begin{aligned} Q(\boldsymbol{x}) &= \left(\boldsymbol{z} + \Sigma^{-\frac{1}{2}}\boldsymbol{\mu}\right)^T \Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}} \left(\boldsymbol{z} + \Sigma^{-\frac{1}{2}}\boldsymbol{\mu}\right) \\ &= (\boldsymbol{u}+\boldsymbol{b})^T \operatorname{diag}(\lambda_1,\ldots,\lambda_n)(\boldsymbol{u}+\boldsymbol{b}) \end{aligned} \tag{10.17}$$

where $\boldsymbol{u} = P^T\boldsymbol{z} = (u_1,\ldots,u_n)^T$ and $\boldsymbol{b} = P^T\Sigma^{-\frac{1}{2}}\boldsymbol{\mu} = (b_1,\ldots,b_n)^T$. The expression in (11.10) can be written concisely,

$$Q(\boldsymbol{x}) = \boldsymbol{x}^T A\boldsymbol{x} = \sum_{i=1}^{n} \lambda_i (u_i + b_i)^2 \tag{10.18}$$

It is easily verified that the left hand side of (12.4), that is $\boldsymbol{w}^T\boldsymbol{w}$, is in the quadratic form $Q(\boldsymbol{w})$ with $A = I$, that is, the identity matrix. Thus the collision probability can be computed from the cdf of the quadratic form.

10.2 Series Expansion for the Quadratic Form

We describe below the most general method used to obtain a series expansion for the pdf and cdf of the quadratic form in random variables. Various other methods exists in the literature and we refer the interest readers to [90] for a brief survey. The series expansion that we seek for the pdf of the quadratic form is of the form

$$p_{\mathbf{y}}(y) = p(\mathbf{y} = y) = \sum_{k=0}^{\infty} c_k h_k(y) \tag{10.19}$$

where c_k is a sequence of complex number and $\{h_k\}$ is a known sequence of the form y^k. Let the Laplace transform of $h_k(y)$ be denoted by $L(h_k(y))$. In the expansion sought here, the Laplace transform is of the special form ([59])

$$L(h_k(y)) = \xi(s)\eta^k(s) \tag{10.20}$$

where, for $Re(s) > \alpha$ and α being a real constant, $\xi(s)$ is a non-vanishing (non-zero everywhere) analytic function and $\eta(s)$ is an analytic function with an inverse function $\eta(\zeta(\theta)) = \theta$. Now we are interested in the case where the series expansion is convergent, that is, $\sum_{k=0}^{\infty} c_k h_k(y) < \infty$. For any real number β, let us define

$$\sum_{k=0}^{\infty} c_k h_k(y) \leq \sum_{k=0}^{\infty} |c_k||h_k(y)| \leq \alpha e^{\beta y},\ y \in [0,\infty] \tag{10.21}$$

If the above equation is satisfied almost everywhere, then computing the Laplace transform, we have

$$\int_0^{\infty} e^{-sy}\alpha e^{\beta y} = \alpha \int_0^{\infty} e^{-(s-\beta)y} dy < \infty \tag{10.22}$$

if $(s-a) > 0$. Therefore, from Lebesgue's dominated convergence theorem, we have the following lemma.

Lemma 2. *Let $\{h_k\}_0^{\infty}$ be a sequence of measurable complex valued functions on $[0,\infty]$ and $\{c_k\}_0^{\infty}$ be a sequence of complex numbers such that almost everywhere the following is*

satisfied

$$\sum_{k=0}^{\infty} |c_k||h_k(y)| \leq \alpha e^{\beta y},\ y \in [0,\infty] \tag{10.23}$$

where α, β are real constants. Then when $s > 0$ and $p_{\mathbf{y}}(y) = \sum_{k=0}^{\infty} c_k h_k(y)$, we have

$$L\left(p_{\mathbf{y}}(y)\right) = \sum_{k=0}^{\infty} c_k L(h_k(y)) \tag{10.24}$$

The implications of the above lemma are twofold. The first is that the series expansion is convergent. This however is rather straightforward from our construction of the series expansion. The second is the fact that the Laplace transform of the series $p_{\mathbf{y}}(y)$ can be obtained by taking the Laplace transform of the individual terms of the series. This fact will be used below to derive the pdf and the cdf of the quadratic from. We now state the following theorem without proof. The proof may be found in [90].

Theorem 1. *For $Q(\mathbf{x}) = \mathbf{y} = \mathbf{x}^T A\mathbf{x}$ with $A = A^T > 0, \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma), \Sigma > 0$, the moment generating function $M_Q(t)$ of Q is given by*

$$M_Q(t) = \exp\left(-\frac{1}{2}\sum_{i=1}^{n} b_i^2\right) \exp\left(\frac{1}{2}\sum_{i=1}^{n} b_i^2 (1-2t\lambda_i)^{-1}\right) \prod_{i=1}^{n} (1-2t\lambda_i)^{-\frac{1}{2}} \tag{10.25}$$

where the b_i, λ_i are the parameters of the quadratic form as defined in Section 10.1. Let us now define the series $M(\theta)$ such that

$$M(\theta) = \sum_{k=0}^{\infty} c_k \frac{L(h_k(y))}{\xi(\zeta(\theta))} = \sum_{k=0}^{\infty} c_k \theta^k \tag{10.26}$$

where the infinite series is a uniformly convergent series for θ in some region with $M(\theta) > 0$, $M(0) = c_0$ and $s = \zeta(\theta)$. We note here that if $p_{\mathbf{y}}(y) = 0$ for $y < 0$, then $M_Q(-t)$ represents the Laplace transform of $p_{\mathbf{y}}(y)$. Thus, from (10.25) we have

$$L\left(p_{\mathbf{y}}(y)\right) = \exp\left(-\frac{1}{2}\sum_{i=1}^{n} b_i^2\right) \exp\left(\frac{1}{2}\sum_{i=1}^{n} b_i^2 (1+2s\lambda_i)^{-1}\right) \prod_{i=1}^{n} (1+2s\lambda_i)^{-\frac{1}{2}} \tag{10.27}$$

Using $\zeta(\theta) = \theta^{-1}$, we have

$$L\left(p_{\mathbf{y}}(y)\right) = s^{-\frac{n}{2}} M(\theta) \tag{10.28}$$

Thus we obtain,

$$c_0 = M(0) = \exp\left(-\frac{1}{2}\sum_{i=1}^{n} b_i^2\right)\prod_{i=1}^{n}(2\lambda_i)^{-\frac{1}{2}} \tag{10.29}$$

Differentiating the natural logarithm of $M(\theta)$, we get the following form

$$\ln M(\theta) = d_0 + \sum_{k=1}^{\infty} d_k \frac{\theta^k}{k} \tag{10.30}$$

where

$$\begin{aligned} d_0 &= -\frac{1}{2}\sum_{i=1}^{n} b_i^2 + \ln\prod_{i=1}^{n}(2\lambda_i)^{-\frac{1}{2}} \\ d_k &= \frac{1}{2}\sum_{i=1}^{n}\left(1 - kb_i^2\right)(2\lambda_i)^{-k} \end{aligned} \tag{10.31}$$

From (10.28), we have the following lemma.

Lemma 3.

$$L\left(p_{\mathbf{y}}(y)\right) = \sum_{k=0}^{\infty} c_k(-1)^k s^{-\left(\frac{n}{2}+k\right)} \tag{10.32}$$

We now obtain the required expressions for the pdf and cdf of the quadratic form of $Q(\mathbf{x})$.

Lemma 4. *The cdf of* $Q(\mathbf{x}) = \mathbf{y} = \mathbf{x}^T A\mathbf{x}$ *with* $A = A^T > 0, \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma), \Sigma > 0$ *is*

$$F_{\mathbf{y}}(y) = p(\mathbf{y} \le y) = \sum_{k=0}^{\infty}(-1)^k c_k \frac{y^{\frac{n}{2}+k}}{\Gamma\left(\frac{n}{2}+k+1\right)} \tag{10.33}$$

and its pdf is given by

$$p_{\mathbf{y}}(y) = p(\mathbf{y} = y) = \sum_{k=0}^{\infty}(-1)^k c_k \frac{y^{\frac{n}{2}+k-1}}{\Gamma\left(\frac{n}{2}+k\right)} \tag{10.34}$$

where Γ denotes the gamma function, c_0 and d_0, d_k are the terms defined in (10.29) and (10.31), respectively. The expression for c_k is given by (see Appendix C for derivation)

$$c_k = \frac{1}{k}\sum_{j=0}^{k-1} d_{k-j}c_j,\ k \ge 1 \tag{10.35}$$

Proof. From Lemma 3, we have $L\left(p_{\mathbf{y}}(y)\right) = \sum_{k=0}^{\infty} c_k(-1)^k s^{-\left(\frac{n}{2}+k\right)}$. The lemma is proved by noting that $s^{-\left(\frac{n}{2}+k\right)}$ is Laplace transform of $y^{\frac{n}{2}+k-1}/\Gamma\left(\frac{n}{2}+k\right)$. Integrating the expression for $p_{\mathbf{y}}(y)$, we obtain the required expression for $F_{\mathbf{y}}(y)$.

Theorem 2. *The collision probability for the collision constraint formulated in (12.4) is given by*

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = \sum_{k=0}^{\infty}(-1)^k c_k \frac{y^{\frac{n}{2}+k-1}}{\Gamma\left(\frac{n}{2}+k\right)} \tag{10.36}$$

where $y = (r_1+s_1)^2$.

Proof. From (12.4), the collision constraint is in the quadratic form $Q(\boldsymbol{y})$, with $\boldsymbol{w} \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{x}_k} - \boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{x}_k} + \Sigma_{\mathbf{s}_k}\right)$. We recall here that $\boldsymbol{w} = \mathbf{x}_k - \mathbf{s}_k$, where $\mathbf{x}_k$ and $\mathbf{s}_k$ are the random vectors that denote the robot and obstacle locations, respectively and are distributed according to $\mathbf{s}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{s}_k}\right)$ and $\mathbf{x}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{x}_k}, \Sigma_{\mathbf{x}_k}\right)$. As noted before, the collision probability is the cdf of the quadratic form $Q(\boldsymbol{y})$. Thus from Lemma 4, the above theorem is proved.

10.3 Revisiting Convergence of the Series Expansion

As seen in Lemma 2, the cdf and the pdf of the quadratic form is convergent. In the following, we will derive upper bounds for the truncation error of the series expansions for the pdf and the cdf of the quadratic form.

If the infinite series pdf in (11.16) is truncated after N terms, the truncation error is

$$e(N) = \sum_{k=N+1}^{\infty} |c_k h_k(y)| = \left| \sum_{k=N+1}^{\infty} c_k \frac{y^{\frac{n}{2}+k-1}}{\Gamma\left(\frac{n}{2}+k\right)} \right| \tag{10.37}$$

Using Cauchy's inequality, we get

$$|c_k| \leq \frac{m(\rho)}{\rho^k}, \;\; m(\rho) = \max_{|\theta|=\rho} |M(\theta)| \tag{10.38}$$

Thus we have

$$e(N) \leq \frac{m(\rho)}{\rho^k} \left| \sum_{k=N+1}^{\infty} \frac{y^{\frac{n}{2}+k-1}}{\Gamma\left(\frac{n}{2}+k\right)} \right| \leq$$
$$m(\rho) \left(\Gamma\left(\frac{n}{2}\right) N! \right)^{-1} \left(\frac{y}{2}\right)^{\frac{n}{2}-1} \left(\frac{y}{2\rho}\right)^{N+1} \exp\left(\frac{y}{2\rho}\right) \tag{10.39}$$

where we have used the gamma function identity, $\forall \varsigma > 0,\ \Gamma(\varsigma+1) = \varsigma\Gamma(\varsigma)$. In a similar manner, we obtain the truncation error for the infinite series cdf in (11.15)

$$E(N) \leq m(\rho)\left(\Gamma\left(\frac{n}{2}\right)(N+1)!\right)^{-1}\left(\frac{y}{2}\right)^{\frac{n}{2}}\left(\frac{y}{2\rho}\right)^{N+1}\exp\left(\frac{y}{2\rho}\right) \tag{10.40}$$

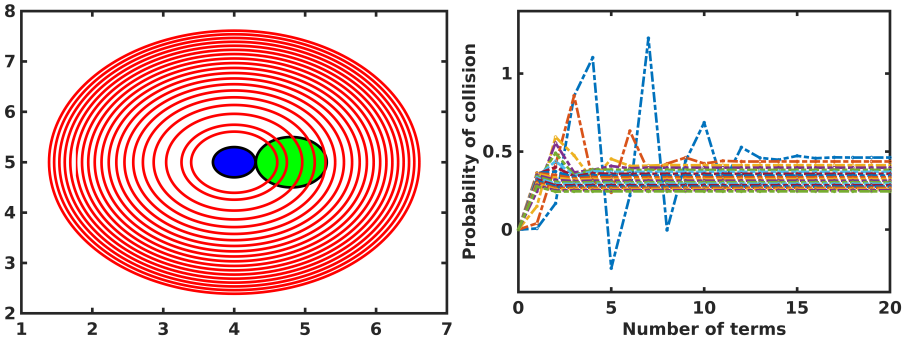
The expression for $m(\rho)$ is obtained from [60],

$$m(\rho) = \prod_{i=1}^{n}\lambda_i^{-\frac{1}{2}}\exp\left(-\frac{1}{2}\sum_{i=1}^{n}\frac{b_i^2\lambda_i}{\lambda_i+\rho}\right)\prod_{i=1}^{n}(1-\frac{\rho}{\lambda_i})^{-\frac{1}{2}} \tag{10.41}$$

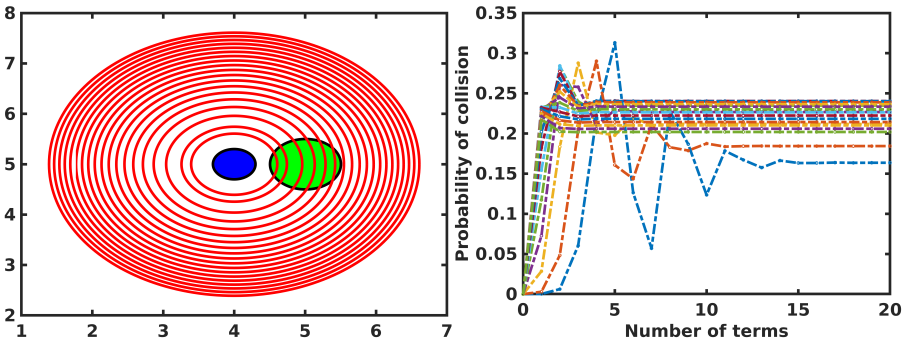
For the expression in (10.41) to be valid, it is required that $\rho < \lambda_i$ and therefore we have $\rho < \min \lambda_i$. As a result, $m(\rho)$ vanishes with $\sum_{i=1}^{n} b_i^2 \to \infty$. We recall here that $\boldsymbol{b} = P^T\Sigma^{-\frac{1}{2}}\boldsymbol{\mu} = (b_1,\ldots,b_n)^T$. Thus, larger the distance from the obstacles and lower the uncertainty in the robot and obstacle positions, the greater is the b_i value. In such scenarios, based on our experience, convergence is often attained within the first few terms of the series.

It is worth noting that for a given robot configuration and obstacle parameters, the varying term in (10.40) is $(y/2\rho)^{N+1}/(N+1)!$. This term is inversely proportional to the parameter ρ. As discussed above ρ depend on λ_i's, that is, the eigenvalues of $\Sigma_k + \Sigma_s$. Thus at time instant k, the parameter that influences the convergence is the degree of uncertainty in both the robot and obstacle locations, that is, $\Sigma_k + \Sigma_s$. This is visualized for different configurations in Fig. 10.1. The blue and green circles represent a robot and an obstacle, respectively. The red ellipses corresponds to the 3σ uncertainties for different covariances $diag(0.04, 0.04)$, $diag(0.08, 0.08)$, $\ldots$, $diag(0.74, 0.74)$. For all the scenarios discussed we choose $E(N) = 0.001$. In Fig. 10.1(a) the robot and the obstacle are touching each other. For each of these covariances, the number of terms for convergence is shown in Fig. 10.1(b). The worst case corresponds to the covariance of $diag(0.04, 0.04)$, requiring 16 terms for convergence (dashed blue line with spikes in Fig. 10.1(b)). In Fig. 10.1(c) the distance between the robot and the obstacle is increased by $0.2m$ and the covariance $diag(0.04, 0.04)$ needs 12 terms for convergence. The distances are further increased by $0.4m$ and $0.8m$ in Fig. 10.1(e),(g) and their worst case convergences are 9 and 5 respectively, as seen in Fig.10.1(f),(h). The number of terms for the worst case convergence corresponds to covariance $diag(0.04, 0.04)$ and the respective timings for collision probability computation are shown in Table 12.1.

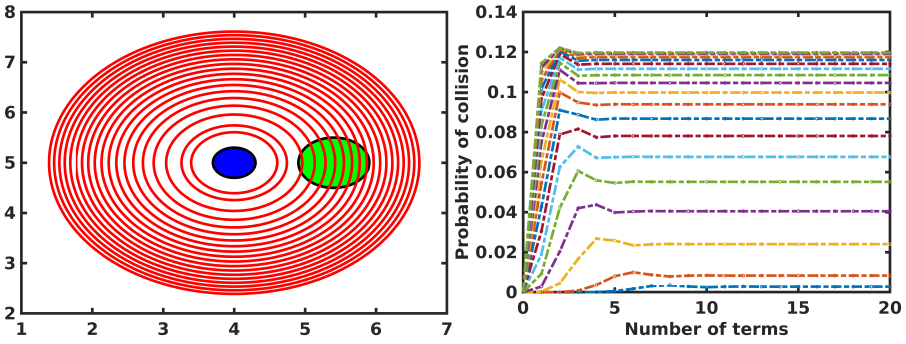
Similarly, the term $(y/2\rho)^{N+1}/(N+1)!$ is directly proportional to y which quantifies the size of the robot and the obstacle. We recall here from (10.14) that $y = (r_1 + s_1)^2$, that is,



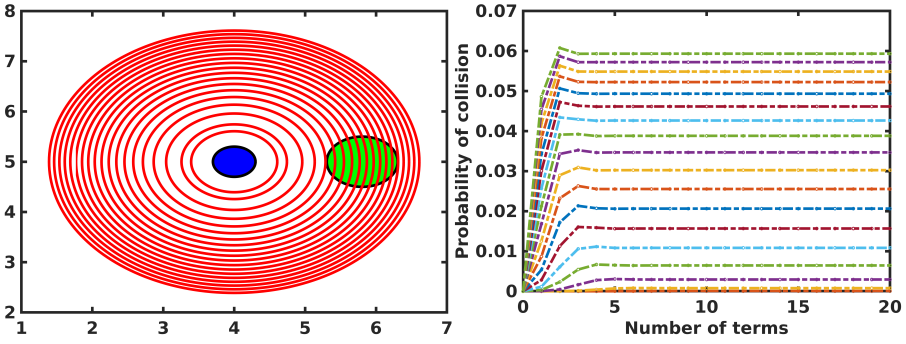
(a) Configuration A **(b)** Collision probability evolution



(c) Configuration B **(d)** Collision probability evolution

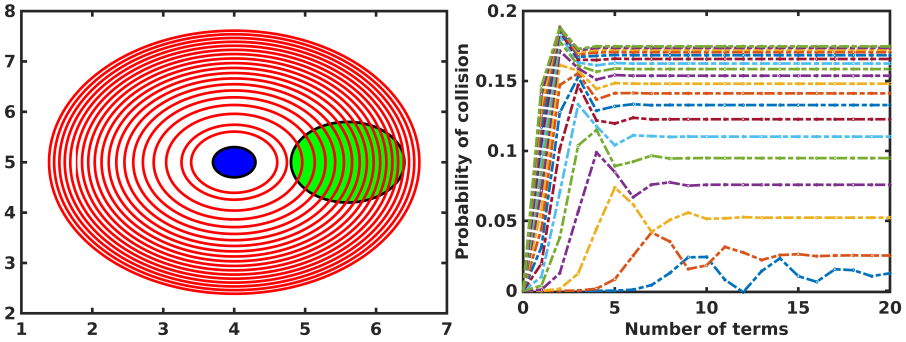


(e) Configuration C **(f)** Collision probability evolution

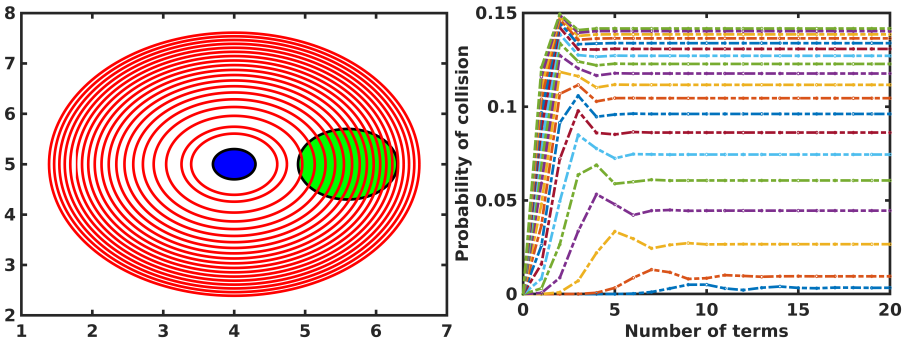


(g) Configuration D **(h)** Collision probability evolution

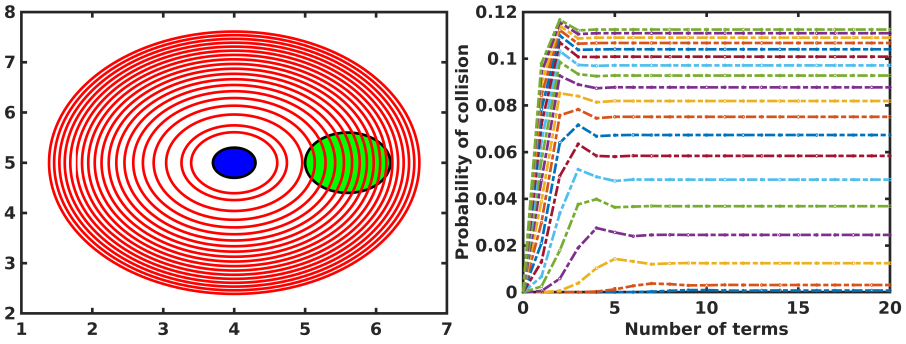
Figure 10.1 Different configurations for a robot of radius $0.3m$ and obstacle of radius $0.5m$. For each configuration the evolution of collision probability is plotted for different covariances. In each of the 4 configurations, the maximum terms for convergence correspond to the minimum covariance of $diag(0.04, 0.04)$.



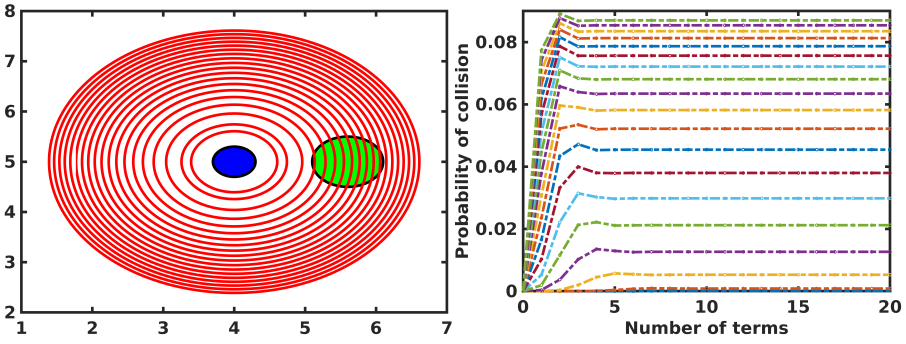
(a) Configuration E **(b)** Collision probability evolution



(c) Configuration F **(d)** Collision probability evolution



(e) Configuration G **(f)** Collision probability evolution



(g) Configuration G **(h)** Collision probability evolution

Figure 10.2 Different configurations for a robot of radius $0.3m$ and obstacle of radius (a) $0.8m$, (c) $0.7m$, (e) $0.6m$ and (g) $0.5m$. In the second column, for each of these configurations the evolution of collision probability is plotted against different covariances.

Configuration	Terms for convergence	Computation time (s)
A	16	0.0412 ± 0.0086
B	12	0.0044 ± 0.0041
C	9	0.0008 ± 0.0003
D	5	0.0004 ± 0.0002

Table 10.1 The maximum number of terms required for convergence and the corresponding collision probability computation time. The values correspond to the covariance $diag(0.04, 0.04)$ for each of the configurations.

Configuration	Terms for convergence	Computation time (s)
E	7	0.0006 ± 0.0005
F	4	0.0004 ± 0.0002
G	3	0.0004 ± 0.0001
H	2	0.0001 ± 0.0000

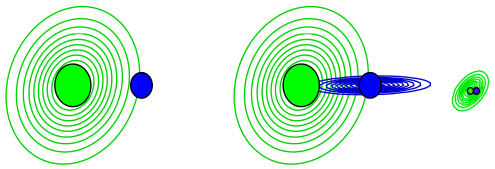
Table 10.2 The maximum number of terms required for convergence and the corresponding collision probability computation time. Each configuration corresponds to different y values with the robot and obstacle locations remaining the same; only obstacle size varies.

the square of the sum of robot and obstacle radius. By keeping the robot size constant and varying the obstacle size, the influence of y on convergence is visualized for four different configurations in Fig. 10.2. In Fig. 10.2(a) $y = 1.1^2(m^2)$ and convergence is obtained within 7 terms. In Fig. 10.2(c),(e),(g) we have $y = 1^2, 0.9^2, 0.8^2$ and the number of terms required for convergence are 4, 3 and 2, respectively. The collision probability computation times are as given in Table 10.2. For $y > 1.1^2$, it can be seen that the number of terms for convergence did not exceed 7 and for $y < 0.8^2$ convergence is achieved with the first two terms. Thus this shows that ρ plays a much larger role in convergence than y.

10.4 Safe Configurations

In the presence of perception and motion uncertainty, providing safety guarantees for robot navigation is imperative. In this Section, we certify safety by defining the notion of a “safe" robot configuration. Let us assume that the obstacle position is known with high certainty as a result of perfect sensing, that is, no significant noise is present. However, since the true state of the robot is not known and only a distribution of these states can be estimated, collision checking has to be performed for this distribution of states. Moreover, in practice, the observations are noisy and this renders the estimated obstacle location (and shape) uncertain. Hence, this uncertainty should be taken into account while considering collision avoidance.

