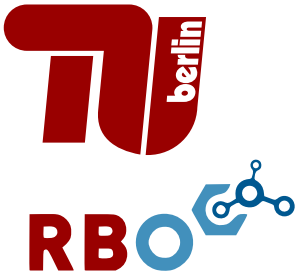
Robust Motion Generation for Mobile Manipulation — Integrating Control and Planning under Uncertainty



vorgelegt von
Arne Sieverling, MSc
ORCID: 0000-0002-2073-0938

von der Fakultät IV — Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation

Prüfungsausschuss

Vorsitzender:	Prof. Dr. Jörg Raisch
Gutachter:	Prof. Dr. Oliver Brock
Gutachter:	Prof. Dr. Marc Toussaint
Gutachter:	Thierry Siméon, PhD, DR

Tag der wissenschaftlichen Aussprache: 23.11.2018

Berlin 2019

Prepublication and Statement of Contribution

Parts of this thesis have been previously published in the following peer-reviewed articles:

(A) **Arne Sieverling**, Nicolas Kuhnen, and Oliver Brock. Sensor-Based, Task-Constrained Motion Generation Under Uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4348–4355. Hong Kong, China. 2014

I (AS) was sole first author and main contributor to paper writing and experimental evaluation. AS and NK equally contributed to implementation and algorithmic ideas. OB gave scientific advice and contributed to paper writing.

(B) Peter Lehner, **Arne Sieverling**, and Oliver Brock. Incremental, Sensor-Based Motion Generation for Mobile Manipulators in Unknown, Dynamic Environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4761–4767. Seattle, USA. 2015

I (AS) was second author and equally contributed with PL to experimental evaluation and paper writing. PL was the main developer of algorithms and implementation. OB gave scientific advice and contributed to paper writing.

(C) Markus Rickert, **Arne Sieverling**, and Oliver Brock. Balancing Exploration and Exploitation in Sampling-Based Motion Planning. In *IEEE Transactions on Robotics.* 30(6):1305-1317, 2014.

I (AS) was second author and contributed to the experimental evaluation. MR was main contributor to algorithmic design and paper writing. OB gave scientific advice and contributed to paper writing.

(D1) Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, **Arne Sieverling**, Vincent Wall, and Oliver Brock. Lessons from the Amazon Picking Challenge: Four Aspects of Building Robotic Systems. In *Proceedings of Robotics: Science and Systems*. Ann Arbour, USA. 2016.

(D2) Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, **Arne Sieverling**, Vincent Wall, and Oliver Brock. Four Aspects of Building Robotic Systems: Lessons from the Amazon Picking Challenge 2015. In *Autonomous Robots*. Springer, US. 2018.

(D2) is an extended version of (D1). I (AS) was joint first author with CE, SH, RJ, RMM, and VW. I was the main developer of the motion generation system and main contributor to the experimental evaluation. All first authors contributed equally to paper writing. OB gave scientific advice and contributed to paper writing.

(E) **Arne Sieverling**, Clemens Eppner, Felix Wolff, and Oliver Brock. Interleaving Motion in Contact and in Free Space for Planning Under Uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4011–4017. Vancouver, Canada. 2017.

I (AS) was sole first author and main contributor to implementation, paper writing, and experimental evaluation. CE assisted in experimentation and writing. FW implemented a prototype and assisted in writing. OB gave scientific advice and contributed to paper writing.

(F) Előd Páll, **Arne Sieverling**, and Oliver Brock. Contingent Contact-based Motion Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain, 2018.

I (AS) was second author and contributed equally with EP to paper writing. EP was the main developer of the method and ran the experimental evaluation. OB gave scientific advice and contributed to paper writing.

Chapters 1 and 2 are original to this thesis. Chapter 3 contains excerpts from (D1) and (D2) but the discussion of the motion generation system was extended. Chapter 4 is original to the thesis, but contains results from (C). The results of Chapters 5 and 6 were previously published in (E) and (F), respectively. Chapter 7 is original to the thesis. Chapter 8 is a combination of publications (A) and (B). Chapter 9 and Chapter 10 are original contributions of this thesis.

Acknowledgements

My first thanks go to Prof. Oliver Brock for giving me the chance to finish this long endeavor and supporting it to the very end. I highly value your commitment to get the best out of every student. Your focus on the big picture made my research more interesting and stronger.

Thanks to Marc Toussaint, Nic Siméon, and Jörg Raisch for being my committee. I am happy that I was part of great projects that showed me the value of integrating systems, ideas, and people. This was only possible thanks to funding from the European Commission and the members of the FIRST-MM and SOMA projects.

I was part of an amazing generation of PhD students at RBO. Building reliable infrastructure from the ground up was the result of amazing teamwork, and I enjoyed every single minute of working with you. Thanks Clemens, Ines, Michael, Raphael, Rico, Roberto, and Sebastian. I am also glad to see the new generation taking over. Thanks Can, Divyam, Előd, Jessica, Manuel, Stefan, Steffen, and Vincent. Thanks Kolja and Mahmoud for being an awesome teaching team. Thanks Alexander, Janika, Wolf, Thomas, and Lizzy for their great support. And thanks to all the other RBO members, visitors, and friends not mentioned. I had the privilege to work with great students. Thanks Nico, Peter, Roman, Gabriel, Rieke, and Felix. Developing ideas with students was one of my biggest motivators, and without your work this thesis would have been very different.

Thanks to my family for their support in this long time. I wish I could catch up on the visits I missed in the last months. Thanks Claudia, Uwe, and Sabine. Thanks Ben, Matze, Steffen, Ireen, Jenny, Johannes, and Georg for being great friends and for always being excited about my work.

My final, biggest thanks go to Annika. You are my greatest motivation, role model, and friend. I am truly happy to navigate this world with you and can't wait to explore all the other worlds.

Abstract

This thesis contributes to algorithmic approaches for the motion generation problem for mobile manipulators. This problem is unsolved in unstructured environments, where the robot does not have access to precise models but must infer the state of the world with its sensors. The challenges for motion generation in these problems arise from the uncertainty prevalent in real world sensors, the different modalities that need to be considered, and real time constraints. Our approach in this thesis is to combine local feedback control with global planning under uncertainty to solve three different applications in manipulation.

In the first part of this thesis we show the feasibility of a feedback-driven approach on a real world manipulation problem. We present an autonomous mobile manipulation system for bin picking. This system was an entry in the "Amazon Picking Challenge", where it outperformed 25 contenders. We evaluate strength and weaknesses of feedback and planning-based methods by comparing our system to others.

In the second part, we review planning-based approaches, using sampling to efficiently search high-dimensional spaces. We present a novel planner for motion that exploits contact to reduce uncertainty. We propose a particle-based uncertainty model and search the combined space of configurations in free space and in contact. Our experiments show that our planners strategies are more robust strategies than solutions of traditional sampling-based planners because of the use of contact to reduce uncertainty. We extend this planner with a model for tactile feedback which allows it to localize objects just using the signal of contact sensors.

In the third part, we discuss the continuous integration of sensor data into plans. To move efficiently in unstructured environments, robots must continuously adapt their plans in response to sensor data. We review trajectory optimization as a tool for path adaptation. We propose a novel approach to sensor-based motion generation based on a factorization in three tasks: 1) continuous path adaption, 2) continuous local planning of new motion alternatives 3) global planning with a model of an uncertain environment. This factorization allows us to generate robust motion in initially completely unknown environments with dynamic obstacles. In addition, we introduce an online learning method for manipulation control based on multi-modal sensors feedback. We conclude this thesis by combining all introduced techniques into a novel unifying framework for motion generation.

Zusammenfassung

Inhalt dieser Arbeit ist ein neuer algorithmischer Ansatz zur effizienten Bewegungsgenerierung von mobilen Manipulatoren. Dieses Problem ist ungelöst in alltäglichen, unstrukturierten Umgebungen. In diesen Umgebungen hat der Roboter kein Zugriff auf exakte Modelle sondern muss Entscheidungen aufgrund seiner Sensordaten treffen. Die Schwierigkeit daran ist die Unsicherheit in den Daten. Unsicherheit erschwert das Planen von robuster Bewegung. Unser Ansatz vereint Planungs- und Regelungbasierte Methoden für drei verschiedene Manipulationsanwendungen.

Im ersten Teil der Arbeit stellen wir ein Fallbeispiel für ein Robotersystem welches hauptsächlich auf lokaler, sensorbasierter Regelung beruht. Wir stellen einen autonomen mobilen Manipulator vor, der autonom bestimmte Objekte aus einem Lagerregal greift. Diese Aufgabe ist für die Intralogistik hochrelevant und wurde als internationaler Wettbewerb "Amazon Picking Challenge" ausgeschrieben. Unser regelungsbasierter Ansatz konnte sich gegen 25 internationale Teams durchsetzen. Wir vergleichen unser System zu den Lösungen anderer Teilnehmer um eine Überblick über die Stärken von regelbasierten und planungbasierten Ansätzen zu geben. Im Falle der "Amazon Picking Challenge" ist die Bewegung für eine bestimmte Aufgabe vorprogrammiert. In den weiteren Teilen der Arbeit stellen wir Ansätze vor, die ein Vielzahl von Aufgaben erledigen.

Im zweiten Teil der Arbeit betrachten wir Planungsbasierte Ansätze die mit Samplingmethoden effizient hochdimensionale Räume durchsuchen. Wir präsentieren einen neuartigen Planer für Manipulation mit Kontakt. Ein Roboter kann Kontakt mit der Umgebung herstellen um Unsicherheit zu reduzieren. Traditionelle Bahnplaner probieren jedoch Kontakt zu verhindern, da sie keine Unsicherheit in ihren Modellen betrachten. Unser Planer benutzt ein Partikel-basiertes Modell und durchsucht sowohl den freien Raum als auch den Raum der Konfigurationen in Kontakt. Wir zeigen in Experimenten, dass unser Planer robustere Strategien findet als Planer die Kontakt und Unsicherheit ignorieren. Eine weitere Variante dieses Planers erstellt Pläne, die taktile Sensoren ausnutzten um die Position von Objekten zu bestimmen.

Der dritte Teil der Arbeit beschäftigt sich mit der effizienten Integration von Sensordaten in Manipulationssysteme. Um sich in unstrukturierten Umgebungen fortzubewegen, müssen

Roboter ihre Pläne effizient anhand von Sensordaten anpassen. Als wichtiges Werkzeug dazu besprechen wir die Pfadoptimierung. Wir betrachten danach einen neuartigen dreigeteilten Ansatz des Problems in 1. kontinuierliche Pfadoptimierung, 2. lokaler Planung von neuen Bewegungsalternativen, und 3. Planung unter Berücksichtigung einer unsicheren dynamischen Welt. Die Kombination dieser drei Ebenen erlaubt uns, Roboter in komplett unbekannten Umgebungen sicher zu manövrieren. Wir stellen außerdem vorläufige Ergebnisse zu einem lernbasierten Ansatz für sensorbasierte Manipulation vor. Wir vereinen alle neuen Konzepte in einer neuen Faktorisierung des Bewegungsgenerierungsproblems.

Contents

List of Figures

Chapter 1

Introduction

Robotic technology promises to assist humans in workplaces such as households or factories. This promise has yet to be fulfilled. Except for vacuum cleaners and lawn mowers, no useful robot operates autonomously in everyday environments. To become truly useful, robots must safely move in human-inhabited environments and assist by changing the world's state. The problem of controlling the state of the environment through selective contact is the **manipulation** (Mason, 2018) problem. In robotics, manipulation is most often implemented on robotic arms and appropriate grippers or hands. These robotic systems that change the environment by applying forces are called **manipulators**. Today, robotic manipulation is a core part of the manufacturing industry. The robots in these applications operate reliably when precise world models are available, tasks are repetitive, and uncertainty sources such as humans are kept away by placing the robot in cages. However, even today, robotic manipulation is stuck in these cages and not used in other applications.

Mobile manipulators are manipulators that also have mobility, i.e. the ability to move freely in space, only constrained by obstacles. The term mobile manipulation was coined around the turn of the millennium when industrial robot arms were mounted on mobile platforms in the Stanford Robotics Lab (Khatib, 1999). The additional mobility should move manipulators out of the caged factory environments into everyday environment such as households, hospitals, and streets. Instead of performing a single task in a controlled environment, a mobile manipulator should assist in a variety of tasks in many different locations. However, this requires to relax the assumption of a perfectly predictable world. Mobile manipulators must deal with dynamic and unknown environments. This requires including sensing capabilities to perceive the world. Therefore, in addition to a change of hardware, mobile manipulation refers to a paradigm shift in software. The programs that run these robots must relax the assumption of perfectly known models which requires algorithms that incorporate perception.

In this thesis we present methods and algorithms for the **motion generation** problem. This problem is, given a description of a manipulation task and a stream of sensor data, to compute motor commands that let the robot reliably fulfill the task while avoiding collisions. In this thesis, we [1] do not distinguish between motion for mobility (i.e. moving the base) and motion for manipulation (i.e. moving the arm). Instead we treat the whole mobile manipulator as one system that should solve manipulation tasks, using all available means. Current motion generation methods reliably solve manipulation tasks once complete information about the environment is available. This can be seen by the impressive motion skills of current robots in controlled settings such as factory floors. However, mobile manipulation systems are still far away from delivering on their promises. The reason for the discrepancy in performance when moving these methods out of factories is the lack of structure in these real world environments.

1.1 Challenges for motion generation

We now want to discuss three challenges for motion generation methods that result from **unstructured environments**. These challenges pose significant difficulties for algorithms and for system design of mobile manipulators.

C1 **Uncertainty:** Motion generation algorithms in unstructured environments need to make decisions about an unpredictable world based on noisy sensors with limited range. To act, robot must take their sensor limitations into account and either take actions that are safe despite high uncertainty or take actions that reduce uncertainty. A robot can find such actions by reasoning explicitly about uncertainty and incorporating the reduction of uncertainty into its objectives. If the robot reasons about the things it knows and does not know, it can either choose actions that will succeed, given its current knowledge, or actions that improve the current knowledge. Reasoning about uncertainty is a complex task because it requires predicting a multitude of possible robot and world states and then taking actions that are expected to succeed in all of them. For long time horizons, the number of possible world states quickly becomes intractable to enumerate.

C2 **Multi-modality:** Manipulation requires a robot to forcefully interact with the world. To do so at a basic skill level, it should be able to monitor the position of objects and

[1] For consistency, the pronoun 'we' will be used to address all contributions, even if they are just the author's personal views. To clarify the author's unique individual contributions, each chapter begins with a discussion. In these parts, 'I' will be used to explain the author's individual contributions to the different chapters.

also observe the forces it is executing. This makes manipulation an inherently multi-modal problem which requires the robot to make sense of different sensor channels with particular strengths (vision, proprioception, tactile sensing, ...) and balance the weaknesses of one channel with the strengths of others. Fusing the information of these different channels is challenging as they measure very different phenomena that cannot easily be merged into a common representation of the world.

C3 **Real-Time requirements:** Unstructured environments require the robot to react timely as obstacles can move unexpectedly and tasks must be fulfilled efficiently. Therefore, all decisions are subject to time constraints. The robot should never stand and think for longer durations, even if a task requires complex, computationally intensive reasoning. Thus real time constraints pose an architectural challenge for motion generation system as the system must process large amount of information at reactive rates.

1.2 Planning vs. feedback

Solutions to the motion generation problem lie on a spectrum between two approaches: feedback and planning-based methods. We will now discuss the two extremes of this spectrum, and then discuss the need for hybrid approaches that take both sides into account.

1.2.1 Planning

Planning divides motion generation into three steps: First, the robot first builds a model of the world by integrating all available sensor data. Second, it finds a course of action that works best in the model. Third, it executes these action in the real world. Planning has many applications in robotics. For example motion planning (LaValle, 2004) casts motion generation as a search problem and finds global solutions, often with theoretical guarantees. The planning approach is particularly useful if good models exist. In such cases, practitioners can calibrate the robot (Hollerbach et al., 2008), localize it in the environment (Thrun et al., 2005), and use planners as general and versatile black-box solvers. Consequently, motion planning methods have been used successfully in many industrial applications (Laumond, 2006). Research in biology also supports the importance of planning. For example, there is evidence that animals reason about the shortest paths between known food sources (Tomasello and Call, 1997).

Planning suffers from the presence of uncertainty, high dimensionality, long time horizons, and inaccurate models. To overcome these challenges, planners can take the uncertainty

of the models into account to find robust plans. However, this makes planning exponentially harder. In fact, complete and optimal planning under uncertainty is computationally intractable (Papadimitriou and Tsitsiklis, 1987).

1.2.2 Feedback

Feedback control implements a sensory-motor loop - the robot continuously extracts task-relevant information from its sensor stream and adapts its actions in response. In manipulation, we can understand feedback control as a continuous sensor-based behavior such as "maintain constant force" or "track an object visually". Feedback control is the basis of many robust motion generation methods and has many applications in manipulation (Kragic and Christensen, 2002). **Force control** (Siciliano and Villani, 2012) closes a loop around force measurements. **Visual servoing** (Chaumette and Hutchinson, 2006) closes a loop around motion of visual features. Feedback from contact with the environment (Lozano-Pérez et al., 1984) was shown to greatly simplify manipulation.

The biggest advantage of the feedback approach is its robustness against uncertainty. Feedback has no requirements for precise models as it continuously extracts the relevant information from sensor data. It does not need to predict many possible futures, as it only reacts to things that actually happen in the real world. There is evidence that feedback is crucial for motion generation in nature too. Humans and bees reliably avoid collisions by using continuous visual feedback (Srinivasan et al., 1996; Warren Jr et al., 2001). Humans exploit contact to reduce uncertainty when grasping (Eppner and Brock, 2015).

However, taken on its own, feedback is limited in its applicability. As perception is limited, the robot can not infer the full state of the world just from its sensors. Additionally, the robot must be able to predict the outcome of its actions in the future. This requires some form of abstraction which leads back to the necessity for planning.

To conclude, we have seen that, on their own, neither feedback or planning is able to deal with the challenges of motion generation in unstructured environments. Thus, one of the main challenges for robust motion generation lies in finding the right factorization of the problem into feedback and planning.

1.3 The hybrid approach

There are many methods between the extremes of planning and feedback. **Feedback motion planning**, which includes methods such as optimal control or model predictive control try

to get the best of both worlds by pre-computing control strategies for many possible states the robot could be in. However, as the complexity of the task increases, these methods get prohibitively complex as they need to cover many eventualities with appropriate reactions. Ultimately, all hybrid methods face a practical trade-off: should they reason quickly and react continuously to incoming data, or should they rather plan for longer horizons and more possibilities.

In this thesis we will propose a novel hybrid approach that integrates planning and feedback on different levels. The approach in this thesis is to exploit feedback as much as possible while performing planning only to the degree needed to solve complex tasks. We will now introduce our approach on a high level.

1.3.1 Sequencing funnels

The metaphor of a **funnel** (Mason, 1985) helps to visualize the uncertainty reduction of feedback. A funnel takes a grain of sand from a large region (the funnel entrance) and moves it to a smaller known region (the funnel exit). Analogously, feedback controllers bring a system from a wide range of possible states (the controller region of attraction), to a well defined goal state (the controllers area of convergence). The funnel view also visualizes the robustness of feedback approaches against uncertainty: if the system is anywhere within the funnel entrance, using feedback control it is guaranteed to end up in a smaller region of space. This holds even if the system does not know its exact position in the funnel entrance.

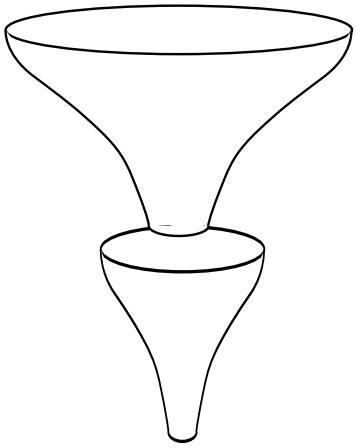


Figure 1.1 The sequential execution of two controllers can be visualized as funnels.

In addition to continuous feedback implemented by funnels, there is also discrete feedback such as the change of contact state when a robot touches the world. These discrete feedback events signal the end of a funnel. To continue with the manipulation, a second controller is needed. This can be visualized by a transition from one funnel to another (Fig. 1.1). Such combined continuous/discrete feedback systems are known as **hybrid systems**. We will model their behavior with state machines that sequence controllers based on discrete sensor events, so-called **hybrid automata**.

Part I of this thesis will present the motion generation system based on hybrid automata for a mobile manipulator. The main contribution is a feedback-based motion generation system that operates in a simplified but realistic application: it reliably picks items from warehouse shelves. The system is robust against real-world uncertainty (C1) by relying on

feedback from different sensors (C2). We show how we manually assemble funnels into complex behavior which lets the system operates reliably without planning. This makes it operate naturally under real-time constraints (C3). We evaluated the system in a robotics competition which allowed us to compare our system to others, in particular systems that are based on planning approaches. The results show the feasibility of applying feedback-based systems in real world tasks.

1.3.2 Reasoning probabilistically about funnels

In Part I, we established hybrid automata as a robust framework for motion generation, however, the behavior was manually specified for a single task. Our next step is to generate hybrid automata for complex motion strategies automatically. A fundamental insight for planning with funnels is that we can quantify the quality of funnels by computing the reduction of uncertainty (see Fig. 1.2). A planner can estimate this reduction by sampling from a distribution that matches the funnel entrance, simulating the execution of the controller, and then estimating the uncertainty of the resulting distribution. Thus, to reason about funnels, we use distributions over state space. Planning with probability distributions is called **belief space planning**. Unfortunately, a very hard problem and computationally intractable to solve in general. To make belief space planning work, we must make further assumptions that simplify reasoning.

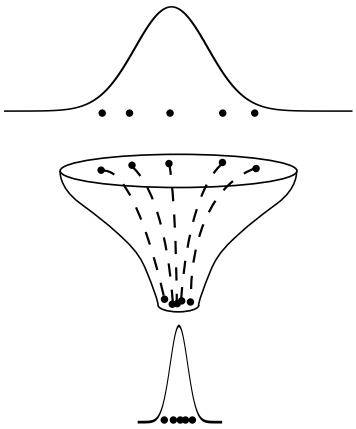


Figure 1.2 The execution of funnels reduces uncertainty which we estimate by simulating controller executions (dashed lines).

In Part II of the thesis, we will show assumptions that pertain to the manipulation planning problem that make reasoning about uncertainty tractable. The main contribution of Part II is a novel manipulation planning framework that exploits contact sensing to reduce uncertainty. The key assumption of our method is that contact sensing can completely reduce uncertainty of the robot relative to the environment in at least one dimension. Because of the efficient reasoning, the method finds robust strategies under significant uncertainty (C1). It simulates the uncertainty reduction of contact sensing, which allow it to combine contact sensing with proprioception (C2). However, as the simulation of funnels is quite complex it does not obey real time constraints.

1.3.3 Maintaining funnels in dynamic environments

Belief space planning is a feasible approach to find complex manipulation strategies. However, it assumes that the world is static and fully known. This contradicts our assumptions about mobile manipulators which should operate in unknown environments using information from their on-board sensors. To operate these robots in unknown environments, we need methods that, based on sensor data, can quickly find hybrid automata that execute basic motion skills. Due to the complexity of the planning problem, we cannot replan continuously, as that would make the robot too slow to react. Instead we will propose an **incremental planner** which continuously adapts a plan in reaction to incoming new sensor information. This plan can be seen as a covering of state space by funnels (Fig. 1.3), however as the environment continuously changes due to dynamic obstacles, the covering also changes continuously in response to sensor data.

The third contribution of this thesis is an incremental, reactive motion planning system for manipulation in dynamic environments. The method continuously generates hybrid automata that capture the possible motions. The method adapts the plan continuously in response to slow changes in the environment. We extend this method to reason explicitly about uncertainty from fast obstacles, such as moving people (C1). The method continuously integrates vision and laser scanner data as feedback sources on different levels of abstraction (C2). The system can perform long term reasoning under real-time constraints (C3) by factoring the planning into reactive control, continuous path adaptation, and global search under uncertainty.

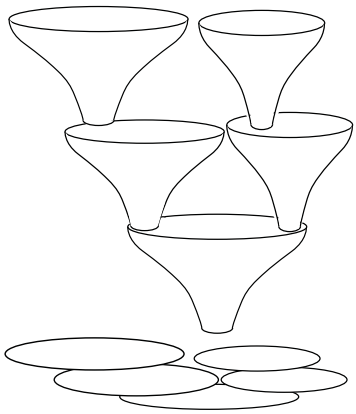


Figure 1.3 Covering the space with funnels.

1.4 Thesis structure

This thesis is split into three parts corresponding to the three main contributions: 1) The description and analysis of a feedback-based mobile manipulation system for logistics applications. 2) A sampling-based belief-space planning framework for contact-based motion 3) An incremental motion generation system for unknown, dynamic environments. Each part starts with a chapter reviewing the background. The later chapters give details for the contributions evaluate them experimentally.

Part I will introduce the feedback-based motion generation system in a real world manipulation tasks. **Chapter 2** will discuss the basics of motion control for robotic manipulators.

We will discuss how to robustly attain goals specified in terms of geometry or desired sensor signals. Based on these methods, in **Chapter 3** we will present and evaluate a feedback-based mobile manipulation system hat picks items from warehouse shelves. We evaluated the system in a competition: the Amazon Picking Challenge 2015. This allows us to compare the feedback-based approach to the state-of-the-art. In addition to presenting the specific system, the outcome of part I is a list of requirements for autonomous motion generation.

Part II will introduce planning methods that fulfill the requirements found in part I. To do so, **Chapter 4** discusses planning approaches from the literature that find feedback strategies using global search. We therefore review sampling-based motion planning for problems with and without uncertainty. We also discuss justified assumptions from the literature that simplify reasoning over uncertainty. Using these concepts, **Chapter 5** presents a novel sampling-based motion planner for manipulation. The planner finds sequences of actions that move the robot in free space but also deliberately seek contact to reduce uncertainty. In **Chapter 6**, we extend this planner to a contingent version that predicts the sensation of contact as measured by the robot and plans appropriate reactions to different signals.

Part III looks at motion generation from a systems point of view, taking all requirements for perception, uncertainty, and real-time constraints into account.**Chapter 7** first gives an overview about sensor-based motion generation architectures. We then discuss optimization-based methods which integrate perception and planning by continuously adapting motion plans to he changing environment. **Chapter 8** presents a novel incremental motion planning system. This system continuously integrates sensor data and maintains a global plan, while at the same time reasoning about the uncertain environment. **Chapter 9** contains preliminary results for a controller based on multi-modal feedback. Finally, in **Chapter 10** we summarize the contributions and hypothesize a general architecture for sensor-based motion generation systems.[2]

[2]This thesis contains three quotes and figures from Carroll (1865) which is in the public domain.

Part I

A case study for feedback-based mobile manipulation

The White Rabbit put on his spectacles. 'Where shall I begin, please your Majesty?' he asked. 'Begin at the beginning,' the King said gravely, 'and go on till you come to the end: then stop.'

Chapter 2

Fundamentals of motion control for mobile manipulation

In this chapter we will briefly introduce the reactive controllers to move a mobile manipulator. We will introduce methods from the literature to achieve user-specified motion tasks with mobile manipulators based on geometric and force feedback. We will also discuss how to assemble robust controllers using hybrid automata. For a more thorough introduction, we refer to robotics textbooks (Craig, 2005; Lynch and Park, 2017). The content of this chapter is unique to this thesis and not published before.

2.1 Configuration

We model a robot as a set of rigid bodies called **links** connected to each other with movable **joints**. For manipulators, some of these links, such as grippers or hands, are designed to apply forces to the environment. We call these links **end-effectors**. In this thesis, we only consider single-arm robots so it will suffice to model the robot as a sequence of joints and links with one end-effector at the end of the chain.The robots we consider in this thesis will be composed of rotational joints that rotate around one axis and translational joints that linearly move along an axis.

The minimum number of real values to specify the position of all the joints of a robot is called the robot's **degrees of freedom (dof)**. The position of each point on a robot with n degrees of freedom is fully specified by its n-dimensional **configuration** $q \in \mathcal{C}$, where $\mathcal{C} \subseteq \mathbb{R}^n$ is called the **configuration space** (Lozano-Pérez, 1983), or **joint space**. We assume the robot can always measure q using encoders that return the joint positions.

The robotic arms we consider in this thesis are a chain of seven rotational joints. They are mounted on a mobile platform which has three degrees of freedom: two translational degrees in the plane and one rotational degree of freedom. In this thesis we only consider **holonomic** bases that can move instantaneously in all three dimensions in the plane (examples for non-holonomic bases are cars which can not move sideways instantaneously). Holonomic bases can be modeled as a sequence of two translational joints and one rotational joint. A mobile manipulator now is a chain of the three mobile degrees of freedom and the seven degrees of freedom of the arm, leading to a single chain with a total of ten degrees of freedom.