Given a robot configuration $\mathbf{x}_k$, we define the following notion of $\varepsilon-$safe configuration.



(a) **(b)** **(c)**

Figure 10.3 Comparison of our approach with [65, 113, 82, 19]. (a) The robot state (in blue) is known perfectly, however the obstacle location (in green) is uncertain. (b) Robot state uncertainty is considered. The approaches in [19, 82, 113] computes higher values. (c) Point-like robot and obstacle considered. The values computed with [19, 82] are much lower than expected while that of [82] is very high.

Definition 11. *A robot configuration* $\mathbf{x}_k$ *is an* $\varepsilon-$*safe configuration with respect to an obstacle configuration* $\mathbf{s}$*, if the probability of collision is such that* $P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}}\right) \leq 1-\varepsilon$.

For example, a $0.99-$safe configuration implies that the probability of this configuration colliding with the obstacle is at most 0.01. On the one hand, sampling-based motion planning approaches such as the PRM ([56]) consider a discrete state space or a set of controls. As a result, it can only guarantee probabilistic completeness for returning $\varepsilon-$safe configurations since the PRM motion planner is probabilistically complete ([55]), that is the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap. As a result, for sampling-based BSP approaches ([1, 89]), the failure to find an $\varepsilon-$safe configuration might be because such a configuration indeed does not exist or simply because there are not enough samples. On the other hand, continuous state and action space BSP approaches ([109, 88, 85, 43]) do not always guarantee $\varepsilon-$safe configurations. This is merely because there might not be enough measurements to localize the robot or to estimate obstacle locations or both and hence this may preclude computing appropriate control commands.

10.5 Comparison to Other Approaches

[65] compute the collision probability by performing MCI. The joint distribution between the robot and the obstacle $p(\mathbf{x}_k, \mathbf{s}_k)$ is simplified as the product of the individual distributions. This MCI approach results in an expression with double summation for computing the probability of collision. However, [65] approximate this to a single summation expression to decrease computational complexity. [113] compute an upper bound for collision probability using Gaussian chance constraints. [82] compute the collision probability by finding the

Case	Algorithm	Collision probability	Computation time (s)	Feasible
(a)	Numerical integral	4.62%	0.8648 ± 0.0418	Yes
	[65]	4.41%	0.0272 ± 0.0023	Yes
	[19]	5.84%	0.0017 ± 0.0002	Yes
	[82]	33.26%	0.2495 ± 0.3093	No
	[113]	9.60%	0.0021 ± 0.0003	No
	Our approach	4.61%	0.0254 ± 0.0034	Yes
(b)	Numerical integral	8.25%	1.1504 ± 0.0318	Yes
	[65]	7.87%	0.0325 ± 0.0024	Yes
	[19]	14.20%	0.0011± 0.0002	No
	[82]	36.31%	0.2156 ± 0.4068	No
	[113]	16.73%	0.0013 ± 0.0003	No
	Our approach	8.22%	0.0216 ± 0.0023	Yes
(c)	Numerical integral	14.82%	1.1341 ± 0.0211	No
	[65]	15.26%	0.0287 ± 0.0059	No
	[19]	0.46%	0.0015 ± 0.0007	Yes
	[82]	0.61%	0.3233 ± 0.5405	Yes
	[113]	50.00%	0.0018 ± 0.0007	No
	Our approach	14.83%	0.0280 ± 0.0093	No

Table 10.3 Comparison of collision probability approaches.

$\mathbf{x}_k$ that maximizes $p(\mathbf{x}_k, \mathbf{s}_k)$ and formulate the problem as an optimization problem with a Lagrange multiplier. Unlike in [82], which computes the maximum density, [19] use the density associated with the center of the robot. However, this approximation can either be smaller (if covariance is small) or larger than the true value. We formulate the problem as exactly given in each of the works mentioned above to compare it with our approach[1]. The MCI approach of [65] is evaluated using $10,000$ samples. Finally, to validate the value computed using our approach, we also perform the numerical integration of the expression in (11.13), using Monte Carlo method with $10,000$ samples.

Three different cases are considered as shown in Fig. 10.3. The solid green circle denotes an obstacle of radius $0.5m$ and its corresponding uncertainty contours are shown as green circles. The solid blue circle denotes a robot of radius $0.3m$ with the blue circles showing the Gaussian contours. We define a collision probability threshold of 0.09, that is, a $0.91-$safe configuration. The collision probability values and the computation times are summarized in Table 10.3. In Fig. 10.3(a), the robot position is known with high certainty and our approach computes collision probability as 4.61% and hence the given configuration is a $0.91-$safe configuration. The numerical integration of (11.13) gave a value of 4.62%. The

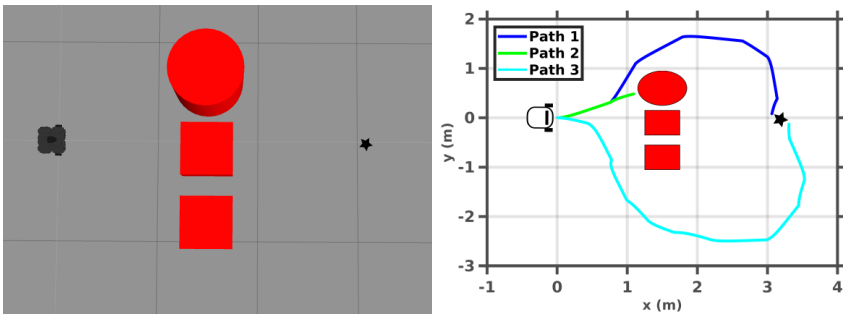
[1]For comparison, the computation of other approaches have been reproduced to the best our understanding and the reproduced codes can be found here- https://bitbucket.org/1729antony/comparison/src/master/

approach of [65] and [19] also estimates the configuration to be feasible, giving a collision probability value of 4.41% and 5.84%, respectively. The collision probability computed as given in [82] is 33.26% (not a $0.91-$safe configuration). Moreover, the value computed is almost seven times higher than the one computed using our approach. Similarly, the value computed using the approach in [113] is 9.60%, predicting the configuration to be unsafe. The higher values are due to the overly conservative nature of the estimates as loose upper bounds are computed. In Fig. 10.3(b), there is robot uncertainty along the horizontal axis and the collision probability computed using our approach is 8.22% and the numerical integration gave a collision probability value of 8.25%. As compared to the previous case, the probability has almost doubled. This is quite intuitive as seen from the robot location uncertainty spread and hence there is greater chance for intersection between the two spheres. The increased chance for collision is also rightly communicated by the values computed using other approaches. The value computed using the approach in [82] gave a much higher value of 36.31%, an increase by 342% as compared to our approach. As in the previous case, the approaches in [19], [113] also gave higher values of 14.20% and 16.73%, respectively, while [65] gave a feasible value of 7.87%.

The approaches in [19, 82] assume that the robot radius is negligible and that the obstacle size is relatively small compared to their location uncertainties. We also compute the collision probabilities for a robot and an obstacle with radius $0.05m$ each, where the robot and the obstacle are touching each other (Fig. 10.3(c)). The obstacle location is also much more certain, with the uncertainty reduced by 97% as compared to cases in Fig. 10.3(a),(b). The numerical integration gave a collision probability value of 14.82%. The probability of collision computed using our approach is 14.83%, whereas, using the approach in [82], the computed value is 0.61%. A lower value of 0.46% is obtained using the approach in [19]. The approach of [113] gave an overly conservative estimate of 50%. The value computed using [65] is 15.26%. To get a sense of the actual value, we compute the area of the covariance matrix, which is $6.28 \times 10^{-4} m^2$. This clearly indicates that 0.61%, 0.46% and 1.69% are too small values while 50.00% is a very high value. Our approach computes the exact probability of collision and outperforms the approaches in [65, 19, 82, 113].

We now provide a comparison in simulation using a scenario shown in Fig. 10.4(a). The robot has to reach the goal position (black star) by avoiding the obstacles in-between. To make the implications of overly conservative estimates (loose upper bounds) explicit, we make the following assumptions. During each planning session[2], the robot can choose from

[2]By a planning session we mean an L look-ahead step planning at the current time and choosing an optimal control. Thus if n planning sessions are required to reach a goal this means that the control action was executed n times.



(a) **(b)**

Figure 10.4 Comparison to other approaches in simulation: (a) Top view of the environment in gazebo. (b) Path 1 is the trajectory executed by the robot following our approach. The trajectory is executed in seven planing sessions. Path 2 leads to collision as upper bounds are computed by other approaches deeming the plans infeasible. The action set is extended and a longer trajectory is executed by the robot using the approach of [113]. The goal is reached in 15 planning sessions.

a set of nine different actions. A description of the motion and observation models can be found in Section 10.10. An action is chosen based on an additive cost of distance to the goal and the collision probability. If the collision probability for an action is greater than 0.01, then this is penalized by redefining the value to be equal to a large number M. The trajectory executed by the robot using our collision probability computation approach is shown in blue (Path 1) in Fig. 10.4(b). The goal was reached in seven planning sessions. For all the other approaches the robot executes the green trajectory (Path 2), leading to a collision. The control action as a result of the first planning session is the same as in Path 1, but the control resulting from the second planning session leads to collision. This is because for all the actions the computed collision probability is greater than 0.01 and hence the values in the cost are redefined to M. Therefore the action chosen is the one that results in minimum distance to the goal and leads to collision. The collision probability values are greater than 0.01 as other approaches compute overly conservative bounds. Our approach computes the exact value and hence the robot reaches the goal safely. We note that collision occurred due to our restrictive action set. If this restriction is relaxed then all the other approaches are able to compute a path with a greater curve than Path 1. One such trajectory is shown in cyan (Path 3), with the collision probabilities computed using the approach by [113]. Thus it is seen that loose upper bounds for collision probability can lead to longer trajectories or in some cases deem all plans to be infeasible.

10.6 Non Circular Geometry

Given two objects (represented as convex polygons), in this Section we derive the collision constraint as a measure of the distance between the mid points of the objects. As before, let us consider two objects, $\mathscr{R} \subset \mathbb{R}^2$ and $\mathscr{S} \subset \mathbb{R}^2$. Let us assume that $\mathscr{S}$ is static and $\mathscr{R}$ can perform translational motions and is approaching $\mathscr{S}$. Then, subtracting $\mathscr{R}$ from $\mathscr{S}$ gives a convex polygon $\mathscr{P}$ such that for any $c \in \mathscr{P}$, then $\mathscr{R} \cap \mathscr{S} \neq \{\phi\}$ ([71]), that is, the convex polygon $\mathscr{P}$ is the set of configurations of $\mathscr{R}$ that leads to collision with obstacle $\mathscr{S}$. Note that, $\mathscr{R}$ and $\mathscr{S}$ are essentially two sets whose elements are the (x,y) pairs belonging to the respective polygons that they represent. Therefore, $\mathscr{P}$ is essentially the Minkowski difference between the two sets $\mathscr{R}$ and $\mathscr{S}$.

Definition 12. *The Minkowski sum of two sets $\mathscr{S}$, $\mathscr{R} \subseteq \mathbb{R}^d$ is*

$$\mathscr{S} + \mathscr{R} = \{s + r \mid s \in \mathscr{S},\ r \in \mathscr{R}\} \tag{10.42}$$

Definition 13. *The Minkowski difference of two sets $\mathscr{S}$, $\mathscr{R} \subseteq \mathbb{R}^d$ is*

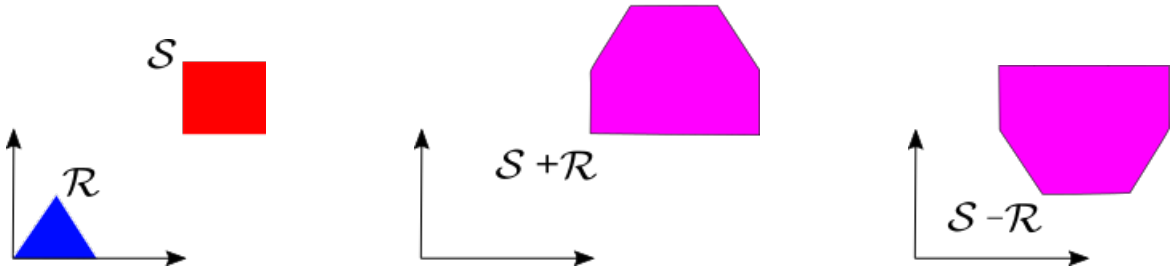
$$\mathscr{S} - \mathscr{R} = \{s - r \mid s \in \mathscr{S},\ r \in \mathscr{R}\} \tag{10.43}$$

The Minkowski sum and difference of two objects are visualized in Fig. 10.5. The Minkowski difference between the two sets $\mathscr{S}$ and $\mathscr{R}$, also called the configuration space obstacle, is the set of (translational) configurations of $\mathscr{R}$ that brings it into collision with $\mathscr{S}$ ([71, 11]). However, we would like to obtain a collision constraint of the form (12.3). In order to obtain such a constraint, we first compute the Minkowski difference between the set $\mathscr{S}$ and the mid-point of $\mathscr{R}$. This gives a new convex set $\mathscr{P}'$ whose elements are formed by subtracting each element of the set $\mathscr{S}$ by the mid-point of object $\mathscr{R}$. In other words, the set $\mathscr{P}'$ is the set of all configurations of the mid-point of $-\mathscr{R}$[3] obtained by shifting/translating this point by each element in the set $\mathscr{S}$.

Lemma 5. *The maximum distance from a point to a line segment occurs at either of the two end-points of the line segment.*

Proof. Given a point P, its distance to line segment AB is PC; see Fig. 10.6. Using Pythagoras theorem, we get $PA > PA'$ since $CA > CA'$. Similarly for $CA \geq CB$, we get $PA \geq PB$, with the equality obtained when PC divides line segment AB equally. Thus the maximum distance from P to AB is either PA or PB.

[3] It holds that $-\mathscr{R} = \{-r \mid r \in \mathscr{R}\}$



(a) Convex objects **(b)** Minkowski sum **(c)** Minkowski difference

Figure 10.5 (a) Two convex objects and the object formed by considering the (b) Minkowski sum and (c) Minkowski difference.

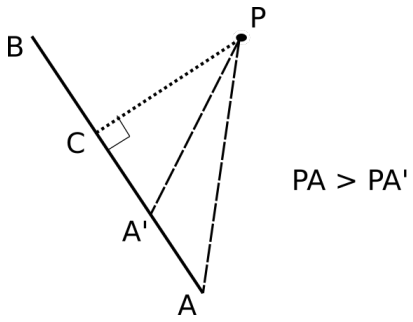


Figure 10.6 Illustration of Lemma 8. The maximum distance from a point P to line segment AB is either PA or PB.

Lemma 6. *The maximum distance from a point P to any other point on a polygon $\mathcal{Q}$ is obtained by computing*

$$\sup\{PV_i \mid V_i \text{ is the vertex of } \mathcal{Q}\} \tag{10.44}$$

where PV_i denotes the line segment from point P to the vertex V_i.

Proof. Given a point inside (or outside) a polygon, the maximum distance to any other point on the polygon is obtained when these points fall on the boundary of the polygon. The boundary of a polygon is formed by edges or line segments and from Lemma 8, the maximum distance from a point to a line segment occurs at the end-points of the line segment. The end-points of these line segments form the vertices of the polygon. Hence it is only sufficient to compute the distances to the vertices of the polygon. Hence the distance from the given point to the vertex of the polygon that is farthest from the point gives the required distance.

For convex polygons $\mathscr{R}$ and $\mathscr{S}$, the boundary configurations of the Minkowski difference represents configurations that lead to contact between $\mathscr{R}$ and $\mathscr{S}$ ([11]), that is, the

configurations where $\mathscr{R}$ and $\mathscr{S}$ touch each other. Also note that the polygon $\mathscr{P}'$ obtained by computing the Minkowski difference between the mid-point of $\mathscr{R}$ and the set $\mathscr{S}$ is fundamentally the set of all translated configurations of the mid-point of $-\mathscr{R}$ in the set $\mathscr{P} = \mathscr{S} - \mathscr{R}$. Thus the collision constraint is obtained by computing the maximum distance between the mid-point of the obstacle $\mathscr{S}$ and the polygon $\mathscr{P}'$.

Theorem 3. *Given a convex polygonal set $\mathscr{R}$ and an obstacle set $\mathscr{S}$, the collision constraint is given by*

$$\sup\{SV_i \mid V_i \text{ is the vertex of } \mathscr{P}'\} \tag{10.45}$$

where S is the mid-point of $\mathscr{S}$ and $\mathscr{P}'$ is the set obtained by computing the Minkowski difference between $\mathscr{S}$ and the mid-point of $\mathscr{R}$.

Proof. We saw above that the collision constraint is obtained by computing the maximum distance between the mid-point of the obstacle $\mathscr{S}$, that is S and the polygon $\mathscr{P}'$. From Lemma 6, the maximum distance is achieved at the vertices of the polygon. Hence, it follows from Lemma 6 that the collision constraint is $\sup\{SV_i \mid V_i \text{ is the vertex of } \mathscr{P}'\}$.

Thus, if $\mathscr{R}$ and $\mathscr{S}$ correspond to the set of points occupied by the robot and the obstacle, respectively, the collision constraint in (12.3) can be written as $\|\mathbf{x}_k - \mathbf{s}_k\|^2 \leq \left(\sup\{SV_i\}\right)^2$.

The Minkowski sum or difference are not invariant to rotations and hence rotation about a reference axis elicits different sets. The resulting sets are obtained by pre-multiplying the starting configuration with the standard rotation matrix of the corresponding angle. This renders different collision constraints for the two given sets. However, while planning for future control commands, the robot pose is often estimated using the motion model and by simulating possible future observations. As a result, an estimate of the robot orientation is computed. Moreover, for static obstacles, both in known and unknown environments, the geometry of the obstacle is a constant[4]. In the case of dynamic obstacles, the orientation of this geometry changes. Thus, assuming that the orientation of the obstacle is known and using the estimated robot orientation, the collision constrained is obtained as elucidated in Theorem 3.

10.7 Complexity Analysis

Finding a trajectory to the goal requires performing Bayesian (EKF) update operations. This involves performing matrix operations, that is, matrix multiplication and inversion of

[4]In this work we assume non-deformable objects.

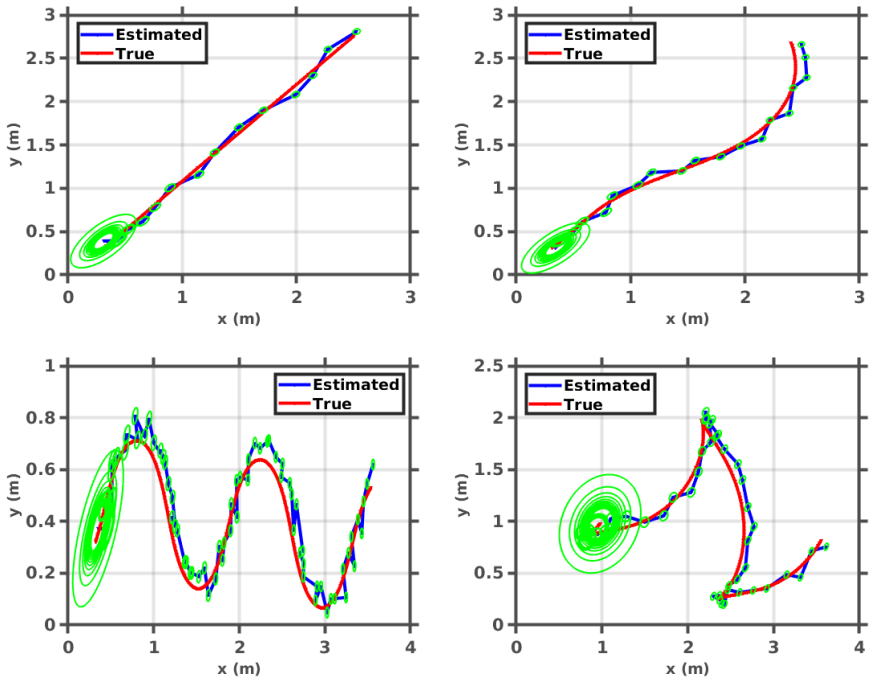


Figure 10.7 True obstacle trajectories plotted along with the estimated obstacle trajectories. In all the cases the linear velocity of the obstacle is greater than or equal to $0.5m/s$. A laser rangefinder is placed at the origin pointing towards the north-east direction. The green ellipses show the estimated covariances. Initial large ellipses correspond to the prior uncertainties. The prior uncertainties shrink as measurements are obtained due to obstacle detection.

matrices. For a state of size n, the covariance matrix is of size $O(n^2)$. Therefore, each step of the Bayesian update has a complexity of $O(n^3)$. Let L denote the number of time steps in the trajectory or the look-ahead horizon, then the overall computational complexity is $O(n^3L)$. Note that this is the complexity while computing the objective function at each time step. The number of times the computation is to be performed cannot be expressed beforehand as it depends on the specific application and objective to be achieved. Let us now analyze the complexity of collision probability computation. From (10.40) we see that for each iteration, the truncation error varies with $(y/2\rho)$. Therefore, to achieve $E(N) \leq \delta$, for an $\varepsilon-$safe configuration, $k = O\left(\log \frac{\delta\rho}{y(1-\varepsilon)}\right)$ iterations are required. We note that for each obstacle, the runtime is increased by this factor.

10.8 Obstacle State Estimation

We adapt the approaches in [97, 82] and describe below the process for estimating future obstacle states. Let us consider that at time instant k, the robot at state is $\mathbf{x}_k$ and the estimated

obstacle location is $\mathbf{s}$. Since the obstacle is following an unknown trajectory, the robot receives a series of measurements $\mathbf{z}_k^1, \ldots, \mathbf{z}_k^n$. Note that since the obstacle is moving, then each measurement $\mathbf{z}_k^i$ corresponds to a different location of the dynamic obstacle. Given the robot pose $\mathbf{x}_k$ and the measurement $\mathbf{z}_k^i$, the obstacle location $\mathbf{s}^i$ can be estimated using the Bayesian approach,

$$p(\mathbf{s}^i|\mathbf{x}_k, \mathbf{z}_k^i) = \eta \; p(\mathbf{z}_k^i|\mathbf{x}_k, \mathbf{s}^i) \; p(\mathbf{s}^i|\mathbf{x}_k) \tag{10.46}$$

where $\eta = 1/p(\mathbf{z}_k^i|\mathbf{x}_k)$ is the normalization constant. Since the obstacle state $\mathbf{s}^i$ is independent of the robot state $\mathbf{x}_k$, we obtain

$$p(\mathbf{s}^i|\mathbf{x}_k, \mathbf{z}_k^i) = \eta \; p(\mathbf{z}_k^i|\mathbf{x}_k, \mathbf{s}^i) \; p(\mathbf{s}^i) \tag{10.47}$$

where $p(\mathbf{s}^i)$ is the prior density. Given the current robot belief $b[\mathbf{x}_k]$ and the measurement $\mathbf{z}_k^i$, the expression $p(\mathbf{z}_k^i|\mathbf{x}_k, \mathbf{s}^i)$ is computed using the measurement model (8.2). Therefore, the mean of the obstacle state $\mathbf{s}^i$ is obtained as

$$\bar{\mathbf{s}}^i = \arg\max_{\mathbf{s}^i} \; p(\mathbf{s}^i|\mathbf{x}_k, \mathbf{z}_k^i) \tag{10.48}$$

with the covariance matrix defined accordingly. Once n measurements are acquired at time k, we use it to estimate future obstacle states within in Model Predictive Control (MPC) strategy, where the robot plans for an optimal sequence of controls for L look-ahead steps. At each look-ahead step, the second term in (10.47), that is the obstacle belief has to be updated as per the obstacle motion model which is unknown. Given the state $\mathbf{s}^n$ obtained from the last measurement $\mathbf{z}_k^n$, the new state $\mathbf{s}'$ can then be predicted as

$$p(\mathbf{s}') = \int_{\mathbf{s}^n} p(\mathbf{s}'|\mathbf{s}^n) \; p(\mathbf{s}^n) \tag{10.49}$$

whose state space form is given by

$$\mathbf{s}(t+1) = A\mathbf{s}(t) + B\mathbf{u}(t) + \nu(t) \tag{10.50}$$

where $\mathbf{u}(t)$ is the control and $\nu(t)$ is the process noise and A, B are matrices which will be defined later. Now we discuss how this prediction can be achieved. From each estimated location $\mathbf{s}^i$ we can then compute the approximate velocities in the x and y directions using the forward difference method. Note that we assume that the obstacle does not change its velocity very drastically and that any two consecutive velocities differ by an $\varepsilon \ll 1m/s$.

Therefore, given $\mathbf{s}^1,\ldots,\mathbf{s}^n$ we obtain the sets

$$\frac{\Delta \mathbf{x}}{\Delta t} = \left\{ \frac{\bar{x}^2 - \bar{x}^1}{\Delta t}, \ldots, \frac{\bar{x}^n - \bar{x}^{n-1}}{\Delta t} \right\} = \left\{ \frac{\Delta \bar{x}_1^2}{\Delta t}, \ldots, \frac{\Delta \bar{x}_{n-1}^n}{\Delta t} \right\}$$
$$\frac{\Delta \mathbf{y}}{\Delta t} = \left\{ \frac{\bar{y}^2 - \bar{y}^1}{\Delta t}, \ldots, \frac{\bar{y}^n - \bar{y}^{n-1}}{\Delta t} \right\} = \left\{ \frac{\Delta \bar{y}_1^2}{\Delta t}, \ldots, \frac{\Delta \bar{y}_{n-1}^n}{\Delta t} \right\} \quad (10.51)$$

where $\bar{x}^i, \bar{y}^i$ are the two components of $\bar{\mathbf{s}}^i$ and Δt is the time between two measurements. In a similar way we also compute the rate of change of the velocities in the x and y directions. From this computed sets, we choose the maximum change in velocities in both directions and denote the corresponding covariances[5] as Σ_v^x and Σ_v^y. From the Taylor series, each component of $\mathbf{s}'$ can be written as

$$x'(t') = x^n(t+\Delta t) \approx x^n(t) + \frac{\Delta \bar{x}_{n-1}^n}{\Delta t}\Delta t + \frac{1}{2}\frac{\Delta^2 \bar{x}_{n-1}^n}{\Delta t^2}\Delta t^2$$
$$y'(t') = y^n(t+\Delta t) \approx y^n(t) + \frac{\Delta \bar{y}_{n-1}^n}{\Delta t}\Delta t + \frac{1}{2}\frac{\Delta^2 \bar{y}_{n-1}^n}{\Delta t^2}\Delta t^2 \quad (10.52)$$

Note that the above equation is in the form of (10.50). The process noise is hence defined as

$$\nu(t) \sim \mathcal{N}\left(0, \begin{bmatrix} \frac{1}{4}\Sigma_v^x(\Delta t)^4 & 0 \\ 0 & \frac{1}{4}\Sigma_v^y(\Delta t)^4 \end{bmatrix}\right) \quad (10.53)$$

We use a 2D laser scanner to estimate the state of dynamic obstacles. It is assumed that the geometry of the obstacle is spherical and is known beforehand. From each scan of the laser rangefinder, the ray with the minimum distance r_j and the corresponding orientation is computed to form a measurement $\mathbf{z}_k^i$. This is repeated to obtain n distinct measurements. Given these measurements and the current robot state estimated using the standard EKF, the x and y components of the obstacle location are estimated. These estimated values are then used to compute the respective velocities using (10.51). The location estimates of the last scan $\mathbf{z}_k^n$ is then used as the prior in (10.49) to estimate future obstacle states. The respective mean and covariance are computed using (10.52) and (10.53). To illustrate our approach, in Fig. 10.7 we plot the true and estimated locations for different obstacle trajectories.

The approach is readily extended to estimate the state of all obstacles detected by the laser scanner. We note here that advanced strategies exits in the literature to efficiently segment

[5]Note that since each variable is Gaussian, their differences are also Gaussian and the corresponding covariances can be computed trivially.

laser rangefinder's scans, but it is not the main focus of the work discussed herein. We therefore employ a rather simpler method sufficient to demonstrate the approach discussed herein. The laser rangefinder returns a sequence of distance measurements and these distances are less than the maximum range when obstacles are encountered. We assume that the obstacles are not too close, that is, there is a least one distance measurement between two obstacles that gives the maximum range. This discontinuity in the distance measurements between two obstacles allows us to separate the laser scanner measurements into different clusters belonging to different obstacles. From each cluster, we estimate the state of the corresponding obstacle. Note that it does not guarantee estimating the state of all the obstacles since some of them could be completely occluded by the others. It is also worth mentioning that estimating the location of static obstacles is a special case of the approach discussed here since for static obstacles both $\frac{\Delta \mathbf{x}}{\Delta t}$ and $\frac{\Delta \mathbf{y}}{\Delta t}$ equate to zero.

10.9 Objective Function

At each time instant k, the robot plans for L look-ahead steps to obtain a control policy $\mathbf{u}^{\star}_{k:k+L-1}$ given by

$$\mathbf{u}^{\star}_{k:k+L-1} = \underset{\mathbf{u}_{k:k+L-1}}{\arg\min} J_k(\mathbf{u}_{k:k+L-1}) \tag{10.54}$$

where $J_k(\mathbf{u}_{k:k+L-1})$ is the objective function. As per the standard MPC, at each time step the first control command $\mathbf{u}^{\star}_k$ is then applied. At each time step, the robot is required to minimize its control usage and proceed towards the goal $\mathbf{x}^g$ avoiding collisions, while minimizing its state uncertainty. We quantify the state uncertainty by computing the trace of the marginal covariance of the robot position. As a result, we have the following objective function

$$\begin{aligned} J_k(\mathbf{u}_{k:k+L-1}) \doteq \sum_{l=0}^{L-1} \left\| \xi(\mathbf{u}_{k+l}) \right\|^2_{M_u} + tr\left(\|M_\Sigma\|^2_{\Sigma_{k+l}} \right) + M_C P(\mathscr{C}_{\mathbf{x}_{k+l}, \mathbf{s}_{k+l}}) \\ + \underset{\mathbf{z}_{k+L}}{\mathbb{E}} \left[\|\mathbf{x}_{k+L} - \mathbf{x}^g\|^2_{M_g} + tr\left(\|M_\Sigma\|^2_{\Sigma_{k+L}} \right) \right] \end{aligned} \tag{10.55}$$

where $\|x\|_S = \sqrt{x^T S x}$ is the Mahalanobis norm, M_u, M_g, M_C are weight matrices and $\xi(\cdot)$ is a function that quantifies control usage. The choice of weight matrices and the control function vary with the application. The term $tr\left(\|M_\Sigma\|^2_{\Sigma_k} \right) = tr\left(M_\Sigma^T \Sigma_k M_\Sigma \right)$ returns the marginal covariance of the robot location. Therefore, $M_\Sigma = \tau \bar{M}_\Sigma$, where τ is a positive scalar and $\bar{M}_\Sigma$ is a matrix filled with zero or identity entries. M_C penalizes the belief states with higher

Algorithm 3 Safe motion planning.

Input: $b[\mathbf{x}_k], L, \varepsilon, N$, radii, $\mathbf{z}_k^1, \ldots, \mathbf{z}_k^n$

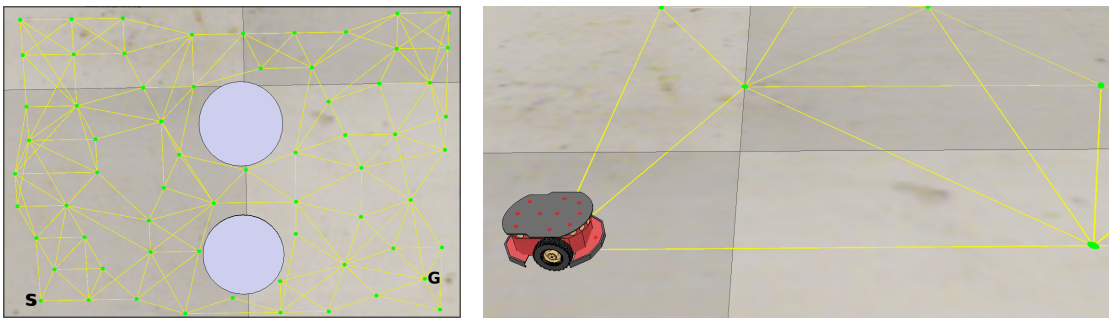
1: $J_k = 0$, $l = 0$
2: compute $p(\mathbf{s}^i|\mathbf{x}_{k+l}, \mathbf{z}_{k+l}^i)\ \forall i, 1 \leq i \leq n$ and $\frac{\Delta \mathbf{x}}{\Delta t}, \frac{\Delta \mathbf{y}}{\Delta t}$
3: **while** true **do**
4: $\quad b[\mathbf{x}_{k+l+1}^-] \leftarrow b[\mathbf{x}_{k+l}] p(\mathbf{x}_{k+1+l}|\mathbf{x}_{k+l}, \mathbf{u}_{k+l})$
5: $\quad \{\mathbf{z}_{k+l+l}\} \leftarrow$ simulate future observations
6: $\quad$ **for** each $\{\mathbf{z}_{k+l+l}\}$ **do**
7: $\quad\quad$ compute $b[\mathbf{x}_{k+l+1}]$
8: $\quad\quad$ predict obstacle state $\mathbf{s}_{k+l}$ (using (10.52))
9: $\quad\quad$ compute $\sum_j P(\mathscr{C}_{\mathbf{x}_{k+l+1}, \mathbf{s}_{k+l}})$
10: $\quad\quad$ compute total_cost (using (10.55))
11: $\quad$ **end for**
12: $\quad J_k \leftarrow J_k +$ total_cost
13: **end while**
14: $\mathbf{u}_{k:k+L-1}^\star \leftarrow \arg\min_{\mathbf{u}_{k:k+L-1}} J_k(\mathbf{u}_{k:k+L-1})$
15: **return** $\mathbf{u}_{k:k+L-1}^\star$

collision probabilities. Since future observations are not available at planning time and are stochastic, the expectation is taken to account for all possible future observations.

Our approach is summarized in Algorithm 3. At each time instant, the robot state is estimated using EKF (lines 4, 7). As described in the previous Section, obstacles are detected using a laser rangefinder. For the $j-$th detected obstacle, its future state is then estimated (line 8) using the approach discussed in Section 10.8. The total collision cost is then computed by adding the collision cost with each obstacle (line 10). Please note that if no $\varepsilon-$safe configuration exists then the algorithm terminates. Finally the total cost is computed as given in (10.55). This is repeated for each horizon step to obtain the optimal control policy $\mathbf{u}_{k:k+L-1}^\star$. The control command $\mathbf{u}_k^\star$ is then applied and the process is repeated till the goal is reached.

10.10 Results

In this section we describe our implementation and then illustrate and explore the capabilities of our proposed approach. First, we present a theoretical example to conceptually understand the proposed approach. Next, we consider both single and multi-robot experiments, which are performed using different Gazebo-based realistic simulations. For all the experiments we use a TurtleBot3 Waffle robot with a radius of $0.22m$. The robot is equipped with a Laser



(a) **(b)**

Figure 10.8 Simulation environment. (a) Scaled-down ($\times\frac{1}{4}$) top view of the environment with the sampled roadmap and start and goal locations of the robot. (b) Pioneer robot at the starting node of the roadmap.

Approach	Robot radius	Obstacle uncertainty	Beacon (object) uncertainty	Planned trajectory
Our	Point	No	No	Fig. 10.10(a)
Our	Point	Yes	No	Fig. 10.10(b)
Our	0.3 m	No	No	Fig. 10.10(a)
[19]	0.3 m	No	No	Fig. 10.10(a)
[82]	0.3 m	No	No	Fig. 10.10(d)
Our	0.3 m	Yes	No	Fig. 10.10(c)
Our	0.3 m	No	Yes	Fig. 10.10(e)
Our	0.3 m	No	No (true beacon location)	Fig. 10.10(f)
Our	0.3 m	No	No (mean beacon location)	Fig. 10.10(a)

Table 10.4 Different configurations used for the 2D environment domain.

Distance Sensor LDS-01 and we use the same to acquire obstacle range and bearing. The performance is evaluated on an Intel® Core i7-6500U CPU@2.50GHz×4 with 8GB RAM under Ubuntu 16.04 LTS. In all the Gazebo based experiments, the initial uncertainty in robot pose is $\Sigma_0 = diag(0.1m, 0.1m, 0.02rad)$. The LDS detections/measurements are only from the obstacles whose motion is unknown and the EKF is employed to predict the robot state at each time step. The ground truth odometry from Gazebo is used to measure the pose of the robot, mimicking a motion capture system. This measurement is then corrupted with noise to perform state estimation. However, this estimation is not performed at each time step and we randomly select the times steps to carry out the same. In this way we explore the robustness of our approach to localization uncertainties.

10.10.1 Theoretical Example 1

We consider the case of a environment where a mobile robot moving in an environment of $30m \times 20m$. A scaled-down top view is seen in Fig. 10.8(a). The underlying PRM graph, the

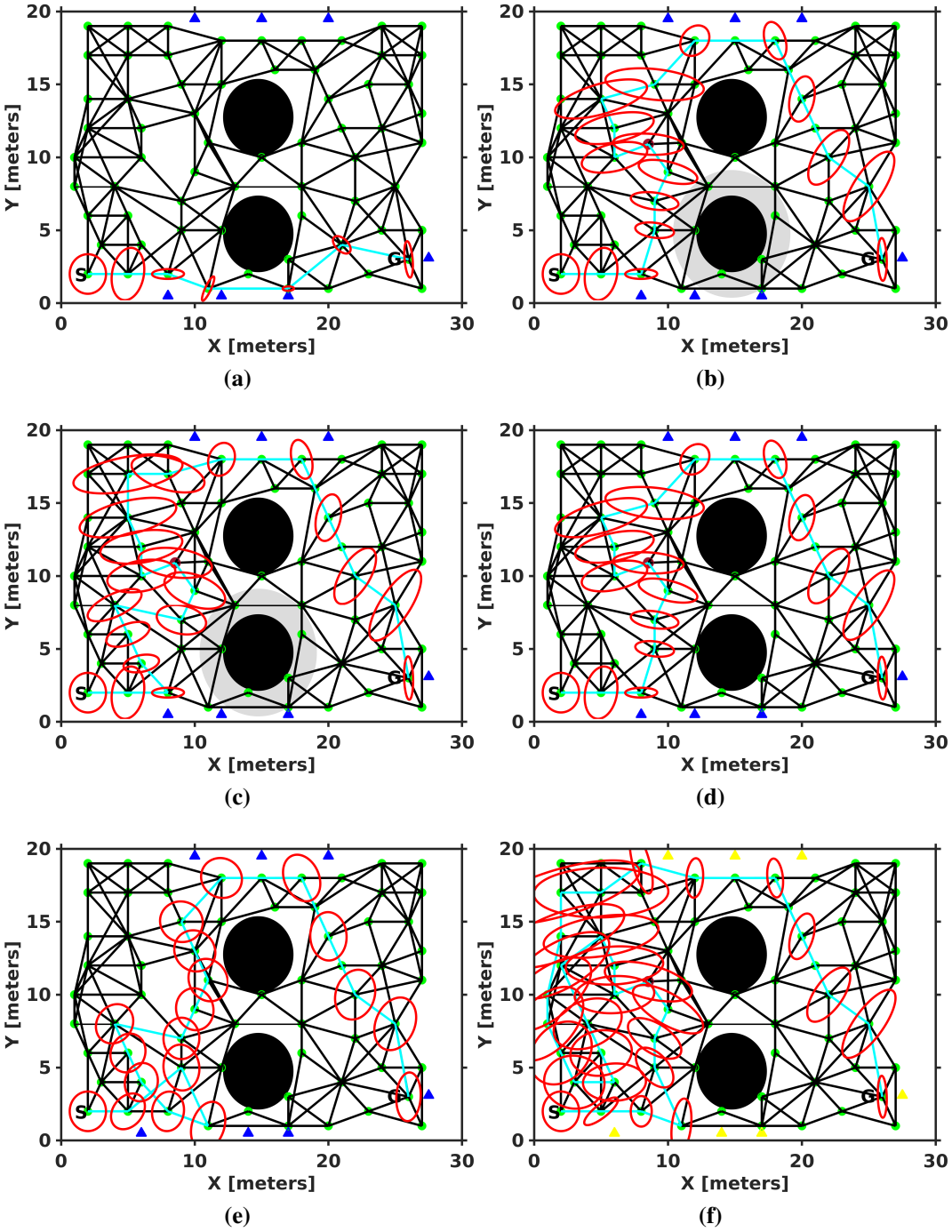


Figure 10.9 Trajectory and the covariance evolution for single planning instantiations are shown. Different cases with obstacle uncertainty for a point robot and a robot of radius $0.3m$ are shown in (a), (b), (c) and (d). (e) The planned trajectory when there is uncertainty in beacon locations. (f) True beacon locations are shown in yellow.

start (S in the figure) and goal (G in the figure) locations can also be seen. The gray circles denote the obstacles in the environment. Fig. 10.8(b) shows a Pioneer P3DX robot at the start location. For the robot motion model, we consider the following non-linear dynamics [105]

$$\begin{aligned} x_{k+1} &= x_k + \delta_{trans}\cos(\theta_k + \delta_{rot1}) \\ y_{k+1} &= y_k + \delta_{trans}\sin(\theta_k + \delta_{rot1}) \\ \theta_{k+1} &= \theta_k + \delta_{rot1} + \delta_{rot2} \end{aligned} \tag{10.56}$$

where $\mathbf{x}_k \doteq (x, y, \theta)$ is the robot pose at time k and $\mathbf{u}_k \doteq (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$ is the applied control. The model assumes that the robot ideally implements the following commands in order: rotation by an angle of δ_{rot1}, translation of δ_{trans} and a final rotation of δ_{rot2} orienting the robot in the required direction. The robot accrue translational and rotational errors while executing $\mathbf{u}_k$ and localizes itself by estimating its position using signal measurements from beacons $\bar{b}_1, \ldots, \bar{b}_7$, which are located at $(x_{\bar{b}_1}, y_{\bar{b}_1}), \ldots, (x_{\bar{b}_7}, y_{\bar{b}_7})$. The signal strength decays quadratically with the distance to the beacon, giving the following observation model with sensor noise v_k,

$$\mathbf{z}_k = \begin{bmatrix} 1/\left((x_k - x_{\bar{b}_1})^2 + (y_k - y_{\bar{b}_1})^2 + 1\right) \\ \vdots \\ 1/\left((x_k - x_{\bar{b}_7})^2 + (y_k - y_{\bar{b}_7})^2 + 1\right) \end{bmatrix} + v_k \tag{10.57}$$

We validate our approach in the above discussed environment by varying different parameters, a summary of which is provided in Table 10.4. Below we detail each of cases considered in Table 10.4. We first consider the motion planning approach for a point-like robot. The cost function is of the form in (10.55) with $M_u = 0.3$, $M_g = diag(0.8, 0.8)$, $M_\Sigma = diag(1,1)$ and $M_C = 10$. The underlying PRM graph with 65 nodes is shown in Fig. 10.10, with the green dots denoting the sampled nodes. The robot, starting from its initial belief state (mean pose denoted by S in the figure) has to reach the node $\mathbf{x}_g$ (G in the figure), while reducing its uncertainty. The blue triangles denote the beacons that aid in localization. The solid black circles with radius 0.5m, represent obstacles in the environment and the red ellipses denote the 3σ covariances (only the (x,y) portion is shown). Unless otherwise mentioned, in all the experiments, $0.99-$safe configurations are solicited and the total planning time is the average time for 25 different runs.

We first consider a case with a point robot and no uncertainty in obstacle location. The planned trajectory in this case is seen in cyan in Fig. 10.10(a) with total planning time of $0.0051s(\pm 0.0008s)$. Please note that the total planning time also includes the collision probability computation time. Next, we consider uncertainty in one of the obstacle

location, whose covariance ellipse is shown in gray. The planned trajectory is seen in cyan in Fig. 10.10(b) and the planning was completed under $0.0279s(\pm 0.0043s)$. Due to the uncertainty in the obstacle location, the robot takes a longer route to avoid collision. A robot of radius $0.3m$ and certain (negligible uncertainty) obstacles gave the same trajectory as in Fig. 10.10(a) with a planning time of $0.0055s(\pm 0.0009s)$. However, when the obstacle location is uncertain the resulting trajectory is as shown in Fig. 10.10(c). A change in the trajectory is observed, as compared to the case of a point robot in Fig. 10.10(b). The planning time in this case is $0.0294s(\pm 0.0047s)$. It is also worth mentioning that in Fig. 10.10(b) and (c), the roadmap was updated by adding a node since a $0.99-$safe configuration could not be found. The added node is seen in brown, with its coordinates being approximately $(9,11)$. We also run the case with no obstacle uncertainty and a robot of radius $0.3m$ using the approach of Park *et al.* [82]. In this case the planned trajectory is as given in Fig. 10.10(d). Note that using our approach, the same scenario gives a shorter trajectory (Fig. 10.10(a)). The longer trajectory computed using the approach in [82] is due to the fact that a loose upper bound is computed for the collision probability. As a result a longer trajectory is obtained. Contrary to this, we compute the exact collision probability and hence a shorter trajectory is synthesized. The same scenario is also run with the approach in [19] and produced a trajectory similar to ours. However, since the uncertainties are significantly lower, the approximate collision probability values computed using [19] are much smaller than the actual values.

Next, we consider the case with uncertainty in the location of the beacons. The considered robot radius is $0.3m$ with the bottom obstacle being uncertain with covariance $diag(0.49, 0.49)$. Taking object uncertainty into account, the planned trajectory with covariance evolution is as shown in Fig .10.10(e). Fig. 10.10(f), shows the trajectory planned with true beacon locations. The beacons are shown in yellow to denote the true location. Considering only the mean position of the beacons and neglecting the position uncertainty, the planned trajectory is as shown in Fig. 10.10(a). Actual execution of this would lead to collision with the bottom obstacle. However, executing the planned trajectory obtained by considering the uncertainty in beacon locations does not violate the $\varepsilon-$safety criterion and all the configurations are $0.99-$safe.

10.10.2 Theoretical Example 2

We consider the case of a mobile robot navigating in a 2D environment of $20m \times 40m$. Fig. 10.10 shows the underlying PRM graph (sampled nodes in green, connected by edges) with 90 nodes. In this domains, the robot (radius $0.3m$), starting from its initial belief state

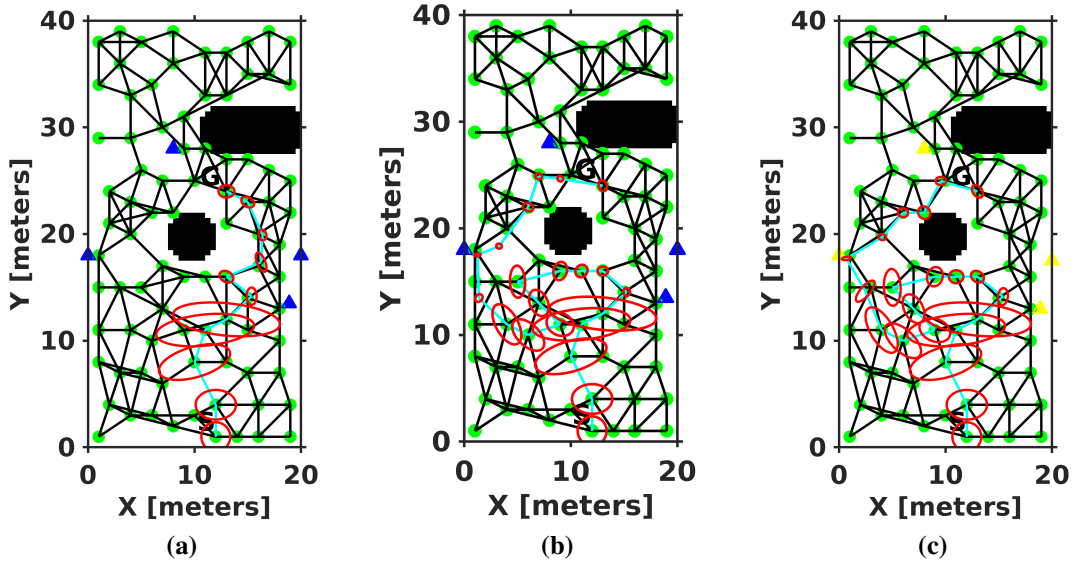


Figure 10.10 Trajectory and the covariance evolution for single planning for the 2D environment. (a) Plan obtained when object uncertainty is not considered. (b) The planned trajectory when object uncertainty is considered (c) Planned trajectory with true landmark locations.

(mean pose denoted by S in the figure) has to reach the node $\mathbf{x}_g$ (G in the figure), minimizing its cost function (11.42). The blue/yellow triangles denote the landmarks in the environment and the solid black blobs represent the obstacles in the environment. The red ellipses denote the 3σ covariances (only the (x,y) portion is shown). Unless otherwise mentioned, in all the experiments, $0.99-$safe configurations are solicited and the total planning time is the average time for 25 different runs.

The state $\mathbf{x}_k \doteq (x_k, y_k, \theta_k)$ is the robot pose (position and orientation) at time k. The applied control vector $\mathbf{u}_k \doteq (\delta_{rot1}, \delta_{trans}, \delta_{rot2})$ consists of an initial rotation δ_{rot1}, followed by a translation of δ_{trans} and a final rotation of δ_{rot2}, orienting the robot in the required direction. The non-linear robot dynamics is thus the same as (10.56).

For robot localization, we consider a landmark based measurement model that returns the range and bearing. The measurement model with noise is thus obtained as

$$\mathbf{z}_k = \begin{bmatrix} r_k^i = \sqrt{(O_k^i(1) - x_k(1))^2 + O_k^i(2) - x_k(2))^2} \\ \\ \phi_k^i = \arctan(\frac{O_k^i(2) - x_k(2)}{O_k^i(1) - x_k(1)}) - x_k(3) \end{bmatrix} + v_k \ , \ v_k \sim \mathcal{N}(0, Q_k) \qquad (10.58)$$

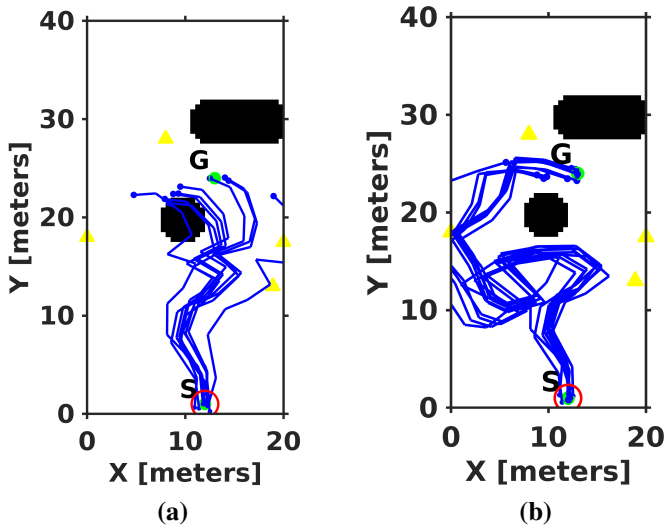


Figure 10.11 Execution traces of robot's true state across ten simulation with initial state drawn from the known initial belief. (a) Computed control when object uncertainty not considered is followed. (b) Traces of robot's true state while following the computed control considering object uncertainty.

where r_k^i and ϕ_k^i are the range and bearing of the i-th object O_k^i (at time k) relative to the robot frame.