In this thesis we also assume the robot is **fully-actuated** which means that each joint of the robot can be controlled individually. We also assume that the motors of each joint allow direct control of torques. We represent the control torques for all joints in a single vector $\Gamma \in \mathbb{R}^n$.

2.2 Feedback control

Feedback control methods rely on sensor data to generate motion. The basic idea of feedback control is to continuously compute an error between the current state and the desired state of the robot and compute a motor command to counteract the error. A simple feedback controller to reach a desired joint configuration $q_d \in \mathcal{C}$ is the proportional controller. This controller continuously reads the current configuration $q \in \mathcal{C}$, computes an error $e = q_d - q$, and then reduces the error by applying a proportional torque

$$\Gamma = K_p e \tag{2.1}$$

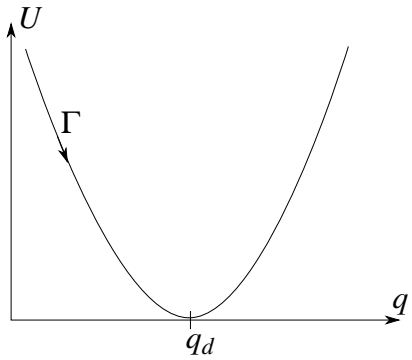


Figure 2.1 A 2D potential function with minimum at q_d. Gradient descent on U makes a robot converge to q_d.

The constant positive definite matrix $K_p \in \mathbb{R}^{n \times n}$ is called proportional gain and usually contains values on the diagonal that specify how strong each joint should react to an error. This controller is equivalent to performing gradient descent on an attractive potential function $U_{\text{att}} : \mathcal{C} \to \mathrm{R}$.

$$\Gamma = -\nabla U_{\text{att}}(q) \tag{2.2}$$

$$U_{\text{att}}(q) = \frac{1}{2} K_p e^{\intercal} e \tag{2.3}$$

$U_{\text{att}}(q)$ is shaped like a bowl with minimum at q_d. The control law will take any state in $\mathcal{C}$ and moves it towards q_d, like a funnel. When executing proportional control, the robot will oscillate around the minimum due to its kinetic energy. To prevent these oscillations we can add derivative error term to the controller that damps the system. This leads to proportional-derivative control (**PD-control**):

$$\Gamma = K_p e + K_d \dot{e} \tag{2.4}$$

with $\dot{e} = \dot{q}_d - \dot{q}$, and a constant $K_d \in \mathbb{R}^{n \times n}$ called derivative gain. This law can be understood as a linear spring with constant K_p that pulls the system to the desired state, and a damping factor K_d that prevents oscillations.

2.2.1 Potential fields

Motion generation methods should move to a goal while avoiding obstacles. The **artificial potential field** method (Khatib, 1985) combines the attractive potential field for the goal with a second repulsive potential field U_{rep} that keeps the robot away from obstacles.

$$U_{\text{rep}} = \begin{cases} \frac{1}{2}\nu\left(\frac{1}{d(q)} - \frac{1}{d_0}\right)^2, & \text{if } d(q) \leq d_0 \\ 0, & \text{else} \end{cases} \tag{2.5}$$

where $d : \mathcal{C} \to \mathbb{R}$ returns the distance from the robot to the closest obstacle, $\nu \in \mathbb{R}$ gives the strength of the repulsive force, and $d_0 \in \mathbb{R}$ is a minimum distance from which the repulsive force works. The robot can now move to the goal while avoiding obstacles by performing gradient descent on the combined potential

$$\Gamma = -\nabla(U_{\text{att}} + U_{\text{rep}}) \tag{2.6}$$

The artificial potential field is the prototype for reactive motion generation. The robot can compute F_{att} and F_{rep} continuously using only local sensor data, for example from laser scanner data. As the computations are easy to carry out, the robot can continuously compute the forces and react instantaneously to obstacles. Unfortunately, the potential field method is not the solution to motion generation. The reason for this are local minima (see Fig. 2.2). Local minima appear when attractive and repulsive forces cancel out, making the robot stop. This points to a general issue for local control methods: all methods that act based solely on *local* information can be subject to local minima. To handle local minima, we need potential fields that are globally convergent to the goal, which are called **navigation functions** (Rimon and Koditschek, 1992). Unfortunately, computing such functions for

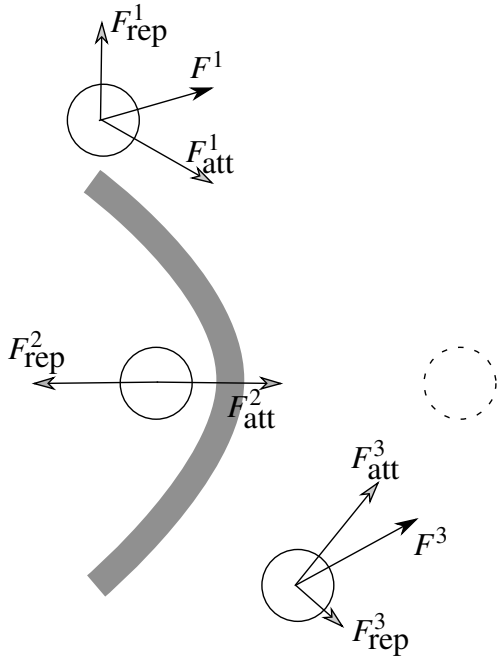


Figure 2.2 The forces F^1, F^2, F^3 computed by the artificial potential field method for a disc-shaped robot in three configurations. The obstacle is shown in grey and the goal is shown with dashed lines. The second robot is in a local minimum as repulsive and attractive forces cancel out, leading to $F^2 = 0$.

high-dimensional systems is intractable. The approach in this thesis is to sequence local controllers into global strategies without local minima.

2.3 Joint-space control

The motion of manipulators is strongly influenced by dynamic effects such as inertia and gravity. To control manipulators we cannot ignore these effects but must model them and integrate them into our controllers. The dynamics of a kinematic chain can be written as:

$$\Gamma = A(q)\ddot{q} + b(q,\dot{q}) + g(q) \tag{2.7}$$

where $A(q) \in \mathbb{R}^{n\times n}$ is called **joint-space inertia matrix**, or simply **mass matrix** and expresses the inertial properties of each link in joint space, $b(q,\dot{q}) \in \mathbb{R}^n$ captures the combined Coriolis and centrifugal forces, and $g(q) \in \mathbb{R}^n$ is the torque needed to compensate gravity. These properties depend on the current robot configuration and velocity. If each link's mass, center of mass, and inertia tensor are known, all quantities can be computed using efficient recursive Newton-Euler Inverse Dynamics methods (Featherstone, 2014).

We will describe with a PD controller how our system should behave without dynamics and then use (2.7) to modify the behavior to compensate all dynamic effects. In addition to desired joint position q_d and joint velocity $\dot{q}_d$, we now also assume a desired joint acceleration $\ddot{q}_d \in \mathbb{R}^n$. Together we then can specify a PD-controller that executes these nominal values as:

$$\ddot{q}_{\text{ref}} = \ddot{q}_d + K_p e + K_d \dot{e} \tag{2.8}$$

and insert this reference acceleration into the dynamics equation to obtain the **inverse dynamics controller**, or **computed torque controller**:

$$\Gamma = A(q)\Big(\ddot{q}_d + K_p(q_d - q) + K_d(\dot{q}_d - \dot{q})\Big) + b(q,\dot{q}) + g(q) \tag{2.9}$$

In the remainder of this thesis, whenever we mention a **joint space controller**, we use this law. In practice, to satisfy the robots actuation constraints, we often specify the desired behavior over time as a **trajectory** $\tau : [0,T] \to \mathcal{C}$, which is a function that returns the desired configuration the robot should reach at time step t. From the trajectory, we can compute the desired velocity as the first time derivatives $\dot{q}_d(t)$ and the desired acceleration as the second derivative $\ddot{q}_d(t)$ and insert them in the joint space controller. Trajectories can be computed in a multitude of ways, e.g. by using initial and final configuration of the robot, as well as maximum velocities and accelerations of the joints as constraints for spline interpolation (see Craig (2005) for details).

2.4 Task-space control

Many manipulation tasks can be specified as constrained motion. For example maintaining the orientation of a grasped glass of water or writing with a pen on a sheet of paper. These tasks have in common that they constrain certain types of motion around a specific point on the robot. E.g. To keep the glass upright, we need constrain two dimensions of hand orientation. To write with a pen we need to constrain the three dimensional position of the tip of the pen.

The introduced joint space controllers do not directly allow to execute these tasks as we cannot express the imposed constraints easily in configuration space. In this section, we will give control laws to control the position, orientation, and force of parts of the robot. We call the point we wish to specify our constraints on the **operational point**. For a given task, the associated **task space** $\mathcal{T}$ is the natural space the task is expressed in. For positioning the tip of the pen (ignoring the orientation), we would choose $\mathcal{T} = \mathbb{R}^3$. For maintaining the

orientation, we would choose $\mathcal{T} = SO(3)$ – the three-dimensional space of orientations of a rigid body. We call the minimum number of dimensions needed to express a task the **task dimensionality** and denote it with m. If the number of degrees of freedom n of a robot is higher than the task dimensionality ($n > m$) the robot is **redundant**. Mobile manipulators are redundant robots. They have ten degrees of freedom but only six dimensions are needed to fully specify position and orientation of their end-effector. Redundancy implies that a robot can fulfill a given task in a multitude of ways.

2.4.1 Forward kinematics

Forward kinematics is the mapping from configuration space to task space $f : \mathcal{Q} \to \mathcal{T}$, i.e. for a robot in configuration q, $f(q) = x$ would return a parameter vector $x \in \mathcal{T}$ of the current task, e.g. the position of the finger tip.

To compute the forward kinematics, we use homogeneous transforms to compute the positions of all links of the robot. We denote with $T_i^j(q_i) \in SE(3)$ the homogeneous transform describing the relative pose of robot links i and j. This pose depends only on q_i – the position of the i-th joint. We can compute the position of link k of the robot (relative to the base link 0) T_0^k by multiplying the relative transforms of all links of the robot along the kinematic chain.

$$T_0^k(q) = T_0^1(q^{(0)})T_1^2(q^{(1)})\cdots T_n^k(q^{(n)}) \tag{2.10}$$

Where $q^{(i)}$ refers to the i-th entry of q. We assume we can extract the current task space coordinates $x \in \mathcal{T}$ of a task specified relative to link k of the robot $T_0^k(q)$ with a given function $g : SE(3) \to \mathcal{T}$. Together, we can compute the forward kinematics as

$$f(q) = g(T_0^i(q)) \tag{2.11}$$

2.4.2 Jacobian

Inverse kinematics now asks for the inverse mapping: for given desired task coordinates $x_d \in \mathcal{T}$, find a configuration q_d so that $f(q_d) = x_d$. There is no general solution to the inverse kinematics problem. Not all tasks x_d are actually reachable and for reachable tasks there might be multiple solutions, for example due to redundancy.

To circumvent these issues, we do not compute f^{-1} in closed form but rather use a feedback controller that regulates an error in task space. For a robot at a given configuration q_0, with $x = f(q_0)$, we compute an error e_x from the current task coordinates x to the desired task coordinates x_d. This error function is dependent on the chosen task-space

parametrization. We can then use a feedback controller to reduce this error. To do so we use the **Jacobian** $J(q_0) \in \mathbb{R}^{m \times n}$ which is the matrix of partial differentials $J(q_0) := \frac{\partial f}{\partial q}\Big|_{q_0}$ of f at configuration q_0. The Jacobian captures the relation between joint space velocities and task space velocities:

$$\dot{x} = J(q)\dot{q} \tag{2.12}$$

For brevity, we will from now on drop the index and write short J for $J(q)$. To map from task space velocity to joint space velocity, we can invert the, generally non-square, Jacobian J using a Pseudoinverse J^+. Together, this gives us a desired velocity

$$\dot{q}_d = J^+(K_{x,p}e_x) \tag{2.13}$$

where $K_{x,p} \in \mathbb{R}^{m \times m}$ is a proportional gain in task space. By integrating and differentiating $\dot{q}_d$ we can compute desired positions q_d and accelerations $\ddot{q}_d$, respectively, and use the values $(q_d, \dot{q}_d, \ddot{q}_d)$ as input for a joint space controller (2.9). We call this controller the **velocity-based task-space controller**. A drawback of this method is that velocity-based controllers do not allow to specify damping behavior, as this would require controlling accelerations. To control accelerations instead, we can use the relation $\ddot{x} = \frac{d}{dt}J(q)\dot{q} = J\ddot{q} + \dot{J}\dot{q}$ to compute

$$\ddot{q}_d = J^+(K_{x,p}e_x + K_{x,d}\dot{e}_x - \dot{J}\dot{q}) \tag{2.14}$$

this **acceleration-based task-space controller** controller adds a damping term with derivative gain $K_{x,d} \in \mathbb{R}^{m \times m}$.

For all these control laws, the choice of pseudoinverse is important for the behavior. For redundant robots such as mobile manipulators, $\dot{x} = J\dot{q}$ is an underdetermined equation system, which means that infinitely many joint space velocities result in the same task-space velocity. In this case, there exist infinitely many different pseudoinverses and the choice of pseudoinverse dictates how the robot will distribute the required task space velocities along the kinematic chain. Therefore often it is useful to chose a weighted pseudoinverse:

$$J_W^+ = W^{-1}J^\intercal(JW^{-1}J^\intercal)^{-1} \tag{2.15}$$

Here, the symmetric positive definite matrix $W \in \mathbb{R}^{n \times n}$ can be used to weigh individual joints, i.e. picking a relatively high value for entry $W_{i,i}$ will result in joint i moving less than the other joints. The **dynamically consistent pseudoinverse** $\bar{J}$ (Khatib, 1987) is a weighted pseudoinverse using the joint space inertia matrix A as weighting. The special property of this inverse is to minimize the instantaneous kinetic energy, making heavy joints move less.

2.4.3 Operational space control

The previous methods were feedback controllers based on velocities and accelerations. Manipulators must establish and maintain contact, hence they must be able to control forces. For a given task space $\mathcal{T}$, we denote with $F \in \mathbb{R}^m$ the generalized forces at the operational point. These forces act in the task space dimensions, i.e. if $\mathcal{T} = \mathbb{R}^3$, F would be a three-dimensional force vector. If the task also contains rotations, the associated vector F would also contain torques around the rotation axes.

The **operational space control** method (Khatib, 1987) computes F to control the robot. We will first present operational space control for non-redundant robots ($m = n$) and generalize it to redundant robots later. Operational space control is based on the following relation between torques Γ and the force F for a task with Jacobian J.

$$\Gamma = J^\intercal F \tag{2.16}$$

This equation cannot be used directly to compute command torques, as dynamic effects must be compensated. The key insight of operational space control is that the robot dynamics (2.7) can be projected into the task space, giving the **end-effector equations of motion**:

$$F = \Lambda(x)\ddot{x} + \mu(x,\dot{x}) + p(x) \tag{2.17}$$

$\Lambda \in \mathbb{R}^{m\times m}$ given by $\Lambda = J^{-\intercal} A J^{-1}$ is the task-space inertia matrix, $\mu \in \mathbb{R}^m$ are the Coriolis and Centrifugal forces expressed in task space and $p \in \mathbb{R}^m$, $p = J^{-\intercal} g$ the gravitational forces expressed in task space.[1]

This equation *dynamically decouples* the end-effector, which means that all dynamical effects are compensated in task space. This allows to specify a desired behavior for the end-effector as if it was a free-flying unit-mass rigid body. For example, to move the end-effector along a task space specific trajectory we can define a PD-controller in task space:

$$\ddot{x}_{\text{ref}} = \ddot{x}_d + K_p e_x + K_d \dot{e}_x \tag{2.18}$$

by inserting $\ddot{x}_{\text{ref}}$ as $\ddot{x}$ into (2.17), we can compensate all dynamic effects, resulting in the end-effector force F_d we need to apply to move the end-effector on the trajectory. We then transform this force into control torques using $\Gamma = J^T F_d$.

[1] We write $J^{-\intercal}$ as shorthand notation for $(J^\intercal)^{-1}$. See Khatib (1987) for derivation and details on the μ-term

2.4.4 Force feedback

Controlled manipulation in contact is facilitated by using a force-torque sensor mounted on the robot as feedback source. Using operational space control, it is straightforward to close a loop around measured force. **Impedance control** makes the end-effector interact with the environment like a spring-mass damper system with user-specified gains that specify the mass $M \in \mathbb{R}^{m\times m}$, damping $D \in \mathbb{R}^{m\times m}$, and spring constant $K \in \mathbb{R}^{m\times m}$.

$$\ddot{x}_d = M^{-1}(D\dot{e}_x + Ke_x + F_m) \tag{2.19}$$

where F_m is the measurement of the force-torque sensor transformed into the end-effector frame.

2.4.5 Multi-objective control

The redundancy of mobile manipulators allows them to execute multiple tasks at once, for example positioning the end-effector while placing the mobile base away from obstacles. **Multi-priority control** imposes a hierarchy between different controllers running at the same time. Higher priority controllers are executed using all available degrees of freedom while lower-priority controllers only operate using the remaining degree of freedom, without interfering with high-priority tasks. The idea of multi-priority control schemes is to compute projector matrices N that project commands into the space where they do not affect higher priority tasks. We call N the **nullspace projector** of the task and for a task with Jacobian J, we can compute it by $N = (I - J^{+}J)$, where I is the $n \times n$ identity matrix. Now, for example, we can use this projector to combine an end-effector positioning task with an additional joint velocity $\dot{q}_0 \in \mathbb{R}^n$ which should be executed using the remaining degrees of freedom:

$$\dot{q} = J^{+}\dot{x} + (I - J^{+}J)\dot{q}_0 \tag{2.20}$$

Nullspace projections can be cascaded to satisfy multiple tasks at once. We now assume a list of n tasks, each given by reference velocity x_i and Jacobian J_i, $i = 0\ldots,n$.

$$\dot{q} = {J_0}^{+}\dot{x}_0 + N_0\left[J_1\dot{x}_1 + N_1\left[\ldots + N_{n-1}\left[{J_n}^{+}\dot{x}_n\right]\ldots\right]\right] \tag{2.21}$$

Here, task 0 is executed using all available degrees of freedom. Tasks with higher indices are executed in the combined nullspace of all other tasks with lower indices.

Nullspace projections can also be computed in the operational space formalism. In the velocity-based control law, the choice of pseudoinverse is up to the control designer. However, in operational space control, the subtasks should generate no velocity and additionally no

accelerations that interfere with the higher priority tasks. This is only the case for the dynamically consistent pseudoinverse (Khatib, 1987) $\bar{J}$ from Sec. 2.4.2. To control a primary task and a secondary torque, we can compute

$$\Gamma = J^\mathsf{T} F + (I - J^\mathsf{T} \bar{J}^\mathsf{T}) \Gamma_0 \tag{2.22}$$

By cascading nullspace projections, multiple tasks can be controlled in the operational space framework too. This leads to the **whole body control** framework (Sentis and Khatib, 2005):

$$\Gamma = J_0^\mathsf{T} F_0 + N_0^\mathsf{T} \left[J_1^\mathsf{T} F_1 + N_1^\mathsf{T} \left[\ldots + N_{\text{n-1}}^\mathsf{T} \left[J_\text{n}^\mathsf{T} F_\text{n} \right] \ldots \right] \right] \tag{2.23}$$

We use a shorter notation for the hierarchical execution of tasks (Huber and Grupen, 1997):

$$\Gamma = \Phi_n \triangleleft \ldots \triangleleft \Phi_1 \triangleleft \Phi_0 \tag{2.24}$$

Here a task Φ_i is a tuple (x_i, J_i, U_i) combination task space coordinates $x_i \in \mathcal{T}_i$, Jacobian J_i, and potential function U_i. Some often tasks are given in Table 2.1. The $\triangleleft$-sign between denotes that Φ_{i+1} is executed in the nullspace of Φ_i. Using this framework we can now easily specify multiple tasks for the robot to achieve at the same time.

symbol	description	dim	J	potential
Φ_{task}	end-effector position	6	manipulator Jacobian	U_{att}
Φ_{obs}	obstacle avoidance	1	J projected on normal of closest obst.	U_{rep}
Φ_{pos}	arm posture	7	unit matrix with 0 on base joint indices	U_{att}

Table 2.1 Useful task potentials in whole body control. See Sentis (2007) for implementation details.

A typical control structure using these potential is $\Phi_{\text{pos}} \triangleleft \Phi_{\text{obs}} \triangleleft \Phi_{\text{task}}$. This implements a task on the robot's end-effector, while the remaining degrees of freedom avoid collisions with the mobile base. The posture specifies the remaining degrees of freedom and is chosen to keep the robot away from joint limits.

2.4.6 Practical considerations

On an ideal robot, whole body control satisfies all requirements for manipulation in a clean formalism. However, the practical performance of control laws can differ a lot on different robots. To control positions, higher-order task space control laws based on force or acceleration often perform worse than simple velocity-based control schemes (Nakanishi

et al., 2008). The reason for the drop in performance is that the higher-order controllers rely much more on accurate dynamic models. In operational space control, for example, the inertia matrix and its inverse appears in many places due to the use of the dynamically consistent pseudoinverse. Uncertainties in the inertia matrix are critical as it is often ill-conditioned (Featherstone, 2004). If the inertia matrix is not modeled well, or if there are a lot of non-linear effects in the actuation (such as static friction in the joints), errors will accumulate and make the controller unstable.

Another example for non-linear effects degrading performance from our practical experiments is the wrist of the Barrett WAM robot. The seventh joint of the WAM manipulator is actuated directly by a relatively weak motor compared to the rest of the arm. Consequently, the encoder readings are much more noisy and the last joint has a high amount of static non-linear friction. To overcome this static friction, an operational space controller generates a high task-space error that influences all joints. However, this increases the total torque which makes the lower, much stronger, joints unstable. Often, simpler control laws can overcome these issues. We observed that the WAM is much more stable when controlled with a velocity-based control law where an additional integral term counters the static friction of the wrist joint without influencing other joints as much. Therefore, we use the following mixed control law which is a combination of velocity-based task space control and an additional feedback term outside of the dynamics compensation (Righetti et al., 2014):

$$\begin{aligned} \dot{q}_d &= J^+(\dot{x} + K_{x,p}(x_d - x)) \\ \Gamma &= A(q)\ddot{q}_d + b(q,\dot{q}) + g(q) + K_{q,d}(\dot{q}_d - \dot{q}) + K_{q,p}(q_d - q) \end{aligned} \tag{2.25}$$

$\ddot{q}_d$ and q_d are computed by differentiating or integrating $\dot{q}_d$, respectively.

2.5 Hybrid automata

Complex behavior is combined of different modes of operation. As all local controllers are subject to local minima, we need a way to compose them into long term plans. Not all sensor modalities can easily be used in feedback controllers. The reason is that much of the useful feedback in the world is not continuous, but discrete. Examples for these discrete feedback events are contact sensors detecting that the robot makes or breaks contact, or a visual processing algorithm method detecting an object. This requires to combine the continuous feedback of controllers with discrete feedback of sensors. The **hybrid automaton** (Henzinger, 2000) is a formalism to describe the combination of discrete and continuous feedback.

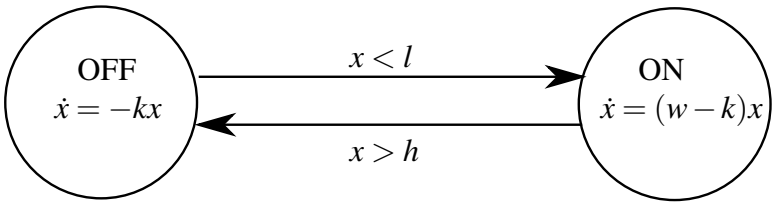


Figure 2.3 A hybrid automaton for a heating system. In the OFF state the room temperature x decreases exponentially until it is below a threshold l. This activates a mode switch to the ON state, increasing the temperature exponentially, until an upper threshold h is reached.

Formally, a hybrid automaton is a graph $G = (V, E)$ where the nodes V (called **control modes**) are continuous feedback controllers and edges E (called **control switches**) specify the transition rules between two controllers. The transition is defined by a discrete sensor feedback event. Fig. 2.3 shows a simple hybrid automaton with two states modeling a heating system controlling the temperature x. [2]

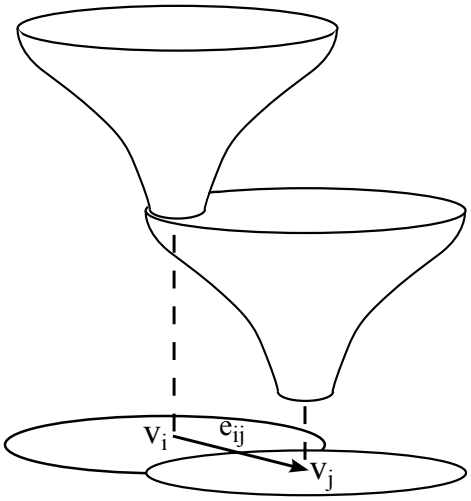


Figure 2.4 Two control modes v_i and v_j of a hybrid automaton and a control switch e_{ij} connecting them: the funnels visualize the controllers' domain of attraction; the exit of the funnel is the controller's attractor.

The hybrid automaton describes the sequential execution of funnels (see Fig. 2.4). Each mode is a funnel, moving the robot. A control switch signals the convergence of the first controller which moves the system into the second funnel. Hybrid automata have several advantages, in particular for robotic manipulation:

[2] The combination of continuous and discrete feedback makes analysis and verification of hybrid automata non-trivial. For example, a given hybrid automaton can transition infinitely often between modes in finite time (a property called *Zeno*), which makes it effectively non-implementable. In this thesis, we will use hybrid automata only as a data structure but mostly ignore the control-theoretic framework behind.

- **Modularity:** The modes and switches are well-defined interfaces and their interactions are easy to specify. The simple structure allows to compose complex motion intuitively, i.e. the robot could operate on a nominal path. However, in case of unexpected sensor events it will switch to other branches in the graph that handle the failure case safely.
- **Heterogeneous control:** A hybrid automaton can easily switch between joint space control to task space control. This allows to specify subtasks in their natural possible space, i.e. a standard grasping hybrid automaton moves the robot with a joint space controller to an initial configuration, from there visually servos to approach an object, then applies constant force with the end-effector while closing the hand for a grasp.

In conclusion, we discussed feedback control methods for mobile manipulators based in joint space, task space, and feedback from force and vision sensors. We introduced the hybrid automaton framework to combine continuous and discrete feedback into complex strategies. In the next section, we will show a manipulation system based on manually defined hybrid automata. In all later parts of the thesis we will discuss methods that generate such hybrid automata autonomously, just from a description of the robot and the environment.

Chapter 3

A mobile manipulation approach to the Amazon Picking Challenge 2015

The **Amazon Picking Challenge** (APC) tested the ability of robotic systems to fulfill a fictitious order by autonomously picking the ordered items from a warehouse shelf (Fig. 3.1). In this chapter, we provide a detailed technical description and experimental evaluation of this system. The challenge is an instance of the mobile manipulation problem including the challenges from Section 1.1: it requires to robustly solve tasks under uncertainty (C1), as picking objects and their positions are not known a priori, combine sensor data from multiple modalities (C2), as both visual and tactile perception are required, and it must operate in a limited time frame (C3).

The system presented here was able to solve the task and outperformed the system of 25 contenders on the challenge, winning by a significant margin. Interestingly, our system diverged from the motion generation approach taken by most other challenge participants. Compared to other contenders in the challenge, who relied to a large degree on planning methods, we picked a mostly feedback-based architecture.

The goal of this chapter is twofold: first, we describe and evaluate a real world manipulation system based on hybrid automata. Second, we extract insights from our system that should lead to more robust motion generation. We will collect a list of requirements for feedback-based manipulation system to operate robustly in the real world. In parts II and III, we show how to fulfill these requirements in motion generation methods that can generalize to a wide range of tasks beyond the Amazon Picking Challenge.