The mean landmark locations are $(0,18),(8,28),(20,18),(18.9,13.5)$. The landmarks at $(20,18),(18.9,13.5)$ are not precisely known and has an associated uncertainty of $diag(0.02,0.02)$ in each of their locations. We first neglect the uncertainty and plan using the mean landmark locations. The planned trajectory is seen in cyan in Fig. 10.10(a) and the associated beliefs are seen in red. The overall planning time is $0.0041s(\pm 0.0003s)$. We note here that the overall planning time also includes the collision probability computation time. Next, we consider the landmark uncertainty during planning. The planned trajectory and the associated beliefs are seen in Fig. 10.10(b). We note here that there is a significant change in the planned trajectory. The total planning time in this case is $0.0042s(\pm 0.0008s)$. Finally, we plan using the true landmark locations of $(0,18),(8,28),(20,17.5),(18.9,13)$ which are seen in yellow in Fig. 10.10(c). The overall planning time is $0.0044s(\pm 0.0011s)$. As seen in the figure, the planned trajectory is similar to the case when landmark uncertainty is considered. However, executing the plan synthesized by not considering the landmark uncertainty (scenario in Fig. 10.10(a)) would lead to collision and larger goal state covariance. This is visualized in Fig. 10.11. The traces of true robot state across ten simulations while executing the plan synthesized by neglecting object uncertainty (scenario in Fig. 10.10(a)) is shown in Fig. 10.11(a). The initial state is sampled from the known initial belief and 60% of the execu-

tions lead to collision. Fig. 10.11(b) shows the traces of true robot state across ten simulations while executing the computed control policy by considering object uncertainty (scenario in Fig. 10.10(b)). Therefore, not considering the object uncertainty lead to localization errors and thereby synthesize inefficient plans.

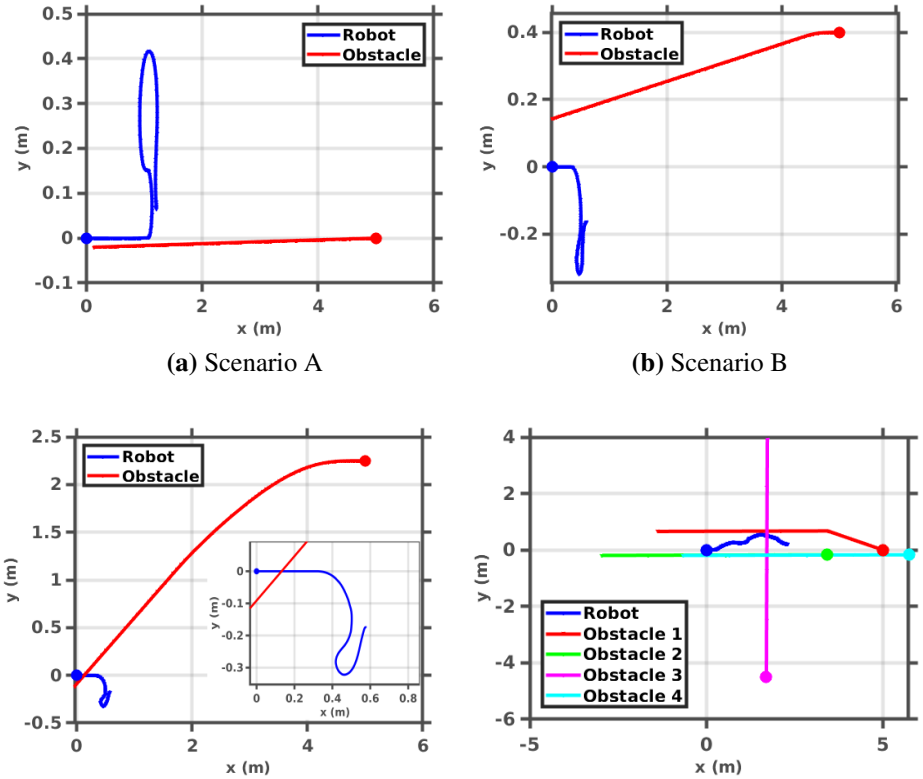
10.10.3 Single-robot Scenarios

In this Section, we discuss our collision avoidance approach considering single-robot scenarios in the Gazebo simulator. Dynamic obstacles are simulated using different robots whose motion model is unknown to the considered robot. The robot kinematics is as follows

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_k - \frac{V_k}{\omega_k}\sin(\theta_k) + \frac{V_k}{\omega_k}\sin(\theta_k + \omega_k \Delta t) \\ y_k + \frac{V_k}{\omega_k}\cos(\theta_k) - \frac{V_k}{\omega_k}\cos(\theta_k + \omega_k \Delta t) \\ \theta_k + \omega_k \Delta t \end{bmatrix} + w_k \tag{10.59}$$

where the applied control $\mathbf{u}_k = (V_k, \omega_k)^T$ is made up of the linear and angular velocities and w_k is the noise as defined in Chapter 8. We define the prior uncertainty in the obstacle location as $diag(0.1m, 0.1m)$ and corrupt the range data returned by LDS with varying noise with variance $0.1 \times rand(1)m^2$. By default, $0.99-$safe configurations are solicited and we use a look-ahead horizon of $L = 7$. Since the TurtleBot3 robot is used, the collision constraint is $\|\mathbf{x}_k - \mathbf{s}_k\|^2 \leq (0.22 + 0.22)^2$. In each experiment, we consider a robot starting from the location $(0,0)$ and having to reach the goal location of $(3,0)$, subject to minimizing the objective function in (11.42).

First, we consider three scenarios (A, B and C) where the robot has to avoid head-on collisions with dynamic obstacles. The obstacle linear velocities in each scenario are $0.5m/s$, $1.0m/s$ and $2.5m/s$, respectively. This is however unknown to the robot and at each time step obstacle states are estimated using the approach detailed in Section 10.8. The robot and obstacle trajectories for the three scenarios are shown in Fig. 10.12(a)-(c). Note that evading a collision is the main focus of these experiments and hence only the relevant trajectories are plotted. Snapshots of four different stages during each the trajectory execution are shown in Fig. 10.13. The first row corresponds to scenario A, the second to scenario B and the third row displays snapshots of scenario C. For all the scenarios, stage 1 shows the initial configuration of the robot and the obstacle. Once the obstacle is detected, a control command for evading the obstacle is computed to move towards a $0.99-$safe configuration. The beginning of execution of such a control command is seen in stage 2. Stage 3 shows the snapshot when



(a) Scenario A **(b)** Scenario B

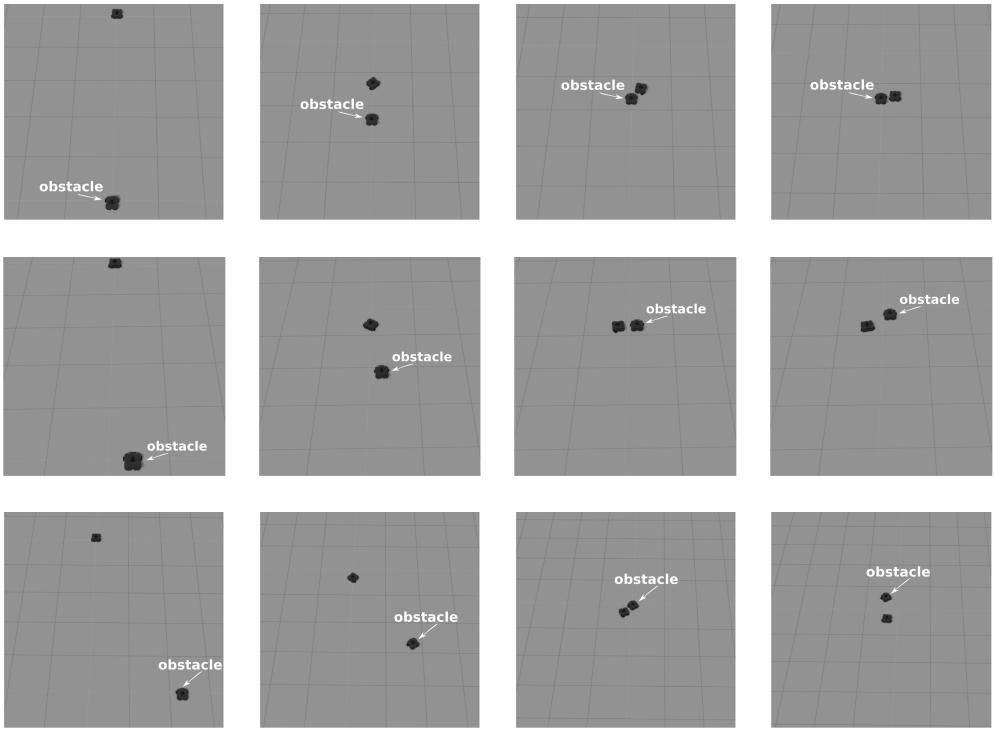
(c) Scenario C **(d)** Scenario D

Figure 10.12 Top view of robot and obstacle trajectories are plotted with the starting locations marked as round blobs. The robot trajectory is shown in blue. (a) Single obstacle with velocity of $0.5m/s$. (b) Obstacle velocity is $1.0m/s$. (c) Obstacle velocity is $2.5m/s$ and the zoomed figure is shown in the inset. (d) Four obstacles with different velocities.

the obstacle and the robot are very close to each other with the robot evading the obstacle to avoid collision. In stage 4 it is seen that the robot has successfully avoided collisions.

For each scenario, the experiment is performed 50 times and the average time for computing the associated collision probability is shown in the first three rows of Table 10.5. The last row corresponds to Scenario D, a multi-obstacle scenario which will be described soon. As a safety metric, the minimum distance between the two robots is also measured and the results are shown in Table 10.5. For all the scenarios a success rate of 100% is achieved, that is, in all the 50 experiments, there were no collisions. However, lower look-ahead horizon, that is, $L < 7$ did not give 100% success rate as most often the obstacles were too close before executing the appropriate control command. Another parameter that affects the success rate is the value of ε. For example, a $0.4-$safe configuration always resulted in collision for scenarios B and C.

In scenario *D*, we consider four obstacles, each with different velocities. The robot successfully evades collision with all the four obstacles and the results are shown in the last row of Table 10.5. The trajectories of the robot and the obstacles can be seen in Fig. 10.12(d). Aerial snapshots at different time instants are shown in Fig. 10.14. Stage 1 corresponds to the initial configuration of the robot and the obstacles. Stages 2 and 3 show the robot moving to evade a head-on collision with obstacle 2 (obstacle numbers in Fig. 10.12(d)). Stages 4 through 7 show different instances while the robot tries to evade the remaining obstacles. Finally, in stage 8, the robot has successfully avoided potential collisions. The mean computation time for collision probability is $0.3682s$ and the computation time of the entire framework is $0.4230s$. The entire framework time includes the time for collision probability computation, uncertainty propagation, and obstacle state estimation.

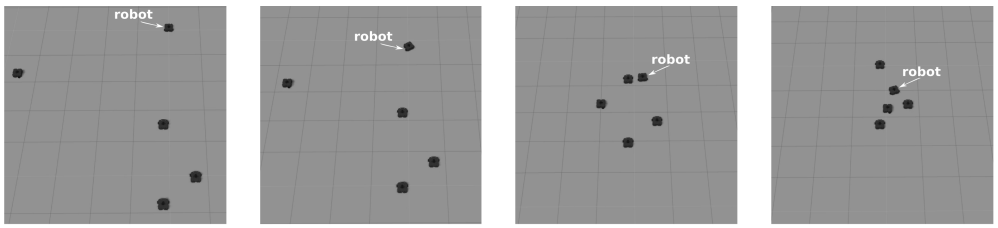


(a) Stage 1 **(b)** Stage 2 **(c)** Stage 3 **(d)** Stage 4

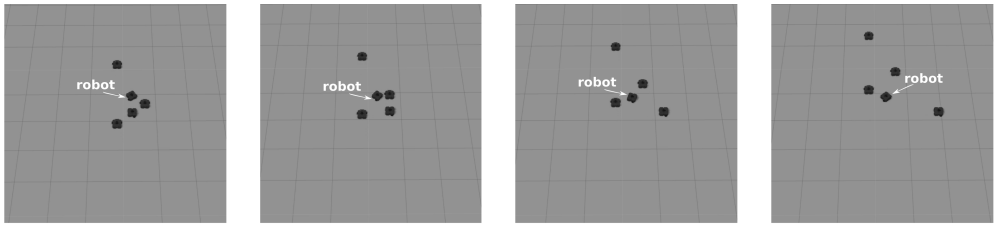
Figure 10.13 Top view snapshots of the robot and the obstacle at four different stages (from left to right) of the experiment in scenarios *A* (row 1), *B* (row 2) and *C* (row 3). Positive x-axis is vertically downwards.

Scenario	Minimum distance (m)	Collision probability computation time (s)
A	0.16	0.0267 $\pm$ 0.0078
B	0.31	0.0189 $\pm$ 0.0074
C	0.12	0.0191 $\pm$ 0.0072
D	0.20	0.0368 $\pm$ 0.0023

Table 10.5 The minimum distance between the robot and the obstacles and the collision probability computation time for four different scenarios. The minimum distance corresponds to the minimum among all the distances between robots and the obstacles.



(a) Stage 1 **(b)** Stage 2 **(c)** Stage 3 **(d)** Stage 4



(e) Stage 5 **(f)** Stage 6 **(g)** Stage 7 **(h)** Stage 8

Figure 10.14 Top view snapshots of the robot and the obstacles in the Gazebo environment at different stages of the experiment in scenario *D*.

10.10.4 Multi-robot Scenarios

In this Section we demonstrate our approach with multi-robot planning scenarios. In this setting, each robot considers all other robots as dynamic obstacles. However, there is no communication between the robots and the obstacle/robot states are estimated using the approach described in Section 10.8.

We first consider different scenarios with two robots. The initial pose of the robots are $(0,0,0)$ and $(3,0,-\pi)$ and the goal for each robot is to navigate towards the starting location of the other robot. The starting configuration and the executed trajectory of scenario I can be seen in Fig. 10.15(a), (b). Scenario II, which includes a cube and a cylinder as static obstacles, is shown in Fig.10.15(c), (d). It can be seen that the robots evade collision with each other and the static obstacles and navigate between the obstacles. The locations of

Scenario	Minimum distance (m)	Collision probability computation time (s)
I	0.33	0.0117 ± 0.0044
II	0.08	0.0137 ± 0.0123
III	0.51	0.0099 ± 0.0013
IV	0.10	0.0211 ± 0.0052

Table 10.6 Minimum distance between the robot and the obstacles in four scenarios and the corresponding collision probability computation time.

the static obstacles are unknown to the robots and they are estimated using the approach discussed in Section 10.8. However, we assume known data association and we apply the collision constraint derived in Theorem 3. The obstacles in scenario II are pulled closer in scenario III (Fig. 10.15(e),(f)) to prevent the robots from passing between the obstacles. This is rightly estimated by the robots and they navigate around the obstacles to reach the goal. However it was seen that for $L < 7$, both robots turned to the same side and 20% (10 out of 50) of the time this leads to collision. This is so because, as the robots turn to the side of the cube, the cube occludes one robot from the other. By the time each robot turns around the cube and see the other, they are already too close to avoid collision. In scenario IV (Fig. 10.15(g),(h)), we consider four robots, where the robots facing each other are required to swap their positions. The initial poses of each robot are $(0,0,0)$, $(1.5,-1.5,\frac{\pi}{2})$, $(3,0,-\pi)$ and $(1.5,1.5,-\frac{\pi}{2})$, respectively.

Table 10.6 shows the statistics for the four scenarios discussed above. The minimum distance between the robot and the obstacle and the average computation time for evaluating the collision probability are reported. In scenario IV, it was seen that $\varepsilon < 0.99$ leads to collision in 80% of the experiments. For the other scenarios, $\varepsilon < 0.9$ successfully evaded collision in all the experiments.

10.11 Discussion

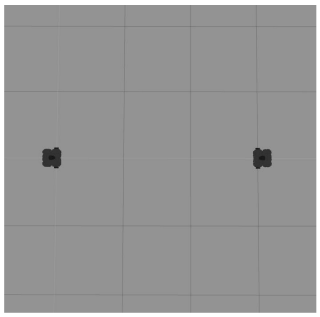
In Section 10.5, we have compared our approach to other similar techniques ([65, 19, 82, 113]) and it is seen that our approach outperforms them. In this section we outline few limitations and discuss how to overcome them by proposing suitable extensions. These extensions would enhance the capability and robustness of our approach in challenging scenarios.

In Section 10.8, we have modelled the object uncertainty as a Gaussian. The assumption is justified in the case of Gaussian belief states, and works for all practical situations. Yet,

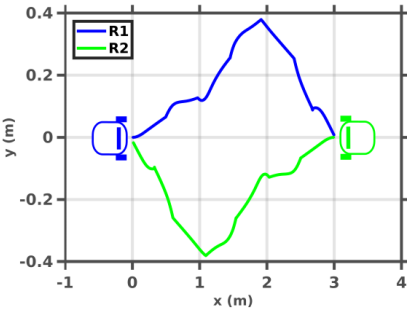
in general, the model might not be Gaussian and it has to be determined based on the environment, sensing model and the robot task that needs to be achieved. Finding an appropriate model, especially for non-Gaussian belief states is a work for the future.

The collision probability approach discussed in this chapter is not restrictive to mobile robots and is readily extended to any 3D rigid body robot. For example, a quad-rotor can be approximated using a minimum volume enclosing sphere and therefore our approach can be used directly. Similarly, in the a manipulator robot each link is approximated by minimum volume bounding spheres that tightly enclose the link. For such robots, the collision with an obstacle has to be checked for each bounding volume. For example, let us consider a manipulator robot with l bounding spheres. Then the collision condition for the $i-$th sphere is given by $\mathscr{C}_{\mathbf{x}_k^i,\mathbf{s}_k}$, where $\mathbf{x}_k^i$ is the center of the $i-$th sphere. Furthermore, an alternative and more appropriate approach is to consider the minimum-volume enclosing ellipsoid for each link ([91]). For every convex polyhedron, there exists a unique ellipsoid of minimal volume that contains the polyhedron and is called the *Löwner-John ellipsoid* of the polyhedron ([31]). Thus each link can be represented by their corresponding Löwner-John ellipsoids. However, the collision condition in (12.3) is no longer valid. The collision condition should be reformulated using the distance between two ellipsoids. Please note that the representation using Löwner-John ellipsoid is also extended to the 3D obstacles.

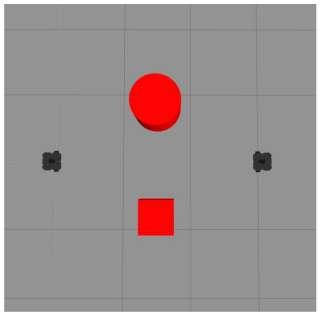
While formulating the objective function in Section 10.9, we assume that the set of actions from which the robot can plan its future control is known *a priori*. In other words, a finite action set is considered. This justifies the inclusion of the collision cost term $P(\mathscr{C}_{\mathbf{x}_{k+l},\mathbf{s}_{k+l}})$ in (11.42). However, our approach is not limited to any specific set of actions or trajectories. The general approach would be to include the set of all possible control actions. The objective function in (11.42) is then reformulated as an optimization problem with the collision cost term included as a constraint to keep the collisions within the $1-\varepsilon$ bound.



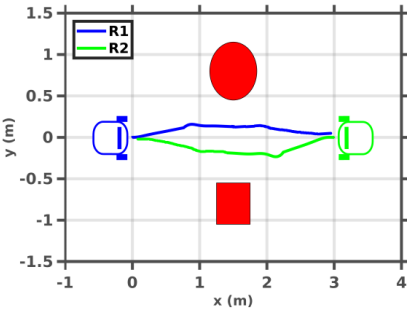
(a) Scenario I



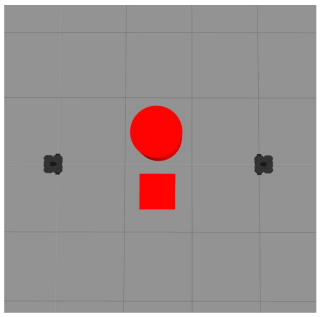
(b) Executed trajectory



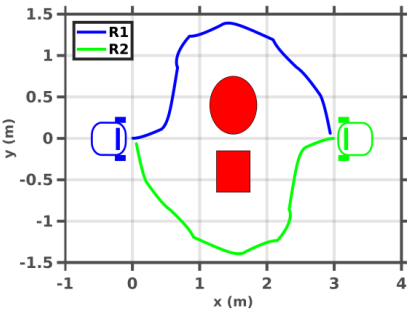
(c) Scenario II



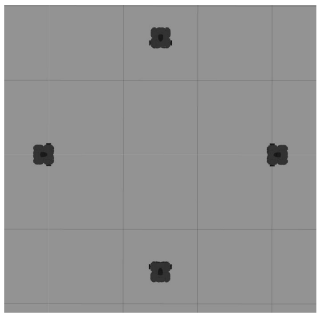
(d) Executed trajectory



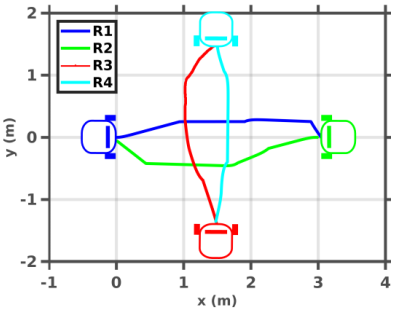
(e) Scenario III



(f) Executed trajectory



(g) Scenario IV



(h) Executed trajectory

Figure 10.15 Different multi-robot scenarios and the corresponding trajectories executed by the robots.

Chapter 11

Bounded Collision Probability

In this chapter, we relax the spherical geometry assumption of the previous chapter and consider the minimum-volume enclosing ellipsoids. For every convex polyhedron, there exists a unique ellipsoid of minimal volume that contains it and is called the *Löwner-John ellipsoid* of the polyhedron [31]. The ellipsoid representation provides a much better approximation of the polyhedron as compared to the spherical representation. This chapter is thus **C9** of our contribution. The collision constraint is thus formulated as the distance between *Löwner-John ellipsoids*. Furthermore, we derive a tight upper bound for fast approximation of the collision probability for 3D motion planning.

11.1 Distance Between Two Ellipsoids

Definition 14. *For a real $m \times n$ matrix A and a vector $\mathbf{b} \in \mathbb{R}^m$, the solution set $\{\mathbf{x} \in \mathbb{R}^n | A\mathbf{x} \leq \mathbf{b}\}$ is called a polyhedron $\mathscr{P}$.*

We note that, a scalar-valued function $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function if $f(\lambda \mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) \leq \lambda f(\mathbf{x}_1) + (1-\lambda) f(\mathbf{x}_2), \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^n$ and $0 \leq \lambda \leq 1$.

Definition 15. *A set $\mathscr{E}^n(A, \boldsymbol{a}) \subseteq \mathbb{R}^n$ is an $n-$dimensional ellipsoid if there exists a vector $\boldsymbol{a} \in \mathbb{R}^n$ and a positive definite $n \times n$ matrix A such that*

$$\mathscr{E}^n(A, \boldsymbol{a}) = \{\mathbf{x} \in \mathbb{R}^n | \, (\mathbf{x} - \mathbf{a})^T A \, (\mathbf{x} - \mathbf{a}) \leq 1\} \tag{11.1}$$

Theorem 4. *For every convex polyhedron $\mathscr{P} \subseteq \mathbb{R}^n$, there exits a unique ellipsoid $\mathscr{E}$ of minimal volume that contains $\mathscr{P}$, called the **Löwner-John ellipsoid** of $\mathscr{P}$.*

Proof. The proof may be found in [48] and a comprehensive treatment is available in [31]. □

For computing collision-free paths it is imperative that distance between the robot and its environment is known. The geometry of robots and other objects in the environment can be expressed using polyhedrons or a combination of polyhedrons. From Theorem 4, there exists a unique minimum volume ellipsoid (Löwner-John ellipsoid) that encloses a convex polygon $\mathscr{P}$. It is also noteworthy that non-convex polyhedrons can be decomposed into overlapping convex polyhedrons. As result it is suffices to compute the distance between the Löwner-John ellipsoids of the robot and the obstacle, respectively.

A convex optimization approach for computing Löwner-John ellipsoids can be found in [91]. In this work we assume that these ellipsoids are known to us. Note that for an ellipsoid $\mathscr{E}(A,\mathbf{a})$, $\mathbf{a}$ is the center of the ellipsoid. Thus our assumption of known ellipsoids mean that at each planning instant the matrix A and the vector $\mathbf{a}$ is known to us. Typically, $\mathbf{a}$ is the center of the robot or the objects that the ellipsoid encloses. Thus, once a Löwner-John ellipsoid $\mathscr{E}(A,\mathbf{a})$ is computed, at each planning instant, the variable quantity is the matrix A, that varies with orientation. It is noteworthy that while planning for future control commands, the robot state is often estimated using the motion model and by simulating possible future observations. As a result, the quantity $\mathbf{a}$ as well as the orientation is known. An appropirate rotation matrix is then used to compute A. Moreover, for static obstacles, both in known and unknown environments, the geometry of the obstacle is a constant and hence the corresponding Löwner-John ellipsoids remain constant. In the case of dynamic obstacles, the state of these obstacles change and a state estimation technique is required to compute the varying enclosing ellipsoids.

We now ompute the distance between two ellipsoids $\mathscr{E}_1,\mathscr{E}_2$ which will be denoted by $d(\mathscr{E}_1,\mathscr{E}_2)$. This will serve as a precursor to obtaining the collision condition. First, we will compute the distance of a point $\mathbf{x}_0 \in \mathbb{R}^n$ from an $n-$dimensional ellipsoid $\mathscr{E}^n(A,\mathbf{a})$. For this purpose, we will use the method outlined in [91].

We wish to compute the minimum distance of a point $\mathbf{x}_0$ from the ellipsoid $\mathscr{E}^n(A,\mathbf{a})$. We assume that the point lies outside the ellipsoid and further without loss of generality assume that $\mathbf{x}_0$ is at the origin of the global $\mathbb{R}^n$ frame. Thus the problem reduces to

$$\text{minimize } \|\mathbf{x}\|, \text{ subject to } (\mathbf{x}-\mathbf{a})^T A(\mathbf{x}-\mathbf{a}) = 1 \tag{11.2}$$

We will now state two lemmas without proof through the remainder of this section. The proofs may be found in [91].

Lemma 7. *Let* $\mathbf{x}^*$ *be the solution to (11.2). Then*

$$\mathbf{x}^* = \lambda_0(M)\left[\lambda_0(M)A - I\right]^{-1} A\mathbf{a} \tag{11.3}$$

where $\lambda_0(M)$ *is the eigenvalue with minimal real part (minimal eigenvalue) of a* $2n \times 2n$ *matrix M given by*

$$M = \begin{bmatrix} \tilde{A} & -I \\ -\tilde{a}\tilde{a}^T & \tilde{A} \end{bmatrix} \tag{11.4}$$

with $\tilde{A} = A^{-1}$ *and* $\tilde{a} = A^{-1/2}\mathbf{a}$*. Therefore the distance of a point* $\mathbf{x}$ *from the ellipsoid* $\mathscr{E}^n(A,\mathbf{a})$ *is given by* $\|\mathbf{x} - \mathbf{x}^*\|$.