In Section 3.2, we describe the implementation of our system. Section 3.3 then describes the performance of the system, both in the competition and in additional experiments conducted in our lab. We then discuss in Section 3.4 the main difference of our motion generation approach relative to other contenders. Finally, in Section 3.5, we collect a list of

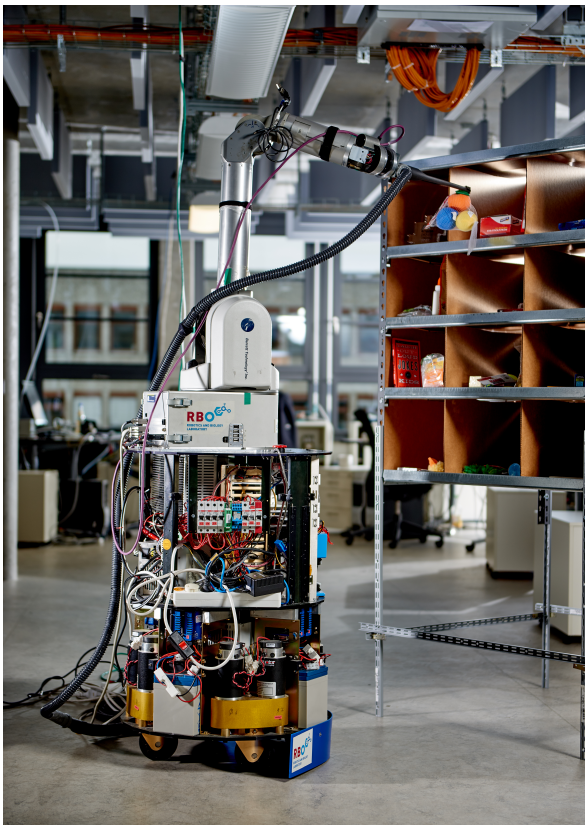


Figure 3.1 Our mobile manipulator picking an object from the Amazon Picking Challenge shelf ©*TU Berlin/PR/Philipp Arnoldt*

general requirements for motion generation methods in mobile manipulation, which will be tackled with all further methods in this thesis.

Sections 3.1, 3.2, 3.3, and 3.4 are extracted from a previous publication (Eppner et al., 2018) although modified with focus on the motion generation aspects. The publication contains equal contribution by six other authors. The described system was a team effort, combining many people's work. My unique contribution to the system is the implementation of the hybrid automaton-based motion generation and performing the experimental evaluation. Section 3.5 is not published anywhere else.

3.1 The Amazon Picking Challenge 2015

In the Amazon Picking Challenge, the task consists of autonomously picking twelve out of 25 objects (Fig. 3.2) from a warehouse shelf and placing them into a storage container (Fig. 3.1) within 20 minutes. The robot has knowledge of which objects are contained in each of the shelf's twelve bins, but not of their exact arrangement inside the bin.

Figure 3.2 The 25 objects from the Amazon Picking Challenge

Evaluation criteria For each successfully picked target object, the robot receives 10, 15, or 20 points, depending on how many additional objects were in the same bin (between none and three). Objects that were considered difficult to pick obtained up to three bonus points. Picking the wrong object results in -12 points.

Objects The 25 competition objects varied widely in size and appearance (Fig. 3.2). Objects ranged from small spark plugs to bulky boxes, from hard cases to soft plush toys, and from loose objects to those wrapped in plastic. This variety presented a challenge for grasping and perception.

Environment The items were placed in the twelve different bins of the shelf. The robot was allowed to operate in a $2\,\text{m} \times 2\,\text{m}$ area in front of this shelf. The bins had a rather small opening ($21\,\text{cm} \times 28\,\text{cm}$) but extend quite far to the back ($43\,\text{cm}$). The small opening was problematic for many state-of-the-art robotic hands as they barely fit into the bins, once the shelf is stuffed. As a result, motion capabilities within the bins were greatly reduced. Since objects could be surrounded by other objects or could lie close to the side wall of the bin, grasping opportunities were restricted. Moreover, the bins had a lid in the front, providing an opportunity for the robot or the grasped object to get caught while retracting from the bin.

The shelf also posed significant challenges for perception. Due to the narrow bins, objects were often partially obstructed. The reflective metal floor of the shelf rendered color and depth information unreliable. These perceptual challenges made vision unreliable and led to uncertainty in the estimated object positions which must be handled by motion generation.

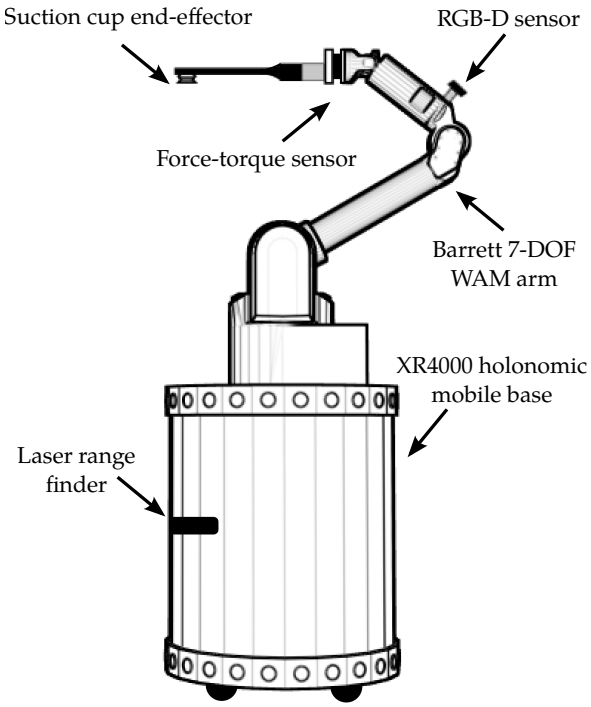


Figure 3.3 Schematic drawing of the WAM+XR4000 robot used in the Amazon Picking Challenge

3.2 Technical system description

We now describe the hardware and algorithmic components of our solution.

3.2.1 Hardware components

Our robot is composed of a mobile manipulator, a custom suction-cup gripper, and various on-board sensors (Fig. 3.3).

Mobile manipulator

We use a seven degree-of-freedom Barrett WAM mounted on a Nomadic XR4000 mobile base. Four caster wheels in the base with eight actuatable axes provide holonomic motion capabilities (Holmberg and Khatib, 2000). The entire kinematic chain possesses ten holonomic degrees of freedom.

The inclusion of (holonomic) mobility—a choice of embodiment that set our solution apart from most other participants in the Amazon Picking Challenge—greatly facilitated the generation of motion. The ability to reposition the base enabled the arm to easily reach inside all of the bins (Fig. 3.4), leading to simpler arm motion than with a static base.

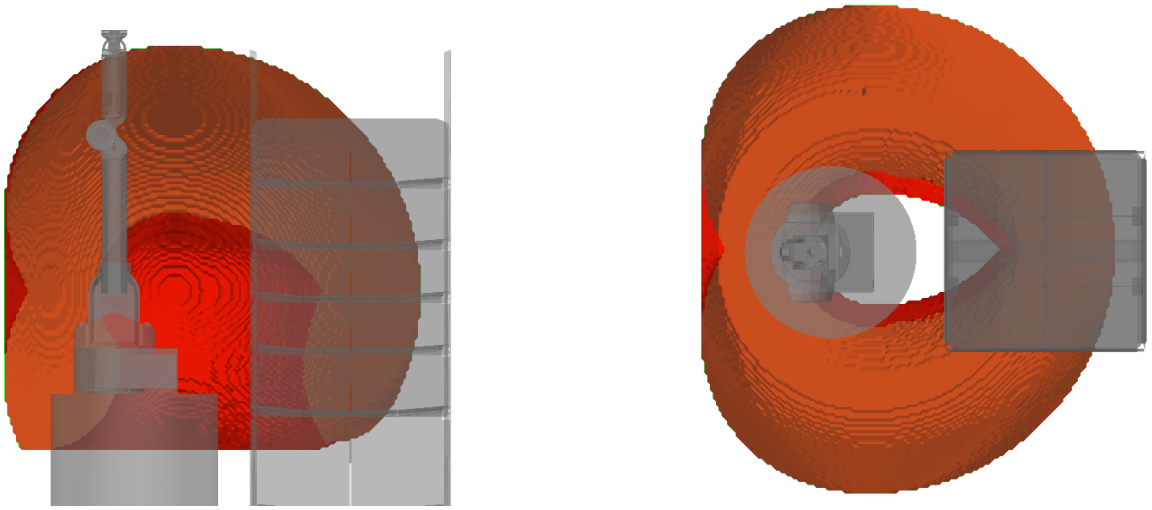


Figure 3.4 The workspace of our 7-dof arm next to the competition shelf, shown from the side and top-down. The workspace volume is the set of all reachable positions of the end-effector in which it points forward. To facilitate viewing, the workspace volume is cut open (the outer surface is red, while the inner one is orange). The shape of the workspace indicates that the addition of a mobile base significantly relaxes the kinematic constraints that are present when reaching in one of the bins.

End-effector

Our end-effector consists of a modified crevice nozzle from a vacuum cleaner with a suction cup mounted at its tip (Fig. 3.5). An off-the-shelf vacuum cleaner, operating at $250W$, generates sufficient air flow (and suction in the case of a tight fit between suction cup and object) to lift up to 1.5 kg.

This simple end-effector can reliably pick all but one of the challenge objects (the pencil cup) from the narrow, deep shelf bins. Grasping success is rather insensitive to the exact contact location with the object, leading to reduced requirements for perception. At the same time, the end-effector's thin shape reduces the need for complex collision avoidance or pre-grasp object manipulation, as it easily fits in between objects, pushing them aside if necessary.

Sensors

All sensors are on-board: a base-mounted SICK LMS-200 laser range finder, an Asus XTion Pro RGB-D camera on the robot's forearm, a six-axis ATI Gamma force-torque sensor on the wrist, and a pressure sensor inside the tube connecting the vacuum cleaner to the end-effector. These sensors provide feedback to monitor and guide task execution and to reduce uncertainty. The robot uses the laser range finder for particle-filter-based localization with respect to a

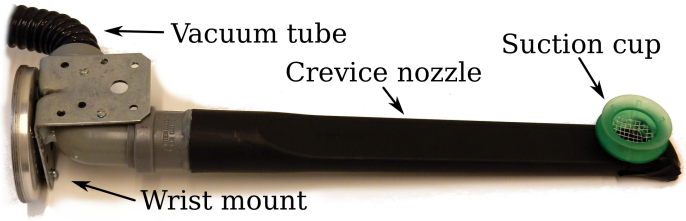


Figure 3.5 The suction cup end-effector of the WAM+XR4000 manipulator used to pick objects

2-D map of the shelf's base, it uses the force-torque sensor to guide the end-effector motion, and the pressure sensor to detect picking failure.

3.2.2 Object recognition

To successfully pick specific objects, the robot must recognize and locate the target object. Our system captures images with an RGB-D camera mounted on the robot's forearm and return bounding boxes for all segmented objects. The object recognition is not part of this thesis and is explained in detail in (Jonschkowski et al., 2016).

3.2.3 Grasping

The grasp planner initially sorts the list of target items based on heuristics, in decreasing order of expected number of points to be gained. The robot then chooses between two picking strategies:

- *Top-down:* The end-effector is positioned above the object, moves downward until a force measurement signals contact with the object and activates suction. The protruding metal edges on the left- and rightmost bins of the shelf are avoided by slightly rotating the end-effector around the vertical axis in these bins (Fig. 3.6, left).
- *Side:* The end-effector approaches the object from the left or right. The robot executes a force-guarded motion orthogonal to the bin wall, pushing the object to the side until it senses contact, and then activates suction (Fig. 3.6, right).

Both grasping primitives deliberately move the end-effector *into* the object and push it against the floor, walls, or other objects. Aligning the objects this way simplifies the picking action. This is an example of exploiting the environment to guide manipulation (Eppner and Brock, 2015) using haptic feedback.

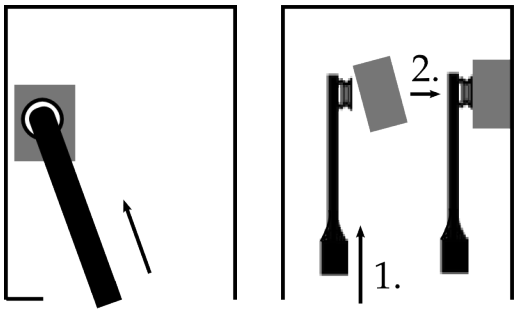


Figure 3.6 Top-down view of then end-effector picking and object (grey) from the bin: *left:* Top down picking; The end-effector approaches the object from the top, avoiding the left frontal side lip. *right:* Side picking. First the robot moves the end-effector next to the object. Second, it aligns the object with the wall by pushing.

To select one of the two primitives, we developed a scoring method that estimates their chance of success. The score is based on *(i)* how well the given side of the object can be grasped and *(ii)* the amount of free space to bring the end-effector into position. For this, the robot determines the orientation of objects by estimating their bounding box. This is important, for example, in the case of books which must be picked up from the front or back cover.

3.2.4 Motion generation

The execution of picking motions is realized with continuous feedback controllers. We transition between these controllers based on discrete sensor events. We used hybrid automata (Sec. 2.5) to specify this behaviour. States in a hybrid automaton correspond to one or several feedback controllers, while state transitions are triggered by sensor events. For each object, we generate and execute a hybrid automaton that implements the selected picking primitive (see example in Fig. 3.7). The automaton describes the actions for a successful grasp but also contains countermeasures for common failures. The resulting hybrid automaton consists of 26 states and 50 transitions. 34 of these transitions just deal with error handling. Error handling occurs in response to sensor input. For example, if the robot detects an undesired contact with the shelf, it retracts from the bin and reattempts to pick the object later.

The hybrid automaton not only enables us to efficiently exploit feedback but also to compose motions defined in joint space as well as in task space. We implemented the following controllers:

- joint space control (2.9) involving base and arm to move precisely and repeatably into view position

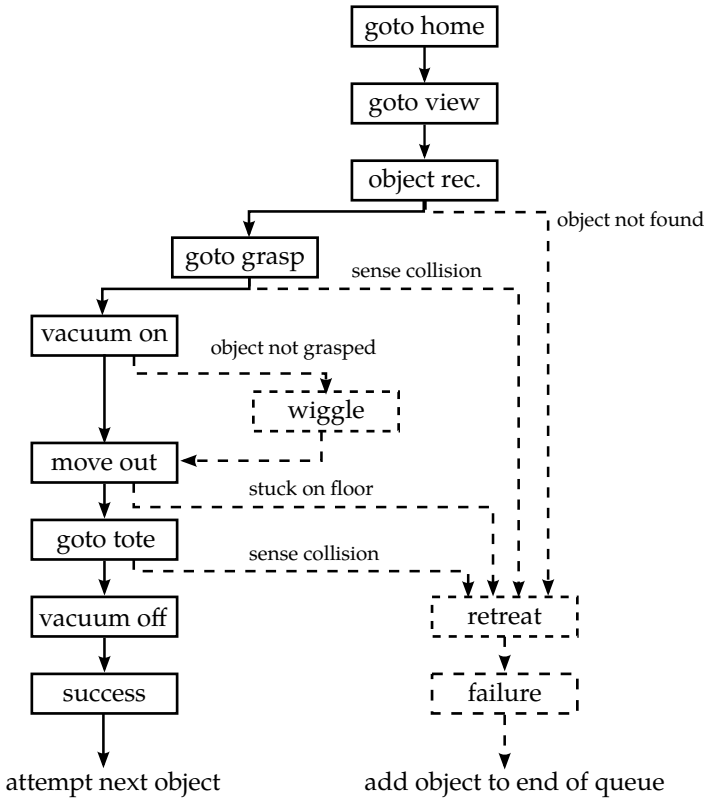


Figure 3.7 Simplified hybrid automaton for one picking primitive. Nodes are controllers and edges sensor-based transitions. The dashed nodes and edges handle failures during execution.

- task space control (2.25) involving base and arm ($\Phi_{\text{pos}} \triangleleft \Phi_{\text{obs}} \triangleleft \Phi_{\text{task}}$) for reaching in and out of the shelf, and moving to the drop-off location)
- task-space control (2.25) on the arm ($\Phi_{\text{pos}} \triangleleft \Phi_{\text{task}}$) for fine-positioning inside the shelf.

3.3 Evaluation

The Amazon Picking Challenge provides an in-depth evaluation of our system, comparing its performance with 25 teams from around the world in real-world conditions. We complement the competition results with nine additional experiments, using all object configurations used during the competition. We also discuss capabilities and limitations of our system.

3.3.1 Quantitative evaluation

Competition result

In our competition performance, we scored 148 out of 190 points (77.8%). We attempted to pick all twelve objects and were successful for ten. An average picking motion took 87 seconds. This allowed us to maximally attempt 14 picks within the 20 minutes of challenge duration. The scored points put us well above the competitors who scored 88 points for second, and 35 points for third place. A video of complete run can be found at https://www.youtube.com/watch?v=DuFtwpxQnFI.

Fig. 3.8 shows the competition setup from the robot's perspective including the estimated segments of the target objects. Our system picked the wrong object in bin F (row 2, column 3) and was not able to retrieve the bulky box from bin K (row 4, column 2) (more details in Sec. 3.3.3).

Post-hoc evaluation

To gather more data about our system's performance, we reenacted all five shelf configurations that were used in the challenge. We performed two trials per setup, using the robot system from the challenge without modifications. The total testing time was 200 minutes in which the robot picked 95 objects, of which 85 were target objects. We reached an average point count of 117.6 ($\sigma = 29.2$) which is 62.5% of all available points. This shows that the competition run was on the upper end of the system's capabilities. Still, out of the ten trials, only one (72 points) would have lead to our team placing second.

3.3.2 System capabilities

Object type: Our robot can pick 24 of the 25 objects, irrespective of whether they are soft, rigid, heavy, or light. It cannot pick the pencil cup, as its meshed structure foils the suction-based gripper (Fig. 3.9d, Sec. 3.3.3).
Object placement: The system is able to execute all grasp types in all bins and can even pick up objects close to the back wall.
Lighting conditions: The object recognition is invariant to extreme lighting condition variations, such as the ones encountered during the competition.
Shelf pose: The system can handle displacements of the shelf with continuous feedback from localization.
Shelf collision: Collisions with the shelf occurred during the tests but they do not lead to system failure.

Figure 3.8 Target objects and segmentation results during the APC run; the blue line outlines the segments returned by our method.

Long-term operation: During the 200 minutes of running time, we did not encounter a single total system failure, although we encountered several unexpected events in the experiments (see next section). We attribute this to the many failure cases covered by the hybrid automaton (Fig. 3.7).

3.3.3 System limitations

From the 120 attempted picks, the system picked ten wrong objects and failed to pick 25 objects. Table 3.1 shows the six failure categories. We will discuss these failures in detail and find that most failure cases can be addressed by inserting more feedback into the system.

(a) (b) (c) (d)

Figure 3.9 Failure cases of our system: (a) Wrong object boundary (b) Mistaking part of another object (right) for the target object (left) (c) The plush balls (right) were picked instead of the small spark plug (middle). (d) The pencil cup cannot be picked up with our end-effector.

85	successful picks
13	object recognition failures
9	bulky objects stuck at removal
9	missing small objects due to end-effector imprecision
2	displacing objects during approach
2	meshed pencil cup (cannot be picked with our end-effector)

Table 3.1 Failure cases for 120 picking attempts

Object recognition failures

Perception was one of the main challenges in the competition. We attribute 13 failed picking attempts to the object recognition pipeline. These failures occur when vision fails to discriminate objects in the target bin, resulting in wrong object boundaries (Fig. 3.9a) or mistaking another object for the target object (Fig. 3.9b). Consequently, the robot sometimes chooses the wrong pre-pick pose or the wrong picking primitive. We believe that object recognition can be improved by adding more sources of feedback. For example, by weighing objects or visually inspecting them after the pick.

Bulky objects stuck at removal

Eight scenarios contained a large box which could only be removed from the bin by tilting it. Our system failed on all eight attempts. The long bottle brush also got stuck once on the shelf lip and dropped in the process of removal. One way to address these failures is by using additional (motion) planning (chapter. 4). Planning would allow us to reason how to reorient objects to remove them from the bin.

Small objects

Out of ten attempts, the robot failed nine times attempting to pick up the small spark plugs. In the competition run, the robot even picked up a non-target object instead (Fig. 3.9c). These failures result from the fact that the reaching movement is executed open-loop, accumulating a significant error in forward kinematics of the arm, resulting in a pose error of up to 1 cm. This can be addressed easily by adding more feedback, for example by using visual servoing.

Displacing objects

In five out of ten attempts, the robot toppled over the glue bottle. The bottle then required a reattempt from the top. In two cases the robot did not have enough time for a reattempt and lost points. As before, this failure case can be alleviated by additional feedback; the robot could detect the tumble and immediately change the strategy. Alternatively, a planner could reason about the tipping point of objects to pick a more stable approach direction for the pick.

Pencil cup

The meshed metal pencil cup (Fig. 3.9d) does not have enough solid surface to pick it with suction. This failure mode shows a limitation of our chosen embodiment and is outside of the hardware's capabilities.

3.4 Comparison to other systems

We now want to analyze our and other systems on the spectrum of feedback and planning approaches.

Our design choice on the spectrum

Our system relies on very simple planning. We use on-line grasp approach planning and execute the motions using pre-defined, feedback-guided motion primitives, thus avoiding configuration-space motion planning altogether (Sec. 3.2.4). Feedback control is so successful in the Amazon Picking Challenge setting because the task only requires a limited range of motions, and the shelf provides plenty of contact surfaces to generate useful feedback. However, our evaluation (Sec. 3.3.3) shows that some shortcomings of our system, such as the lack of in-bin reorientation of objects, must be addressed by some form of planning.

Other team's motion generation systems

In the 2015 challenge, most contenders employed some form of configuration space planning (80% of the teams used motion planning, 44% used MoveIt (Correll et al., 2018)). In the 2016 challenge, all teams exploited some form of feedback from in-hand contact sensing, i.e. air pressure sensors that signal contact with objects. Beyond this simple feedback source, solutions ranged from classical sense-plan-act implementations to more feedback driven systems.

Far on the feedback side, Team MIT tried to exploit sensing by installing a large number of sensors on their gripper (Yu et al., 2016). These included pressure sensors for the suction-based grasping as well as hall effect and strain gauge sensors to measure gripper contact. Team MIT used closed-loop force control to scoop under objects with a spatula-like gripper.

Similar to our approach regarding the use of feedback, Team NimbRo (Schwarz et al., 2017) replaced complex motion planning with keyframe-based motion generation which included simple linear interpolation and inverse kinematics that avoids collisions with the shelf. Also similar to our solution, the team used joint torque sensors to perceive and react to unwanted collisions at execution time.

On the other end of the spectrum, in 2016 also purely planning-based approaches proved to be feasible strategies. Team Delft's motion generation followed the classical "sense-plan-act" paradigm (Hernandez et al., 2016). They decomposed motion trajectories into reach, grasp, retreat, and drop-off, and calculated most of the parts offline, especially those that were happening outside the shelf.

Similarly, Team PFN executed their motion without further feedback. Open loop execution of planned motion requires careful calibration. In the 2016 challenge, Team PFNs placed 2nd because of a long time to first pick which was a consequence of a long calibration phase. Team Delft stated that in tightly filled bins, pure collision-avoiding planning will start to fail and therefore

> "Force-feedback and compliance in the gripper seem unavoidable to achieve a reliable solution." (Hernandez et al., 2016)

In conclusion, the top-ranking solutions of the 2015 and 2016 Amazon Picking Challenge vary a lot on the spectrum of planning and feedback. This shows that reliable solutions for the Amazon Picking Challenge can reside on both ends of the spectrum. Purely planning-based approaches succeeded because the environment is static and perfectly known so that open-loop execution is feasible for a calibrated system. Consequentially, teams that rely on planning avoided contact. They delayed the contact with the object until the last possible moment.

On the other end of the spectrum, teams that relied on feedback often did not plan. These teams usually implemented hand-coded motions exploiting the structured environment. As most of the strategies are coded for the specific application they will not easily generalize to other scenarios. There is consensus among all teams that feedback leads to more robust behavior which will be required in a more realistic setting.

To conclude, our entry to the Amazon Picking Challenge was based on a feedback-driven approach using no autonomous planning. We observed other teams in the 2015 challenge most often picked planning-based architectures, indicating that the use of feedback was important for our success.

3.5 Requirements for robust motion generation

The strategies of our system were hand-designed by roboticists. Every single action and reaction was manually programmed and optimized for the task at hand. Simple changes such as a changing the shelf-layout would make the system fail completely. To generalize the lessons learned from the Amazon Picking Challenge to arbitrary tasks, we require methods that can generate hybrid automata autonomously from sensor data. We now want to enumerate three requirements for motion generation that need to be solved to make robots operate more robustly.

Contact

The system in the Amazon Picking Challenge reliably and robustly contacted the environment. Contact is a challenge for many planning methods as it is discontinuous. In our system, we handled contact with a hybrid automaton approach, where continuous motions are sequenced by discrete switches based on force. We believe that motion generation methods must be able to come with such strategies autonomously. We will address planning with contact in Chapter 5, presenting a planner that generates such contact-based hybrid automata.

Contingent planning

The hybrid automata solving the picking task were composed of a main sequence of events, but also contained branches for failure cases. For example, unwanted collisions with the shelf could be measured by the exerted force which brings the arm into a mode where it safely retracts. We believe a planner must be able to predict possible failure cases, find out how to detect them using discrete feedback, and plan possible reactions. A planner predicting

possible failure modes and suitable reactions is called **contingent planner**. In Chapter 6, we will present contingent planning for manipulation with contact.

Incremental planning

The real world is dynamic, even in the Amazon Picking Challenge where the robot is the only moving agent. Objects will move after picking or unwanted collisions might displace the shelf. Therefore, after each picking attempt, our picking system restarts the perception pipeline and updates picking locations i the hybrid automaton. Planners should not need to restart whenever small things in the world change. Rather, they should be able to continuously integrate new sensor data and adapt their plans. We call a planner that continuously integrate data **incremental planner** and will present incremental motion planners for mobile manipulation in Chapter 8.

Part II

Contact-based manipulation planning under uncertainty

“Would you tell me, please, which way I ought to go from here?’
’That depends a good deal on where you want to get to,’ said the Cat.
’I don’t much care where -’ said Alice.
’Then it doesn’t matter which way you go,’ said the Cat. ’- so long as I get SOMEWHERE,’
Alice added as an explanation.
’Oh, you’re sure to do that,’ said the Cat, ’if you only walk long enough.”

Chapter 4

Background in sampling-based motion planning

In the previous chapters, we introduced the feedback controllers that let robots execute manipulation tasks. These methods were based on local potential functions, describing behavior by forces. The limitations of this local control approach came from local minima. On their own, purely reactive controllers are unable to execute long term motion without getting stuck. In this chapter, we will approach the motion generation problem from the global side and ask: How can we find strategies that do not get stuck in local minima? Ultimately, to answer this question we need methods that understand the **connectivity** of the space. This knowledge allows us to avoid dead-ends and will solve problems that require long, complex motions.

In this section, we present sampling-based methods for motion planning. The first half will introduce tree and graph-based approaches for motion planning.[1] We will also discuss how to plan motions under task-space constraints. In the second half of this section, we will transfer the sampling-based approach to planners that reason explicitly about uncertainty. Planning under uncertainty is a PSPACE-complete problem and thus it can only be solved by making reasonable assumptions that simplify the problem. We will discuss these assumptions at the end of the section. The contents of this chapter are unique to this thesis, and a condensed overview of the planning literature. For more details we refer to AI and planning textbooks (Bertsekas, 1995; LaValle, 2004; Choset et al., 2005; Russell and Norvig, 2016).

[1] Although often referred to as planning, we will not discuss optimization-based methods here but in Chapter 7.2, where we focus on the continuous integration of sensor data.

4.1 Planning as search

Motion planning is a search problem. In classic AI planning (Russell and Norvig, 2016), the task of moving a robot is solved by first discretizing the state space and then using graph search algorithms such as A^* (Hart et al., 1968). The advantage of this approach is its **completeness**: After a limited time the planner will either return the optimal solution or, if start and goal are not connected, return failure. However, in mobile manipulation applications, this approach is not feasible due to the high dimensionality of the configuration space. The number of states needed to discretize an n-dimensional state space increases exponentially with n. Discretizing the configuration space of a 10-dof mobile manipulator into 100 bins for each dimension and storing one integer value for each state would result in a memory requirement of 800 exabyte, which is in the order of magnitude of all available memory on the world today. This problem is referred to as the **curse of dimensionality**.

The sampling-based approaches we will introduce in this chapter can efficiently plan paths for robots with many degrees of freedom, despite the curse of dimensionality. Their strategy is, not to fix a discretization of the state space a priori, but to make the discretization part of the problem. Sampling-based approaches incrementally build a problem-specific discretization of the state space. The key to the success of sampling-based approaches is that even in high-dimensional space, a low number of well-chosen states, and actions to move between them, suffices to solve a given problem. By doing so, sampling-based planners give up on completeness, trading it in for weaker guarantees such as probabilistic completeness and asymptotic optimality.