Thus, the distance of a point from an ellipsoid is computed as an eigenvalue problem. This can be leveraged to arrive at the collision condition between two ellipsoids. Let us consider two ellipsoid, $\mathscr{E}_1 = \mathscr{E}^n(B,\mathbf{b})$ and $\mathscr{E}_2 = \mathscr{E}^n(C,\mathbf{c})$. We compute a point $\mathbf{x}^* \in \mathscr{E}_2$ such that the ellipsoid level surface surrounding $\mathscr{E}_1$ first touch $\mathscr{E}_2$ at $\mathbf{x}^*$.

Lemma 8. *Given two ellipsoids* $\mathscr{E}^n(B,\mathbf{b})$ *and* $\mathscr{E}^n(C,\mathbf{c})$*, the point* $\mathbf{x}^* \in \mathscr{E}_2$ *at which the ellipsoid level surface surrounding* $\mathscr{E}_1$ *first touch* $\mathscr{E}_2$ *is given by*

$$\mathbf{x}^* = \mathbf{b} + \lambda_0(M')B^{-1/2}\left[\lambda_0(M')I - \tilde{C}\right]^{-1}\bar{c} \tag{11.5}$$

where $\lambda_0(M')$ *is the minimal eigenvalue of a* $2n \times 2n$ *matrix* M' *given by*

$$M' = \begin{bmatrix} \tilde{C} & -I \\ -\tilde{c}\tilde{c}^T & \tilde{C} \end{bmatrix} \tag{11.6}$$

with $\tilde{C} = \bar{C}^{-1}$ *and* $\tilde{c} = \bar{C}^{-1/2}\bar{c}$*, where* $\bar{C} = B^{-1/2}CB^{-1/2}$ *and* $\bar{c} = B^{1/2}(\mathbf{c} - \mathbf{b})$.

11.2 Collision Condition

For two ellipsoids $\mathscr{E}_1 = \mathscr{E}^n(B,\mathbf{b})$ and $\mathscr{E}_2 = \mathscr{E}^n(C,\mathbf{c})$, a collision between them occurs if $\mathscr{E}_1 \cap \mathscr{E}_2 \neq \{\phi\}$. We will denote this collision condition by

$$\mathscr{C}_{\mathbf{b},\mathbf{c}} \doteq \left\{\mathscr{E}_1, \mathscr{E}_2 | \mathscr{E}_1 \cap \mathscr{E}_2 \neq \{\phi\}\right\} \tag{11.7}$$

We now define a quadratic form in random variables and later show that that the collision condition can be written in terms of the quadratic form in random variables of the difference in

robot and obstacle locations. Note that the concepts of quadratic form in random variables and its cdf and pdf were already presented in Chapter 10. However, for the sake of completeness and the ease of the readers we present them here again.

Definition 16. *Let* $\mathbf{x} = (x_1, \ldots, x_n)^T$ *denote a random vector with mean* $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^T$ *and covariance matrix* Σ*. Then the quadratic form in the random variables* $x_1, \ldots, x_n$ *associated with an* $n \times n$ *symmetric matrix* $A = (a_{ij})$ *is*

$$Q(\mathbf{x}) = Q(x_1, \ldots, x_n) = \mathbf{x}^T A \mathbf{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j \tag{11.8}$$

Let us define $\mathbf{y} = \Sigma^{-\frac{1}{2}} \mathbf{x}$ and define a random vector $\mathbf{z} = \left(\mathbf{y} - \Sigma^{-\frac{1}{2}} \boldsymbol{\mu} \right)$. The resulting distribution of $\mathbf{z}$ is thus zero mean with covariance being the identity matrix. Thus the quadratic form becomes

$$Q(\mathbf{x}) = \left(\mathbf{z} + \Sigma^{-\frac{1}{2}} \boldsymbol{\mu} \right)^T \Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}} \left(\mathbf{z} + \Sigma^{-\frac{1}{2}} \boldsymbol{\mu} \right) \tag{11.9}$$

Suppose there exists an orthogonal matrix P, that is, $PP^T = I$ which diagonalizes $\Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}}$, then $P^T \Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}} P = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of $\Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}}$. The quadratic form can now be written as

$$\begin{aligned} Q(\mathbf{x}) &= \left(\mathbf{z} + \Sigma^{-\frac{1}{2}} \boldsymbol{\mu} \right)^T \Sigma^{\frac{1}{2}} A \Sigma^{\frac{1}{2}} \left(\mathbf{z} + \Sigma^{-\frac{1}{2}} \boldsymbol{\mu} \right) \\ &= (\mathbf{u} + \mathbf{b})^T \operatorname{diag}(\lambda_1, \ldots, \lambda_n) (\mathbf{u} + \mathbf{b}) \end{aligned} \tag{11.10}$$

where $\mathbf{u} = P^T \mathbf{z} = (u_1, \ldots, u_n)^T$ and $\mathbf{b} = P^T \Sigma^{-\frac{1}{2}} \boldsymbol{\mu} = (b_1, \ldots, b_n)^T$. The expression in (11.10) can be written concisely,

$$Q(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} = \sum_{i=1}^{n} \lambda_i (u_i + b_i)^2 \tag{11.11}$$

Using Lemma 8, for two ellipsoids $\mathscr{E}^n(B, \mathbf{b})$, $\mathscr{E}^n(C, \mathbf{c})$, we can compute the point $\mathbf{x}^* \in \mathscr{E}_2$ at which the ellipsoid level surface surrounding $\mathscr{E}_1$ first touch $\mathscr{E}_2$. Now, suppose that the two ellipsoids touch each other, then $\mathbf{x}^*$ satisfies the equation of $\mathscr{E}_1$. Thus we have, $(\mathbf{x}^* - \mathbf{b})^T B (\mathbf{x}^* - \mathbf{b}) = 1$. Substituting for the value of $\mathbf{x}^*$ from Lemma 8, expanding and rearranging, it follows that $\mathbf{y}^T D^T B D \mathbf{y} = 1/\lambda_0^2(M')$, where $\mathbf{y} = \mathbf{c} - \mathbf{b}$ and $D = B^{-1/2}(\lambda_0(M')I - \tilde{C})^{-1} B^{1/2}$ and M' is a $2n \times 2n$ matrix as defined in (11.6). A collision between the two ellipsoid occur when they touch or intersect each other. Thus the collision condition, that is, $\mathscr{C}_{\mathbf{b},\mathbf{c}}$ can be

written as

$$\mathbf{y}^T D^T B D \mathbf{y} = \mathbf{y}^T A \mathbf{y} \leq \frac{1}{\lambda_0^2(M')} \tag{11.12}$$

where $A = D^T B D$. In motion planning, the ellipsoid collision that we have discussed so far correspond to robot-obstacle or robot-robot collision. While planning under motion and sensing uncertainty, the robot and obstacle states can only be estimated in probabilistic terms. This renders $\mathbf{b}$, $\mathbf{c}$ as random vectors. As discussed before, in this work we model these probabilities as Gaussian distributions. As a result, the difference vector $\mathbf{c} - \mathbf{b}$ is also a Gaussian (the difference of Gaussian random variables is also a Gaussian random variable). Since B, C is positive definite, the matrix A is symmetric and therefore $\mathbf{y}^T A \mathbf{y}$ is in a quadratic form in the random variables of $\mathbf{y} = \mathbf{c} - \mathbf{b}$. Thus, the expression $\mathbf{y}^T A \mathbf{y}$ in (12.3) is in the quadratic form in the Gaussian random vector $\mathbf{y}$. Thus the collision probability can be written as

$$P(\mathbf{y}^T A \mathbf{y} \leq \frac{1}{\lambda_0^2(M')}) \tag{11.13}$$

Let $\mathbf{v} = \mathbf{y}^T A \mathbf{y}$, then

$$P(\mathbf{v} \leq 1/\lambda_0^2(M')) = F_{\mathbf{v}}(1/\lambda_0^2(M')) \tag{11.14}$$

where $F_{\mathbf{v}}$ is the cumulative distribution function (cdf) of $\mathbf{v}$. From the following theorem, an exact expression for $F_{\mathbf{v}}$ is obtained.

Lemma 9. *The cdf of* $Q(\mathbf{y}) = \mathbf{v} = \mathbf{y}^T A \mathbf{y}$ *with* $A = A^T > 0, \mathbf{y} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma), \Sigma > 0$ *is*

$$F_{\mathbf{v}}(v) = P(\mathbf{v} \leq v) = \sum_{k=0}^{\infty} (-1)^k c_k \frac{y^{\frac{n}{2}+k}}{\Gamma\left(\frac{n}{2}+k+1\right)} \tag{11.15}$$

and its pdf is given by

$$p_{\mathbf{v}}(v) = P(\mathbf{v} = v) = \sum_{k=0}^{\infty} (-1)^k c_k \frac{y^{\frac{n}{2}+k-1}}{\Gamma\left(\frac{n}{2}+k\right)} \tag{11.16}$$

where Γ denotes the gamma function and

$$c_0 = \exp(-\frac{1}{2}\sum_{i=1}^{n} b_i^2)\prod_{i=1}^{n}(2\lambda_i)^{-\frac{1}{2}}$$
$$c_k = \frac{1}{k}\sum_{i=0}^{k-1} d_{k-i}c_i$$
$$d_k = \frac{1}{2}\sum_{i=1}^{n}\left(1 - kb_i^2\right)(2\lambda_i)^{-k}$$

Proof. The proof of the above lemma can be found in Chapter 10, Section 10.2 and Appendix A. □

Thus computing the cdf as elucidated in Lemma 9 gives the exact value of collision probability. The cdf is computed as an infinite series; a proof of convergence, an expression for the truncation error and the computational complexity can be found in [104]. In our experience, the convergence is often obtained within the first few terms and hence can be used for online planning. However, during online motion planning it often suffices to compute fast approximate upper bounds for the collision probability. In the next section we derive a tight upper bound that can be computed faster than the cdf in Lemma 9.
Special case: For a Gaussian random variable $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, $\Sigma > 0$ and a matrix $A = A^T$, under certain conditions the quadratic form $\mathbf{x}^T A\mathbf{x}$ is distributed as a noncentral chi-square distribution. We will now state the theorem without proof that states the necessary and sufficient conditions. The proof may be found in [90].

Theorem 5. *Let* $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, $\Sigma > 0$. *Then the necessary and sufficient conditions for* $\mathbf{x}^T A\mathbf{x} \sim \chi_r^2(\Delta^2)$, $A = A^T$ *and* $\Delta^2 = \boldsymbol{\mu}^T A\boldsymbol{\mu}$ *are*

1. $tr(A\Sigma) = r$
2. $A\Sigma A = A$.

Whenever the collision condition in (12.3) satisfy the conditions in Theorem 5, the cdf of the collision probability in (11.13) can be computed exactly as the cdf of the corresponding noncentral chi-square distribution. This can be obtained by using a series approximation or a look-up table.

11.3 Tight Bound for Collision Probability

In this section, we derive a tight upper bound for the collision probability, that is, $P(\mathbf{y}^T A\mathbf{y} \leq 1/\lambda_0^2(M'))$.

Lemma 10. *Let* $\mathbf{v}$ *be a random variable that is never larger than* β*. Then, for all* $\alpha < \beta$

$$P(\mathbf{v} \leq \alpha) \leq \frac{\beta - \mathbb{E}(\mathbf{v})}{\beta - \alpha} \tag{11.17}$$

Proof. From Markov's inequality, for all $\alpha > 0$, we have

$$P(\mathbf{v} \geq \alpha) \leq \frac{\mathbb{E}(\mathbf{v})}{\alpha}$$

Let us now define $\tilde{\mathbf{v}} = \beta - \mathbf{v}$ such that $P(\mathbf{v} \leq \beta) = 1$. Thus, we have

$$P(\mathbf{v} \leq \alpha) = P(\beta - \tilde{\mathbf{v}} \leq \alpha) = P(\tilde{\mathbf{v}} \geq \beta - \alpha) \tag{11.18}$$

Applying Markov's inequality to $\tilde{\mathbf{v}}$, we get

$$P(\tilde{\mathbf{v}} \geq \beta - \alpha) \leq \frac{\mathbb{E}(\tilde{\mathbf{v}})}{\beta - \alpha} = \frac{\beta - \mathbb{E}(\mathbf{v})}{\beta - \alpha} \tag{11.19}$$

This completes the proof. □

Using the above lemma, an upper bound for the collision probability is obtained as

$$P(\mathbf{y}^T A\mathbf{y} \leq \frac{1}{\lambda_0^2(M')}) \leq \frac{\beta - \mathbb{E}(\mathbf{y}^T A\mathbf{y})}{\beta - \frac{1}{\lambda_0^2(M')}} \tag{11.20}$$

In the following, we will elucidate how to compute the expectation in the numerator $\mathbb{E}(\mathbf{y}^T A\mathbf{y})$ and the parameter β.

Lemma 11. *For a Gaussian random variable* $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$*, the expectation of its quadratic from is* $\mathbb{E}(\mathbf{x}^T A\mathbf{x}) = tr(A\Sigma) + \boldsymbol{\mu}^T A\boldsymbol{\mu}$.

Proof.

$$\begin{aligned}\mathbb{E}(\mathbf{x}^T A\mathbf{x}) &= tr\left(\mathbb{E}(\mathbf{x}^T A\mathbf{x})\right) = \mathbb{E}\left(tr(A\mathbf{x}\mathbf{x}^T)\right)\\ &= tr\left(A\,\mathbb{E}(\mathbf{x}\mathbf{x}^T)\right) = tr\left(A(\Sigma+\boldsymbol{\mu}\boldsymbol{\mu}^T)\right)\\ &= tr\,(A\Sigma)+tr\left(A\boldsymbol{\mu}\boldsymbol{\mu}^T\right) = tr(A\Sigma)+\boldsymbol{\mu}^T A\boldsymbol{\mu}\end{aligned}$$

To prove the above, we have used the fact that $tr\left(\mathbb{E}(\cdot)\right) = \mathbb{E}\left(tr(\cdot)\right)$, $tr(AB) = tr(BA)$ and $\Sigma = \mathbb{E}(\mathbf{x}\mathbf{x}^T) - \boldsymbol{\mu}\boldsymbol{\mu}^T$. □

Similarly, for a Gaussian random variable $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, the variance of its quadratic form is given by

$$Var(\mathbf{x}^T A\mathbf{x}) = tr\left(A\Sigma(A+A^T)\Sigma\right) + \boldsymbol{\mu}^T(A+A^T)\Sigma(A+A^T)\boldsymbol{\mu} \tag{11.21}$$

Thus using the expected value and the variance of the quadratic form, an expression for β can be obtained such that $P(\mathbf{x}^T A\mathbf{x} \leq \beta) = 1$.

We now define the following notion of an $\varepsilon-$safe configuration.

Definition 17. *A robot configuration $\mathbf{x}_k$ is an $\varepsilon-$safe configuration with respect to an obstacle configuration $\mathbf{s}_k$, if the probability of collision is such that $P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) \leq \varepsilon$.*

Let us consider the case where the ellipsoids enclose a robot currently at location $\mathbf{x}_k$ and an obstacle at location $\mathbf{s}_k$, respectively. Since the position of robots and obstacles are Gaussian distributed random variables, collision avoidance constraints can only be written in a probabilistic manner. Similar to $\mathscr{C}_{\mathbf{b},\mathbf{c}}$, the collision condition between the robot and obstacle will be written as $\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}$. We are looking for robot positions $\mathbf{x}_k$ such that the probability of collision is at most ε, that is, $P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) \leq \varepsilon$. Thus, from (11.13), we have

$$P(\mathbf{y}^T A\mathbf{y} \leq \frac{1}{\lambda_0^2(M')}) \leq \varepsilon \tag{11.22}$$

Using Lemma 10, the bounded collision (11.22) constraint is obtained as

$$\beta - \mathbb{E}(\mathbf{y}^T A\mathbf{y}) \leq \varepsilon\left(\beta - \frac{1}{\lambda_0^2(M')}\right) \tag{11.23}$$

11.4 Comparison to Other Methods

We provide a comparison with several state-of-the-art methods using a robot and a close-by obstacle. The robot is located at $(0.95, 0.95, 0)$ m with semi-principle axes $(0.18, 0.18, 0.22)$ m and covariance $diag(0.41, 0.41, 0.21)$ m^2. The obstacle is located at the origin with semi-principle axes $(0.6, 0.6, 1.2)$ m. The collision probability values can be seen in Table 12.1. We define $\varepsilon = 0.09$ and this configuration is $0.09-$safe or is feasible. The value computed using the upper bound (11.20) is very close to the actual value computed using the cdf of the quadratic form (11.15). We note here that we use the distance between two ellipsoids to compute the collision probability. Most approaches approximate an integral of the joint distribution between the robot and the obstacle to compute the collision probability. Given the current robot state $\mathbf{x}_k$ and the obstacle state $\mathbf{s}_k$, the collision probability is given by

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = \int_{\mathbf{x}_k}\int_{\mathbf{s}_k} I_c(\mathbf{x}_k,\mathbf{s}_k)p(\mathbf{x}_k,\mathbf{s}_k) \tag{11.24}$$

where I_c is an indicator function defined as

$$I_c(\mathbf{x}_k,\mathbf{s}_k) = \begin{cases} 1 & \text{if } \mathscr{R}\cap\mathscr{S} \neq \{\phi\} \\ 0 & \text{otherwise.} \end{cases} \tag{11.25}$$

and $p(\mathbf{x}_k,\mathbf{s}_k)$ is the joint distribution of the robot and the obstacle. The numeric integral of (12.17) gives the exact value and we compute the same to validate our approach. As seen from Table 12.1 the difference in values is very negligible. The double summation of numerical integration is approximated to a single summation in [65] and gives a feasible result. The approach in [104] also gives a feasible configurations. Other approaches compute loose upper bounds and hence determine the configuration infeasible. Our approach thus computes a tight upper bound.

11.5 Belief Dynamics

We consider a Gaussian parametrization for the probability distribution over the robot (or object) state which is known as the *belief* state. Though BSP has been researched extensively in the past [89, 109, 52, 1, 62, 83, 102, 29], below we provide a brief overview of the same.

We will now state the Gaussian belief state dynamics. The motion (8.1) and observation (8.2) models can be written probabilistically as $p(\mathbf{x}_{k+1}|\mathbf{x}_k,\mathbf{u}_k)$ and $p(\mathbf{z}_k|\mathbf{x}_k)$, respectively. Given an initial distribution $p(\mathbf{x}_0)$ (Gaussian), and the Gaussian motion and observation

Methods	Collision probability	Computation time (s)	Feasible ?
Numerical integral	0.0568	4.3619 ± 1.1784	Yes
Approximate Numerical integral [65]	0.0773	1.7932 ± 0.1927	Yes
Bounding volume [81, 54]	1	0.0003 ± 0.0016	No
Center point approximation [19]	0.1027	0.0004 ± 0.0001	No
Maximum probability approximation [82]	0.7168	0.2288 ± 0.1626	No
Chance constraint [113]	0.1894	0.0013 ± 0.0000	No
Rectangular bounding box [33]	0.1582	0.0056 ± 0.0006	No
Sphere approximation [104]	0.0898	0.0198 ± 0.0309	Yes
Our approach– exact	0.0572	0.0044 ± 0.0042	Yes
Our approach– upper bound	0.0660	0.0009 ± 0.0003	Yes

Table 11.1 Comparison of collision probability methods

models, the posterior probability distribution at time k is the belief $\mathbf{b}(\mathbf{x}_k)$ and is be written as

$$\mathbf{b}(\mathbf{x}_k) = p(\mathbf{x}_k|\mathbf{z}_{0:k}, \mathbf{u}_{0:k-1}) \tag{11.26}$$

where $\mathbf{z}_{0:k} \doteq \{\mathbf{z}_0, ..., \mathbf{z}_k\}$ and $\mathbf{u}_{0:k-1} \doteq \{\mathbf{u}_0, ..., \mathbf{u}_{k-1}\}$. The posterior distribution can be expanded using Bayes rule and theorem of total probability giving

$$p(\mathbf{x}_k|\mathbf{z}_k, \mathbf{z}_{0:k-1}, \mathbf{u}_{0:k-1}) = \eta_k p(\mathbf{z}_k|\mathbf{x}_k) \int_{\mathbf{x}_{k-1}} p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) \mathbf{b}(\mathbf{x}_{k-1}) \tag{11.27}$$

where $\eta_k = 1/p(\mathbf{z}_k|\mathbf{z}_{0:k-1}, \mathbf{u}_{0:k-1})$ is the normalization constant and $\mathbf{b}(\mathbf{x}_{k-1})$ is the belief at time $k-1$. Since the belief $\mathbf{b}(\mathbf{x}_k) \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$ is a Gaussian with mean $\boldsymbol{\mu}_k$ and covariance Σ_k we will denote the belief state by a vector

$$\mathbf{b}(\mathbf{x}_k) = \left[\boldsymbol{\mu}_k^T, \mathbf{s}_k^T\right]^T \tag{11.28}$$

where $\mathbf{s}_k^T = [s_{k_1}^T, \ldots, s_{k_n}^T]$ is vector composed of the n columns of Σ_k. Equivalently, $\mathbf{s}^T$ will also be denoted by $vec(\Sigma_k)$ To implement the Bayes filter in (11.27) we use the technique of EKF. The belief state parameters are then given by the EKF update equations

$$\boldsymbol{\mu}_k = \bar{\boldsymbol{\mu}}_k + K_k \left(\mathbf{z}_k - h(\bar{\boldsymbol{\mu}}_k)\right) \tag{11.29}$$

$$\Sigma_k = (I - K_k H_k) \bar{\Sigma}_k \tag{11.30}$$

where

$$\begin{aligned}\bar{\boldsymbol{\mu}}_k &= f(\boldsymbol{\mu}_{k-1}, \mathbf{u}_{k-1}) \\ \bar{\Sigma}_k &= F_{k-1}\Sigma_{k-1}F_{k-1}^T + R_{k-1} \\ K_k &= \bar{\Sigma}_k H_k^T \left(H_k \bar{\Sigma}_k H_k^T + Q_k\right)^{-1}\end{aligned} \tag{11.31}$$

with H_k being the Jacobian of $h(\cdot)$ with respect to $\mathbf{x}_k$ and F_{k-1} is the Jacobian of $f(\cdot)$ with respect to $\mathbf{x}_{k-1}$. The second term in (11.29) depends on the measurement $\mathbf{z}_k$ and is often referred to as the *innovations process*. Since future observations are unknown at the planning time, the innovations process is stochastic. As a result the belief state dynamics is stochastic in nature.

Theorem 6. *The innovations process is a zero-mean Gaussian white noise sequence with*

$$\mathbb{E}\left(\left[\mathbf{z}_k - h(\bar{\boldsymbol{\mu}}_k)\right]\left[\mathbf{z}_k - h(\bar{\boldsymbol{\mu}}_k)\right]^T\right) = H_k\bar{\Sigma}_k H_k^T + Q_k \tag{11.32}$$

Proof. The proof may be found in [76], page 234. □

From Theorem 6, the stochastic belief state dynamics can be written as

$$\mathbf{b}(\mathbf{x}_k) = g\left(\mathbf{b}(\mathbf{x}_{k-1}), \mathbf{u}_{k-1}\right) + W\left(\mathbf{b}(\mathbf{x}_{k-1}), \mathbf{u}_{k-1}\right) w_{k-1} \tag{11.33}$$

where

$$g\left(\mathbf{b}(\mathbf{x}_{k-1}), \mathbf{u}_{k-1}\right) = \begin{bmatrix}\bar{\boldsymbol{\mu}} \\ \mathbf{s}_k^T\end{bmatrix} = \begin{bmatrix} f(\boldsymbol{\mu}_{k-1}, \mathbf{u}_{k-1}) \\ vec\left((I - K_k H_k)\bar{\Sigma}_k\right)\end{bmatrix} \tag{11.34}$$

$$W\left(\mathbf{b}(\mathbf{x}_{k-1}), \mathbf{u}_{k-1}\right) = \begin{bmatrix} K_k \\ \mathbf{0}\end{bmatrix} \tag{11.35}$$

$$w_{k-1} \sim \mathcal{N}(\mathbf{0}, H_k\bar{\Sigma}_k H_k^T + Q_k) \tag{11.36}$$

Thus the innovation term $K_k(\mathbf{z}_k - h(\bar{\boldsymbol{\mu}}_k))$ is distributed according to

$$\mathcal{N}(\mathbf{0}, K_k(H_k\bar{\Sigma}_k H_k^T + Q_k)K_k^T) \tag{11.37}$$

The assumption of maximum likelihood observation was first relaxed by Van Den Berg *et al.* [109]. They used a similar approach as discussed above to arrive at the stochastic belief dynamics. However, in [109] a first-order approximation is used rendering the innovation

term to be distributed according to $\mathcal{N}(\mathbf{0}, K_k H_k \bar{\Sigma}_k)$. It is also noteworthy that most approaches prior to and after [109] assumed maximum likelihood observations [89, 88, 67, 84]. The maximum likelihood observation at time k is $\mathbf{z}_k = h(\bar{\boldsymbol{\mu}}_k)$, which reduces the innovation term in (11.29) to zero. This assumption thus eliminates the stochasticity from the belief state dynamics. Other approaches that relax the maximum likelihood assumption treat either simulate future measurements or treat them as random variables [43, 108, 83].

11.6 Objective Function

We formulate the collision avoidance in belief space planning as an optimization problem. At each time instant k, the robot plans for L look-ahead steps and minimizes an objective function J_k, subject to certain constraints. We consider the objective function

$$J_k = \mathop{\mathbb{E}}_{\mathbf{z}_{k+1:k+L}} \left[\sum_{l=0}^{L-1} c_l(\mathbf{b}(\mathbf{x}_{k+l}), \mathbf{u}_{k+l}) + c_L(\mathbf{b}(\mathbf{x}_{k+L})) \right] \tag{11.38}$$

where c_l is the cost for each look-ahead step and c_L is the terminal cost. Since future observations are not available at planning time and are stochastic, the expectation is taken to account for all possible future observations. The optimization problem can then be formally stated as

$$\begin{aligned} &\min_{\mathbf{b}_{k:k+L-1}, \mathbf{u}_{k:k+L-1}} \quad J_k \\ &s.t. \quad \mathbf{b}(\mathbf{x}_{k+l}) = g\left(\mathbf{b}(\mathbf{x}_{k+l-1}), \mathbf{u}_{k+l-1}\right) + W\left(\mathbf{b}(\mathbf{x}_{k+l-1}), \mathbf{u}_{k+l-1}\right) w_{k+l-1} \\ &\qquad \mathbf{u}_{k+l} \in \mathbf{U} \\ &\qquad P\left(\mathscr{C}_{\mathbf{x}_k, \mathbf{s}_k}\right) \leq \varepsilon \end{aligned} \tag{11.39}$$

where $\mathbf{b}_{k:k+L-1} = \{\mathbf{b}(\mathbf{x}_k), \mathbf{b}(\mathbf{x}_{k+1}), \ldots, \mathbf{b}(\mathbf{x}_{k+L-1})\}$, $\mathbf{u}_{k+l} \in \mathbf{U}$ constraints the control input to lie within the feasible set of control inputs $\mathbf{U}$ and $P\left(\mathscr{C}_{\mathbf{x}_k, \mathbf{s}_k}\right) \leq \varepsilon$ enforces $\varepsilon-$safe configurations, bounding the collision probability. The expectation is taken to account for the fact that the observation is unknown at the planning time.

At each time step, the robot is required to minimize its control usage and proceed towards the goal $\mathbf{x}^g$ avoiding collisions. As a result, we have the following immediate and terminal costs

$$c_l(\mathbf{b}(\mathbf{x}_{k+l}), \mathbf{u}_{k+l}) = \left\| \xi(\mathbf{u}_{k+l}) \right\|^2_{M_u} \tag{11.40}$$

$$c_L(\mathbf{b}(\mathbf{x}_{k+L})) = \left\| \mathbf{x}_{k+L} - \mathbf{x}^g \right\|^2_{M_g} \tag{11.41}$$

where $\|x\|_S = \sqrt{x^T S x}$ is the Mahalanobis norm, M_u, M_g are weight matrices and $\xi(\cdot)$ is a function that quantifies control usage. The choice of weight matrices and the control function vary with application. The objective function in (11.38) can now be explicitly written as

$$\begin{aligned} J_k &= \mathop{\mathbb{E}}_{\mathbf{z}_{k+1:k+L}} \left[\sum_{l=0}^{L-1} \left\| \xi(\mathbf{u}_{k+l}) \right\|_{M_u}^2 + \|\mathbf{x}_{k+L} - \mathbf{x}^g\|_{M_g}^2 \right] \\ &= \sum_{l=0}^{L-1} \left\| \xi(\mathbf{u}_{k+l}) \right\|_{M_u}^2 + \mathop{\mathbb{E}}_{\mathbf{z}_{k+L}} \left[\|\mathbf{x}_{k+L} - \mathbf{x}^g\|_{M_g}^2 \right] \end{aligned} \tag{11.42}$$

The expectation is discarded from the first term as it does not depend on the future observations. Let us now proceed by evaluating the term with expectation. Using (11.29), we have

$$\begin{aligned} \|\mathbf{x}_{k+L} - \mathbf{x}^g\|_{M_g}^2 &= \left\| \bar{\boldsymbol{\mu}}_{k+L} + K_{k+L}\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right) - \mathbf{x}^g \right\|_{M_g}^2 \\ &= \left\| \bar{\boldsymbol{\mu}}_{k+L} - \mathbf{x}^g + K_{k+L}\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right) \right\|_{M_g}^2 \\ &= \left\| \bar{\boldsymbol{\mu}}_{k+L} - \mathbf{x}^g \right\|_{M_g}^2 + \left\| K_{k+L}\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right) \right\|_{M_g}^2 + \\ &\left(\bar{\boldsymbol{\mu}}_{k+L} - \mathbf{x}^g\right)^T M_g \left(K_{k+L}\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right) \right) + \left(K_{k+L}\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right) \right)^T M_g \left(\bar{\boldsymbol{\mu}}_{k+L} - \mathbf{x}^g\right) \end{aligned} \tag{11.43}$$

Computing the expectation of the expression in (11.43) and using $\mathbb{E}\left[\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right)\right] = \mathbf{0}$, we have

$$\mathop{\mathbb{E}}_{\mathbf{z}_{k+L}} \left[\|\mathbf{x}_{k+L} - \mathbf{x}^g\|_{M_g}^2 \right] = \left\| \bar{\boldsymbol{\mu}}_{k+L} - \mathbf{x}^g \right\|_{M_g}^2 + \mathop{\mathbb{E}}_{\mathbf{z}_{k+L}} \left[\left\| K_{k+L}\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right) \right\|_{M_g}^2 \right] \tag{11.44}$$

For any random vector $\mathbf{y}$ and a matrix A of appropriate dimension, we have

$$\mathbb{E}\left[\mathbf{y}^T A \mathbf{y}\right] = tr\left(A \mathrm{Var}(\mathbf{y})\right) + \mathbb{E}[\mathbf{y}]^T A\, \mathbb{E}[\mathbf{y}] \tag{11.45}$$

where $\mathrm{Var}(\cdot)$ denotes the variance. Using (11.45) and using $\mathbb{E}\left[\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right)\right] = \mathbf{0}$, the expression in (11.44) simplifies to

$$\begin{aligned} \mathop{\mathbb{E}}_{\mathbf{z}_{k+L}} \left[\|\mathbf{x}_{k+L} - \mathbf{x}^g\|_{M_g}^2 \right] &= \left\| \bar{\boldsymbol{\mu}}_{k+L} - \mathbf{x}^g \right\|_{M_g}^2 + tr\left(K_{k+L} \mathrm{Var}\left(\mathbf{z}_{k+L} - h(\bar{\boldsymbol{\mu}}_k)\right) \right) \\ &= \left\| \bar{\boldsymbol{\mu}}_{k+L} - \mathbf{x}^g \right\|_{M_g}^2 + tr\left(K_{k+L}\left(H_{k+L} \bar{\Sigma}_{k+L} H_{k+L}^T + Q_{k+L} \right) \right) \end{aligned} \tag{11.46}$$

where the expression for the variance is obtained from (11.32). We will now derive the expression for the third constraint in (12.15), that is $P\left(\mathscr{C}_{\mathbf{x}_{k+l},\mathbf{s}_{k+l}}\right) \leq \varepsilon$. Let the Löwner-John ellipsoids of the robot and the obstacle be $\mathscr{E}(X,\mathbf{x}_{k+l})$ and $\mathscr{E}(S,\mathbf{s}_{k+l})$, respectively. From (11.23), we have

$$P\left(\mathscr{C}_{\mathbf{x}_{k+l},\mathbf{s}_{k+l}}\right) \leq \varepsilon \equiv$$
$$\beta - \mathbb{E}\left[(\mathbf{s}_{k+l}-\mathbf{x}_{k+l})^T A_{k+l}(\mathbf{s}_{k+l}-\mathbf{x}_{k+l})\right] \leq \varepsilon\left(\beta - \frac{1}{\lambda_0^2(M')}\right) \quad (11.47)$$

From (11.6), we recall that

$$M'_{k+l} = \begin{bmatrix} \tilde{S} & -I \\ -\tilde{\mathbf{s}}_{k+l}\tilde{\mathbf{s}}_{k+l}^T & \tilde{S} \end{bmatrix} \quad (11.48)$$

with $\tilde{S} = \bar{S}^{-1}$ and $\tilde{\mathbf{s}}_{k+l} = \bar{S}^{-1/2}\bar{\mathbf{s}}_{k+l}$, where $\bar{S} = X^{-1/2}SX^{-1/2}$ and $\bar{\mathbf{s}}_{k+l} = X^{1/2}(\boldsymbol{\mu}_{\mathbf{s}_{k+l}} - \boldsymbol{\mu}_{\mathbf{x}_{k+l}})$. Similarly, from (12.3), we have $A_{k+l} = D_{k+l}^T X D_{k+l}$, with $D_{k+l} = X^{-1/2}(\lambda_0(M'_{k+l})I - \tilde{S})^{-1}X^{1/2}$ and $\lambda_0(M'_{k+l})$ being the minimal eigenvalue of M'_{k+l}. The expectation in (11.47) is then evaluated following (11.45). Thus we obtain

$$\mathbb{E}\left[(\mathbf{s}_{k+l}-\mathbf{x}_{k+l})^T A_{k+l}(\mathbf{s}_{k+l}-\mathbf{x}_{k+l})\right] =$$
$$tr\left(A_{k+l}\left(\Sigma_{\mathbf{s}_{k+l}} + \Sigma_{k+l}\right)\right) + \left(\boldsymbol{\mu}_{\mathbf{s}_{k+l}} - \boldsymbol{\mu}_{\mathbf{x}_{k+l}}\right)^T A_{k+l}\left(\boldsymbol{\mu}_{\mathbf{s}_{k+l}} - \boldsymbol{\mu}_{\mathbf{x}_{k+l}}\right) \quad (11.49)$$

where we have used the fact that $\mathbf{s}_{k+l} - \mathbf{x}_{k+l}$ is Gaussian distributed as $\mathbf{s}_{k+l} - \mathbf{x}_{k+l} \sim \mathscr{N}(\boldsymbol{\mu}_{\mathbf{s}_{k+l}} - \boldsymbol{\mu}_{\mathbf{x}_{k+l}}, \Sigma_{\mathbf{s}_{k+l}} + \Sigma_{k+l})$.

11.7 Results

In this section we describe our implementation and then evaluate the capabilities of our proposed approach. Simulations are performed in the Gazebo environment with a quadrotor of semi-principle axes $(0.18, 0.18, 0.06)$ m. We refer the readers to [22] for the quadrotor dynamics. The ground truth odometry from Gazebo is used to measure the pose of the robot, mimicking a motion capture system. This measurement is then corrupted with noise which is zero mean with covariance $\Sigma = diag(0.05\text{m}^2, 0.05\text{m}^2, 0.05\text{m}^2, 0.1\text{deg}^2, 0.1\text{deg}^2, 0.1\text{deg}^2)$. The optimization of (12.15) is performed using the MPC based approach developed in [22], which is based on ACADO [39] and qpOASES [23]. In all the experiments we use a collision probability bound of $\varepsilon = 0.05$. A look-ahead horizon of two seconds is used with

	Measurement noise															
	Σ				2Σ				3Σ				4Σ			
Method	d	l	T	sp	d	l	T	sp	d	l	T	sp	d	l	T	sp
Bounding volume [81, 54]	0.70	7.72	6.6	100	0.93	8.15	6.19	100	1.38	8.78	6.54	100	1.51	9.54	7.04	100
Our method	0.40	7.48	6.01	100	0.45	8.05	6.13	100	0.76	8.21	6.19	100	0.79	8.37	6.43	100
Center point approximation [19]	-	-	-	0	-	-	-	0	0.10	7.02	6.01	55	0.13	7.10	6.08	35

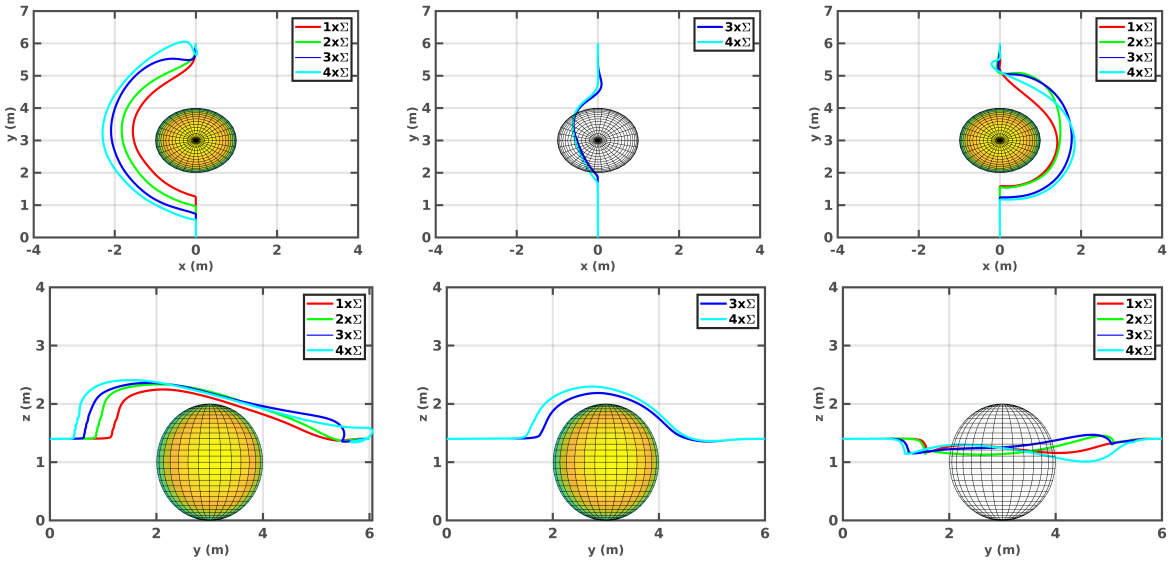
Table 11.2 Collision probability efficiency with varying measurement noise. The minimum distance between the quadrotor and the obstacle is denoted by d (m). l (m) is the total trajectory length and T (s) is the total trajectory duration. sp denotes success percentage.

a discretization of 0.1 seconds. The performance is evaluated on an Intel® Core i7-6500U CPU@2.50GHz×4 with 8GB RAM under Ubuntu 16.04 LTS.

Comparison to bounding volume approaches: We compare our approach to bounding volume methods [81, 54] wherein robots and obstacles are enlarged with their 3-σ confidence ellipsoids. Computation time and complexity are greatly reduced with bounding volumes. However, plans tend to be overly conservative and suboptimal. In the experiment, we consider a quadrotor at $(0, 0, 1.4)$ m moving to the goal location at $(0, 13, 1.4)$ m. An obstacle of semi-principle axes $(1, 1, 1)$ m is located in between at $(0, 3, 3)$ m. We conduct the experiment with varying measurement noise of $\Sigma, 2\Sigma, 3\Sigma,$ and 4Σ. To compare the efficiency of our approach we define the compute the following metrics: (a) d– minimum distance between the quadrotor and the obstacle, (b) l– total trajectory length, and (c) T– total trajectory duration.

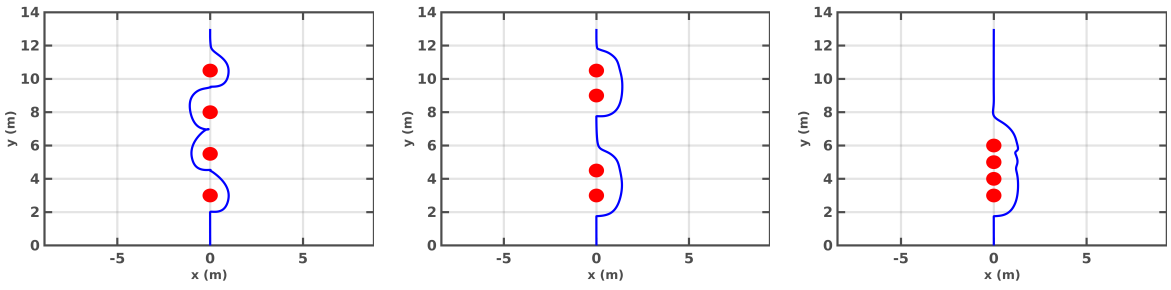
The results can be seen in the first two rows of Table 11.2. In all the cases, our approach is more efficient as can be seen from the shorter average trajectory length and duration. As the measurement noise increases this becomes more evident. We also provide a comparison to center point approximation [19] discussed in Section 8.2 and Table 11.1, which is also computationally less intense. However, as recognized by [82], if the covariance is small, the approximated probability can be much smaller than the exact probability. Moreover, the approach work well only when the sizes of objects are relatively very small compared with their position uncertainties [113]. This is seen in the last row of Table 11.2. For measurement noise Σ and 2Σ the approach resulted in collision for all the runs. For measurement noise 3Σ and 4Σ, the approach succeeded in 55% and 35% of the runs, respectively. This reduced success percentages are due to lower values of the collision probabilities computed. The executed trajectories for all the three approaches in Table 11.2 are shown in Fig. 11.1.

Four obstacles: In this experiment, we consider four obstacles which are placed (a) far apart from each, (b) two obstacles are placed close to each other, and (c) all obstacles are placed close to each other. In each of the cases, the quadrotor starting from $(0, 0, 1.4)$ m has to reach the goal location at $(0, 13, 1.4)$ m. The quadrotor successfully reaches the goal



(a) Bounding volume **(b)** Center point approximation **(c)** Our method

Figure 11.1 Top view (x-y) of four obstacles in different locations. The solid blue lines represent the trajectories executed by the quadrotor and the red blobs represent the obstacles.



(a) **(b)** **(c)**

Figure 11.2 Simulation with varying measurement noise. The upper plots shows the top view (x-y) and the lower plots show the side view (y-z). The solid lines represent the trajectories executed by the quadrotor.

Obstacle location	l (m)	T (s)	d (m)
(a)	17.78	14.94 ± 0.03	0.59
(b)	16.10	14.54 ± 0.01	0.88
(c)	14.57	14.36 ± 0.02	0.82

Table 11.3 Trajectory results with varying obstacle configurations.

location avoiding collisions. The respective trajectories in each of the three cases can be seen in Fig. 11.2. The change in configuration of the obstacles affects the collision probability computation which is reflected in the respective executed trajectories. The results are shown in Table 11.3.

Column domain: In this experiment, we test the adaptability of our approach to cluttered and challenging environments. Fig. 11.3 shows a snapshot of the column domain which consists of 19 cylindrical columns that obstruct the quadrotor path. We test the We consider two different scenarios where the quadrotor need to avoid collision with the columns to reach the goal location. In scenario 1, the quadrotor starting from $(-2.5, 0, 1.4)$ m has to reach the goal location at $(-2.5, 12, 1.4)$ m. The trajectory followed is seen in Fig. 11.4a, with an average trajectory length of 13.72 m, and trajectory time of 13.14 ($\pm$0.02) seconds. The average minimum distance between the quadrotor and the obstacles is 0.21 m. In scenario 2, the quadrotor starting from $(0, 0, 1.4)$ m has to reach the goal location at $(1, 12, 1.4)$ m. The trajectory followed is seen in Fig. 11.4b, with an average trajectory length of 13.42 m, and trajectory time of 13.37 ($\pm$0.05) seconds. The average minimum distance between the quadrotor and the obstacles is 0.66 m. We note here that there since there are 19 columns, there are 19 constraints of the form $P(\mathscr{C}_{\mathbf{x}_k, \mathbf{s}_k}) \leq \varepsilon$. Our method is computationally less intense and hence solvable in real time.

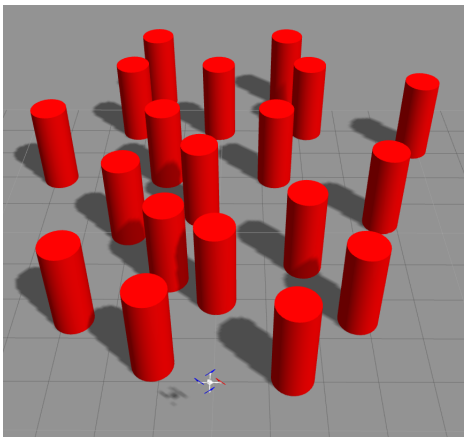
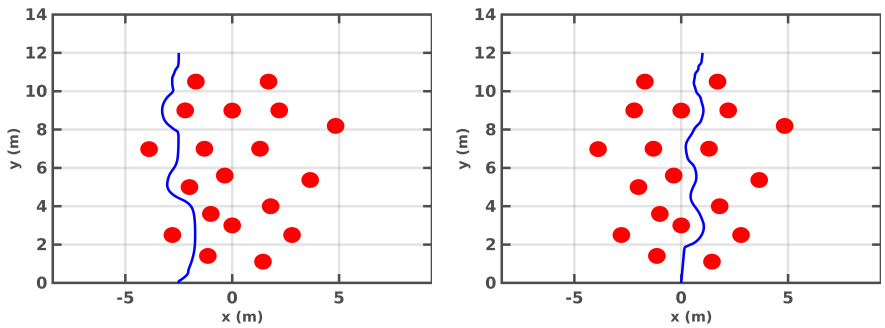


Figure 11.3 Column domain in Gazebo consisting of 19 cylindrical columns.



(a) Scenario 1 **(b)** Scenario 2

Figure 11.4 Top view (x-y) of the column world scenarios. The solid blue lines represent the trajectories executed by the quadrotor and the red blobs represent the obstacles. Quadrotor successfully evades collision in both the scenarios.

Chapter 12

Fast and Bounded Collision Constraint

In this chapter, we discuss our contribution **C10** by developing an accurate constraint for collision avoidance during motion planning which is much faster when compared to the approaches discussed in Chapter 10 and Chapter 11. For a specific collision probability threshold, the collision avoidance constraint can then be used for online MPC optimization. To be robust to uncertain environments, robot motion and sensing uncertainties (and obstacle uncertainties) are incorporated by propagating the uncertainties within the MPC framework.

12.1 Collision Constraint for Motion Planning

We denote by $\mathscr{R}$ the set of all points occupied by a rigid-body robot at any given time. Similarly, let $\mathscr{S}$ represent the set of all points occupied by a rigid-body obstacle. A collision occurs if there exits a point such that it is in both $\mathscr{R}$ and $\mathscr{S}$. Thus the collision condition is defined as

$$\mathscr{R} \cap \mathscr{S} \neq \{\phi\} \tag{12.1}$$

and we denote the probability of collision as $P\left(\mathscr{R} \cap \mathscr{S} \neq \{\phi\}\right)$. In this work we assume spherical geometries for $\mathscr{R}$ and $\mathscr{S}$ with radii r_1 and s_1, respectively. We assign body-fixed reference frames to robot and obstacle centers located at $\mathbf{x}_k$ and $\mathbf{s}_k$, respectively in the global frame. By abuse of notation we will use $\mathbf{x}_k$ and $\mathbf{s}_k$ equivalently to $\mathscr{R}$ and $\mathscr{S}$. However, when we talk about the distribution of their locations, we refer to the distribution of their centers (the body-fixed frame). The collision condition is thus defined in terms of the body-fixed frames as

$$\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k} : \mathscr{R} \cap \mathscr{S} \neq \{\phi\} \tag{12.2}$$

We recall here that the locations of the robot and the obstacles are in general uncertain. Let us now consider a robot and an obstacle at any given time instant k, distributed according to the Gaussians $\mathbf{x}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{x}_k}, \Sigma_{\mathbf{x}_k}\right)$ and $\mathbf{s}_k \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{s}_k}\right)$, respectively. Since the robot and the obstacles are assumed to be spherical objects, the collision constraint is written as

$$\|\mathbf{x}_k - \mathbf{s}_k\|^2 \leq (r_1 + s_1)^2 \tag{12.3}$$

Thus 12.3 is equivalent to $\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}$. Let us denote the difference between the two random variables by $\mathbf{w} = \mathbf{x}_k - \mathbf{s}_k$. Using the expression for the difference between two Gaussian distributions, we have $\mathbf{w} \sim \mathscr{N}\left(\boldsymbol{\mu}_{\mathbf{x}_k} - \boldsymbol{\mu}_{\mathbf{s}_k}, \Sigma_{\mathbf{x}_k} + \Sigma_{\mathbf{s}_k}\right)$. The collision constraint in (12.3) can now be written in terms of $\mathbf{w}$,

$$\mathbf{y} = \|\mathbf{w}\|^2 = \mathbf{w}^T\mathbf{w} \leq (r_1 + s_1)^2 \tag{12.4}$$

where $\mathbf{y}$ is a random vector distributed according to the squared L_2-norm of $\mathbf{w}$.

Proposition 1. *A symmetric matrix $A \in \mathbb{R}^{n\times n}$ with orthonormal eigenvectors q_i can be factorized as*

$$A = Q\Lambda Q^T \tag{12.5}$$

where the columns of Q correspond to the orthonormal eigenvectors q_i and Λ is a diagonal matrix comprised of the corresponding eigenvalues of A.

Lemma 12. *For a symmetric matrix $A \in \mathbb{R}^{n\times n}$ and a random vector $\mathbf{x}$, we have*

$$\mathbf{x}^T A\mathbf{x} \leq \lambda_{max}\|\mathbf{x}\|^2 \tag{12.6}$$

where λ_{max} is the maximum eigenvalue of A.

Proof. From Proposition 1, we have

$$\begin{aligned}\mathbf{x}^T A\mathbf{x} = \mathbf{x}^T Q\Lambda Q^T\mathbf{x} = \left(Q^T\mathbf{x}\right)^T \Lambda \left(Q^T\mathbf{x}\right) = \sum_{i=1}^{n} \lambda_i (q_i^T\mathbf{x})^2 \\ \leq \lambda_{max}\sum_{i=1}^{n}(q_i^T\mathbf{x})^2 = \lambda_{max}\|\mathbf{x}\|^2\end{aligned}$$

where we have used the fact that the eigenvectors are orthonormal. □

Proposition 2. *For any random variable* $\mathbf{x}$*, the probability of an event* $P(\mathbf{x} \leq x)$ *is given by its cumulative distribution function (cdf)* $F_{\mathbf{x}}(x)$*, that is,*

$$F_{\mathbf{x}}(x) = P(\mathbf{x} \leq x), \quad -\infty < x < +\infty \tag{12.7}$$

Proposition 3. *For n-dimensional* $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$

$$y = (\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \sim \chi^2_n \tag{12.8}$$

where χ^2_n *denotes the chi-squared distribution with n degrees of freedom.*

Let F_{chi} be the cdf of a chi-squared distribution with n degrees of freedom and let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, then for any $-\infty < x < +\infty$ we have

$$P((\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \leq x) = F_{chi}(x) \tag{12.9}$$

Alternatively, for any $0 \leq \varepsilon \leq 1$, we have

$$P((\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \leq F^{-1}_{chi}(\varepsilon)) = \varepsilon \tag{12.10}$$

Lemma 13. *Let* $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$*,* $\|\mathbf{x}\|^2 \leq \alpha$*,* F_{chi} *be the cdf of a chi-squared distribution and*

$$P((\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \leq F^{-1}_{chi}(\varepsilon)) = \varepsilon$$

Then $\lambda_{max}\left(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu}\right) \leq F^{-1}_{chi}(\varepsilon)$*, where* λ_{max} *is the maximum eigenvalue of* Σ^{-1}*.*

Proof. From Lemma 12, it follows that

$$(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \leq \lambda_{max} \|\mathbf{x} - \boldsymbol{\mu}\|^2 \tag{12.11}$$

where λ_{max} is the maximum eigenvalue of Σ^{-1}. Expanding the right-hand side of (12.11) and using the fact that $\|\mathbf{x}\|^2 \leq \alpha$, we get

$$(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \leq \lambda_{max}\left(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu}\right) \tag{12.12}$$

Thus, for $(\mathbf{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}) \leq F^{-1}_{chi}(\varepsilon)$ it suffices that $\lambda_{max}\left(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu}\right) \leq F^{-1}_{chi}(\varepsilon)$. □

Note that the collision constraint in (12.4) is of the form $\|\mathbf{x}\|^2 \leq \alpha$ and this allows us to define a notion of maximum allowable collision probability. We remind the readers that $\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}$ represents the collision condition and is therefore equivalent to (12.3). We now define the collision constraint that satisfy the required collision probability threshold.