4.2 Configuration space obstacles

The goal of motion planning is to move a robot through the obstacle-free part of the space. We denote with $O \subset \mathbb{R}^3$ the **obstacle region**. The set $\mathcal{A}(q) \subset \mathbb{R}^3$ denotes all points that are occupied by the robot in configuration $q \in \mathcal{C}$. We can also express this region in configuration space: $\mathcal{C}_{\text{obs}} = \{q \in \mathcal{C} | \mathcal{A}(q) \cap O \neq \emptyset\}$. The part of configuration space where the robot can move without colliding is called **free space**: $\mathcal{C}_{\text{free}} = \mathcal{C} \setminus \mathcal{C}_{\text{obs}}$. The set of all configurations where the robot is in contact with the environment is the boundary of the free space, i.e. the "hull" of the obstacles and denoted with $\partial\mathcal{C}_{\text{free}}$.

Using this definition, a continuous path through $\mathcal{C}_{\text{free}}$ connecting two configurations specifies the motion of a robot. The difficulty of finding such paths lies in the shape of $\mathcal{C}_{\text{free}}$, which is fairly complex even for simple workspace obstacles (see Fig. 4.1). For simple free-flying robots $\mathcal{C}_{\text{free}}$, can be constructed geometrically (Lozano-Pérez, 1983; Latombe,

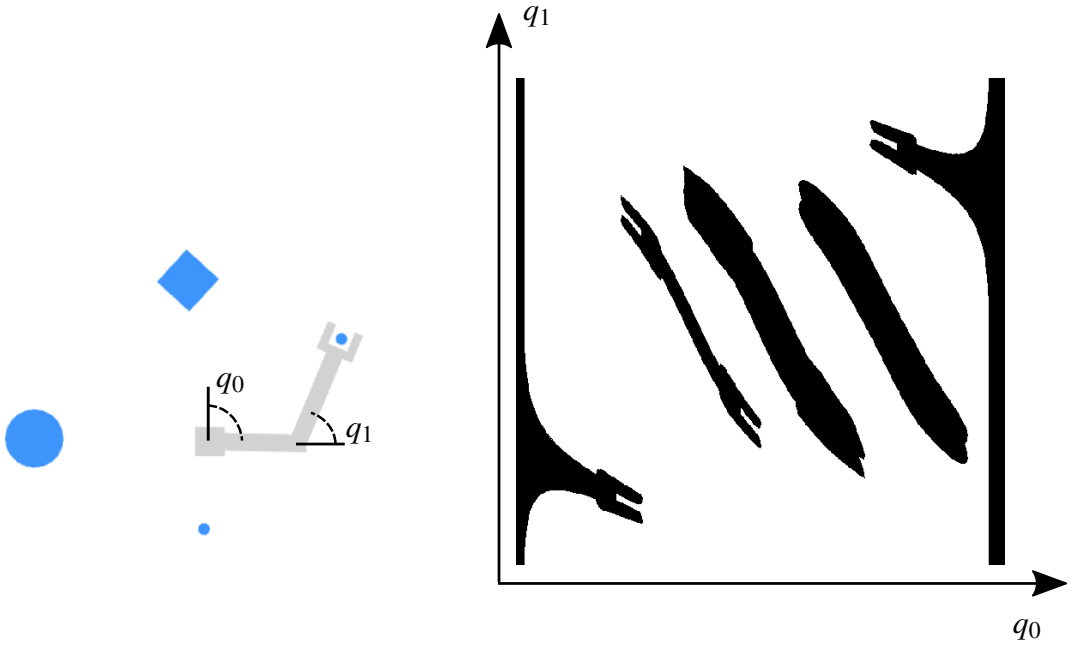


Figure 4.1 *Left*: A 2-dof RR manipulator operating in a 2D workspace. $\mathcal{A}(q)$ is shown in grey, the workspace obstacles O are shown in blue. *Right*: A plot of the corresponding configuration space. The black regions show $\mathcal{C}_{\text{obs}}$.

2012), however, in general no closed form description of $\mathcal{C}_{\text{free}}$ exists. Sampling-based motion planners circumvent the problem of describing $\mathcal{C}_{\text{free}}$ explicitly by using collision checking algorithms such as Gilbert–Johnson–Keerthi (GJK) (Gilbert et al., 1988). These algorithms can efficiently compute intersections between a number of convex polyhedra. This is then used to compute if there is a collision for a robot in a given configuration. This is equivalent with sampling $\mathcal{C}_{\text{free}}$.

4.3 Probabilistic roadmaps

The definition of free configuration space allows the following classic definition of the motion planning problem (also called the *Piano Mover's Problem* (Reif, 1979)):

Path planning problem: Given a start configuration $q_{\text{start}} \in \mathcal{C}_{\text{free}}$ and a goal configuration $q_{\text{goa}}, \in \mathcal{C}_{\text{free}}$, find a continuous path $\tau : [0,1] \to \mathcal{C}_{\text{free}}$ such that $\tau(0) = q_{\text{start}}$ and $\tau(1) = q_{\text{goal}}$.

Probabilistic Roadmaps (PRM) (Kavraki et al., 1996) were the first sampling-based planners in general configuration space. The PRM algorithm has two phases, a construction phase (see Alg. 1) and a query phase.

Construction: The PRM method incrementally constructs a roadmap which is a graph $G = (V,E)$ that is initially empty. The method samples in each iteration a configuration

$q_{\text{rand}} \in \mathcal{C}_{\text{free}}$. This sample can come from a random or deterministic process that covers $\mathcal{C}_{\text{free}}$. The algorithm adds q_{rand} as a vertex into the graph and then checks for each of the k nearest neighbours (where k is a fixed constant) if the new vertex can be connected to any other vertex in the graph with a straight line that lies completely in $\mathcal{C}_{\text{free}}$. If this is the case, the straight line connection is added to E. The construction phase repeats this process, usually for a fixed number of iterations.

Query: the planner takes as input the goal and start configurations and tries to connect them to nodes in the existing roadmap. If start and goal can be connected to the roadmap, a shortest path algorithm (Dijkstra, 1959) is invoked to search the graph for the shortest path between q_{start} and q_{goal}. If there is such a shortest path, the path is returned.

Algorithm 1 Basic PRM construction

Input: $q_{\text{start}}, q_{\text{goal}}$
Output: $G = (V, E)$
1: $V \leftarrow \{q_{\text{start}}\}$ *init tree with start configuration*
2: **while** no path between q_{start} and q_{goal} **do** *search until goal reached*
3: $q_{\text{rand}} \leftarrow$ SAMPLE()
4: $\mathcal{Q}_{\text{near}} \leftarrow$ NEAREST_NEIGHBOURS(q_{rand}, G)
5: **for all** $q_{\text{near}} \in \mathcal{Q}_{\text{near}}$ **do**
6: **if** CONNECT$(q_{\text{near}}, q_{\text{rand}})$ **then**
7: $V \leftarrow V \cup \{q_{\text{rand}}\}$
8: $E \leftarrow E \cup \{(q_{\text{near}}, q_{\text{rand}})\}$

The PRM is a **probabilistic complete** planner. This means that if a motion planning problem is solvable, the PRM will find a solution with probability approaching 1 if the number of iterations approaches infinity. The PRM is also **asymptotically optimal** which means that it converges to the shortest possible path if the number of iterations approaches infinity. This property however only holds for an increasing search radius $k > \mathrm{e}(1 + 1/d)\log(|V|)$, where e is the Euler number, and d the dimensionality of $\mathcal{C}$ (Karaman and Frazzoli, 2011)[2]. Once a PRM is constructed, it can, for any given static environment, solve a multitude of planning problems. This property is referred to as **multi-query** planning.

PRMs can efficiently solve path planning in complex, high-dimensional spaces because they are not directly affected by the curse of dimensionality. The reason for this is that the minimal number of nodes needed to cover the environment is strongly related to the visibility (Siméon et al., 2000), i.e. the size of the regions that can be connected with straight-line motions. The **ε-goodness** (Barraquand et al., 1997) is a measure of visibility. It describes the volume fraction of the free space that is visible from configuration. The key

[2] a similar result exists for an alternative version of the PRM, where all neighbours in a fixed radius r are candidates for connection (Karaman and Frazzoli, 2011)

to the success of PRMs is the observation that the *minimal* ε-goodness of typical motion planning problems does not increase exponentially with configuration-space dimension. Even for high-dimensional problems, roadmaps with few nodes can cover the space well. However, finding these positions is not trivial. This challenge has been tackled with a multitude of different approaches based on adaptive sampling (Boor et al., 1999; Siméon et al., 2000; Hsu et al., 2003), workspace geometry analysis (Amato et al., 1998; Holleman and Kavraki, 2000; Brock and Kavraki, 2001; Yang and Brock, 2004), or statistical learning (Burns and Brock, 2005; Knepper and Mason, 2012).

4.4 Rapidly-exploring random trees

Rapidly-Exploring Random Trees (RRT) (LaValle, 1998) are single-query planners, i.e. they must be restarted for every combination of q_{start} and q_{goal}. Originally, they were designed to handle problems with dynamic constraints but they proved to be very efficient solvers for unconstrained path planning problems too. One iteration of the RRT planner is an extension of one branch of a search tree towards a sample. In each iteration, a sampler generates a random configuration q_{rand}, the planner finds the nearest node q_{near} in the current search tree, and from this closest node, the tree simulating one step moving towards the random sample. The resulting node q_{new} is then checked for collision and added to the tree if it is in free space.

Algorithm 2 Basic RRT planner

Input: $q_{\text{start}}, q_{\text{goal}}$
Output: $T = (V, E)$
1: $V \leftarrow \{q_{\text{start}}\}$ *init tree with start configuration*
2: **while** true **do** *search until goal reached*
3: $q_{\text{rand}} \leftarrow$ SAMPLE()
4: $q_{\text{near}} \leftarrow$ NEAREST_NEIGHBOUR(q_{rand}, G)
5: $q_{\text{new}} \leftarrow$ EXTEND$(q_{\text{near}}, q_{\text{rand}})$
6: **if** IS_VALID(q_{new}) **then**
7: $V \leftarrow V \cup \{q_{\text{new}}\}$
8: $E \leftarrow E \cup \{(q_{\text{near}}, q_{\text{new}})\}$
9: **if** $\|q_{\text{new}} - q_{\text{goal}}\| < \varepsilon_{\text{goal}}$ **then**
10: **return** T

The key to the RRT is the connection to the *nearest* neighbour. This step lets the RRT quickly grow into space at first and then grow inwards to fill gaps. This behaviour can be understood by noticing that, given uniform random sampling, the probability to extend a

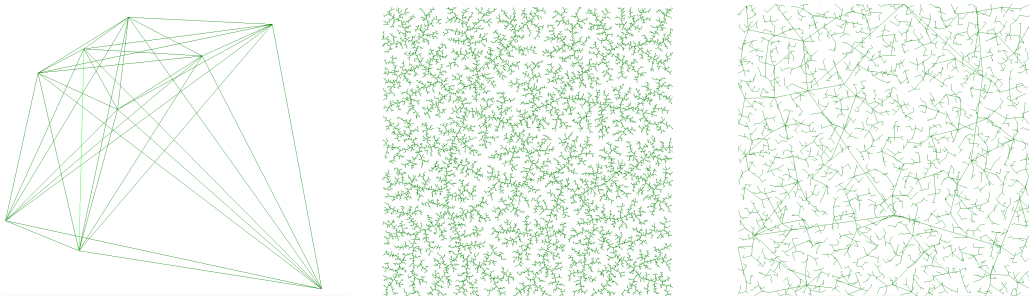


Figure 4.2 Sampling-based motion planners exploring a 2D configuration space without obstacles. *Left*: PRM creates one graph with high connectivity *Middle*: RRT creates a tree of short extension steps *Right*: RRT-Connect uses longer straight-line connections to span the space quickly

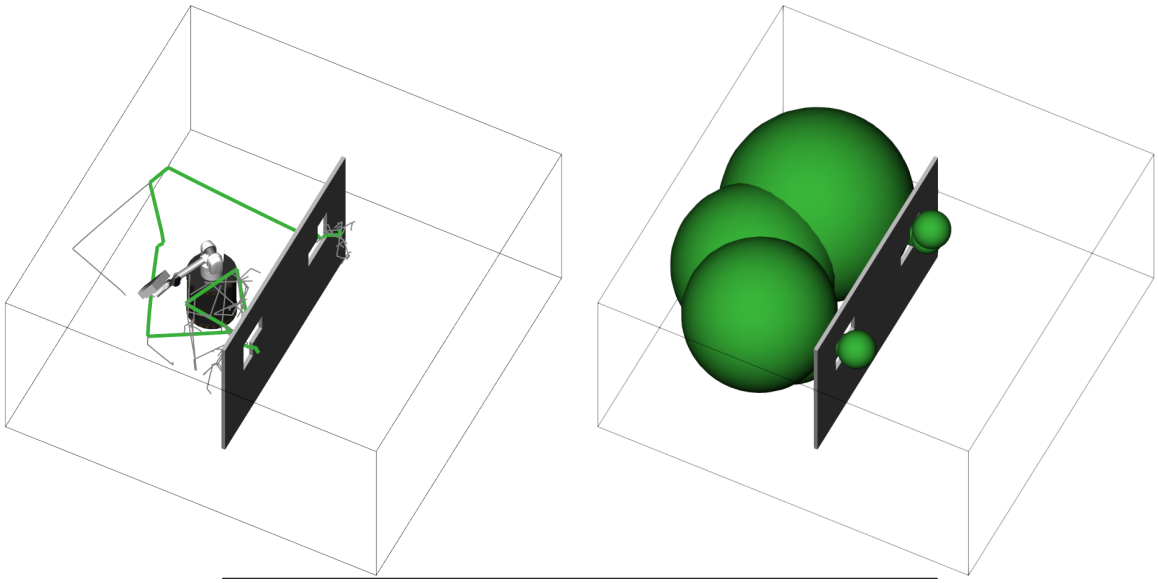
given node is proportional to the size of its Voronoi region. Nodes on the border of the tree have bigger regions. This exploration behaviour is known as the **Voronoi-bias**.

RRTs becomes a very efficient path planner in a variant for planning without kinodynamic constraints called **RRT-Connect** (Kuffner and LaValle, 2000). This variant replaces the one step forward simulation of an action by a *connect*-move which is the straight line connection between nearest node and sample, analogously to the PRM. In practice, often two trees grow from start and goal and try to connect to each other.

Fig. 4.2 shows the search trees for different sampling-based planers. RRTs do not have the multi-query property of the PRM as the trees are rooted in the start and goal configurations of a problem. Compared to PRMs, RRTs have an advantage of only searching the reachable space. If large parts of configuration space are blocked for a robot, the RRT will not search them while the PRM might spend substantial effort sampling it. Just like the PRM, the RRT is probabilistic complete. A modified version called RRT* is asymptotically optimal (Karaman and Frazzoli, 2011).

4.5 Exploration and exploitation in motion planning

There are countless extensions to the PRM and RRT planners and depending on the problem, their performance can vary widely. On some problems they find paths in milliseconds, while on others they are too slow to find any solutions in realistic time. The performance of a planner is strongly linked to its ability to identify the parts of the space that are relevant for a particular problem and focus search on these areas. A planner that has information about



Planner	Vertices	Collision Checks	Time [s]	%
PRM	32649	4510812	1200.0	0
RRT-Connect1	93745	6053905	1200.0	0
RRT-Connect2	17791	1053950	203.9	100
EET	285	12732	16.9	100

Figure 4.3 Planning results for a mobile manipulation planning problem (Rickert et al., 2014); Shown top left is the search tree and trajectory found by the EET planner; Top right is the workspace tunnel used in the EET. Shown in the table are the number of vertices in the search tree, collision checks, computation time, and success rate for the different planners, averaged over 20 attempts.

the relevant subspace will outperform more problem-agnostic and probabilistic complete planners.

In realistic applications, perfect information about the relevant configuration space is not available. To plan efficiently, it is therefore crucial to balance the exploitation of gathered information with exploration, which is gathering new information. The **Exploring-Exploiting-Tree** planner (EET) (Rickert et al., 2014) exemplifies this concept. It balances the exploitation of information with explorative behaviour. The EET's information source is a workspace decomposition which guides the search from start to goal. This decomposition represents a tunnel of free workspace. While in many cases the guidance of free workspace is helpful, sometimes it can lead the planner in wrong directions. The key of the EET planner is that it adaptively balances the usage of information (which is exploitation) with a broader search in configuration space (which is exploration). This balancing lets the planner rely on workspace information as much as possible if the information is helpful. In regions where the information is misleading the planner relies less on it and searches more broadly.

The adaptive usage of workspace information lets the EET perform up to three orders of magnitude better than standard RRT planners (Rickert et al., 2014), on a large number of realistic problems. The EET will also be a component of a motion generation system for dynamic environments we will introduce in Chapter 8.

To illustrate the feasibility of sampling-based planning for mobile manipulation, we show in Fig. 4.3 a comparison of the PRM, RRT-Connect, and EET planners for a mobile manipulation planning problem. In this problem, the 10-dof WAM+XR4000 manipulator has a large L-shaped workpiece at its end-effector. It starts with the workpiece placed inside a slightly bigger L-shaped hole. The goal configuration is inside another hole, i.e. the robot must overcome two narrow passages to solve the task. The results show a large gap in performance, with EETs outperforming dual tree RRT-Connects by a factor of 13, which outperform single-tree RRT-Connects and PRMs by more than two orders of magnitude (the planner was stopped if no solution was found after 20 minutes).

4.6 Task-constrained planning

For robotic manipulation, the path planning problem formulation is not directly applicable. The reason is that manipulation is inevitably linked to the geometry of the real world. As discussed in section 2.4, many tasks are specified as subspaces of $SE(3)$. Therefore, motion for manipulation is almost always subject to extra constraints which must be taken into account while planning.

An m-dimensional constraint can be expressed as a **task function** $T : \mathcal{C} \to \mathbb{R}^m$, such that $T(q) = 0$ when the constraint is satisfied. This definition of constraint allows to define the **task manifold** $\mathcal{M}_T$, which is the part of configuration space that fulfills all constraints. Fig. 4.4 shows the task manifold for a two-dimensional constraint.

$$\mathcal{M}_T = \{q \in \mathcal{C} : T(q) = 0\} \tag{4.1}$$

We now can define the **constrained path planning problem**: Given a start configuration $q_{\text{start}} \in \mathcal{C}_{\text{free}}$, a goal state $x_{\text{goal}} \in \mathbb{R}^m$ and a constraint function T, find a continuous path $\tau : [0,1] \to \mathcal{M}_T$ such that $\tau(0) = q_{\text{start}}$ and $T(\tau(1)) = x_{\text{goal}}$. Two types of constraints are relevant for our applications:

Pose: The constrained motion used in task-space controllers are closely related to constraints functions for planning. Every potential function can be trivially turned into a constraint function by setting it to zero at its minimum. Thus, a robot carrying a tray of objects must keep two degrees of rotation fixed. A robot wiping a window has constraints on

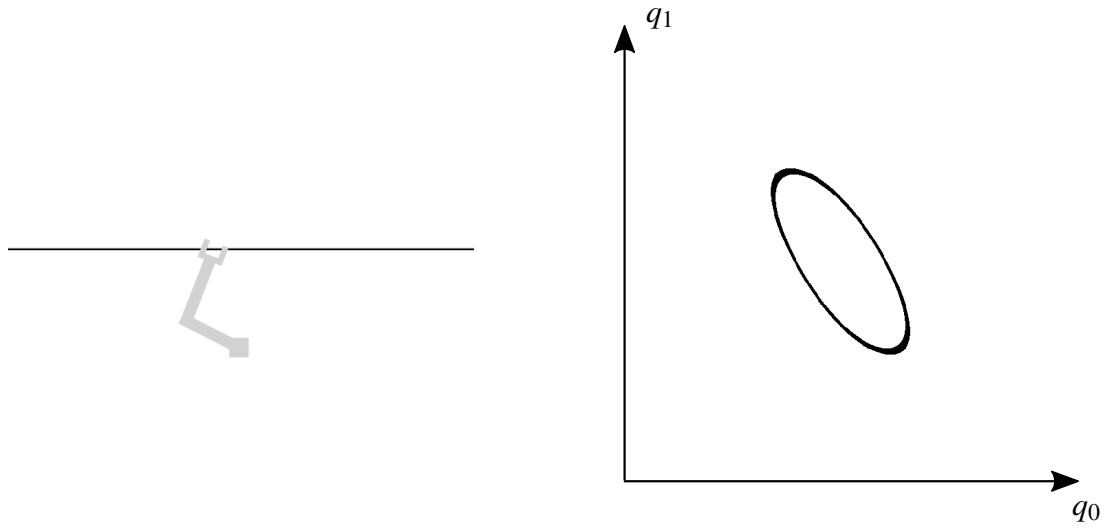


Figure 4.4 *Left*: A 2-dof RR manipulator operating in a two-dimensional workspace. A one-dimensional task constraint T is shown in black. *Right*: A plot of the corresponding configuration space. The black shape shows the corresponding task manifold $\mathcal{M}_T$.

the end-effector position and orientation. In general, such constraints can be specified as

$$T(q) = 0 \text{ if } f(q) \in \mathcal{X} \subset SE(3)$$

Contact: Configurations in contact also define a manifold, given by

$$T(q) = 0 \text{ if } q \in \partial\mathcal{C}_{\text{free}}$$

Other possible constraints not relevant for this thesis are visibility (can a sensor see a feature), balance, or dynamic constraints such as the maximum liftable weight for a given configuration.

These constraints define a lower-dimensional task manifold of the configuration space. Sampling-based approaches cannot directly be applied to these constraints as the probability to randomly sample a configuration on the manifold is effectively zero. To generate paths on a lower dimensional manifold, samples must be dragged onto it. There are two main to achieve this: **Direct sampling** samples a random point on the manifold $x_{\text{rand}} \in \mathcal{M}_T$. It then takes an existing configuration in the graph and moves it towards that point using the Jacobian (either transpose or Pseudoinverse (see Sec. 2.4)). This is comparable to simulating the execution of a task space controller moving towards the sample:

$$q_{n+1} = q_n + \delta J^{+}_{\mathcal{M}}(f(q_n) - x_{\text{rand}}) \tag{4.2}$$

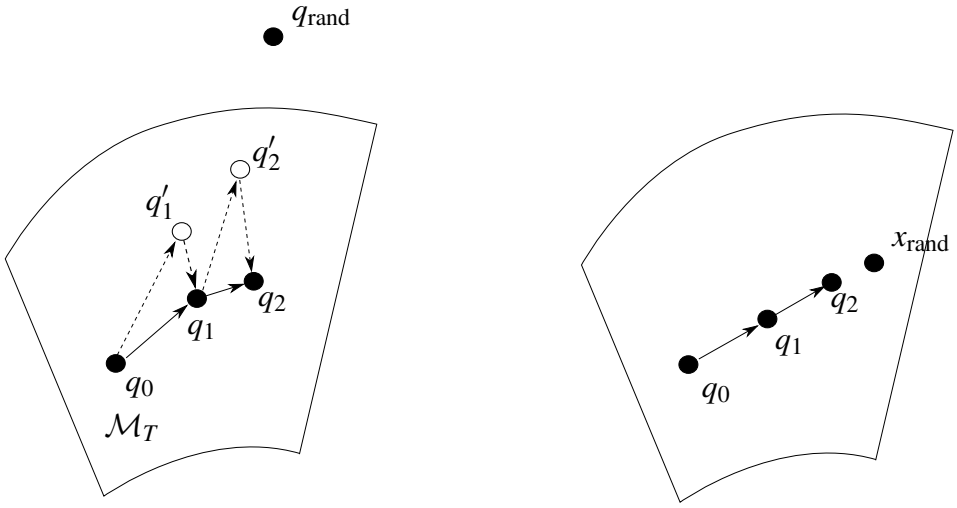


Figure 4.5 *Left*: The projection method. Samples on the manifold move towards a random sample q_{rand} and then are projected back on the manifold. *Right:* Direct sampling: the sample lies on the manifold and the samples get directly extended towards it.

Projection methods (Stilman, 2007; Berenson et al., 2009) do not sample the task manifold directly but sample random configurations. They then move the existing node on the manifold towards the random sample configuration and then, in a second step, project it back onto the manifold:

$$\begin{aligned} q'_{n+1} &= q_n + \delta(q_{\text{rand}} - q_n) \\ q_{n+1} &= q'_n + J^+_{\mathcal{M}}(\Delta x_{\text{disp}}) \end{aligned} \tag{4.3}$$

where Δx_{disp} is the vector between closest points on robot and constraint manifold.

The advantage of the direct method is that it samples a parametrization of the manifold and thus will uniformly sample the task space. Additionally, direct sampling is not directly affected by the dimensionality of the configuration space and can be applied to robots with many degrees of freedom (Shkolnik and Tedrake, 2009). The disadvantage of direct sampling is that the method is not probabilistic complete (Stilman, 2007). For completeness, the sampler must reach the whole nullspace of the task manifold. If a task requires many nullspace motions, direct sampling might get stuck.

To alleviate these problems, we propose a third hybrid approach, which directly samples the task manifold but also explores the nullspace of the task. This approach samples a point on the manifold x but also an additional random configuration $q_{\text{exp}} \in \mathcal{C}$. This configuration is projected to the nullspace and added to the random sample.

$$q_{n+1} = q_n + J^+_{\mathcal{M}}(f(q_n) - x_{\text{rand}}) + (I - J^+_{\mathcal{M}} J_{\mathcal{M}})(q_{\text{exp}} - q_n) \tag{4.4}$$

This is comparable to simulating a multi-priority controller that follows the shortest path towards the manifold but also randomly explores the null-space.

4.7 Motion planning under uncertainty

Most of the previous planners' assumptions do not hold in the real world due to uncertainty. The path planning problem requires precise models of the world, it assumes that the robot will move exactly as commanded, and it assumes that the robot knows exactly where it is. Any of these assumptions could lead to a critical failure. This makes path planning solutions often brittle and lets them fail in not perfectly known environments.

To relax the perfect world assumptions, planners can incorporate models of the uncertainty about robot state, action outcome, and world model into the planning problem. Planners then reason about many possible worlds and find a plan that works for all or most of them. We will now discuss motion planners that reason explicitly about uncertainty. Throughout the discussion, we will increase the number of uncertainty sources they take into account.

4.7.1 Planning with uncertain actions

The path planning problem assumes that a robot can perfectly execute any trajectory τ. This assumption conflicts with real world uncertainty as in reality the robot will diverge from the nominal path. Especially minimal length trajectories often exactly touch the borders of $\mathcal{C}_{\text{free}}$. If there is just a tiny deviation from that trajectory, the robot will collide.

One way to counter these problems is to model the robot's actuation uncertainty explicitly and then find motion strategies which are robust to many possible action outcomes. To model the actuation uncertainty, we define a set of actions $\mathcal{U} \subset \mathbb{R}^n$ the robot can execute. These actions can be a velocity or torque commands, depending on the available controllers. We assume that if a robot in configuration q, applies control input $u \in \mathcal{U}$, the configuration changes according to a probability distribution called **motion model** $p(q'|q,u)$.

Planning motions under this model is a **Markov Decision Process (MDP)**. An MDP is defined as the combination of **transition function** p and **reward function** r. The transition function is defined as above and must have the **Markov property**,i.e. the probability of reaching q' after applying action u depends only on the state q, independent of how it was reached exactly. The reward function $r(q)$ maps each state to a real (non-infinite) value. This function defines both target regions for the robot, which have high reward, as well as regions that should be avoided and give negative reward. The solution to an MDP is a **policy** $\pi : \mathcal{C} \to \mathcal{U}$ which is a mapping from states to actions.

The objective of an MDP is to maximize the cumulative discounted rewards. For an action sequence $(q_0, q_1, q_2, \ldots)$, this is defined as

$$\sum_{t=0}^{\infty} \gamma^t r(q_t) \tag{4.5}$$

where γ is a discount factor between 0 and 1. The discount factor γ penalizes rewards that are gathered later. The objective for solving an MDP is now to find an optimal policy π^* which generates the highest expected cumulative discounted rewards.