Lemma 14. *Given n-dimensional* $\mathbf{x} \sim \mathscr{N}(\boldsymbol{\mu},\Sigma)$*, and* $P(\|\mathbf{x}\|^2 \leq \alpha) \leq \varepsilon$*, then*

$$\lambda_{max}\left(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu}\right) \leq F_{chi}^{-1}(\varepsilon) \tag{12.13}$$

where λ_{max} *is the maximum eigenvalue of* Σ^{-1} *and* F_{chi} *is the cdf of the chi-squared distribution with n degrees of freedom.*

Proof. We have

$$\begin{aligned}
P(\|\mathbf{x}\|^2 \leq \alpha) &= P(\mathbf{x}^T\mathbf{x} \leq \alpha) = P((\mathbf{x}-\boldsymbol{\mu}+\boldsymbol{\mu})^T(\mathbf{x}-\boldsymbol{\mu}+\boldsymbol{\mu}) \leq \alpha) \\
&= P((\mathbf{x}-\boldsymbol{\mu})^T(\mathbf{x}-\boldsymbol{\mu}) + 2(\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu} \leq \alpha) \\
&= P(\|\mathbf{x}-\boldsymbol{\mu}\|^2 \leq \alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu}) \\
&= P(\lambda_{max}\|\mathbf{x}-\boldsymbol{\mu}\|^2 \leq \lambda_{max}(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu}))
\end{aligned}$$

where λ_{max} is the maximum eigenvalue of Σ^{-1}. Now from Lemma 12 it follows that

$$\begin{aligned}
&P(\lambda_{max}\|\mathbf{x}-\boldsymbol{\mu}\|^2 \leq \lambda_{max}(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu})) \\
&\quad = P((\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) \leq \lambda_{max}(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu})) = P(\|\mathbf{x}\|^2 \leq \alpha) = \varepsilon
\end{aligned}$$

The required result then directly follows from Lemma 13. □

Since the collision constraint in (12.4) is a Gaussian distribution, for a collision probability threshold of ε we can directly use the constraint in (12.13). We now state the following lemma which is a direct consequence of Lemma 14.

Lemma 15. *Given n-dimensional* $\mathbf{x} \sim \mathscr{N}(\boldsymbol{\mu},\Sigma)$*, and* $P(\|\mathbf{x}\|^2 \leq \alpha) \leq \varepsilon$*, then*

$$P(\|\mathbf{x}\|^2 \leq \alpha) \leq F_{chi}^{-1}\left(\lambda_{max}(\alpha - 2\mathbf{x}^T\boldsymbol{\mu} + \boldsymbol{\mu}^T\boldsymbol{\mu})\right) \tag{12.14}$$

12.2 Objective Function

We formulate the collision avoidance problem as an optimization problem. At each time instant k, the robot plans for L look-ahead steps and minimizes an objective function J_k,

subject to collision and other constraints. The optimization problem can then be formally stated as

$$\begin{aligned} \min_{\mathbf{x}_{k:k+L},\mathbf{u}_{k:k+L-1}} \quad & J_k \\ s.t. \quad & \mathbf{x}_{k+1} = f(\mathbf{x}_k,\mathbf{u}_k) \\ & \mathbf{u}_{k+l} \in \mathbf{U} \\ & P\left(\mathscr{C}_{\mathbf{x}_{k+l},\mathbf{s}_k^i}\right) \leq \varepsilon \end{aligned} \tag{12.15}$$

where

$$J_k = \sum_{l=0}^{L-1} c_l(\mathbf{x}_{k+l},\mathbf{u}_{k+l}) + c_L(\mathbf{x}_{k+L}) \tag{12.16}$$

with c_l denoting the cost term at time $k+l$ and c_L denoting the terminal cost, $\mathbf{x}_{k+1} = f(\mathbf{x}_k,\mathbf{u}_k)$ is the robot dynamics (8.1), $\mathbf{u}_{k+l} \in \mathbf{U}$ constraints the control inputs to lie within the feasible set $\mathbf{U}$ and $P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k^i}\right) \leq \varepsilon$ enforces a collision probability threshold of ε with obstacles $\mathbf{s}_k^i$.

We recall here that to determine the constraint $P\left(\mathscr{C}_{\mathbf{x}_{k+l},\mathbf{s}_k^i}\right) \leq \varepsilon$ in (12.15) it is required to evaluate the constraint in (12.13), which depends on the uncertainty or the covariance at each time step. Thus the uncertainty needs to be propagated at each time step to compute the collision probability constraint. In this work we use the EKF uncertainty propagation; other approaches can be found in [73]. Note that the covariance dynamics dependent on the robot state and control inputs and hence require $\frac{L}{2}(n_{\mathbf{x}}^2+n_{\mathbf{x}})$ [36] ($n_{\mathbf{x}}$ is the dimension of $\mathbf{x}$) additional variables in the optimization problem, increasing the computation time significantly. Thus, similar to [36, 113], we approximate the uncertainty evolution by propagating the robot uncertainties based on its last-loop state and control inputs.

12.3 Comparison to Other Approaches

We provide a comparison with several state-of-the-art methods using a robot and a close-by obstacle. For this comparison, we use a 2D example, however our approach is not limited to 2D scenarios and is equally applicable in 3D scenarios as it can be seen in Section 12.4. The robot is located at $(0.38,0)$ m with radius 0.2 m and covariance $diag(0.04,0.04)$ m^2. The obstacle is located at the origin with radius 0.2 m. The collision probability values can be seen in Table 12.1. To validate the value computed using our approach, we compute the exact collision probability by performing numerical integration. Given the current robot state

Methods	Collision probability	Computation time (ms)
Numerical integral	0.1728	9168.9 ± 258.0
Approximate Numerical integral [65]	0.4280	18.30 ± 3.90
Bounding volume [81, 54]	1	0.1480 ± 0.4411
Maximum probability approximation [82]	1	101.6 ± 23.86
Chance constraint [113]	0.5398	0.3917 ± 0.1278
Rectangular bounding box [33]	0.1601	0.067 ± 0.0070
Our approach	0.1772	0.588 ± 0.13

Table 12.1 Comparison of collision probability methods.

$\mathbf{x}_k$ and the obstacle state $\mathbf{s}_k$, the collision probability is given by

$$P\left(\mathscr{C}_{\mathbf{x}_k,\mathbf{s}_k}\right) = \int_{\mathbf{x}_k}\int_{\mathbf{s}_k} I_c(\mathbf{x}_k,\mathbf{s}_k)p(\mathbf{x}_k,\mathbf{s}_k) \tag{12.17}$$

where I_c is an indicator function defined as

$$I_c(\mathbf{x}_k,\mathbf{s}_k) = \begin{cases} 1 & \text{if } \mathscr{R}\cap\mathscr{S} \neq \{\phi\} \\ 0 & \text{otherwise.} \end{cases} \tag{12.18}$$

and $p(\mathbf{x}_k,\mathbf{s}_k)$ is the joint distribution of the robot and the obstacle. The numeric integral of (12.17) gives the exact value and is used to compare the tightness of the upper bound computed using our approach. As seen from Table 12.1 the value computed using our approach provides a tighter bound when compared to other approaches. The double summation of numerical integration is approximated to a single summation in [65] and this results in a much higher value. Other approaches compute loose upper bounds and hence the resulting values are significantly higher. Our approach thus computes a tighter upper bound.

12.4 Results

In this section we describe our implementation and then evaluate the capabilities of our approach. Simulations are performed in the Gazebo environment with mobile robots as well as quadrotors. The mobile robot kinematics is as follows

$$\mathbf{x}_{k+1} = \begin{bmatrix} x_k - \frac{v_k}{\omega_k}\sin(\theta_k) + \frac{v_k}{\omega_k}\sin(\theta_k+\omega_k\Delta t) \\ y_k + \frac{v_k}{\omega_k}\cos(\theta_k) - \frac{v_k}{\omega_k}\cos(\theta_k+\omega_k\Delta t) \\ \theta_k + \omega_k\Delta t \end{bmatrix} + n_k \tag{12.19}$$

	Measurement noise								
	Σ			4Σ			16Σ		
Method	d(m)	l(m)	T(s)	d(m)	l(m)	T(s)	d(m)	l(m)	T(s)
Bounding volume [81, 54]	0.50	3.31	16.19	0.70	3.90	16.34	0.81	4.26	16.54
Our method	0.49	3.28	16.01	0.58	3.59	16.13	0.61	3.71	16.29

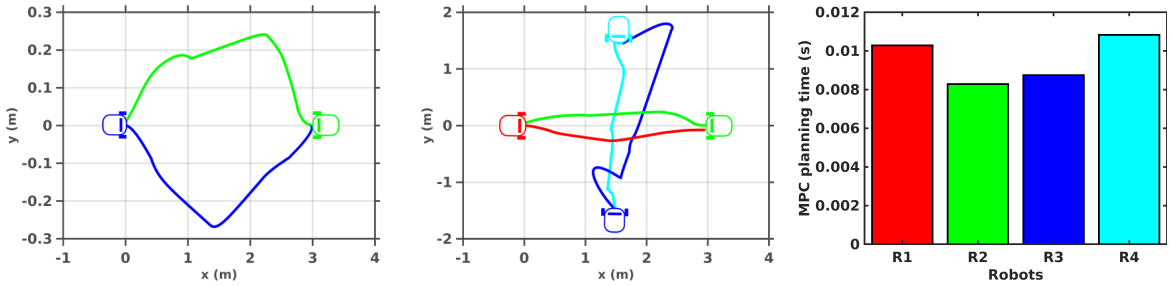
Table 12.2 Collision probability efficiency with varying measurement noise.

where the applied control $\mathbf{u}_k = (v_k, \omega_k)^T$ is made up of the linear and angular velocities and n_k is the zero mean Gaussain noise. We refer the readers to [22] for the quadrotor dynamics. The ground truth odometry from Gazebo is used to measure the pose of the robot, mimicking a motion capture system. This measurement is then corrupted with noise which is zero mean and is used to estimate the state of the robots employing an EKF. The optimization of (12.15) is set up in ACADO [39] which generates a C++ template to run the MPC problem (12.15), and is then modified according to the execution platform. For quadrotor control, we use the publicly available RPG-MPC[1] [22] which is also based on ACADO and modify it to meet our requirements. A look-ahead horizon of $L = 1$ second is used with a discretization of 0.1 seconds for mobile robots and for the quadrotors we use $L = 2$ seconds. The performance is evaluated on an Intel® Core i7-6500U CPU@2.50GHz×4 with 8GB RAM under Ubuntu 16.04 LTS.

Comparison to bounding volume approaches: Bounding volume methods [81, 54] presents a straightforward approach for computing collision probability under uncertainty by enlarging the robot and obstacles by their $3-\sigma$ ellipsoids. We provide an efficiency comparison of such methods with ours. We consider a scenario in which a mobile robot navigates from $(0,0)$ m to $(3,0)$ m with an obstacle of radius 0.2 m at $(1.5,0)$ m. We begin with a measurement noise of $\Sigma = diag(0.02\text{m}^2, 0.02\text{m}^2, 1.2\text{deg}^2)$ and increase it to 4Σ and 16Σ. We define the following metrics to compare the efficiency: $d-$ minimum distance between the robot and the obstacle, $l-$ total trajectory length, and $T-$ total trajectory duration. The results are shown in Table 12.2 where each given value is an average over 10 different simulations. In all the three cases, the trajectory length and duration quantities certify our approach as most efficient. This is more evident as the measurement noise increases as we compute a tight upper bound.

Mobile robot scenarios: In this setting we consider multiple mobile robots exchanging their initial positions with each robot considering every other robot as a dynamic obstacle. To this end, the trajectory (pose and covariance) of each robot is communicated to other robots. The trajectories for two and four mobile robots exchanging their positions can be

[1] https://github.com/uzh-rpg/rpg_mpc

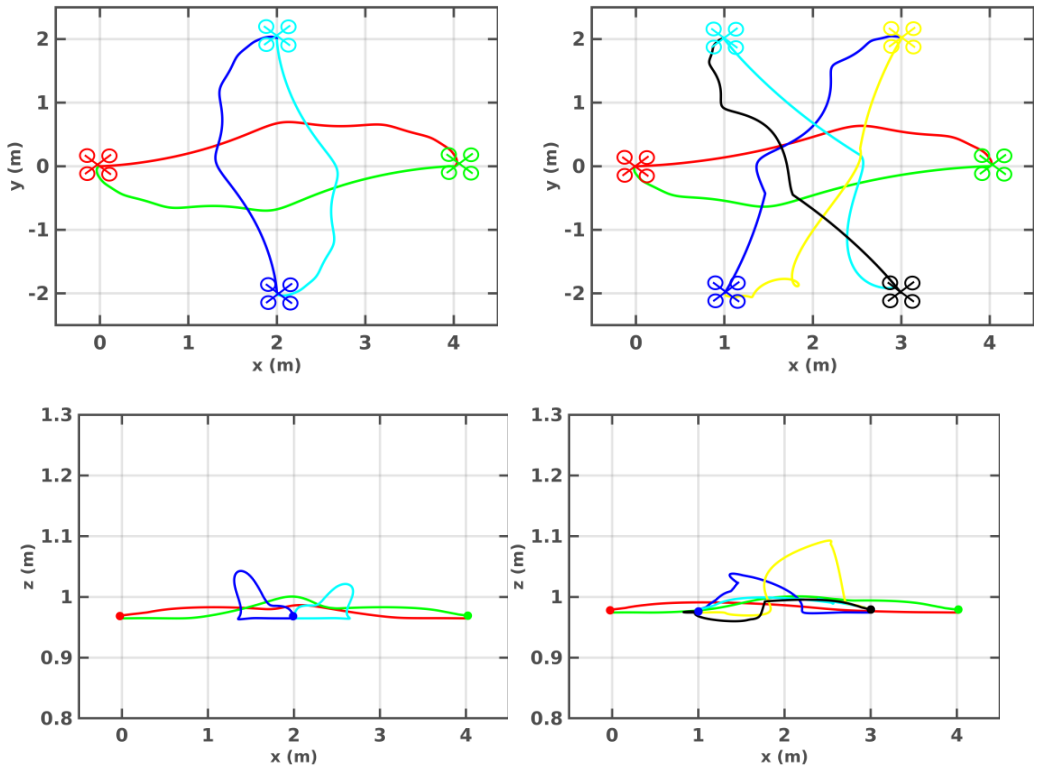


(a) **(b)** **(c)**

Figure 12.1 (a), (b) Simulation results of mobile robots exchanging their positions. The solid lines represent the trajectories executed by the robots. (c) Mean MPC planning time for four robots R1, R2, R3 and R4, respectively.

seen in Fig. 12.1a and Fig. 12.1b. We use a collision probability threshold of 0.1 and a minimum separation of 0.2 meters is achieved between the robots. In both the cases, a measurement noise of $\Sigma = diag(0.02\text{m}^2, 0.02\text{m}^2, 1.2\text{deg}^2)$ is used to corrupt the ground truth odometry which is then used to estimate the robot states using EKF. The simulation was run 10 times and the robots successfully avoided collisions in all the runs. For collision probability thresholds above 0.2 the success rate was less 100%. The average computation time for MPC planning is 9.50 ms and Fig. 12.1c shows the mean MPC planning time for each robot. The low computation time thus allows for real time online planning.

Quadrotor scenarios: Similar to the scenario discussed above, here we consider multiple quadrotors exchanging their initial positions. Each quadrotor communicates its trajectory, both pose and covariance, with others. The top view and side view for four and six quadrotors exchanging their initial positions can be seen in Fig. 12.2. Close distances between quadrotors are observed due to our tight bound. We also observed that a success rate of 100% is achieved for collision probability thresholds below 0.1. Mean computation time for MPC planning is 3.05 ms.



(a) Four quadrotors **(b)** Six quadrotos

Figure 12.2 Simulation results of quadrotors exchanging their positions; trajectories executed visualized by solid lines. The upper plots show the top view (x-y) and the lower plots show the side view (x-z).

Chapter 13

Conclusions

In this part of the thesis, we have first presented an approach that incorporates reasoning regarding the landmark uncertainties within the BSP framework. We consider a Gaussian parametrization of the belief dynamics and derive the corresponding mean and covariance of the belief state when the object uncertainty is considered. We also analyze the effect of adding the object uncertainty for belief estimation and provide the conditions when the effect is negligible.

In Chapter 10 we present a novel approach to compute an exact expression for the collision probability when the robot and obstacle states are uncertain. In contrast, existing works compute an approximation of the actual collision probability. Note that the exact value can be computed using numerical integration but is computationally expensive. We therefore propose an alternate derivation to obtain the exact value by formulating the collision condition as a quadratic form in random variable and the associated collision probability is the cdf of the quadratic from. We derive the cdf is derived as an infinite series and we prove its convergence and provide an upper bound for the truncation error. We further relax the spherical geometry (of robot and obstacles) assumption by considering the exact convex footprints of the robot and the obstacles and derive the collision constraints for convex polygons. A method to estimate the states of dynamic obstacles and further estimate its future states to enable non-myopic planning is also discussed. Gazebo based simulation using single and multi-robot scenarios with both static and dynamic obstacles demonstrate the real-time online capability of our approach.

In many scenarios it might be sufficient to work with approximate collision probability values. Such approximations can be computed much faster when compared to the exact values. Thus in Chapter 11 we derive a tight upper bound for fast approximation of the collision probability for 3D motion planning. We also relax the spherical geometry assumption of

Chapter 10 and consider the minimum-volume enclosing ellipsoids. This representation provides a better approximation of polyhedrons as compared to the spherical representation. Efficiency of our approach with respect to trajectory length and duration is tested in simulation by comparing with bounding volume approaches. Trajectory variation due to change in obstacle configurations have also been tested and it is seen that our method readily adapts to the same. Finally, we also test our approach in a heavily cluttered column domain.

In Chapters 10 and 11 we first derive the collision probability and then formulate it as a constraint for online motion planning. In Chapter 12 we directly derive the collision constraint (a tight upper bound for collision probability can be derived from it) for online MPC planning. The proposed approach is tested in simulation considering mobile robots as well as quadrotors to demonstrate that successful collision avoidance is achieved in real time application.

Though our simulations in multi-robot setting resembles that of Multi-Agent Pathfinding (MAPF), there are certain differences. We consider a combined objective function of cost due to uncertainty and distance traversed by the robot. Moreover, we perform a look-ahead based planning rather than in the combined state space of all agents, as usually done in MAPF. Further, we do not consider the *wait* action in MAPF and we require the robots to be moving unless they have reached the goal. In this work, it was our aim to demonstrate the efficacy of our collision avoidance strategies. Yet, with minor changes, we believe that our approach can be suited to certain facets of MAPF.

Note that currently we assume either spherical or ellipsoid bounding volumes for obstacles and spheres. Though this approximations are valid for all practical purposes the estimates may not fare well in heavily crowded environments or those with narrow passages. In such scenarios the collision constraint is to be formulated by considering the exact geometries of robot and obstacles. As a first step in this direction, in Chapter 10 we consider 2D scenarios by deriving the collision constraint between convex polygons by considering their footprints. This needs to be extended to 3D scenarios by considering the collision constraint between 3D polygons.

Currently we consider all obstacles in the environment while computing the collision probability during each time instant of the planning phase. Thought it does not hinder real time performance as seen from the experiments, scalability to larger domains might be challenging. Considering obstacles only within a certain distance from the robot, for example, within its 3-σ uncertainty region might reduce the computation time to efficiently scale the approach to larger domains (higher number of obstacles). Furthermore, in some cases the assumption of Gaussian distribution for robot state and other noises may not hold.

For example, estimation problems under aliased environments can lead to many distinct hypothesis for obstacle state estimates and this assumption will fare poorly. Another challenge is to determine the number of lookahead steps for MPC planning. Currently we perform the method of trail and error aided with our previous experience. Similar challenge exist in determining the discretization time for each step of the MPC optimization. These variables also change with different robot dynamics, for example, a mobile robot and a quadrotor exhibit different behaviors under the same constants. Note that in all the derivations we have considered two different spheres or ellipsoids interacting. Manipulators present an added challenge that they can be modeled only using a combination of spheres or ellipsoids. Though extension of our methods to such robots are trivial— one need to consider the collision of each sphere or ellipsoid of the robot to that of the obstacle, it would be interesting to see how our methods fare under such robots.

References

[1] Agha-Mohammadi, A.-A., Chakravorty, S., and Amato, N. M. (2014). FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304.

[2] Alatartsev, S., Stellmacher, S., and Ortmeier, F. (2015). Robotic task sequencing problem: A survey. *Journal of intelligent & robotic systems*, 80(2):279–298.

[3] Aoude, G. S., Luders, B. D., Joseph, J. M., Roy, N., and How, J. P. (2013). Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76.

[4] Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J. (2006). *The traveling salesman problem: a computational study*. Princeton university press.

[5] Axelrod, B., Kaelbling, L. P., and Lozano-Pérez, T. (2018). Provably safe robot navigation with obstacle uncertainty. *The International Journal of Robotics Research*, 37(13-14):1760–1774.

[6] Bernardini, S., Fox, M., Long, D., and Piacentini, C. (2017). Boosting Search Guidance in Problems with Semantic Attachments. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 29–37, Pittsburgh, PA, USA.

[7] Blackmore, L., Ono, M., and Williams, B. C. (2011). Chance-constrained optimal path planning with obstacles. *IEEE Transactions on Robotics*, 27(6):1080–1094.

[8] Bry, A. and Roy, N. (2011). Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, pages 723–730.

[9] Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204.

[10] Cambon, S., Alami, R., and Gravot, F. (2009). A hybrid approach to intricate motion, manipulation and task planning. *The International Journal of Robotics Research*, 28(1):104–126.

[11] Cameron, S. and Culley, R. (1986). Determining the minimum translational distance between two convex polyhedra. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 591–596.

[12] Coles, A. J., Coles, A. I., Fox, M., and Long, D. (2010). Forward-chaining partial-order planning. In *Twentieth International Conference on Automated Planning and Scheduling*.

[13] Dantam, N. T., Kingston, Z. K., Chaudhuri, S., and Kavraki, L. E. (2018). An Incremental Constraint-Based Framework for Task and Motion Planning. *International Journal of Robotics Research, Special Issue on the 2016 Robotics: Science and Systems Conference*, 37(10):1134–1151.

[14] de Silva, L., Pandey, A. K., Gharbi, M., and Alami, R. (2013). Towards combining htn planning and geometric task planning. In *RSS Workshop on Combined Robot Motion Planning and AI Planning for Practical Applications*.

[15] Ding, X. C., Pinto, A., and Surana, A. (2013). Strategic planning under uncertainties via constrained markov decision processes. In *IEEE International Conference on Robotics and Automation*, pages 4568–4575.

[16] Dornhege, C., Eyerich, P., Keller, T., Trüg, S., Brenner, M., and Nebel, B. (2009a). Semantic Attachments for Domain-Independent Planning Systems. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 114–121, Thessaloniki, Greece.

[17] Dornhege, C., Gissler, M., Teschner, M., and Nebel, B. (2009b). Integrating symbolic and geometric planning for mobile manipulation. In *Safety, Security & Rescue Robotics (SSRR), IEEE International Workshop on*, pages 1–6. IEEE.

[18] Du Toit, N. E. and Burdick, J. W. (2010). Robotic motion planning in dynamic, cluttered, uncertain environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 966–973. IEEE.

[19] Du Toit, N. E. and Burdick, J. W. (2011). Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 27(4):809–815.

[20] Durrant-Whyte, H. and Henderson, T. C. (2016). Multisensor data fusion. In *Springer handbook of robotics*, pages 867–896. Springer.

[21] Erdem, E., Haspalamutgil, K., Palaz, C., Patoglu, V., and Uras, T. (2011). Combining high-level causal reasoning with low-level geometric reasoning and motion planning for robotic manipulation. In *2011 IEEE International Conference on Robotics and Automation*, pages 4575–4581. IEEE.

[22] Falanga, D., Foehn, P., Lu, P., and Scaramuzza, D. (2018). PAMPC: Perception-aware model predictive control for quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE.

[23] Ferreau, H. J., Kirches, C., Potschka, A., Bock, H. G., and Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363.

[24] Fikes, R. E. and Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208.

[25] Fox, M. and Long, D. (2003). PDDL2. 1: An extension to PDDL for expressing temporal planning domains. *Journal of artificial intelligence research*, 20:61–124.

[26] Fox, M. and Long, D. (2006). Modelling Mixed Discrete-Continuous Domains for Planning. *Journal of Artificial Intelligence Research*, 27(1):235–297.

[27] Frey, K., Steiner, T., and How, J. (2020). Collision Probabilities for Continuous-Time Systems Without Sampling. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA.

[28] Garrett, C. R., Lozano-Perez, T., and Kaelbling, L. P. (2018). FFRob: Leveraging symbolic planning for efficient task and motion planning. *The International Journal of Robotics Research*, 37(1):104–136.

[29] Garrett, C. R., Paxton, C., Lozano-Pérez, T., Kaelbling, L. P., and Fox, D. (2019). Online replanning in belief space for partially observable task and motion problems. *arXiv preprint arXiv:1911.04577*.

[30] Ghallab, M., Nau, D., and Traverso, P. (2016). *Automated planning and acting*. Cambridge University Press.

[31] Grötschel, M., Lovász, L., and Schrijver, A. (1988). *Geometric Algorithms and Combinatorial Optimization*. Springer-Verlag, New York.