The key to solving MDPs is the **Bellman equation** which computes a **value function** $V : \mathcal{C} \to \mathbb{R}$ that captures the expected objective value obtained from executing the optimal policy π^* in each state.

$$V(q) = r(q) + \gamma \max_u \int p(q'|q,u)V(q')dq' \tag{4.6}$$

This equation recursively sums over the reward in the current state and the maximum possible expected reward the robot could get when moving from state q to a successor state q'. For *discrete* state and action spaces the **value iteration** algorithm can efficiently compute this function. It stores an approximation of the value $\hat{V}(q)$ for each state q and recursively updates these approximations for a growing time horizon:

$$\hat{V}_{k+1}(q) = r(q) + \gamma \max_u \sum_{q' \in \mathcal{C}} p(q'|q,u)\hat{V}_k(q') \tag{4.7}$$

The approximations then converge to the true value function, i.e. $\hat{V}_k \to V$, for $k \to \infty$ for any initialization $\hat{V}_0$. After computing V, the optimal policy follows from picking the action that maximizes the expected utility:

$$\pi(q) = \text{argmax}_u \int p(q'|q,u)V(q')dq' \tag{4.8}$$

The solutions to MDPs are qualitatively different to the solutions of path planners. Path planners return one sequence of states, assuming that the robot can execute them exactly as planned. Planning under uncertainty instead returns a policy capturing the optimal action for each state. Even if the robot diverges from its nominal path, it will always know what to do. This mapping acts like a global feedback loop continuously that changes the action continuously depending on the robot's state. In contrast to the feedback controllers from Chapter 2, these policies are guaranteed to lead to the goal and not end up in local minima. Planners that generate policies are referred to as **Feedback motion planners**.

The reward function allows to formulate different kind of different planning problems. One relevant problem that models path planning with actuation uncertainty is the **Stochastic Shortest Path Problem** (Bertsekas, 1995) which is a non-discounted ($\gamma = 1$) MDP with negative reward for every state that is not the goal. The goal state is a sink which means that the robot can stay in it for an indefinite time. The solution to this problem is a policy that moves the robot towards the goal in the lowest possible expected number of actions.

MDPs capture planning under actuation uncertainty well, however, just like the search-based approach for path planning, the challenge lies in the discretization of configuration space, as the number of states grows exponentially with the dimensionality. Therefore, to be applicable for manipulators, also MDP approaches need to construct a sample-based approximation of the state space during planning. This idea led to MDP-based versions of the PRM (Alterovitz et al., 2007) and RRT planners (Melchior and Simmons, 2007). These planners invoke a sampling-based planner and then solve the MDP on the reduced state space given by the nodes of the tree or roadmap. To obtain the transition probabilities, these methods do Monte-Carlo simulation: they sample random, noisy robot actions and gather statistics about the outcome.

4.7.2 Partial observability

To apply the MDP policy $\pi(q)$, a robot must know perfectly where it is in the world. In reality, this assumption does not hold as all of the robots sensors are subject to uncertainty.

Partial Observable Markov Decision Processes (POMDPs) model a robot that does not have access to its ground truth state. The problem is called **partially observable** if the robot must estimate its state from noisy measurements. We denote the the aggregated sensor values of the robot with z. To do so we assume the robot has access to a **sensor model** which is the probability of receiving measurement z in state q and is given by $p(z|q)$.

The robot must now infer its state from the sequence of noisy measurements and at the same time pick the optimal action. To do so, it must reason about the whole observation history which can be very long. The fundamental insight of POMDP planners is that the robot can express its history in a **belief state** $b \in \mathcal{B}$. $\mathcal{B}$ is the space of probability distributions over $\mathcal{C}$ and called **belief space**. The optimal action now depends only on the robots current belief state. This means that the belief sufficiently captures all relevant information in the observation history. To pick the optimal action, the robot does not need to keep track of the full history nor does it need access to the ground truth state.

The solution to a POMDP is a policy π mapping from belief space $\mathcal{B}$ to actions. To execute the policy, the robot alternates between two steps:

1. update the belief state using the last measurement z:

$$b'(q) = \alpha p(z|q) \int_{\mathcal{C}_{\text{free}}} b(q) p(q'|q,u) dq \tag{4.9}$$

   where α is a normalization constant.

2. execute the next action $\pi(b'(q))$.

As the belief state fulfills the Markov property, we can compute the policy by casting the POMDP as an MDP in belief space. However, as the belief space is continuous, value iteration cannot be applied directly. The trick of POMDP solvers is to exploit that the optimal value function in belief space is piecewise linear. This insight leads to solution algorithms that construct the value function incrementally (Kaelbling et al., 1998).

However, even under these insights, planning in belief space is very hard. The dimensionality of belief space is equal to the number of robot states ($\dim(\mathcal{B}) = |\mathcal{C}|$). This exacerbates the curse of dimensionality once more as the high number of states now is leading to an exponential number of belief states. Furthermore, the time complexity of POMDPs depends exponentially on the number of different actions and observations. These issues make optimal solutions to POMDPs computationally intractable. In fact, solving POMDPs optimally is a PSPACE-complete problem (Papadimitriou and Tsitsiklis, 1987). To tackle the high complexity we must make further approximations. In the following, we will discuss some options.

4.8 Strategies for efficient belief-space planning

The key to handle the intractable planning problem under uncertainty is to make the right assumptions, targeted to the specific problem. In the following we will discuss common strategies from the literature.

Sampling

The sampling-based approach also transfer to solving POMDPs. These method also approximate parts of the problem with samples to overcome combinatorial explosions when discretizing the space. A common technique, especially useful for continuous state POMDPs, is to approximate the belief state with a set of samples, so called **particles**. The integral in 4.9 becomes a summation, equivalent to the particle filter algorithm (Thrun et al., 2005). Transitions between states can then be approximated with Monte Carlo Simulation (Thrun, 1999). Another application of sampling in POMDP solvers are point-based solvers (Pineau

et al., 2003; Kurniawati et al., 2008) which gain efficiency by sampling the reachable belief space and only performing value iteration on representative belief points. All these methods relax the optimality requirement and settle for approximate solutions.

Linear models

The planning problem can be simplified drastically by restricting the problem space. Assuming linear motion model and linear sensor model the belief state is Gaussian and many computations simplify a lot. By introducing a quadratic cost term, the planning problem is known as the **Linear Quadratic Regulator** (LQR), which also has closed form solutions given by the Riccatti equation. Even for non-quadratic cost functions, efficient optimization techniques exist (Platt Jr et al., 2010; Prentice and Roy, 2010; Bry and Roy, 2011; Platt Jr et al., 2011; Van Den Berg et al., 2012; Agha-Mohammadi et al., 2014). Linear models also allow to compute regions of attractions of controllers numerically using sums-of-squares programming. This knowledge of funnel entrances allows to compose funnels in a tree-based motion planner (Tedrake et al., 2010).

Ignoring uncertainty

A good heuristic is to ignore all or some sources of uncertainty. **Mixed observability POMDPs** (MOMDPs) (Ong et al., 2010) treat some parts of belief-space as fully observable and can be much easier to solve.

Hindsight optimization

Another useful heuristic is to assume that uncertainty about the state can not be changed (Hindsight optimization (Yoon et al., 2008), QMDPs (Littman et al., 1995)). This corresponds to a pessimistic plan where the robot finds a policy that would work without gathering new information. A dual heuristic to this is to assume that uncertainty disappears completely after picking one action. This corresponds to an optimistic plan. Both assumptions change the POMDP into an MDP and make it computational tractable on large problems but give up all optimality guarantees (Arora et al., 2018). We will see these heuristics in use in Chapter 8 where we use a simplified MDP-based model for uncertain environments.

Replanning

The last, and very powerful strategy is to interleave planning and execution. The insight is that a huge amount of uncertainty will be reduced once the robot starts acting in the real

world. A planner can reason over a short horizon, with simplified models, execute the first actions of the policy, and then update its models and replan. It was shown in planning competitions (Yoon et al., 2007) and robotics problems (Kaelbling and Lozano-Pérez, 2013) that this approach outperforms far more complex reasoning methods that do not update as often. However, replanning comes with multiple difficulties: The planner must be efficient enough to run without blocking the robot for a long timespan. Also, an efficient replanning architecture must continuously integrate sensor data. The whole pipeline from sensor stream to path finding becomes part of the problem. Replanning to counter uncertainty will be our main tool in part III where we integrate simple reasoning over uncertainty with efficient incremental planning.

Chapter 5

Sampling-based belief-space planning in contact

In this chapter we want to use planning to tackle one of the requirements of our case study (Sec. 3.5): the ability to robustly make contact with the environment. Therefore, we will introduce a planner for robust motion under uncertainty that interleaves motion in free space with motion in contact. The need for free-space motion is obvious: motion in free space is efficient, easy to control, and generates low risk for damage to robot and environment. But motion in free space accumulates execution uncertainty, possibly leading to failure to achieve the desired goal. Since contact is a robust and effective way of reducing uncertainty (Lozano-Pérez et al., 1984; Erdmann and Mason, 1988; Eppner et al., 2015), we propose to interleave both types of motions. As a result the planner must reason about the amount of accumulated uncertainty along the path to ensure robust goal attainment, while relying on free-space motion whenever possible.

The advantages of combining motion in contact and in free space are exploited extensively in robotics. For example, strategies that interleave motion in contact and in free space have been the key to success in the DARPA ARM challenge 2011, where robots touched the environment to localize the arm in the world (see Fig. 5.1). We aim to plan such strategies from a description of the scene geometry.

We present a planner called **Contact-Exploiting RRT** (CERRT), based on the RRT planner (Sec. 4.4). This planner finds robust manipulation strategies under uncertainty in robot position, actuation, and world model. The planner scales to high-dimensional configuration spaces. The resulting motions make and break contact with the environment, slide along surfaces, but also avoid collisions with links that have no contact sensing capability.

Our planner overcomes the computational challenges of planning under uncertainties by exploiting two of the strategies discussed in Sec. 4.8: 1) We model the robots state as a set of

Figure 5.1 Contact can efficiently reduce the uncertainty about the robot's state. *Left*: An example from a manipulation challenge, where the robot first touches the door to localize the handle before attempting a grasp (Righetti et al., 2014). *Right*: A similar strategy, generated by our planner CERRT. Shown in green are several executions of the motion under uncertainty, which all first contact the door and then slide down to the handle.

particles ($\rightarrow$ sampling). 2) We assume that sensing contact is reliable and handle it as fully observable ($\rightarrow$ ignoring uncertainty). From this assumption follows that measurable contact eliminates uncertainty completely in one dimension. From a belief space planning perspective this is equivalent to a projection of the n-dimensional belief state to a $(n-1)$-dimensional manifold (see Fig. 5.2).

We evaluate the planner's capability to reason efficiently about high uncertainty on a benchmark manipulation planning problem from the POMDP literature. We show how the planner generalizes to more complex problems by increasing the complexity and the dimensionality of the configuration space. We will validate our planning results with simulation and real world experiments for a manipulation task under significant uncertainty. We also provide quantitative results supporting our claim that contact and free-space motion must be interleaved.

The contents of this section are, to a large extent, published in Sieverling et al. (2017). I conceived the project idea and developed the algorithmic concepts. I was the sole first author of that publication and the main contributor to writing and algorithmic implementation. The other authors assisted in prototyping, experimenting, and writing.

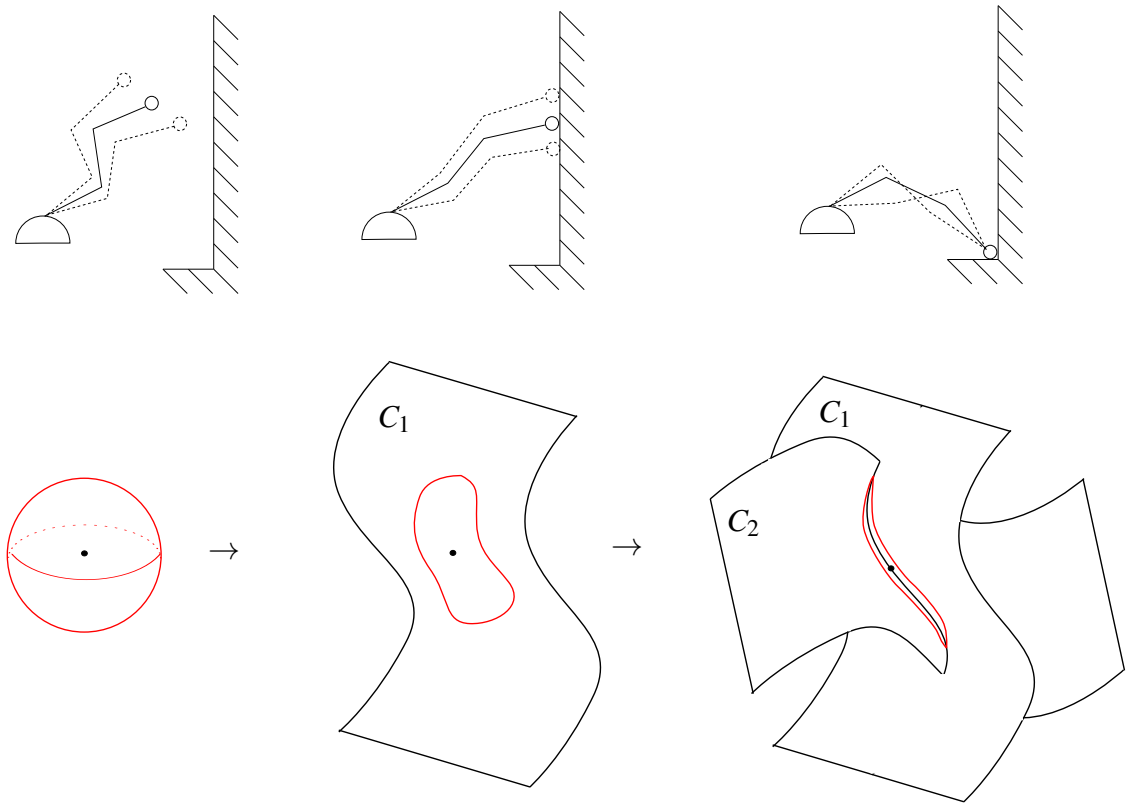


Figure 5.2 Contact reduces uncertainty by projecting a high dimensional belief state onto a lower-dimensional contact manifold. Shown above is a 3-dof manipulator in three different configurations. There is uncertainty in the joint positions which is sketched with dashed lines. Below is a sketch of the configuration space for these three scenes. Shown in red is the distribution of configurations. C_1 and C_2 are the manifolds that describe all configurations in contact with the right or lower wall. In the first step, the robot moves into contact with the right wall, which projects its initial uncertainty onto C_1. In the second step, the robot establishes contact with the lower wall, which projects the distribution on intersection of C_1 and C_2.

5.1 Problem definition

Planning motion for contact-sensing robots requires to combine reasoning about uncertainty with a model for contact sensing. Before presenting our algorithm, we will describe the problem formally.

We plan in the n-dimensional configuration space $\mathcal{C}$. $\mathcal{C}_{\text{valid}}$ is the *valid* configuration space composed of free space $\mathcal{C}_{\text{free}}$ and the configurations in contact at the boundary $\partial\mathcal{C}_{\text{free}}$. The robot can execute actions u which are either straight line joint space motions in free space, guarded motions (moving until the robot is in contact), or compliant slides along surfaces. All motions have an uncertain outcome and the robot can not fully perceive its

configuration but must estimate it from noisy sensors. Additionally, the robot does not fully know its initial configuration. Therefore, instead of planning in configuration space, we plan in belief space $\mathcal{B}$, where each belief $b \in \mathcal{B}$ is a probability distribution over the configuration space. We model the initial state uncertainty as a Gaussian distribution $b_0 := \mathcal{N}(q_0, \sigma_0)$ and use a motion model with independent joint noise $\delta\hat{q} = \delta q + \mathcal{N}(0, \sqrt{|\delta q|}\sigma_\delta)$. We assume the robot has access to reliable contact sensors that reliably detect if the robot is in contact. A key assumptions for efficient planning is fully observable contact sensing, i.e. we assume that no contact is ever detected wrongly. In this first version of the planner we will not use a sensor model for contact sensing, i.e. the robot will not update its belief based on contact measurements.

We call a belief b valid, if it lies mostly in the valid configuration space, i.e.:

$$\int_{q \in \mathcal{C}_{\text{valid}}} b(q)dq > 1 - \varepsilon$$

$\mathcal{B}_{\text{valid}}$ is the space of all valid beliefs. The planning problem is now the following: given a start and goal belief $b_0, b_g \in \mathcal{B}_{\text{valid}}$, find a policy $\pi : \mathcal{B}_{\text{valid}} \to \mathcal{U}$ that brings the robot to the goal belief state with high probability. In this paper we only care about finding feasible policies and do not consider optimality. This problem is a belief space planning problem (Van Den Berg et al., 2011), however, the possibility for the robot to make contact with the environment makes the state non-Gaussian or even multi-modal.

5.2 Contact-exploiting RRT (CERRT)

We now present our motion planner, which finds strategies that move both in free space as well as exploit contact to reduce uncertainty. To capture information about both uncertain configuration and contact, we will define the search space of the planner.

CERRT plans with a combined state of belief over configuration b. We represent the belief with a set of particles $b = \{q_1, \ldots q_N\}$, where each element q is an n-dimensional robot configuration (see Fig. 5.3). We will denote the sample mean and variance of $\mathcal{Q}$ with μ_b and Σ_b. Each belief state is also associated with a fully observable contact $\mathcal{C}(b) = \{c_1, \ldots, c_m\}$. Each contact c is a pair of surfaces in contact $(s_{\text{robot}}, s_{\text{world}})$. s_{robot} is a surface on the robot that has contact sensing capabilities and s_{world} a surface of the environment.

The planner finds strategies that combine free-space and contact motion. In CERRT we assume that free-space motion always increases uncertainty due to the noisy motion model. Because free space motions increase uncertainty, the planner must sequence them with contact motions that reduce uncertainty. Fig. 5.4 shows an example of a decision the

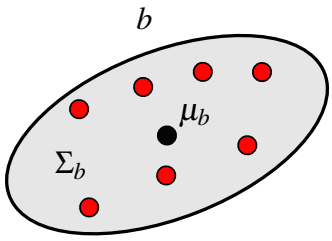


Figure 5.3 The planner's belief state b. The belief is represented by particles q_i (red). The black dot depicts sample mean μ_b and the gray ellipse depicts sample covariance Σ_b.

Algorithm 3 CERRT

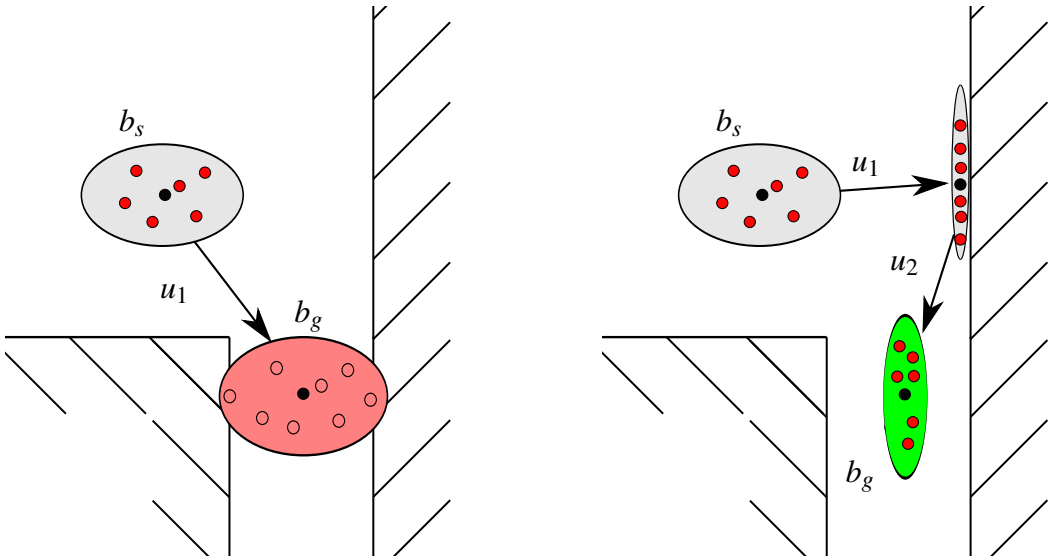
Input: $b_{\text{start}}, b_{\text{goal}}, \varepsilon_{\text{goal}}, \gamma$
Output: $G = (V, E)$

$V \leftarrow \{b_{\text{start}}\}$ *init tree with start state*
$E \leftarrow \emptyset$
while true **do** *search until goal reached*
  $q_{\text{rand}} \leftarrow$ RANDOM_CONFIG() *Sample random node with goal bias*
  $b_{\text{near}} \leftarrow$ NEAREST_NEIGHBOUR$(q_{\text{rand}}, T, \gamma)$ *Find closest belief in tree*
  $u \leftarrow$ SELECT_INPUT$(q_{\text{rand}}, b_{\text{near}}, \gamma)$ *Choose between free space or contact motion*
  $b_{\text{new}} \leftarrow$ NEW_STATE$(b_{\text{near}}, u, q_{\text{rand}})$ *Simulate outcome of action towards* q_{rand}
  if IS_VALID(b_{new}) **then** *Check if belief is valid*
    $V \leftarrow V \cup \{b_{\text{new}}\}$
    $E \leftarrow E \cup \{(b_{\text{near}}, b_{\text{new}})\}$
    $b_{\text{connect}} \leftarrow$ NEW_STATE$(b_{\text{new}}, \text{connect}, \mu_{b_{\text{goal}}})$ *Try to connect to goal state*
    if $\|b_{\text{connect}} - b_{\text{goal}}\| < \varepsilon_{\text{goal}}$ **then** *Test if new node is within goal state*
      return G

planner must take. The robot can not directly enter the narrow passage but must first contact the wall to reduce uncertainty.

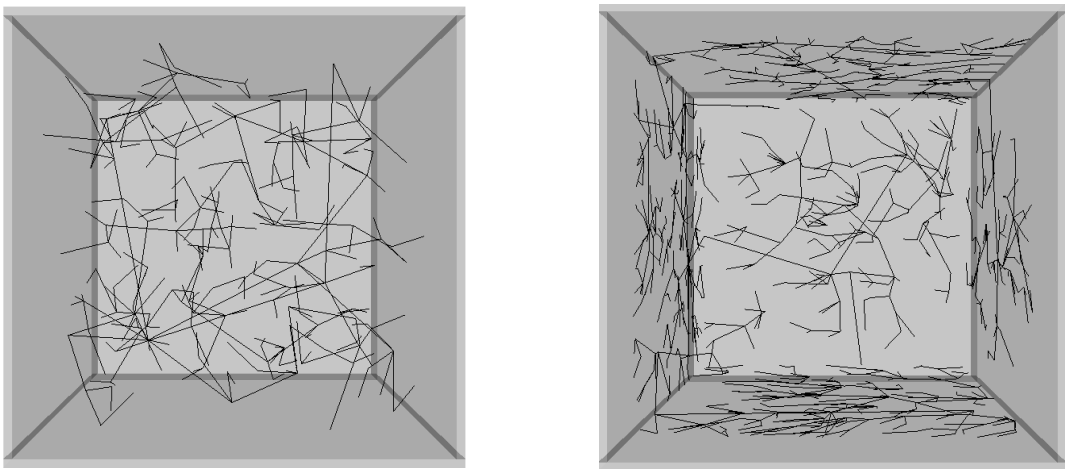
To find such strategies, we grow a tree in the combined space of contact state and belief over configuration. The key to the planners efficiency is a tailored exploration strategy of this space. To adjust the search behaviour of the planner, we introduce a parameter $\gamma \in [0,1]$ that describes the rate with which the planner attempts free-space or contact moves (see Fig. 5.5). If $\gamma = 0$, the planner only explores free space, and behaves like an RRT-Connect with goal bias. If $\gamma = 1$, the planner's only objective is to reduce uncertainty. Values between 0 and 1 balance both objectives.

The Contact-Exploiting RRT (CERRT) is closely related to the kinodynamic RRT (LaValle, 1998) and its structure is given in Algorithm 3 is almost identical to the RRT. However, CERRT differs substantially in the implementation of the subroutines which we will explain in detail in the rest of this section, following the order of the pseudocode in Algorithm 3.



(a) (b)

Figure 5.4 *(a)* To enter the narrow passage, the robot cannot directly take action u_1 because the resulting uncertainty would lead to collision. *(b)* By sequencing a contact move u_1 and a free space move u_2, the robot reduces position uncertainty sufficiently to enter the narrow passage. CERRT finds such sequences of contact and free-space motions.



(a) $\gamma = 0$ (b) $\gamma = 1$

Figure 5.5 The search behavior of CERRT is governed by a free-space/contact exploration bias γ. Shown are two search trees of the CERRT planner exploring the inside of a cube for different values of γ. *(a)* For $\gamma = 0$, the behaviour matches that of a standard RRT. *(b)* For $\gamma = 1$, the planner searches the space of configuration in contact with the walls of the cube. The CERRT planner interleaves both behaviors.

Node selection: NEAREST_NEIGHBOUR

Like the RRT, our planner selects the next node to extend b_{near} with minimal distance to a randomly sampled configuration q_{rand}. Because the node is a belief state we need to define a suitable metric for states x. The choice of metric strongly influences the planning performance (Littlefield et al., 2018). For CERRT, we use a metric that takes the parameter γ into account and can balance the search towards free space or contact motion.

For $\gamma = 0$ we want the tree to expand into free-space quickly, just like the RRT. We achieve this by choosing the node b_n whose mean is closest to q_{rand}. To do so we compute the Euclidean distance $d_\mu(b_n) := \|\mu_{b_n} - q_{\text{rand}}\|$. For $\gamma = 1$ we want to reduce uncertainty by exploring contact space. We achieve this by picking a node with low uncertainty. More specifically, we compute a norm of Σ_{p_n}, which is the covariance matrix of the robot's end-effector position p_n at configuration q_n. We then compute the trace norm, leading to: $d_\Sigma(b_n) := \sqrt{\text{tr}(\Sigma_{p_n})}$. We use this norm mainly because it does not become 0 if the distribution loses support in one dimension (which happens in contact), and also because it is inexpensive to compute. For $0 < \gamma < 1$, we balance the two aforementioned metrics with a convex combination:

$$b_{near} = \text{argmin}_{b_n} \left(\gamma \hat{d}_\Sigma(b_n) + (1-\gamma) \hat{d}_\mu(b_n) \right) \tag{5.1}$$

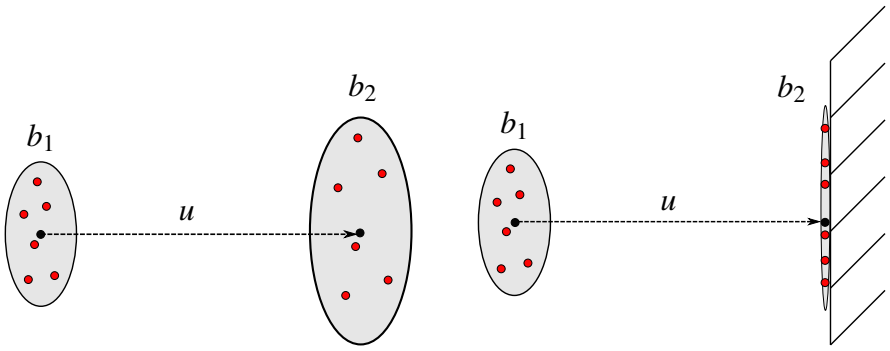
Both distance terms are normalized to the interval $[0, 1]$ by dividing them by the maximum observed value over all samples.

Action selection: SELECT_INPUT

After choosing a node for extension the planner needs to pick the next action. CERRT must have options to move in free space, along contact surfaces, or to switch from free space to contact and vice versa. We implement these options with three different action types. We will briefly introduce them here and give their implementation details in Sec. 5.2.

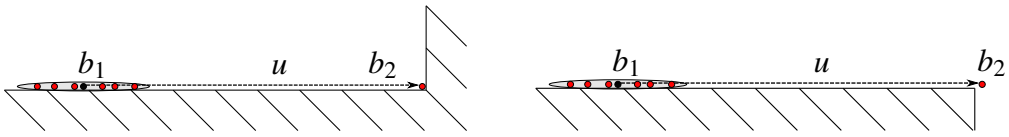
connect: This action attempts a straight line connection in configuration space to the sample q_{rand}. *connect* explores the free space and usually increases position uncertainty (Fig. 5.6a).

guarded: This action moves in the direction of q_{rand} until it establishes contact with the environment. *guarded* is required to switch from free space to contact and always reduces uncertainty in one dimension. (Fig. 5.6b).



(a) Connect (b) Guarded move

Figure 5.6 A free-space move and a move into contact. b_1 and b_2 are the initial and final particle distributions before and after applying action u.



(a) Concave slide (b) Convex slide

Figure 5.7 Two sliding actions. (a) The slide moves the distribution along the surface until it achieves contact with another surface. (b) The slides moves until it loses contact with the surface. Both slides keep uncertainty low in one dimension and reduce it in another dimension.

slide: This action slides along a surface until the contact state changes, either by moving into another contact (Fig. 5.7a) or by leaving the sliding surface (Fig. 5.7b). *slide* explores the space of all contacts, always keeps uncertainty low in one dimension, and can reduce uncertainty in a second dimension.

Our planner selects one of the three actions probabilistically, biased by γ in the following way: if b_{near} is not in contact, it performs a *connect* move or a *guarded* move. If b_{near} is in contact, it *slides* or leaves the contact with a *connect* move. We choose actions based on these distributions:

$$\begin{aligned} p(\text{connect}|\mathcal{C}(b_{\text{near}}) = \emptyset) &= 1-\gamma \\ p(\text{guarded}|\mathcal{C}(b_{\text{near}}) = \emptyset) &= \gamma \\ p(\text{connect}|\mathcal{C}(b_{\text{near}}) \neq \emptyset) &= 1-\gamma \\ p(\text{slide}|\mathcal{C}(b_{\text{near}}) \neq \emptyset) &= \gamma \end{aligned} \tag{5.2}$$

We chose these distributions so that the planner is an RRT-Connect for $\gamma = 0$.

Forward simulation: NEW_STATE

For each of these actions, the planner must be able to reason about the change of uncertainty. We approximate this with a simulation of N noisy actions. The input to the simulation is a motion model $\delta_\alpha(\dot{q})$ with parameter vector α. Examples for the motion model δ are the classical angular and translational motion error for mobile robots or independent error for all joints of the robot.

To extend a node b_1, CERRT samples a particle from $\mathcal{Q}_{b_1}$ and also samples a vector α of parameters of the motion model δ. The extension step then is an invocation of the local planner that executes action u with the motion error δ_α, which we will describe in detail in the next Section. The target of the local planner is q_{rand} with the initial error of the particle added. The extension step is repeated for all particles so that the outcome of the simulation is a new set of particles $\mathcal{Q}_{b_2}$ which is added to the new state b_2.

Algorithm 4 NEW_STATE

Input: b_1, u, q_{rand}
Output: b_2

1: **for** $i \in N_{\text{particle}}$ **do**
2: $q_{\text{near}} \leftarrow \text{SAMPLE}(\mathcal{Q}_{b_1})$ *sample particle from node*
3: $\alpha \leftarrow \text{SAMPLE}(\mathcal{N}(0, \sigma_\delta))$ *sample motion error*
4: $q_{\text{target}} \leftarrow q_{\text{rand}} + (q_{\text{near}} - \mu_{b_1})$ *add the initial error*
5: $q_{\text{sample}} \leftarrow \text{LOCAL\_PLANNER}(a, q_{\text{near}}, q_{\text{target}}, \delta_\alpha)$ *simulate action with one of the local planners (Sec. 5.2)*
6: $\mathcal{Q}_{b_2} \leftarrow \mathcal{Q}_{b_2} \cup \{q_{\text{sample}}\}$
7: **return** b_2

Local planners

Each of the three action types invokes a different local planner. We implement them in the following way:

connect: A connect move is identical to the RRT version. A connect-particle moves on a straight line in configuration space towards the sample q_{rand}, checking for collisions with a fixed resolution of ε. If the particle reaches the sample or moves into contact the motion ends.

guarded: A guarded motion is a connect move in the direction of q_{rand}. A guarded move always ends in contact so it might end before q_{rand} or move beyond q_{rand}.

slide: Sliding motions start with particles in contact and move them along the surface, always maintaining contact. We implement sliding motions as task-space force-feedback controllers with constant orientation. To simulate sliding actions we first choose a random sliding surface (because the node might be in contact with two surfaces at the same time) and then project the end-effector position of the robot in configuration q_{target} onto the sliding surface. The algorithm then alternates between 1) taking a step towards the projected goal, 2) applying the motion error for this step 3) projecting the configuration back on the surface (see Algorithm 5). This is a variant of the projection method (Sec. 4.6), but with the benefit of projecting the effect of the joint-space motion error onto the lower-dimensional manifold of configurations in contact with the environment. This is not easily possible with direct methods. The slide ends if the robot reaches the projected goal, if there is another contact, or if the robot looses contact with the sliding surface (see Fig. 5.7). For all projections we use a damped pseudo-inverse. If the robot is close to a singularity at any step ($\sqrt{\det(JJ^T)} < 0.001$ (Yoshikawa, 1985)) the slide method returns failure.

Algorithm 5 SLIDE

Input: $q_{\text{near}}, q_{\text{sample}}$
Output: q_{real}

$(p_{\text{sample}}, R_{\text{sample}}) \leftarrow T_{EE}(q_{\text{sample}})$
$(p_{\text{surf}}, n_{\text{surf}}) \leftarrow \text{RANDCONTACT}(q_{\text{near}})$ *sample random contact point and surface normal of* q_{near}
$p'_{\text{sample}} \leftarrow p_{\text{sample}} - ((p_{\text{sample}} - p_{\text{surf}}) \cdot n_{\text{surf}}) n_{\text{surf}}$ *project* p_{sample} *on surface*
$\xi \leftarrow T_{\text{near}} - T_{\text{sample}}$
while $\|T_{EE}(q_{\text{robot}}) - T_{\text{sample}}\| > 0$ **do**
  $\Delta q \leftarrow J^{\dagger}(q_{\text{robot}})\,\xi$ *move along surface towards sample*
  $q_{\text{robot}} \leftarrow q_{\text{robot}} + \varepsilon \cdot \Delta\widehat{q}$ *the particles most likely position*
  $q_{\text{real}} \leftarrow q_{\text{robot}} + \delta(\varepsilon \cdot \Delta\widehat{q})$ *the particles actual position*
  while $q_{\text{real}} \in \mathcal{C}_{\text{free}}$ **do**
    $\Delta q_n \leftarrow -J^{+}(q_{\text{new}})\, n_{\text{surf}}$ *move towards surface*
    $q_{\text{real}} \leftarrow q_{\text{real}} + \varepsilon \cdot \Delta\widehat{q_n}$

Node validation: IS_VALID

All nodes in CERRT must have a uniquely defined contact state. To ensure this, we only add those simulation outcomes to the tree that fulfill two requirements:

1. All $q \in \mathcal{Q}_{b_2}$ must either end up in free space or in contact with the same pair of surfaces.

2. If $\mathcal{Q}_{b_2}$ contains configurations in contact, the contact must occur with a link that has a contact-sensor.

The first condition is crucial for our planner's performance because it ensures that the robots contact state is always fully observable. It prevents all actions that end in separate, indistinguishable contacts. In the next chapter, we will not restrict these actions, but rather add contingency branches for the different action outcomes. The second condition allows to treat measurable contact separate from undesired non-observable contact.

After inserting a valid node, the planner attempts to reach the goal state from the newly inserted node, also using forward simulation. If the resulting distribution is close to the desired goal distribution, the planner returns success. Otherwise it moves to the next iteration and picks another sample.

Policy generation

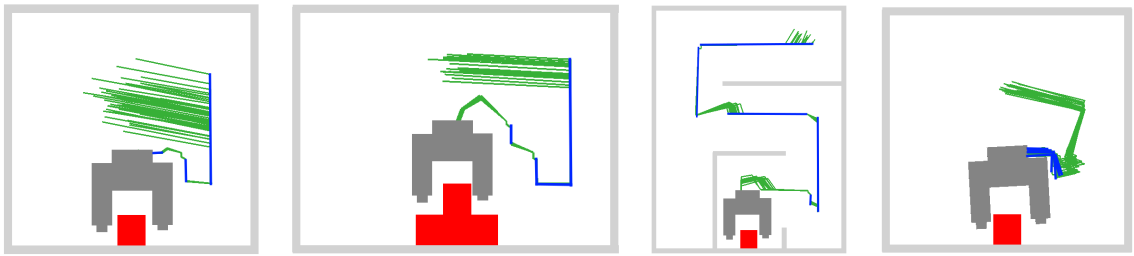
Given a sequence of actions and nodes from start to goal $(u_1, b_1), \ldots, (u_n, b_n)$, we need to generate a policy that can be executed on a robot. This policy is a hybrid automaton that alternates between controllers and contact-based jump conditions. We instantiate one controller followed by one jump condition for each tuple of action and node (u_t, b_t). The type of controller depends on u_t: from *connect* and *guarded* we generate a joint-space velocity controller and from *slide* we generate a compliant task-space controller. The type of jump condition depends on the contact state $\mathcal{C}(b_t)$: If there is contact, the control switch is based on the magnitude of the measured force while for non-contact states, it is based on the covered distance $\|\mu_{b_t} - \mu_{b_{t-1}}\|$. We execute all controllers with low gains to safely make and break contact. This leads to weak tracking performance on the real robot but, as the policy is inherently robust, does not critically affect the outcome.

5.3 Experimental evaluation of the CERRT planner

In this section we will first show policies generated by CERRT for problems from the POMDP literature but also for a high-dimensional manipulation problem. Second, we will analyze the effect of the planner's parameters quantitatively.

Our planner is implemented in the Robotics Library (RL)[1] using the Bullet physics library[2] for collision detection. We executed all experiments on an office PC with a 3.3 GHz Intel Core i5 CPU running the Linux operating system. In all experiments we use a

[1] roboticslibrary.org
[2] bulletphysics.org



(a) (Hsiao et al., 2007) (b) T-Shape (c) Maze (d) $SE(2)$

Figure 5.8 Solutions of the CERRT planner for a grasping scenario adapted from Hsiao et al. (2007). The gripper shows the final configuration of the path. The lines show 20 sampled trajectories, free-space motions are shown in green and slides in blue. The beginning of the paths is always in free space and the end is before grasping. CERRT outperforms POMDP planners on the benchmark (a) and scales to more complex problems.

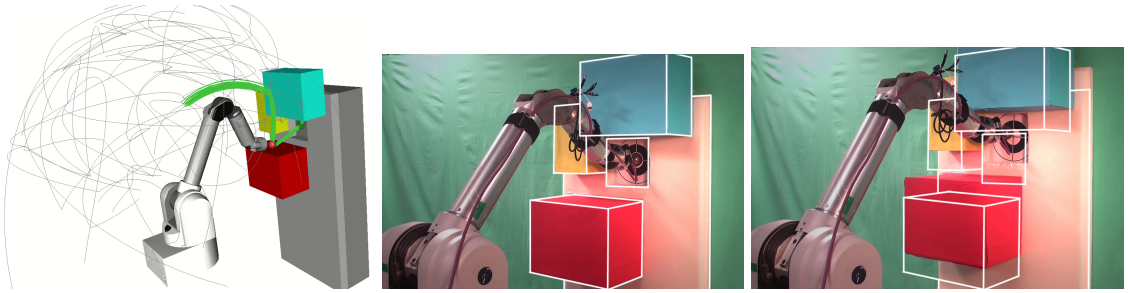
constant number of particles $N = 20$ and a 10% goal bias in the sampler. We always initialize the start belief state x_{start} by sampling N particles from the distribution $\mathcal{N}(q_{\text{start}}, \sigma_{\text{start}})$. All experiments use an independent linear motion error for all joints of $\delta_i(\Delta q) = \mathcal{N}(0, \sigma_\delta \sqrt{\Delta q_i})$.

5.3.1 Performance on manipulation problems

2D grasping: This problem models a gripper picking up a square block at unknown location and is inspired by the POMDP literature (Hsiao et al., 2007; Bai et al., 2010). The gripper has contact sensors at each jaw and can translate in two dimensions. Because of a large initial uncertainty the gripper must contact the object or the walls first and then, after uncertainty is sufficiently reduced, attempt the grasp from the top.

Fig. 5.8a shows one of the solution paths CERRT found on the simple grasping scenario. All policies first establish contact with wall or object and then slide along the ground until contact with the object is perceived. The planning time for this problem averaged over ten runs is 6.8 s (±5.1 s). A POMDP version of the problem with discrete state and actions required an average planning time of 8 s (Kurniawati et al., 2008) and 160 s with continuous state and discrete actions (Bai et al., 2010). Our approach easily scales to more complex scenarios. Fig. 5.8b shows the result for a multi-step piece ($8.2\,\text{s} \pm 6.9\,\text{s}$), Fig. 5.8c a version where the gripper must first navigate through a simple maze ($23.4\,\text{s} \pm 19.3\,\text{s}$), Fig. 5.8d a 3D version of the problem with translation and rotation of the gripper.

7D robot arm motion: CERRT is efficient enough to be directly applied to the seven-dimensional configuration space. We place a 7-dof Barrett WAM robot in front of the wall depicted in Fig. 5.9, similar to the scenario from Phillips-Grafflin et al. (Phillips-Grafflin



(a) (b) (c)

Figure 5.9 The manipulator must touch the target in the square opening of the wall. *(a)* The planner output. Gray lines show all explored motions. The green line is the found path. Our planner finds a strategy that moves to the cyan box, slides down until it loses contact, does a guarded move to the top of the red box, and moves to the target. *(b)* The outcome of executing the strategy on the real robot without uncertainty. The robot reaches the goal precisely. *(c)* We now raise the obstacles by 7 cm (the white overlay shows the wall position from (b)) and execute the policy from (b) again. The robot uses the contact to reduce uncertainty and reaches the target with an error of 2 cm. A video of this experiment can be found at https://www.youtube.com/watch?v=CXaN8ZWRMT0.

and Berenson, 2016). The robot model has an initial uncertainty and a motion uncertainty of $\sigma_{\text{start}} = \sigma_{\delta} = 0.02$. Motion-dependent position error occurs in the real Barrett WAM robot due to stretch of the cables that move the joints. The robot uses a wrist-mounted ATI Gamma force-torque sensor to perceive contact with the end-effector but cannot perceive contact with any other part. We now raise the obstacles by 7 cm (the white overlay shows the wall position from (b)) and execute the policy from (b) again. The robot uses the contact to reduce uncertainty and reaches the target with an error of 2 cm.

The outcome of the planner can be seen in Fig. 5.9. From ten attempts, the planner solved this problem six times within 180 s. The six successful searches required an average time of $23.8\,\text{s} \pm 29.3\,\text{s}$. To validate the robustness of the plan, we introduce an unexpected disturbance. We raise the wall including all obstacles by 7 cm and execute the motion on the robot. The contact with the cyan and red boxes reduces uncertainty and the robot reaches the target with an error of 2 cm which is an effective reduction of 5 cm.

5.3.2 Quantitative analysis of planner parameters

We will now present the results of quantitative experiments that suggest sensible values for the two parameters of the planner: the free-space/contact-space exploration bias γ, and the number of particles N.

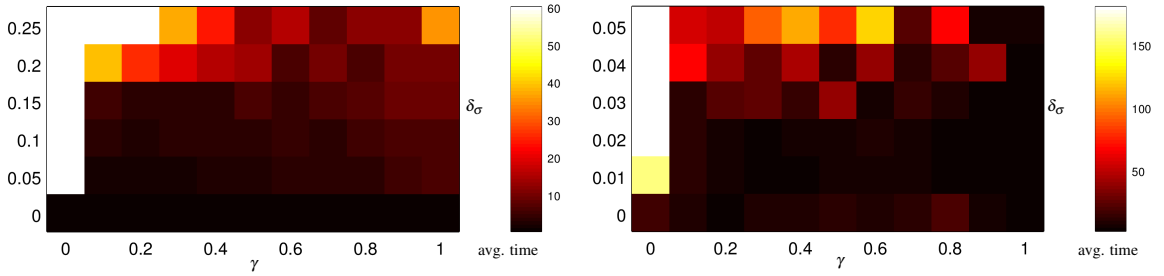


Figure 5.10 Average planning time for different combinations of γ and σ_δ. *Left:* For the 2D scenario from Fig. 5.8b the optimal value of γ depends on the uncertainty. *Right:* for the 7D manipulation scenario from Fig. 5.9, the planner performs best for high values of γ, which lead to a contact-seeking behaviour.

The influence of γ: We executed the planner on two different scenarios: 1) a 2D scenario with narrow passages 2) the 7D manipulation problem from Fig. 5.9. In our analysis we varied γ and the standard deviation of the motion uncertainty σ_δ. We set $\sigma_{\text{start}} = 0$. In both scenarios, we ran the planner ten times each for 66 different combinations of γ and σ_δ. We show the average planning time for each combination in Fig. 5.10. The results show a strong influence of γ on the planning time, depending on the uncertainty.

The border case $\gamma = 0$ corresponds to pure free space search or pure contact motion. For the 2D scenario this is only reliable for problems without uncertainty. The case $\gamma = 1$ corresponds to a pure contact-space exploration. This strategy succeeds in both scenarios because they can be solved by a sequence of sliding motions. For values between 0 and 1 in the 2D scenario the planner always solves the problem. In 2D, free-space exploration is effective as long as uncertainties are low. A value of $\gamma = 0.3$ has the best performance. For high uncertainties more contact must be made and a value of $\gamma = 0.7$ performs best. In the 7D scenario from Fig. 5.9, the planner starts failing for uncertainties higher than 0.02 (we stop the search after 180 s) and free-space exploration is far less effective. We achieved the best results with $\gamma = 0.95$. Our results show that for best planning performance, γ should be tuned to the problem at hand, as some problem require more free-space search and some require more contact.

The number of particles: The second important parameter is the number of particles to consider for planning. A too low number of particles will approximate the belief insufficiently which can lead to a policy with unexpected collisions. The number of particles influences the planning time at least linearly and should be kept low. To find a reasonable number, we ran the manipulator experiment (Fig. 5.9) 20 times varying the numbers of particles. We execute

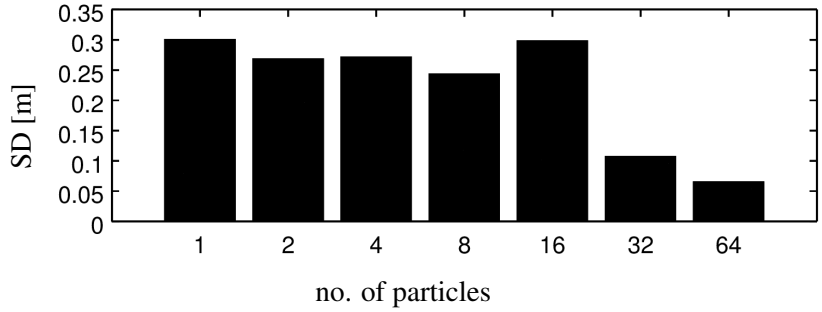


Figure 5.11 The standard deviation of the final position errror drops significantly with 32 particles.

the resulting plans in a dynamic simulation implemented in the RoboticsLab[3] framework and executed each plan 10 times with different motion error Fig. 5.11 shows the results of these experiments. While the average error of the robot's final position is about constant for different runs the standard deviation of the error drops at 32 particles. This suggests that the generated plans are not reliable below 16 particles. A similar number of particles was reported in (Phillips-Grafflin and Berenson, 2016).

5.4 Related contact-based planning approaches

We want to briefly discuss related approaches for motion planning with contact that go beyond the basic methods described in Chapter 4. We will first discuss sampling based motion planners that reason about contact and then belief space planning for manipulation.

5.4.1 Contact-space motion

Sampling-based motion planning can explore the space of configurations in contact (Ji and Xiao, 2001; Siméon et al., 2004; Hauser and Latombe, 2010; Blin et al., 2016) but does not reason about the uncertainty reducing capability. Our planner searches the space of all configurations in contact to exploit its uncertainty-reducing capability.

5.4.2 Belief-space planning in contact

Pre-images and backprojections Classic work in manipulation planning called *fine motion planning* showed how a sequence of compliant motions can be robust to uncertainty. So called Pre-images (Lozano-Pérez et al., 1984) characterize the regions from which compliant actions reach a desired goal state. Chaining them gives uncertainty-tolerant plans. The

[3] simlab.co.kr

pre-image approach generates a sequence of action is guaranteed to lead to a goal state, even under significant uncertainty. This framework was used to find manipulation strategies that bring objects with unknown position into a desired state, without any sensing (Erdmann and Mason, 1988; Goldberg, 1993). Unfortunately, fine motion planning does not directly apply to manipulation problems because computing preimages is a PSPACE-hard problem (Natarajan, 1988). To overcome the computational complexity, instead of backwards-chaining, our CERRT planner uses forward simulation to approximate pre-images. Therefore it can be seen as a sampling-based version of the classic pre-image planning.

Planning with uncertain, compliant actions Some planners in the literature assume uncertain actions which are allowed to make contact. This results in an MDP formulation. The challenge for these approaches is that simple state spaces such as Gaussians do not adequately represent the belief state of configurations in contact. This rules out using efficient LQR-based solutions introduced in the previous chapter.

Most approaches overcome this challenge by approximating the distribution with samples. Very related to our method are particle-RRTs (Melchior and Simmons, 2007; Phillips-Grafflin and Berenson, 2016; Wirnshofer et al., 2018) which represent the outcome of actions as a set of particles, just like our planner. An alternative approach is taken by Guan et al. (2018), who discretize state space first and then construct transition functions from Monte-Carlo simulation.

There are three important differences of particle-RRTs to our work: 1) The particle-RRT assumes perfect knowledge about robot state which CERRT does not. This allows our planner to solve a broader class of problems. 2) Our method explicitly seeks contact to reduce uncertainty, while the particle-RRT just achieves contact randomly. We believe this is the reason for our planner's efficiency. 3) CERRT generates only one sequence of free-space and contact-motions while the particle-RRT has actions with multiple outcomes. This makes the particle-RRT's behaviour more robust to failure. We will extend CERRTs capabilities regarding this limitation in the next chapter.

POMDP planning with contact POMDPs were used to solve simple 2D grasping problems (Hsiao et al., 2007) that use contact as feedback. However these approaches scale badly in high-dimensional spaces. While there exist solvers for continuous state (Bai et al., 2010) and action spaces (Seiler et al., 2015), they can not easily be applied to the high-dimensional configuration space of a robot manipulator. Our method tackles the high complexity by planning with a belief over the robot configuration but a fully observable contact-state. This

moves our problem in the domain of Mixed-observability MDPs (Ong et al., 2010) which are easier to solve.

POMDP solvers were applied to low-dimensional versions of manipulation tasks such as in-hand manipulation to localize an object (Koval et al., 2016b) or pre-grasp manipulation (Hsiao et al., 2007; Bai et al., 2010). The latter application is relevant to our method and we showed the uncertainty-reducing capabilities of our planner on the same problem in Section 5.3, but with two important differences: 1) unlike the POMDP-approaches (Hsiao et al., 2007; Bai et al., 2010) our method does not assume any a priori discretization of state or action space. It can be directly applied using the geometric model of world and robot as input. 2) we do not assume uncertain contact sensors while in the POMDP scenario sensors can return false measurements, which makes up a large part of the complexity. We think that our noise-free assumption is justified for undirected, binary contact-sensing.

Submodularity Touch-based localization of the robot relative to a known environment can be casted as an optimization of a submodular metric (Javdani et al., 2013), which a greedy approach solves efficiently and near-optimally. Combined with trajectory optimization (Vien and Toussaint, 2015) this leads to efficient planners that exploit contact to reduce uncertainty and scale to realistic manipulation problems. However, the submodularity property does not hold in the setting of this paper because all motions in free-space increase uncertainty.

Another related work (Toussaint et al., 2014) optimizes trajectories for contact-exploiting motion by adding uncertainty reduction as an objective. This method does not perform a global search like our method but might be used to post-process the paths found by our planner.

Chapter 6

Contingent contact-based motion planning

The CERRT planner as introduced in the last section produces a single sequence of actions. However, high uncertainty might lead to a single action having possibly several different outcomes. To be prepared for this, a motion planner must in advance determine suitable reactions and capture these alternatives in branches of a motion plan. During the execution of such a plan, the planner can use sensor data to select among these branches the one matching the current situation. A plan able to address such eventualities is called a **contingency plan**; a planner producing it is called a **contingent planner** (Pryor and Collins, 1996). In Part I, we have seen how contingency plans lead to more robust, resilient systems. We identified contingent planning as one of the requirement for robust manipulation extracted from our use case (Sec. 3.5). In this chapter we will extend the CERRT planner from the previous chapter to handle contingencies.

To find contingency plans, a belief-space planner can model the sensing capability of the robot and reason probabilistically about sensor events that might happen during execution. However, taking into account all possible eventualities is not possible. The high-dimensionality and the continuous state and action space in manipulation problems make global, complete solutions to contingent planning intractable. To be effective, a planner must make approximations.

We present the **Contingent Contact-Exploiting RRT** (ConCERRT)—a planner that overcomes this complexity for contingencies based on contact sensing. The planner finds robust strategies in configuration space for problems such as the one shown in Fig. 6.1. These problems require the robot to adapt its motion based on sensor information obtained from contact sensors. The key to our planner is again the assumption that contact sensing is uncertainty-free. This assumption allows to rule out a large part of the robot's state

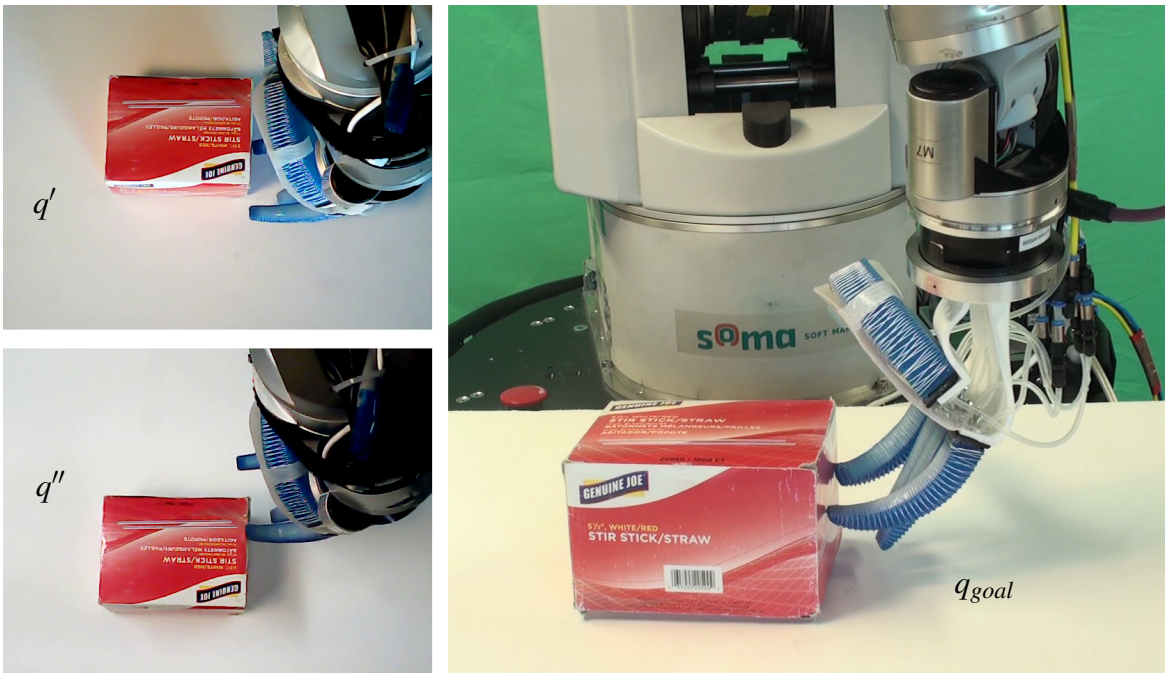


Figure 6.1 A robot performs tactile localization of an object using contact sensors on two fingers of a soft hand. By moving into contact and measuring which finger makes contact first, the robot can estimate the position of the box relative to itself and then adapt its action to reach q_{goal}. Our planner finds such strategies using models of the world and the uncertainty.

space, given a contact measurement. This simplifies planning and leads to a low number of contingencies.

Compared to POMDP approaches, our planner does not require any a priori discretization of configuration or action space. Instead, it builds a task-specific discretization of the state space during planning, informed by the available actions, similar to sampling-based motion planners. In our experiments, we show that our planner consistently finds successful contingency plans in realistic applications. We show solutions for problems with high uncertainty where non-contingent planners fail. In real world experiments, we show that our assumptions about uncertainty hold in realistic scenarios.

The content of this chapter has been published before in Páll et al. (2018). I was the second author of that paper. The first author conceived, implemented, and evaluated the proposed ConCERRT algorithm. I gave scientific advice and helped with experimentation. The first author and me contributed equally to writing.

6.1 ConCERRT

In this section, we present the Contingent Contact-Exploiting RRT planner (ConCERRT), a sampling-based motion planner that finds contingency plans based on contact sensing. Before describing the algorithm, we will first explain the sensor model and two insights that are crucial for understanding our planner.

6.1.1 Sensor model

Compared to the CERRT planner, ConCERRT requires the definition of a sensor model. The output of this model is used to detect contingencies. In our experiments we use two sensor models inspired by haptic sensors.