[32] Hakobyan, A., Kim, G. C., and Yang, I. (2019). Risk-Aware Motion Planning and Control Using CVaR-Constrained Optimization. *IEEE Robotics and Automation Letters*, 4(4):3924–3931.

[33] Hardy, J. and Campbell, M. (2013). Contingency planning over probabilistic obstacle predictions for autonomous road vehicles. *IEEE Transactions on Robotics*, 29(4):913–929.

[34] Hauser, K. and Latombe, J.-C. (2009). Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries. In *In ICAPS Workshop on Bridging the Gap between Task and Motion Planning*.

[35] Hauser, K. and Latombe, J.-C. (2010). Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 29(7):897–915.

[36] Hewing, L., Liniger, A., and Zeilinger, M. N. (2018). Cautious nmpc with gaussian process dynamics for autonomous miniature race cars. In *2018 European Control Conference (ECC)*, pages 1341–1348. IEEE.

[37] Hoffmann, J. (2003). The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *Journal of Artificial Intelligence Research*, 20:291–341.

[38] Hoffmann, J. and Nebel, B. (2001). The FF Planning System: Fast Plan Generation Through Heuristic Search. *Journal of Artificial Intelligence Research*, 14:253–302.

[39] Houska, B., Ferreau, H., and Diehl, M. (2011). ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312.

[40] Imeson, F. and Smith, S. L. (2014). A language for robot path planning in discrete environments: The tsp with boolean satisfiability constraints. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5772–5777. IEEE.

[41] Imeson, F. and Smith, S. L. (2019). An SMT-based approach to motion planning for multiple robots with complex constraints. *IEEE Transactions on Robotics*, 35(3):669–684.

[42] Indelman, V. (2018). Cooperative multi-robot belief space planning for autonomous navigation in unknown environments. *Autonomous Robots*, 42(2):353–373.

[43] Indelman, V., Carlone, L., and Dellaert, F. (2015). Planning in the Continuous Domain: a Generalized Belief Space Approach for Autonomous Navigation in Unknown Environments. *International Journal of Robotics Research*, 34(7):849–882.

[44] Janson, L., Schmerling, E., and Pavone, M. (2018). Monte Carlo motion planning for robot trajectory optimization under uncertainty. In *Robotics Research*, pages 343–361. Springer.

[45] Jasour, A. M. and Williams, B. C. (2019). Risk contours map for risk bounded motion planning under perception uncertainties. *Robotics: Science and Systems*.

[46] Jiang, Y., Yang, F., Zhang, S., and Stone, P. (2019a). Task-Motion Planning with Reinforcement Learning for Adaptable Mobile Service Robots. In *IROS*, pages 7529–7534.

[47] Jiang, Y.-q., Zhang, S.-q., Khandelwal, P., and Stone, P. (2019b). Task planning in robotics: an empirical comparison of PDDL-and ASP-based systems. *Frontiers of Information Technology & Electronic Engineering*, 20(3):363–373.

[48] John, F. (1985). *Extremum problems with inequalities as subsidiary conditions, Studies and Essays Presented to R. Courant on his 60th Birthday, January 8, 1948*. Interscience Publishers, Inc., New York, reprinted in: J. Moser (ed.), Fritz John Collected Papers, Ch. 2, Birkhauser, Boston, pages 543-560.

[49] Johnson, N. L., Kotz, S., and Balakrishnan, N. (1994). Continuous univariate distributions. john wiley& sons. *New York, NY*.

[50] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134.

[51] Kaelbling, L. P. and Lozano-Pérez, T. (2012). Integrated robot task and motion planning in the now. Technical Report 2012-018, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

[52] Kaelbling, L. P. and Lozano-Pérez, T. (2013). Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227.

[53] Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

[54] Kamel, M., Alonso-Mora, J., Siegwart, R., and Nieto, J. (2017). Robust collision avoidance for multiple micro aerial vehicles using nonlinear model predictive control. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 236–243. IEEE.

[55] Karaman, S. and Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894.

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

[57] Khandelwal, P., Zhang, S., Sinapov, J., Leonetti, M., Thomason, J., Yang, F., Gori, I., Svetlik, M., Khante, P., Lifschitz, V., et al. (2017). Bwibots: A platform for bridging the gap between ai and human–robot interaction research. *The International Journal of Robotics Research*, 36(5-7):635–659.

[58] Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2149–2154. IEEE.

[59] Kotz, S., Johnson, N. L., and Boyd, D. (1967a). Series representations of distributions of quadratic forms in normal variables. I. Central case. *The Annals of Mathematical Statistics*, 38(3):823–837.

[60] Kotz, S., Johnson, N. L., and Boyd, D. (1967b). Series representations of distributions of quadratic forms in normal variables ii. non-central case. *The Annals of Mathematical Statistics*, 38(3):838–848.

[61] Kurniawati, H., Du, Y., Hsu, D., and Lee, W. S. (2011). Motion planning under uncertainty for robotic tasks with long time horizons. *The International Journal of Robotics Research*, 30(3):308–323.

[62] Kurniawati, H. and Yadav, V. (2016). An Online POMDP Solver for Uncertainty Planning in Dynamic Environment. In *Robotics Research: The 16th International Symposium ISRR*, pages 611–629.

[63] Lagriffoul, F., Dantam, N. T., Garrett, C., Akbari, A., Srivastava, S., and Kavraki, L. E. (2018). Platform-independent benchmarks for task and motion planning. *Robotics and Automation Letters*.

[64] Lagriffoul, F., Dimitrov, D., Bidot, J., Saffiotti, A., and Karlsson, L. (2014). Efficiently combining task and motion planning using geometric constraints. *The International Journal of Robotics Research*, 33(14):1726–1747.

[65] Lambert, A., Gruyer, D., and Saint Pierre, G. (2008). A fast Monte Carlo algorithm for collision probability estimation. In *10th IEEE International Conference on Control, Automation, Robotics and Vision*, pages 406–411.

[66] Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers.

[67] Lee, A., Duan, Y., Patil, S., Schulman, J., McCarthy, Z., Van Den Berg, J., Goldberg, K., and Abbeel, P. (2013). Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5660–5667.

[68] Lifschitz, V. (2002). Answer set programming and plan generation. *Artificial Intelligence*, 138(1-2):39–54.

[69] Liu, W. and Ang, M. H. (2014). Incremental sampling-based algorithm for risk-aware planning under motion uncertainty. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2051–2058.

[70] Lo, S.-Y., Zhang, S., and Stone, P. (2018). Petlon: Planning efficiently for task-level-optimal navigation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 220–228. International Foundation for Autonomous Agents and Multiagent Systems.

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

[72] Lozano-Pérez, T. and Kaelbling, L. P. (2014). A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*, pages 3684–3691. IEEE.

[73] Luo, Y.-z. and Yang, Z. (2017). A review of uncertainty propagation in orbital mechanics. *Progress in Aerospace Sciences*, 89:23–39.

[74] Martinelli, A., Pont, F., and Siegwart, R. (2005). Multi-robot localization using relative observations. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 2797–2802. IEEE.

[75] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL- The Planning Domain Definition Language. In *AIPS-98 Planning Competition Committee*.

[76] Mendel, J. M. (1995). *Lessons in estimation theory for signal processing, communications, and control*. Pearson Education.

[77] Muñoz, P., R-Moreno, M. D., and Barrero, D. F. (2016). Unified framework for path-planning and task-planning for autonomous robots. *Robotics and Autonomous Systems*, 82:1–14.

[78] Nilsson, N. J. (1984). Shakey the robot. Technical Report 323, Airtificial Intellignece Center, SRI International, Menlo Park, California.

[79] Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE.

[80] Pandey, A. K., Saut, J.-P., Sidobre, D., and Alami, R. (2012). Towards planning human-robot interactive manipulation tasks: Task dependent and human oriented autonomous selection of grasp and placement. In *2012 4th IEEE RAS & EMBS International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, pages 1371–1376. IEEE.

[81] Park, C., Pan, J., and Manocha, D. (2012). ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In *Twenty-Second International Conference on Automated Planning and Scheduling*.

[82] Park, C., Park, J. S., and Manocha, D. (2018). Fast and bounded probabilistic collision detection for high-DOF trajectory planning in dynamic environments. *IEEE Transactions on Automation Science and Engineering*, 15(3):980–991.

[83] Pathak, S., Thomas, A., and Indelman, V. (2018). A unified framework for data association aware robust belief space planning and perception. *The International Journal of Robotics Research*, 37(2-3):287–315.

[84] Patil, S., Duan, Y., Schulman, J., Goldberg, K., and Abbeel, P. (2014a). Gaussian belief space planning with discontinuities in sensing domains. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6483–6490. IEEE.

[85] Patil, S., Kahn, G., Laskey, M., Schulman, J., Goldberg, K., and Abbeel, P. (2014b). Scaling up Gaussian Belief Space Planning Through Covariance-Free Trajectory Optimization and Automatic Differentiation. In *Eleventh International Workshop on the Algorithmic Foundations of Robotics, WAFR, Boğaziçi University, İstanbul, Turkey*, volume 107 of *Springer Tracts in Advanced Robotics*, pages 515–533. Springer.

[86] Patil, S., Van Den Berg, J., and Alterovitz, R. (2012). Estimating probability of collision for safe motion planning under Gaussian motion and sensing uncertainty. In *IEEE International Conference on Robotics and Automation*, pages 3238–3244.

[87] Piacentini, C., Alimisis, V., Fox, M., and Long, D. (2015). An extension of metric temporal planning with application to ac voltage control. *Artificial intelligence*, 229:210–245.

[88] Platt Jr, R., Tedrake, R., Kaelbling, L., and Lozano-Perez, T. (2010). Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain.

[89] Prentice, S. and Roy, N. (2009). The belief roadmap: Efficient planning in belief space by factoring the covariance. *The International Journal of Robotics Research*, 28(11-12):1448–1465.

[90] Provost, S. B. and Mathai, A. (1992). *Quadratic forms in random variables: theory and applications*. M. Dekker.

[91] Rimon, E. and Boyd, S. P. (1997). Obstacle collision detection using best ellipsoid fit. *Journal of Intelligent and Robotic Systems*, 18(2):105–126.

[92] Roumeliotis, S. I. and Bekey, G. A. (2002). Distributed multirobot localization. *IEEE transactions on robotics and automation*, 18(5):781–795.

[93] Sadigh, D. and Kapoor, A. (2016). Safe control under uncertainty with probabilistic signal temporal logic. *Robotics: Science and Systems*.

[94] Saha, M., Sánchez-Ante, G., and Latombe, J.-C. (2003). Planning multi-goal tours for robot arms. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 3, pages 3797–3803. IEEE.

[95] Salzman, O., Hou, B., and Srinivasa, S. (2017). Efficient motion planning for problems lacking optimal substructure. In *Twenty-Seventh International Conference on Automated Planning and Scheduling*.

[96] Schmerling, E. and Pavone, M. (2017). Evaluating Trajectory Collision Probability through Adaptive Importance Sampling for Safe Motion Planning. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts.

[97] Schulz, D. and Burgard, W. (2001). Probabilistic state estimation of dynamic objects with a moving mobile robot. *Robotics and Autonomous Systems*, 34(2-3):107–115.

[98] Shimanuki, L. and Axelrod, B. (2018). Hardness of 3D Motion Planning Under Obstacle Uncertainty. *Workshop on Algorithmic Foundations of Robotics*.

[99] Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S., and Abbeel, P. (2014). Combined task and motion planning through an extensible planner-independent interface layer. In *Robotics and Automation (ICRA), IEEE International Conference on*, pages 639–646. IEEE.

[100] Stilman, M. and Kuffner, J. (2008). Planning among movable obstacles with artificial constraints. *The International Journal of Robotics Research*, 27(11-12):1295–1307.

[101] Sun, W., Torres, L. G., Van Den Berg, J., and Alterovitz, R. (2016). Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In *Robotics Research*, pages 685–701. Springer.

[102] Thomas, A., Mastrogiovanni, F., and Baglietto, M. (2019). Task-Motion Planning for Navigation in Belief Space. In *The International Symposium on Robotics Research*.

[103] Thomas, A., Mastrogiovanni, F., and Baglietto, M. (2020). Towards Multi-Robot Task-Motion Planning for Navigation in Belief Space. In *European Starting AI Researchers' Symposium*. CEUR.

[104] Thomas, A., Mastrogiovanni, F., and Baglietto, M. (2021). An Integrated Localization, Motion Planning and Obstacle Avoidance Algorithm in Belief Space. *Intelligent Service Robotics*, 14(2):235–250.

[105] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT press.

[106] Toussaint, M. (2015). Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[107] Toussaint, M. and Lopes, M. (2017). Multi-bound tree search for logic-geometric programming in cooperative manipulation domains. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4044–4051. IEEE.

[108] Van Den Berg, J., Abbeel, P., and Goldberg, K. (2011). Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913.

[109] Van Den Berg, J., Patil, S., and Alterovitz, R. (2012). Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278.

[110] Van Den Berg, J., Stilman, M., Kuffner, J., Lin, M., and Manocha, D. (2009). Path planning among movable obstacles: a probabilistically complete approach. In *Workshop on the Algorithmic Foundations of Robotics VIII, WAFR, Guanajuato, Mexico*, pages 599–614. Springer.

[111] Weyhrauch, R. W. (1980). Prolegomena to a theory of mechanized formal reasoning. *Artificial Intelligence*, 13.

[112] Wong, C., Yang, E., Yan, X.-T., and Gu, D. (2018). Optimal path planning based on a multi-tree T-RRT* approach for robotic task planning in continuous cost spaces. In *2018 12th France-Japan and 10th Europe-Asia Congress on Mechatronics*, pages 242–247. IEEE.

[113] Zhu, H. and Alonso-Mora, J. (2019). Chance-constrained collision avoidance for mavs in dynamic environments. *IEEE Robotics and Automation Letters*, 4(2):776–783.

Appendix A

Derivation of (9.9)

The mean of $b[\mathbf{x}_{k+1}]$ is the value that minimizes $\mathscr{J}_{k+1}$, and therefore it is obtained by equating its first derivative to zero. The first derivative of $\mathscr{J}_{k+1}$ with respect to $\mathbf{x}_{k+1}$ is obtained as

$$\frac{\partial \mathscr{J}_{k+1}}{\partial \mathbf{x}_{k+1}} = -H_{k+1}^T Q_{k+1}^{-1} \left(\mathbf{z}_{k+1} - h(\bar{\boldsymbol{\mu}}_{k+1}) - H_{k+1}(\mathbf{x}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1})\right) + \Sigma_{O_{k+1}^i}^{-1} \left(\mathbf{x}_{k+1} - \boldsymbol{\mu}_{O_{k+1}^i}\right) + \bar{\Sigma}_{k+1}^{-1} \left(\mathbf{x}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) \quad \text{(A.1)}$$

Since we are evaluating an expression for the mean, we will substitute $\boldsymbol{\mu}_{k+1}$ for $\mathbf{x}_{k+1}$. Thus setting the first derivative of $\mathscr{J}_{k+1}$ to zero, we have

$$\begin{aligned} H_{k+1}^T Q_{k+1}^{-1} \left(\mathbf{z}_{k+1} - h(\bar{\boldsymbol{\mu}}_{k+1})\right) &= H_{k+1}^T Q_{k+1}^{-1} H_{k+1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) + \Sigma_{O_{k+1}^i}^{-1} \left(\boldsymbol{\mu}_{k+1} - \boldsymbol{\mu}_{O_{k+1}^i}\right) \\ &\quad + \bar{\Sigma}_{k+1}^{-1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) \\ &= H_{k+1}^T Q_{k+1}^{-1} H_{k+1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) + \bar{\Sigma}_{k+1}^{-1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) \\ &\quad + \Sigma_{O_{k+1}^i}^{-1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1} + \bar{\boldsymbol{\mu}}_{k+1} - \boldsymbol{\mu}_{O_{k+1}^i}\right) \\ &= H_{k+1}^T Q_{k+1}^{-1} H_{k+1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) + \bar{\Sigma}_{k+1}^{-1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) + \\ &\quad \Sigma_{O_{k+1}^i}^{-1} \left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) + \Sigma_{O_{k+1}^i}^{-1} \left(\bar{\boldsymbol{\mu}}_{k+1} - \boldsymbol{\mu}_{O_{k+1}^i}\right) \\ = \left(H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \bar{\Sigma}_{k+1}^{-1} + \Sigma_{O_{k+1}^i}^{-1}\right) &\left(\boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1}\right) + + \Sigma_{O_{k+1}^i}^{-1} \left(\bar{\boldsymbol{\mu}}_{k+1} - \boldsymbol{\mu}_{O_{k+1}^i}\right) \quad \text{(A.2)} \end{aligned}$$

From (9.8) we have $\Sigma_{k+1}^{-1} = H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \bar{\Sigma}_{k+1}^{-1} + \Sigma_{O_{k+1}^i}^{-1}$. Also using the fact that $K_{k+1} = \Sigma_{k+1} H_{k+1}^T Q_{k+1}^{-1}$, (A.2) simplifies to

$$K_{k+1}\left(\mathbf{z}_{k+1} - h(\bar{\boldsymbol{\mu}}_{k+1})\right) = \boldsymbol{\mu}_{k+1} - \bar{\boldsymbol{\mu}}_{k+1} + \Sigma_{k+1}\Sigma_{O_{k+1}^i}^{-1}\left(\bar{\boldsymbol{\mu}}_{k+1} - \boldsymbol{\mu}_{O_{k+1}^i}\right) \tag{A.3}$$

Rearranging, we get the final expression

$$\boldsymbol{\mu}_{k+1} = \bar{\boldsymbol{\mu}}_{k+1} + K_{k+1}\left(\mathbf{z}_{k+1} - h(\bar{\boldsymbol{\mu}}_{k+1})\right) + \Sigma_{k+1}\Sigma_{O_{k+1}^i}^{-1}\left(\boldsymbol{\mu}_{O_{k+1}^i} - \bar{\boldsymbol{\mu}}_{k+1}\right) \tag{A.4}$$

Appendix B

Derivation of (9.10)

In this Appendix we derive the expression for Σ_{k+1} in terms of the Kalman gain K_{k+1} and the predicted covariance $\bar{\Sigma}_{k+1}$. For convenience we write down the matrix inversion lemma which states that for any invertible matrices B and C and any matrix D with appropriate dimensions, the following holds true

$$\left(B + DCD^T\right)^{-1} = B^{-1} - B^{-1}D\left(C^{-1} + D^T B^{-1} D\right)^{-1} D^T B^{-1} \tag{B.1}$$

We note here that the Kalman gain $K_{k+1} = \Sigma_{k+1} H_{k+1}^T Q_{k+1}^{-1}$ in (9.9) is a function of Σ_{k+1}. Thus we first need to derive an expression for K_{k+1} that does not depend Σ_{k+1}. To obtain such an expression, we follow the approach for the standard EKF case presented in [105]. We begin by post-multiplying $\Sigma_{k+1} H_{k+1}^T Q_{k+1}^{-1}$ with an identity matrix $I = AA^{-1}$, where

$$A = \left(H_{k+1}\bar{\Sigma}_{k+1}\left(\bar{\Sigma}_{k+1} + \Sigma_{O_{k+1}^i}\right)^{-1}\Sigma_{O_{k+1}^i} H_{k+1}^T + Q_{k+1}\right) \tag{B.2}$$

To avoid clutter, let us further define $\tilde{\Sigma}_{k+l} = \left(\bar{\Sigma}_{k+1} + \Sigma_{O^i_{k+1}}\right)^{-1}$. The expression for K_{k+1} can then be written as

$$\begin{aligned}
K_{k+1} &= \Sigma_{k+1} H_{k+1}^T Q_{k+1}^{-1} \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + Q_{k+1}\right) \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T \right. \\
&\qquad \left. + Q_{k+1}\right)^{-1} \\
&= \Sigma_{k+1} \left(H_{k+1}^T Q_{k+1}^{-1} H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + H_{k+1}^T\right) \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + Q_{k+1}\right)^{-1} \\
&= \Sigma_{k+1} \left(H_{k+1}^T Q_{k+1}^{-1} H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + \left(\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}}\right)^{-1} \bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T\right) \\
&\qquad \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + Q_{k+1}\right)^{-1} \qquad \text{(B.3)}
\end{aligned}$$

We will now compute the inverse of the term $\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}}$. This can be done as follows:

$$\begin{aligned}
\left(\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}}\right)^{-1} &= \left(\bar{\Sigma}_{k+1}\left(\bar{\Sigma}_{k+1} + \Sigma_{O^i_{k+1}}\right)^{-1}\Sigma_{O^i_{k+1}}\right)^{-1} = \Sigma_{O^i_{k+1}}^{-1}\left(\bar{\Sigma}_{k+1} + \Sigma_{O^i_{k+1}}\right)\bar{\Sigma}_{k+1}^{-1} \\
&= \Sigma_{O^i_{k+1}}^{-1}\bar{\Sigma}_{k+1}\bar{\Sigma}_{k+1}^{-1} + \Sigma_{O^i_{k+1}}^{-1}\Sigma_{O^i_{k+1}}\bar{\Sigma}_{k+1}^{-1} = \Sigma_{O^i_{k+1}}^{-1} + \bar{\Sigma}_{k+1}^{-1} \qquad \text{(B.4)}
\end{aligned}$$

The expression in (B.3) simplifies to

$$\begin{aligned}
K_{k+1} &= \Sigma_{k+1}\left(H_{k+1}^T Q_{k+1}^{-1} H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + \left(\Sigma_{O^i_{k+1}}^{-1} + \bar{\Sigma}_{k+1}^{-1}\right)\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T\right) \\
&\qquad \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + Q_{k+1}\right)^{-1} \\
&= \Sigma_{k+1}\left(H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O^i_{k+1}}^{-1} + \bar{\Sigma}_{k+1}^{-1}\right)\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T \\
&\qquad \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + Q_{k+1}\right)^{-1} \\
&= \Sigma_{k+1}\left(\Sigma_{k+1}\right)^{-1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + Q_{k+1}\right)^{-1} \qquad \text{(B.5)}
\end{aligned}$$

where we have used the fact that $\Sigma_{k+1}^{-1} = H_{k+1}^T Q_{k+1}^{-1} H_{k+1} + \Sigma_{O^i_{k+1}}^{-1} + \bar{\Sigma}_{k+1}^{-1}$. Thus we obtain

$$K_{k+1} = \bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T \left(H_{k+1}\bar{\Sigma}_{k+1}\tilde{\Sigma}_{k+l}\Sigma_{O^i_{k+1}} H_{k+1}^T + Q_{k+1}\right)^{-1} \qquad \text{(B.6)}$$

Let us now define $\Xi_{k+1} = \Sigma^{-1}_{O^i_{k+1}} + \bar{\Sigma}^{-1}_{k+1}$. Applying the matrix inversion lemma to the right hand side of (9.8), we have

$$\Sigma_{k+1} = \Xi^{-1}_{k+1} - \Xi^{-1}_{k+1} H^T_{k+1} \left(Q_{k+1} + H_{k+1} \Xi^{-1}_{k+1} H^T_{k+1} \right)^{-1} H_{k+1} \Xi^{-1}_{k+1} \tag{B.7}$$

From (B.4), we have

$$\Xi^{-1}_{k+1} = \bar{\Sigma}_{k+1} \tilde{\Sigma}_{k+l} \Sigma_{O^i_{k+1}} \tag{B.8}$$

We note here that the expression Ξ^{-1}_{k+1} appears four times in (B.7). Substituting for Ξ^{-1}_{k+1} using (B.8) in the second and third expression of Ξ^{-1}_{k+1} in (B.7), we get

$$\Sigma_{k+1} = \Xi^{-1}_{k+1} - \bar{\Sigma}_{k+1} \tilde{\Sigma}_{k+l} \Sigma_{O^i_{k+1}} H^T_{k+1} \left(Q_{k+1} + H_{k+1} \bar{\Sigma}_{k+1} \tilde{\Sigma}_{k+l} \Sigma_{O^i_{k+1}} H^T_{k+1} \right)^{-1} H_{k+1} \Xi^{-1}_{k+1} \tag{B.9}$$

From (B.6) and (B.8), it is easily seen that the expression in (B.9) simplifies to

$$\begin{aligned} \Sigma_{k+1} &= \Xi^{-1}_{k+1} - K_{k+1} H_{k+1} \Xi^{-1}_{k+1} = \left(I - K_{k+1} H_{k+1} \right) \Xi^{-1}_{k+1} \\ &= \left(I - K_{k+1} H_{k+1} \right) \bar{\Sigma}_{k+1} \tilde{\Sigma}_{k+l} \Sigma_{O^i_{k+1}} = \left(I - K_{k+1} H_{k+1} \right) \bar{\Sigma}_{k+1} \left(\bar{\Sigma}_{k+1} + \Sigma_{O^i_{k+1}} \right)^{-1} \Sigma_{O^i_{k+1}} \end{aligned} \tag{B.10}$$

This completes the derivation.

Appendix C

Derivation of (10.35)

From (10.30), we have

$$\ln M(\theta) = d_0 + \sum_{k=1}^{\infty} d_k \frac{\theta^k}{k} \tag{C.1}$$

For differentiable $M(\theta)$, we have

$$\frac{d}{d\theta} \ln M(\theta) = \frac{1}{M(\theta)} \frac{d}{d\theta} M(\theta) = \sum_{k=1}^{\infty} c_k \theta^{k-1} \tag{C.2}$$

where we have used the definition of $M(\theta)$ given in (10.26). Also note that by construction $M(\theta) > 0$. Re-arranging (C.2), we obtain

$$M(\theta) \frac{d}{d\theta} \ln M(\theta) = \sum_{k=1}^{\infty} c_k \theta^{k-1} \tag{C.3}$$

From (C.1), we have

$$\frac{d}{d\theta} \ln M(\theta) = \sum_{k=1}^{\infty} d_k \theta^{k-1} \tag{C.4}$$

From (C.2) and (C.4), we thus obtain

$$\left(\sum_{k=0}^{\infty} c_k \theta^k \right) \left(\sum_{k=1}^{\infty} d_k \theta^{k-1} \right) = \sum_{k=1}^{\infty} c_k \theta^{k-1} \tag{C.5}$$

Comparing the coefficient of θ^{k-1} on both sides of the equation, we get the required expression for c_k as

$$c_k = \frac{1}{k} \sum_{j=0}^{k-1} d_{k-j} c_j \tag{C.6}$$