The *Tactile sensor model* assumes the robot can sense binary contact on different parts: $\mathcal{O}_{\text{tactile}}(q) = \{o_1, \ldots, o_k\}$, where each contact observation o_i is a sensor patch indicating contact in the given configuration.

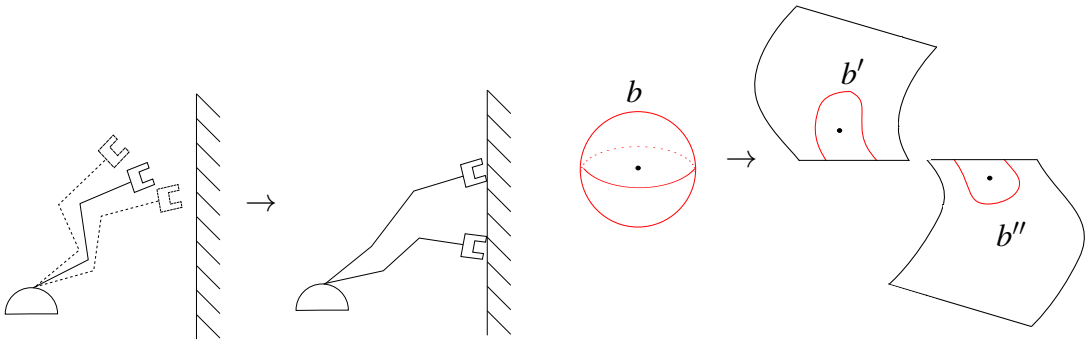
The *Force sensor model* assumes the robot can detect the contact normal: $\mathcal{O}_{\text{force}}(q) = \{o_1, \ldots, o_k\}$. Each observation $o_i = (c_i, n_i)$ is a pair of sensor patch c_i and wall normal n_i.

6.1.2 Belief state partitioning

The first insight is that a robot can effectively reduce uncertainty by moving into contact and observing the resulting contact measurement to rule out parts of its state space. For example, a robot (Fig. 6.2) with uncertain state moves towards a wall until it touches with the left or the right finger. By visualizing this in belief space, the reduction of uncertainty becomes obvious: the contact event partitions the belief in two halves, each with less uncertainty than the original belief.

These belief space partitions can also arise out of contact direction sensing which we show in Fig. 6.3. Here the robot (in this case a 2D point robot) can sense the normal at the point of contact. The robot now moves towards an edge and matches the sensor reading to the wall normals. Just like before, the result of this action is the partition of a large belief into two smaller belief states, which is an efficient reduction of uncertainty.

This is the first insight exploited by the ConCERRT planner: some actions reduce uncertainty by partitioning the belief space. The ConCERRT planner exploits these actions and assembles them into robust policies.



(a) (b)

Figure 6.2 Belief state partitioning with binary contact sensing; *(a)* A 3-dof robot with a two finger hand moves from free space into contact. The initial position uncertainty results in two different contact states. *(b)* This action partitions the belief b into two belief states b' and b'', in contact with the left and right fingers respectively.

6.1.3 Incremental policy construction

Using belief-space partitioning actions in a planner is not straightforward as every added partition adds at least two new belief states to the policy. Both states must be eventually connected to the goal. However, we will show now that this effort can be limited in practice, as a planner can reuse sub-solutions to speed up planning.

The working principle of ConCERRT is shown in Fig. 6.4. ConCERRT maintains two separate lists of belief states:

- $\mathcal{B}_{\text{open}}$ contains all belief states that are yet to be connected to the goal. It is initialized with the initial belief of the robot and increases with every belief space partition. If $\mathcal{B}_{\text{open}}$ is empty, the planner returns success.
- $\mathcal{B}_{\text{connected}}$ contains all beliefs that are already connected to the goal. Initially, it only contains the goal belief b_g, however over time the planner expands the policy and adds elements to $\mathcal{B}_{\text{connected}}$.

ConCERRT now runs for every state in $\mathcal{B}_{\text{open}}$ a separate tree search, attempting to connect to any state in $\mathcal{B}_{\text{connected}}$. If it can connect any node from $\mathcal{B}_{\text{open}}$ to any node from $\mathcal{B}_{\text{connected}}$, it adds the resulting action sequence to the policy and it also adds all nodes visited by that path to $\mathcal{B}_{\text{connected}}$. If this sequence results in any new partitions, it adds it to $\mathcal{B}_{\text{open}}$ and creates a new tree for each of them.

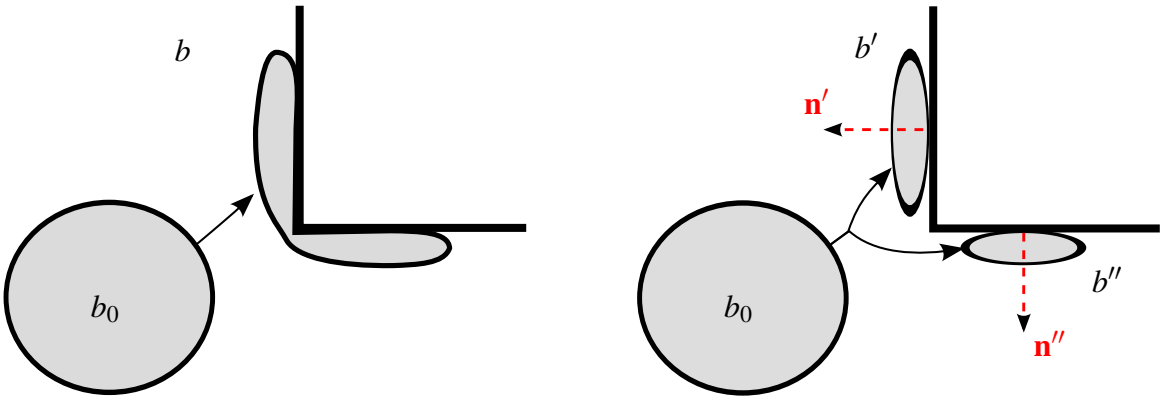


Figure 6.3 *Left*: a robot with state uncertainty moves from state b_0 towards a corner in the world, which projects the uncertainty on the surfaces. *Right*: measuring the contact normal ($\mathbf{n}'$ or $\mathbf{n}''$) allows the robot to partition its belief state b into lower uncertainty states b' and b''.

This parallel search using a whole forest of trees might seem like an overhead. However, the effort is limited in practice, which can be explained by the algorithm moving from exploration to exploitation(see Sec. 4.5). Initially the algorithm must explore most of the space as $\mathcal{B}_{\text{connected}}$ contains only one element. But whenever adding a path to the policy, the algorithm also adds states to $\mathcal{B}_{\text{connected}}$. At some point nodes in $\mathcal{B}_{\text{connected}}$ will cover most of the state space. All these beliefs are opportunities for exploitation which decrease the complexity of later iterations.

6.1.4 Algorithm outline

Based on the two previous insight we can now give the full description of the ConCERRT planner (Alg. 6). Just like CERRT, we represent the belief with a set of particles $b = \{q_1, \ldots, q_N\}$ and denote the sample mean and covariance of b with μ_b and Σ_b respectively.

Different to CERRT, the planner maintains a list of trees $T_i = (V_i, E_i)$. ConCERRT initially samples a fixed number of particles from the initial belief b_{start} and adds them as root to the initial search tree $T_{b_{\text{start}}}$. Then, in every iteration, ConCERRT cycles through all elements of $b_i \in \mathcal{B}_{\text{open}}$ and expands the respective tree T_{b_i}. The expansion of a single tree works exactly like in the CERRT planner: it samples a random configuration, finds the nearest neighbour in the current search tree, chooses an action, simulates the effects of that action, adds the resulting state to the tree, and tries to connect the new state to the goal(s). We will now give implementation details for the expansion. The numbers in parentheses refer to the lines in Alg. 7.

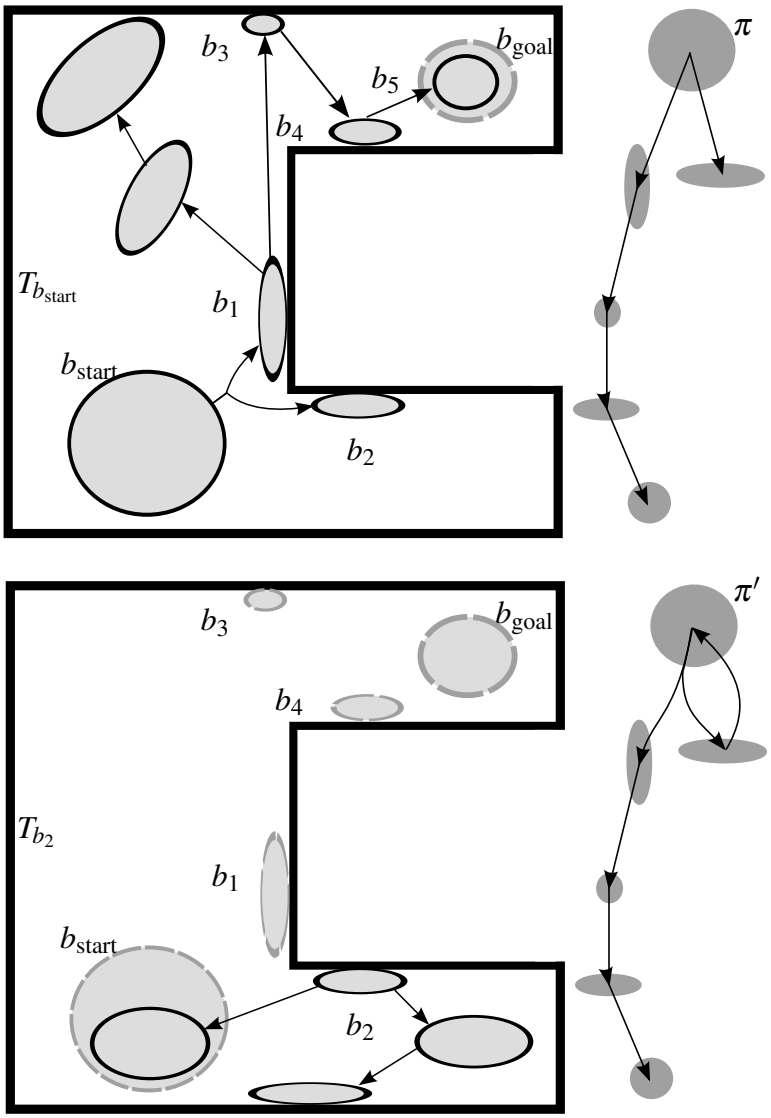


Figure 6.4 Two iterations of the ConCERRT planner: *Top left*: the initial search tree $T_{b_{\text{start}}}$ connecting start and goal beliefs b_{start} and b_{goal}. *Top right*: the resulting policy π, which consists of one path from start to goal but also contains one unconnected partition b_2. *Bottom left*: the next iteration of the algorithm. Starting from b_2, the algorithm builds a new search tree T_{b_2} that can connect to any of the beliefs in π. The planner finds a path that can reconnect by moving back to b_{start}. *Bottom right*: This path is added to the final policy π'.

Algorithm 6 ConCERRT

Input: $b_{\text{start}}, b_{\text{goal}}$
Output: π

$\mathcal{B}_{\text{open}} \leftarrow b_{\text{start}}$
$\mathcal{B}_{\text{connected}} \leftarrow b_{\text{goal}}$
$V_{b_{\text{start}}} \leftarrow \{b_{\text{start}}\}$
$\pi \leftarrow \emptyset$
while $P(\pi) < 1$ **do**
  for all $b \in \mathcal{B}_{\text{open}}$ **do**
    $T_b \leftarrow T_b.\text{EXPAND}(\mathcal{B}_{\text{connected}})$
    $\pi \leftarrow \pi.\text{UPDATE}(T_b)$
    $\mathcal{B}_{\text{open}} \leftarrow \mathcal{B}_{\text{open}}.\text{UPDATE}(T_b)$
    $\mathcal{B}_{\text{connected}} \leftarrow \mathcal{B}_{\text{connected}}.\text{UPDATE}(T_b)$
return π

Sampling (1) The planner samples a random configuration which is the target of the current tree extension. The tree growth is randomly biased towards the goal by choosing $\mu_{b_{\text{goal}}}$ instead of a random sample with a fixed probability $p(b_{\text{goal}})$.

Nearest neighbour search (2) The nearest neighbour selection computes a norm that balances uncertainty and Euclidean distance. Just like in CERRT, we use $d_\Sigma(b) := \sqrt{\text{tr}(\Sigma_b)}$, $d_\mu(b) := \|\mu_b - q_{\text{rand}}\|_2$, and a bias-parameter $\gamma \in [0,\ 1]$:

$$b_{near} = \text{argmin}_b \left[\gamma \left(d_\Sigma(b) + \sum_{b' \in Sib(b)} d_\Sigma(b') \right) + (1-\gamma) d_\mu \right] \tag{6.1}$$

$Sib(b)$ denotes the set of all siblings of b , i.e. the partitions that were reached from the same action. This norm is equivalent to the norm used in CERRT (5.1), except for the summation term. Without this term, the planner would be overly optimistic and prefer actions that lead to low uncertainty states, but might also lead to other states with higher uncertainty. The sum term instead lets the planner take all possible outcomes into account.

Action selection and simulation (3,4,5) The planner chooses an action u randomly and then simulates it. The actions and their implementation is identical to the CERRT planner and discussed in Section 5.2.

Belief state partitioning (6) If b' is valid, we apply the contact sensor model to find potential partitions of the belief state. For each particle $q' \in b'$, we compute $\mathcal{O}(q') = \{o_0, \ldots, o_k\}$. We then cluster the belief b' into $\{b'_{o_0}, \ldots, b'_{o_k}\}$, such that particles with the same measurement are in the same belief. The implementation is different for the two sensor

models: For the *tactile* sensor model, we cluster based on the sensor patches that are in contact. For the *force* sensor model, we cluster two particles into different beliefs if the difference between their measured normals is larger than 15°. We estimate the transition probabilities as $p(b'_{o_i}|b,u) \approx \frac{|\mathcal{Q}(b'_{o_i})|}{N_{\text{particles}}}$

Goal connect (9) We add the new beliefs b'_{o_i} to the tree and try to connect them to any belief in $\mathcal{B}_{\text{connected}}$. To do so we simulate a noisy connect action towards every $b_{\text{goal}} \in \mathcal{B}_{\text{connected}}$ resulting in a new distribution b''. We check if b'' lies within the goal belief by testing if $d_M(q) < \varepsilon_M = 2$ for all $q \in b''$, where $d_M(q)$ is the Mahalanobis distance between q and b_{goal}. If this test succeeds, ConCERRT UPDATEs the policy π with all beliefs on the solution path, $\mathcal{B}_{\text{connected}}$ with all new beliefs that were connected to the goal, and $\mathcal{B}_{\text{open}}$ with all new partitions that are not yet connected to the goal, as described in Section 6.1.3.

Algorithm 7 T.EXPAND()

Input: $\mathcal{B}_{connected}$

```
q_rand ← RANDOM_CONFIG()
b_near ← NEAREST_NEIGHBOUR(q_rand, T)
u ← SELECT_ACTION(q_rand, b_near)
b' ← SIMULATE(q_rand, b_near, u)
if IS_VALID(b') then
    B_contingencies ← BELIEF_PARTITIONING(b')
    for all b'' ∈ B_contingencies do
        T ← T.ADD_BELIEF(b'')
        T ← T.GOAL_CONNECT(b'', B_connected)
return T
```

6.2 Experimental evaluation of the ConCERRT planner

We evaluate the planner in simulation and real world experiments to show that a) ConCERRT scales up to high dimensional problems compared to other belief space motion planners. b) the contingency branches of ConCERRT allow to solve problems with significantly higher uncertainty than comparable non-contingent contact-based planners. c) ConCERRT policies are robust enough to be executed on real world systems. We implemented all experiments using the Robotics Library (Rickert and Gaschler, 2017). All experiments were carried out on a desktop computer with an Intel i5 3.5GHz processor. Table 6.1 gives the values of the planner's parameters for all experiments.

Param.	**Description**	**2D gripper**	**7D sim robot**	**7D real robot**
t [min]	time budget	10	16.66	16.66
γ	contact/free-space bias	0.7	0.9	0.85
N	number of particles	50	40	100
δ_{step}	simulation step size	0.05	0.5	0.5
$p(b_g)$	goal bias probability	0.1	0.1	0.3
σ_{init}	initial uncertainty	$[\sigma,\sigma]$	0	$[2,2,2,3,3,3,3]\times 10^{-2}$
σ_{motion}	motion uncertainty	$[\sigma,\sigma]$	$[\sigma,\sigma,\sigma,\sigma,\sigma,\sigma,0]$	$[1,1,1,2,2,2,0]\times 10^{-2}$
ε_d	dist. threshold to goal	0.2	0.03	0.035

Table 6.1 Planning parameters

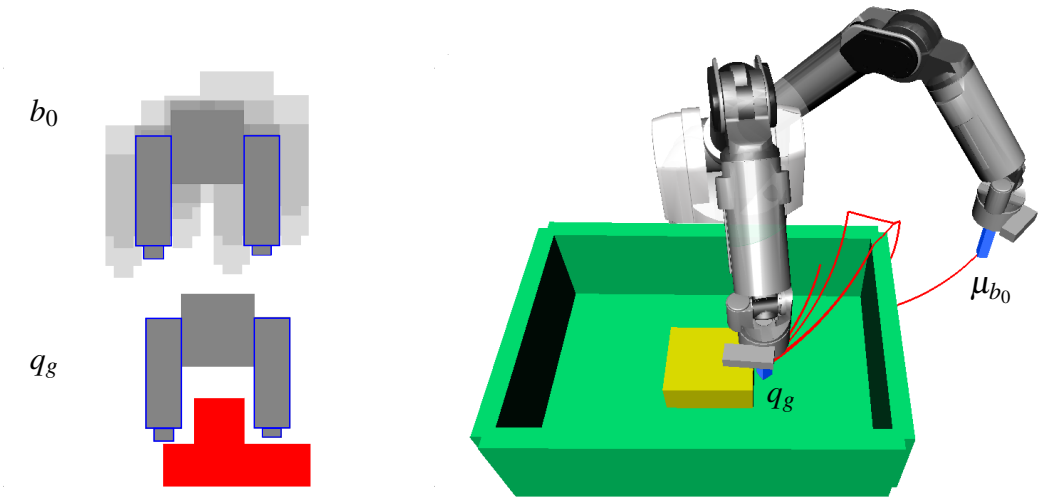


Figure 6.5 *Left*: 2-dof gripper problem with initial distribution b_0 and goal configuration q_g. The finger tips on the gripper (gray) can sense contact with the object (red). *Right*: 7-dof problem. The right configuration shows the mean of the initial belief μ_{b_0}. The goal configuration q_g is inside the green container next to the yellow box. The blue rod at the end-effector is a force sensor. One policy computed by ConCERRT is shown with red lines.

6.2.1 ConCERRT scales to high dimensional problems

POMDP-based belief space planners rely on pre-defined discretization which fails in complex environments. We now validate in simulation that ConCERRT scales to high-dimensional state space with complex contact states. The first experiment (Fig. 6.5 left) models a gripper with contact sensors on the fingers and the fingertips (similar to the setup in Koval et al. (Koval et al., 2016b)). The gripper can translate in two dimensions. Compared to a similar problem from the POMDP literature (Hsiao et al., 2007), there are no outer walls limiting the workspace which increases the difficulty of the problem. In the second experiment (Fig. 6.5 right), a Barrett WAM 7-dof arm must reach into a rectangular container and touch a yellow

obstacle. To reduce uncertainty, it can measure the contact normal of the surfaces with a stick-shaped end-effector.

The results for the second problem (Fig. 6.7) prove that the planner is efficient enough to compute policies directly in 7-dimensional configuration space. The planner finds policies under significant motion uncertainty that slide along the walls of the container to localize the yellow box. We do not show results for uncertainties $\sigma > 0.4$, because the simulation of the sliding action becomes unreliable for extremely high motion uncertainties. Without a fixed discretization, this problem is not solvable for POMDP-based motion planners (Hsiao et al., 2007; Koval et al., 2016a) which become intractable for high number of dof and complex contact manifolds. ConCERRT also handles a significant amount of motion uncertainty. This relaxes the assumptions of related approaches (Koval et al., 2016b) which require the inverse kinematics of the robot and fully observable joint states. Thus, we are able to solve a larger set of problems. ConCERRT also relaxes the assumptions of Particle-RRT motion planners (Phillips-Grafflin and Berenson, 2016) as it does not require reversible actions and fully observable joint states.

6.2.2 Contingent planning increases robustness

Contingency plans capture many possible execution states and find appropriate reactions. Therefore, they should be more robust than plans without contingencies. To validate this, we ran the ConCERRT planner on the simulation scenarios (see Sec. 6.2.1), varying the amount of uncertainty.

We compare our planner against two baselines. The first baseline is an uncertainty-unaware RRT-Connect with goal bias (RRTCon) (Kuffner and LaValle, 2000). The second baseline is the CERRT planner. To compare contingent and non-contingent planners we must define a suitable scoring function taking into account both the planning time and the quality of the plan. Therefore, we compute the score as $P_{\text{success}} = P(\pi) \cdot \frac{N_{\text{succ}}(t)}{N}$, where $P(\pi)$ is the success probability of the policy (equal to 1 for CERRT) and $\frac{N_{\text{succ}}(t)}{N}$ is the ratio of found solutions. We run 30 experiments per setup for the 2-dof and 10 experiments for the 7-dof problems.

In Fig. 6.6 we show results for the 2-dof problem. The RRT Connect always returns a trajectory which fails even with little uncertainty ($N_{\text{succ}} = 0$ for $\sigma \geq 0.05$). The solution quality of the CERRT planner drops to 0% while ConCERRT solutions' quality stays over 50%, even for the highest uncertainty.

The result in Fig. 6.7 shows that ConCERRT substantially outperform the baselines in terms of robustness to uncertainty if the dimensionality increases. In Fig. 6.8 we show how the solution quality improves over planning time for the 7-dof problem with different values

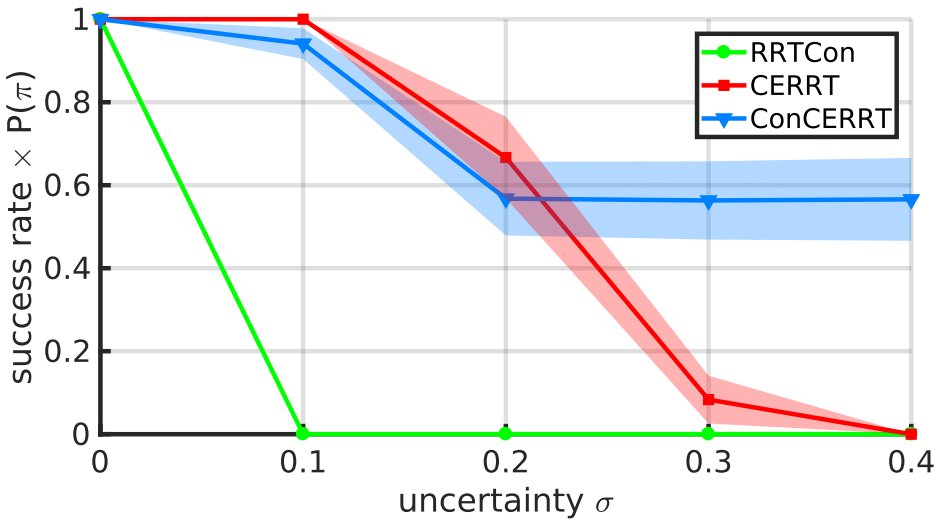


Figure 6.6 Success rate of RRTCon, CERRT, and ConCERRT on the 2-dof gripper problem as function of the position uncertainty. ConCERRT still finds solutions for $\sigma > 0.3$ where the conformant planner fails.

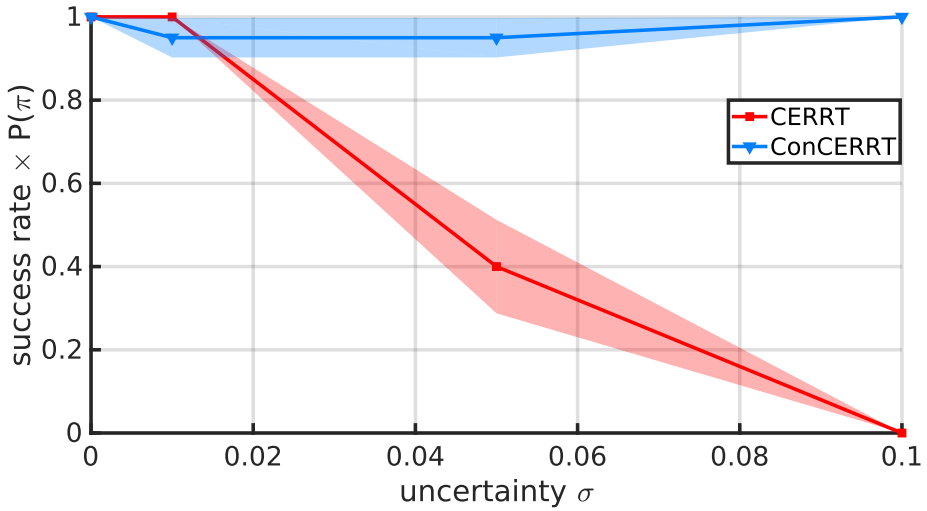


Figure 6.7 Success rate comparison of CERRT and ConCERRT for a 7-dof manipulator with force sensing. Our contingent planner can handle up to ten times higher uncertainty than a conformant planner.

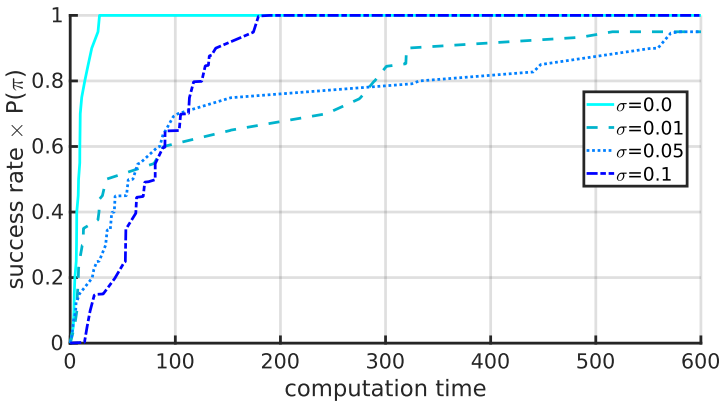


Figure 6.8 The success probability for different ConCERRT policies with increased computation time. The solution quality of the policy increases over time until all possible contact events are covered.

of uncertainty. Interestingly, low uncertainties do not necessarily lead to lower computation times for ConCERRT. We believe this is due to the planner committing to suboptimal contingencies too early, i.e. choosing a contingent plan when a conformant strategy would be possible. The planner finds policies that succeed in 50% of the runs under one minute. The policies improves as time goes on, approaching 100%. This shows the anytime property of ConCERRT.

6.2.3 Validation on real robot

To validate the policies generated by ConCERRT in a real world application, we applied the generated policies on a 7-dof Barrett WAM robot arm with a soft hand (RBO Hand 2 (Deimel and Brock, 2016)) as the end-effector. The experiment is inspired by the problem in Koval et al. (Koval et al., 2016b) where a WAM arm equipped with a contact sensing hand localizes an object on a table surface. Similarly, our task is to sequence contact motions that reduce uncertainty enough such that the hand stops centered in front of the box. Fig. 6.1 shows the experimental setup. The fingers of a soft hand deform when they contact the environment which results in a measurable pressure change. We use large changes in pressure as a proxy for contact sensing. We only use the partially inflated index and little fingers as contact sensors. To find a policy we run the ConCERRT planner as initially but we exclude the slide action from planning as reliable sliding is hard to implement with a soft manipulator.

ConCERRT consistently finds feasible policies in 16.66 minutes. One computed policy π with $P(\pi) = 1.0$ is shown in Fig. 6.9. The policy executes multiple motions in front of the box, expecting no contact, however the policy also contains contingencies for the contact

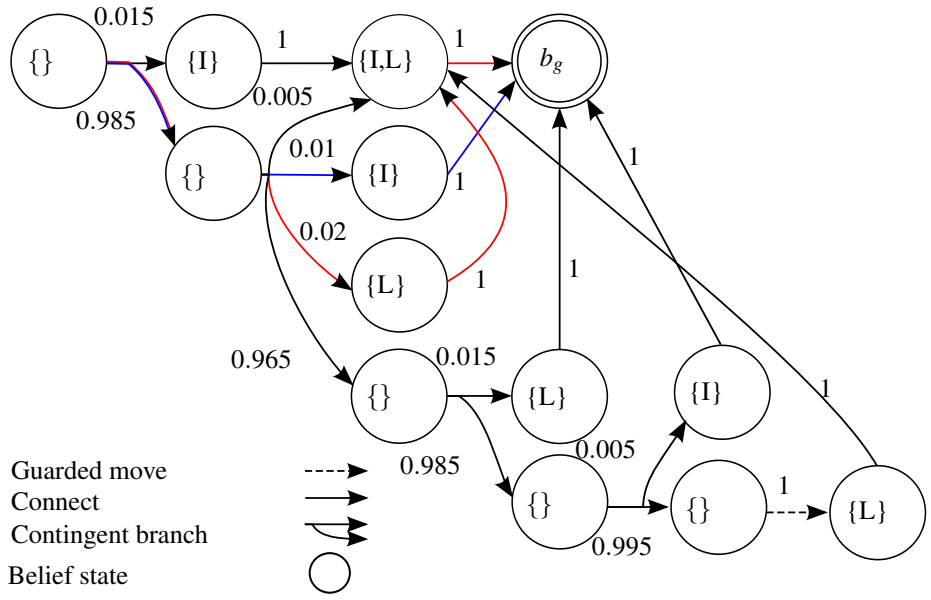


Figure 6.9 The ConCERRT policy as executed on the real robot. Circles are belief states with contact state $\{I\}$ (index finger), $\{L\}$ (little finger), $\{I,L\}$ (both fingers), or $\{\}$ (no contact). The edges with the respective probabilities are actions that move the robot to the goal state b_{goal}. The edge coloring shows the path taken by the robot in Fig. 6.10

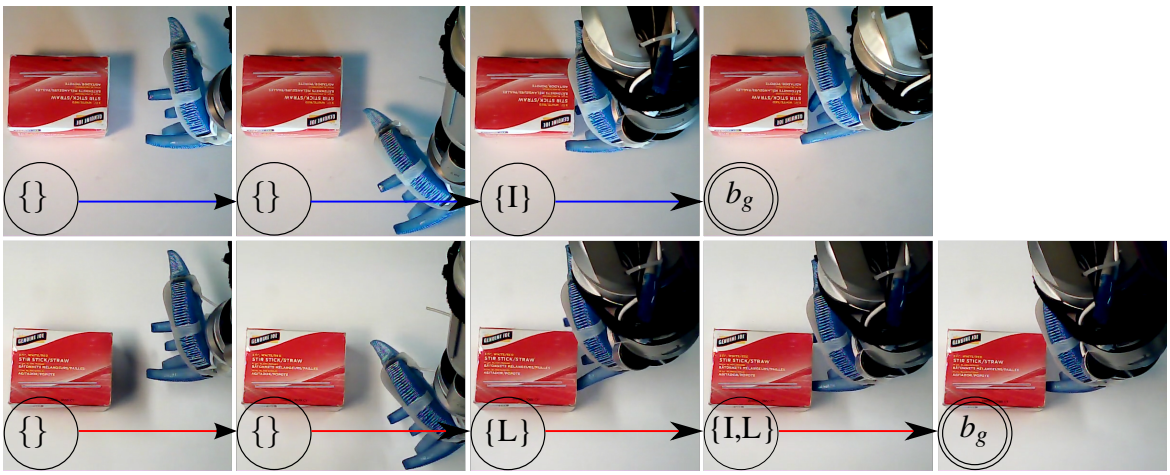


Figure 6.10 Snapshots of two executions of the same policy found by the ConCERRT planner. *Top*: Box displaced +2 cm—the index finger makes contact first. *Bottom*: Box displaced −4 cm—the little finger makes contact first. The shown paths correspond to the blue/red edges in Fig. 6.9. See https://www.youtube.com/watch?v=NaRppcg0CtQ for a video of this experiment.

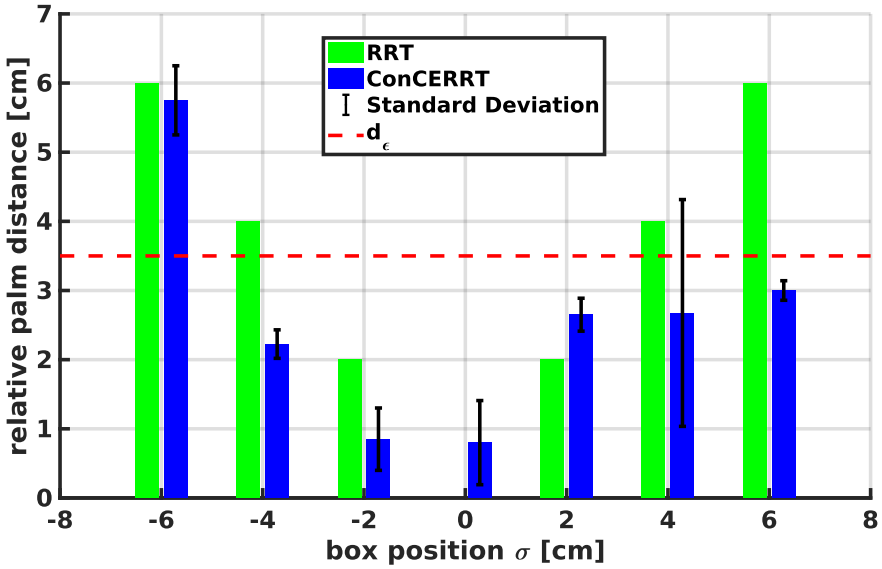


Figure 6.11 Relative position of hand and object after execution of different policies for different object displacements. The real robot can localize the box until up to 4 [cm] position uncertainty.

case. The most likely path through the graph does four of these free space motions and then executes a guarded move to ensure the final contact. To evaluate the robustness of the policy, we move the box 0, 2, 4, and 6 cm to the left and right relative to the hand's initial position. We execute the policy four times for each displacement while keeping the initial hand position constant. Fig. 6.10 shows the execution of two strategies with two different box displacement.

For the given uncertainty model, CERRT is not able to find a conformant solution, thus we can only compare the execution to a uncertainty-unaware planner such as RRT-Connect (RRT). Here we assume that the robot executes the RRT trajectory perfectly but is not aware of the moving box. Thus the position error of the RRT is equal to the position of the box relative to the hand. The results in Fig. 6.11 show that the selected ConCERRT policy is robust up to 4 cm uncertainty and it can handle 6 cm to the right but starts to fail when the box is moved further to the left.

Contact sensing via pressure sensors is not fully reliable. A wrong contact event triggered in six executions out of the 28 runs. In one case this failure could be detected automatically as the contact observation was not part of the policy. In another case, a wrongly detected contact triggered a false reaction, resulting in a high error in the final hand position (at +4 cm), while the other five cases were within the 3.5 cm error bound. We expect to mitigate these failures in the future by equipping the soft fingers with deformation sensors (Wall et al., 2017).

6.3 Related contact-based contingent planning approaches

We reviewed most relevant background in belief-space planning in the previous two chapters. We will now only briefly discuss the literature for planning with contingencies.

Contingency plans have been proven to be more robust solutions for manipulation problems. A classic result in manipulation is that a parallel jaw gripper can rotate polygonal shapes without any sensing (Goldberg, 1993). However, it was shown lately that, to generalize this result to arbitrary shapes and realistic friction models, a contingent plan is needed (Zhou et al., 2017).

Particle-RRT planners (Phillips-Grafflin and Berenson, 2016) add contingencies by reversing and retrying motions that do not lead to the desired outcome. However, in our problem most motions increase uncertainty and therefore are generally irreversible. POMDP policies are contingency plans. One way to use POMDP solvers for manipulation is to discretize the lower-dimensional manifold of configurations in contact (Koval et al., 2016a,b). This allows to solve a tactile localization task, where a robot uses a sequence of contact motions to locate an object. However, the discretization limits the approach to problems with few possible contacts. Our approach does not rely on a predefined discretization but only searches the reachable contact manifold during planning. In our experimental results we showed similar policies on the tactile localization task for more complex contact manifolds.

6.4 Open issues for CERRT and ConCERRT

We presented two novel contact-based motion planners that combines the efficiency of sampling-based motion planners with an efficient uncertainty reasoning. Our first planner, Contact-Exploiting RRT (CERRT) reasoned in belief space about guarded and sliding motions to find efficient plans that reduce uncertainty by exploiting contact. Our second planner, Contingent CERRT (ConCERRT), exploited contact sensing to anticipate informative contact events that can happen during execution of the plan under uncertainty. Two assumptions made our planners operate efficiently in belief space: 1) representing the state by samples and using a heuristic search strategy tailored to the available actions, 2) assuming fully observable contact sensing.

We demonstrated that reasoning about uncertainty allows our planners to solve problems with higher uncertainty than uncertainty-agnostic planners. Compared to POMDP-based motion planners, our planners scale to high dimensional problems with a non-trivial contact manifold. This is because our planners discretize the state space during planning and

only reason about reachable contact events. We demonstrated this by successfully running experiments on a 7-dof robot both in simulation and in the real world.

There are some limitations to our planners, due to the assumption of a fully observable contact: The planner cannot deal with unreliable contact sensors and it needs edges in the environment to disambiguate its state. Additionally, the planner does not take path length into account and might compute long paths. However, it was shown before that asymptotically optimal planning is feasible for similar problems (Wirnshofer et al., 2018). We believe this will also transfer to CERRT and ConCERRT. Another alternative is to use the strategies found by our planner as initialization for optimizers that reason about contact (Posa et al., 2014; Toussaint et al., 2014).

Another limitation is that both CERRT and ConCERRT assume static environments. We assume that the the world will not move when the robot makes contact which is crucial for uncertainty reduction. Dynamic interactions with objects, e.g. pushing an object into an edge are currently not implemented in the planner but in principle the planner should be able to incorporate such actions too. Avoiding dynamic and uncertain environments will be the focus of the next chapters.

Part III

A sensor-based motion generation framework for mobile manipulation

"It sounded an excellent plan, no doubt, and very simply and neatly arranged; the only difficulty was, that she had not the smallest idea how to set about it."

Chapter 7

Background in adaptive motion planning and trajectory optimization

In Part I of this thesis we introduced how robust manipulation arises from describing robot motion as hybrid automata. In Part II we showed how motion planning methods can generate hybrid automata from models of the robot, the world, and estimates of uncertainty. These methods ignored perception, a crucial requirement of mobile manipulation we discussed initially (Sec. 3.5). Mobile manipulators need to capture relevant information with their on-board sensors. Their mobility will bring them to places they have never seen before. Furthermore, as these places will be shared with humans and other robots, many parts of the world can move. We refer to these places as **unknown, dynamic environments**. These environments conflict with the assumptions of planning methods from Part II, where we always assumed the world to be static and models to exist, although respecting that they might be uncertain.

In this chapter we want to discuss the interaction of perception and planning methods. Thus, we will review related work on sensor-based motion generation architectures. In the second half of this chapter, we will review motion generation methods based on optimization. These methods, compared to the search-based approach, have the big advantage of being able to continuously integrate new information to adapt their plans. These techniques will be the foundation for our novel motion generation systems we will present in the next two chapters.

This chapter is a review of related work and unique to this thesis. There are relatively few textbooks on systems design and optimization for robotics. Some background on motion generation architectures is given in Arkin (1998).

7.1 Motion generation architectures

The integration of perception into planning is a complex problem as it requires to efficiently process large amounts of sensor data. Thus, possible architectures for robot systems were highly debated in the 1980s and 1990s. We want to recapitulate the main points of discussion of that time but also discuss more recent developments.

7.1.1 Sense-plan-act

In the early stages of robotics research, discussion revolved around two extreme views on the architecture of robotics systems. The sense-plan-act approach is the straightforward extension of planning methods to incorporate sensor data. This architecture is divided in three steps:

1. *sense:* The robot gathers information about the world with different sensor modalities. It then merges them into a shared representation (model) that captures the real world as close as possible. As no model is perfect, the model includes estimates for its uncertainty.

2. *plan:* The robot uses the model to find a sequence of actions that solve the desired task. The planner reasons explicitly about the uncertainty in the model, i.e. it finds actions that work even if the state of the world is not known completely. The action sequence is specified in the geometry of the world.

3. *act:* The robot executes the action sequence using closed loop controllers that try to follow the plan as close as possible.

This architecture is the dominant paradigm for industrial applications. Its biggest strength is the factorization into intuitive subproblems. The *sense* component generates a model that can be verified by comparing it against the real world. The *plan* can be verified mathematically, as the model is assumed to contain all relevant information. The *act* component's only responsibility is to follow the plan as specified. This factorization allows to specify interfaces between perception, planning, and motion execution and give the problems into the hand of different groups of people that optimize each component or, even easier, just assemble the system from out-of-the-box components. In some areas, where precise models are available, sense-plan-act is a feasible approach (Laumond, 2006; Hernandez et al., 2016).

Consequentially, as sensors are usually noisy, sense-plan-act works well if perception is reduced to a minimum. However, sense-plan-act infrastructures suffer a lot from the challenges inherent to mobile manipulation (see Sec. 1.1). The planner must solve complex

reasoning problems under uncertainty (C1) using sensor data (C2). However the real-time constraints (C3) force it to make decisions quickly. This contradiction is the reason that sense-plan-act does not transfer to the unstructured real world.

7.1.2 Behavior-based

The limitations of the sense-plan-act architecture are known since the 80s. At that time, researchers shifted from model-based reasoning to behavior based robotics which focuses reasoning about the action and the interaction with the world. This idea spawned a multitude of feedback-based architectures known as **behavior-based robotics** (Arkin, 1998). It relies on two key aspects (Brooks, 1991)

- *Situatedness:* Behavior-based robots interact directly with the world and not with abstract models. Thus the key component of a behavior-based system is a feedback loop. Behavior-based approaches specify controllers which, when interacting with the world, lead to motion.
- *Embodiment:* Instead of abstracting away the hardware and morphology of the robotic system, behavior based approaches explicitly focus on the interaction with the hardware. This requires to close a implementation/testing loop early, bringing simple behavior to the real robot, and then adding features to solve more complex tasks.

The classic implementation of these ideas is the subsumption architecture (Brooks, 1986), where multiple feedback controllers run in parallel and a sensor-based arbitration mechanism decides which of the controllers is active. The advantage of such architectures is robustness. As these systems are optimized to react quickly to local changes, they do not suffer from uncertainty. This robustness made behavior-based systems valid solution for simple tasks, in fact, they provide the architecture for robotic vacuum cleaners — the only autonomous robot that reliably moves in unstructured environments today. The hybrid automaton driven system from Part I can also be seen as an instance of this architecture.

The biggest limitation of the behavior-based approach is its scalability. If the system gains complexity, the interactions between feedback controllers become hard to predict. The proposed approach of generating and testing behavior works for systems of limited complexity (see Part I), where the behavior of simple robots can be completely specified by programmers. This approach becomes infeasible for more complex tasks such as (general) manipulation.

7.1.3 Hybrid methods

The discussion between model and behavior-based reasoning is unsolved. 30 years after their introduction both approaches operate well, but only in their respective niches. Since then, many successful motion generation approaches live between these extremes,

Model predictive control (MPC) (Camacho and Alba, 2013), also called **receding horizon control**, is a feedback control method that reason over a longer time horizon. In MPC, a planner continuously finds plans valid for a limited time horizon. It then only executes the first action of that plan and replans with the new information. A famous example of these architectures is the **dynamic window approach** (Fox et al., 1997) which is widely used for reactive obstacle avoidance of fast mobile robots, including cars. To let MPC methods operate outside of their horizon, they can be combined with global planning. Global MPC approaches find a path that minimizes a sum of two costs: the cost inside the horizon and the cost-to-go obtained from the feedback planner at the horizon. This makes the MPC approach converge globally (Brock and Khatib, 1999; Stachniss and Burgard, 2002; Ogren and Leonard, 2005).

Adaptive control (Sastry and Bodson, 2011), or dual control, are feedback control approaches where some model parameters can change throughout execution. Thus, the goal of adaptive control is to perform model identification at the same time as fulfilling the motion task. Another view on adaptive control is to pose it as an online learning problem, and in fact many versions of adaptive control are equivalent to reinforcement learning (Sutton et al., 1992).

Interactive perception (Bohg et al., 2017) is a class of perception problems that are facilitated by the robot interacting with the world. Adaptive control and interactive perception are related: while the goal of interactive perception is understanding the world and not necessarily bringing it into a desired state, both methods share the need of understanding the structure of the combined space of sensor signals and actions over time.

Path adaptation, such as the elastic band (Quinlan and Khatib, 1993) and elastic strips (Brock and Khatib, 2002) methods, continuously adapt a trajectory by computing virtual forces that keep the trajectory short but also at a distance from nearby obstacles. The path adaptation adapts the path using sensor data while the robot is moving. Later, elastic methods were generalized as trajectory optimization, which is non-linear numerical optimization in the space of trajectories. Continuous trajectory optimization in response to moving obstacles was shown to be more robust than both sense-plan-act and purely reactive approaches (Kappler et al., 2018). As trajectory optimization is an important building block for the following contributions, we will now discuss it in more detail.

7.2 Planning as optimization

In trajectory optimization, we formalize path planning as an optimization problem over trajectories τ.

$$\begin{aligned} \text{minimize} \quad & F(\tau(t)) \\ \text{subject to} \quad & G_i(\tau(t)) \leq 0,\ i = 0, \ldots, N_{\text{ineq}} \\ & H_j(\tau(t)) = 0,\ j = 0, \ldots, N_{\text{eq}}. \end{aligned} \tag{7.1}$$

Here $F(\tau)$ is a cost function and $G = \{G_0, \ldots, G_{N_{\text{ineq}}}\}$ and $H = \{H_0, \ldots, H_{N_{\text{eq}}}\}$ define a set of inequality and equality constraints, respectively. (Direct) **trajectory optimization** (also **direct transcription**), is a family of methods that solves this non-linear optimization problem using numerical techniques. Trajectory optimization has a long history, going back to applications in space in the 60s.

All optimization methods share a common functional principle: they always need an initialization with a trajectory which does not need to fulfill the constraints (often the straight line connection between start and goal is used). They then perform some variant of iterative optimization on the cost function, taking collision and other constraints into account. Repeated execution of the optimization step then converges to a (local) minimum. To compute the gradients and higher derivatives, the trajectory is usually discretized into a set of control points, i.e. $\tau = [q_0, \ldots, q_T]$. This framework has been used successfully to find trajectories for satellites, or for finding time-optimal trajectories for industrial manipulators (Von Stryk, 1993), although without reasoning about obstacles.

The path planning problem (Sec. 4.3) can also be casted as a trajectory optimization problem by defining a constraint $H(\tau) \leq 0$ iff $\tau \in \mathcal{C}_{\text{free}}$. A suitable candidate for H is the **signed distance function**, a function that returns the distance to the closest obstacle if the trajectory is in free-space and the penetration distance if the trajectory is in collision. As an optimization objective we can minimize the squared velocities:

$$F(\tau) = \sum_{t=1}^{T} (\tau_{t-1} - \tau_t)^\top (\tau_{t-1} - \tau_t) \tag{7.2}$$

or accelerations:

$$F(\tau) = \sum_{t=2}^{T} (\tau_t - 2\tau_{t-1} + \tau_{t-2})^\top (\tau_t - 2\tau_{t-1} + \tau_{t-2}) \tag{7.3}$$

Optimization with obstacles is hard to solve as the obstacle avoidance constraint makes the problem non-convex. The success of trajectory optimization in manipulation came with

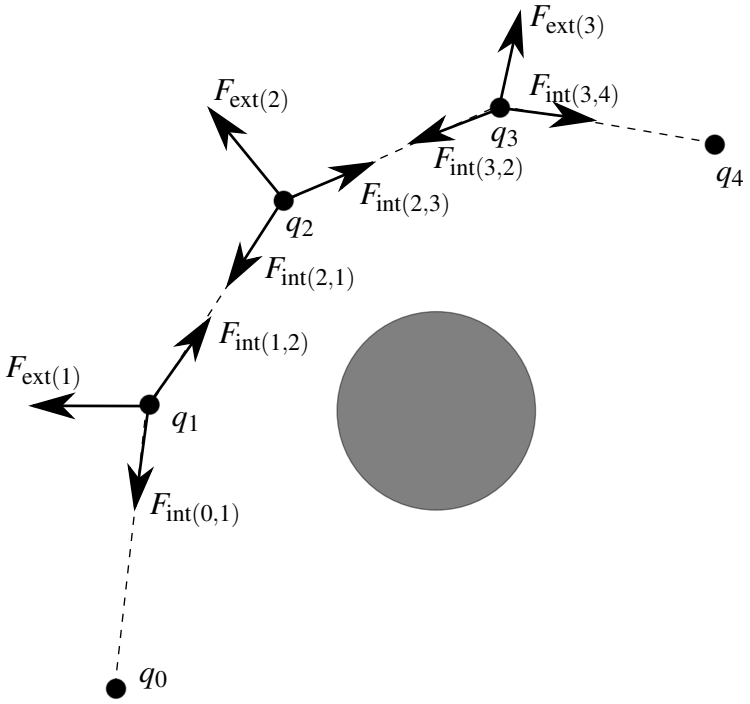


Figure 7.1 The active forces in the elastic band method avoiding a grey circular obstacle. Shown in dashed lines is the trajectory, discretized by five control points $\tau = [q_0, \ldots, q_4]$. The external forces $F_{\text{ext}(i)}$ point away from the obstacle while the internal forces $F_{\text{int}(i,j)}$ minimize the length of the elastic band.

the observation that, especially in high-dimensional configuration spaces, local optimization often will converge to good non-colliding solutions.

Compared to search-based methods, local optimization has no guarantees of finding a collision-free solution but there are two advantages over search: 1) optimization can easily and efficiently incorporate cost function terms beyond obstacle avoidance such as path length, smoothness, or higher derivatives such as jerk. Consequentially, the results are often much smoother than search-based solutions. 2) optimization can be easily applied in changing environments, just by continuously running optimization on the changing scene.

Trajectory optimization methods for path planning differ in the choice of parametrization, the optimization step direction, and the optimizations step size. We will now explain the differences. In Table 7.1, we summarize the discussion.

7.2.1 Elastic methods

The elastic band (Quinlan and Khatib, 1993) method applies two forces on a series of control points along a two-dimensional trajectory. A repulsive force moves each point away from obstacles and an internal force moves each control point towards those before and after in the

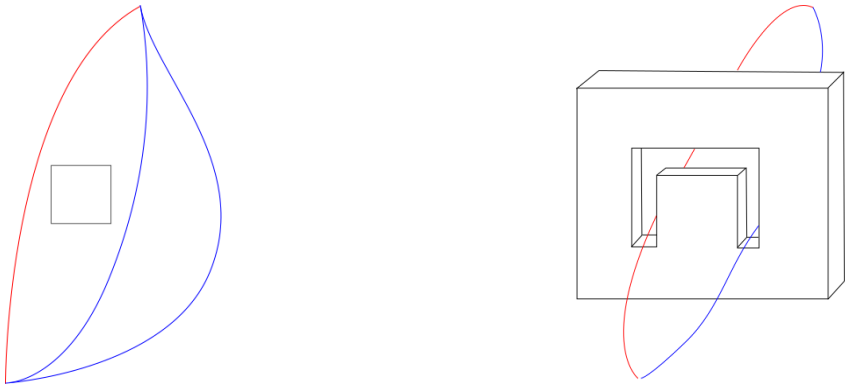
Table 7.1 Overview of trajectory optimization methods

method	discretization	step direction and length
Elastic Bands (Quinlan and Khatib, 1993)	variable (2D) control points	$-\nabla(U_{\text{obs}}+U_{\text{acc}})$, constant gain
Elastic Strips (Brock and Khatib, 2002)	variable control points	$-\nabla(U_{\text{obs}}+U_{\text{acc}})$, constant gain
Chomp (Zucker et al., 2013)	fixed control points	Covariant g.d. $G^{-1}\nabla(U_{\text{obs}}+U_{\text{acc}})$
Stomp (Kalakrishnan et al., 2011)	fixed control points	Stochastic g.d. on $U_{\text{obs}}+U_{\text{acc}}$
trajOpt (Schulman et al., 2014)	fixed control points	SQP of U_{acc}, constrained by U_{obs}
KOMO (Toussaint, 2017)	fixed control points	Aug. Lag. updates for U_{acc}, constrained by U_{obs}
GPMP (Mukadam et al., 2016)	gaussian process	g.d. on $U_{\text{obs}}+U_{\text{prior}}$
GPMP2 (Dong et al., 2016)	gaussian process	probabilistic inference

trajectory (see Fig. 7.1). These forces are equivalent to optimization using gradient descent: The direction of the internal force is equivalent to the gradient of an acceleration-minimizing cost function. The direction of the external forces are equivalent to a gradient on a signed distance function. The step size is constant and given by the user. A crucial part of the implementation is the discretization. Both the position and the number T of control points of the trajectory vary based on local geometry. Elastic methods place control points at the center of free volumes in work space. Thus, it places many control points in narrow regions while only few points specify the trajectory in open spaces. The elastic band method was generalized to manipulators as elastic strips (Brock and Khatib, 2002). Here the forces act on points on the robot and are mapped to control torques using the Jacobian matrices.

7.2.2 Numerical optimization

In robotics, the numerical optimization approach for trajectories gained traction with the CHOMP method (Zucker et al., 2013). CHOMP optimizes a sum of obstacle avoidance and acceleration cost. However, as step direction it uses the Hessian of the acceleration cost term. This direction makes the gradient descent step invariant to the discretization of the trajectory and converges faster. STOMP (Kalakrishnan et al., 2011) performs stochastic gradient descent, which allows to use non-differentiable cost functions. TrajOpt (Schulman et al., 2014) treats obstacle avoidance as a constraint and not as a cost term like the previous methods. It uses sequential quadratic programming to compute step size and direction. The KOMO-framework (Toussaint, 2017) generalizes most of the previous methods as instances of the Newton method. All of these methods discretize the trajectory with control points at fixed steps.



(a) (b)

Figure 7.2 *Left*: In 2D, homotopy classes are good indicators for the local minima of optimizers. A local optimizer can easily deform the blue trajectories into each other because they are in the same homotopy class. Blue and red are in different homotopy classes and thus not deformable into each other. *Right:* In three or more dimensions homotopy is not a good indicator. Blue and red are in the same homotopy class but can not easily be transformed into another by local optimization (adapted from Jaillet and Siméon (2008)

Another branch of optimization methods tackle optimization using inference as optimization tool. AICO (Toussaint, 2009) applied approximate inference in a kinodynamic motion planning setting. The GPMP framework (Mukadam et al., 2016) represents trajectories as Gaussian processes, which allows to adapt the discretization of the trajectory online. Just like elastic methods, the control points of the Gaussian process can be adaptively placed. Together with efficient inference, was shown to reduce computation time drastically, solving 7-dof optimization problems in a fraction of a second (Dong et al., 2016).

7.2.3 Connectivity and homotopy

All previously mentioned methods are local optimizers. They will converge to a local minimum close to the trajectory they have been initialized with. This local minimum might be arbitrarily far from the global optimum, or even in collision. We now want to gain an understanding for the effect of the initialization on local optimizers, and discuss strategies to make local optimizers find globally optimal solutions. We will first discuss the problem in two dimensions and then generalize to problems with higher dimensionality.

Two trajectories are homotopically equivalent if they can be continuously transformed into each other without crossing obstacles. This equivalence relation segments each planning problem into a number of equivalence classes called **homotopy classes**. In two dimensions, homotopy indicates whether optimization will converge to the global optimum as all numerical optimizers continuously deform trajectories but stay away from obstacles. Therefore,

local optimization (with strict collision constraints) will not adapt a trajectory beyond its homotopy class (see Fig. 7.2a). Local optimization will find the global optimum if it is initialized with a path from the right homotopy class.

From this insight follows: if an optimizer is initialized several times with a path from every homotopy class, one of them will converge to the optimal solution. Furthermore, if the obstacles in the world move and all paths are continuously adapted, one of the locally optimal paths will also be globally optimal. Obtaining paths from different homotopy classes is a global search problem and usually done using sampling-based planning. As discussed in Chapter 4, roadmaps capture the connectivity of the space and contain paths from different homotopy classes. Many methods exist to reduce roadmaps to a small set of nodes and edges that capture the connectivity, e.g. generalized Voronoi diagrams (Takahashi and Schilling, 1989), obstacle-based roadmaps (Amato et al., 1998), or visibility-based methods (Jaillet and Siméon, 2008).

Unfortunately, these insights do not directly transfer to more than two dimensions. Here, homotopy alone is not sufficient to tell if an optimizer will end in a local minimum. Fig. 7.2b shows an example where the initial and optimal trajectory are in the same homotopy class but cannot easily be transformed into another by local optimization. Instead, for higher dimensional problems, an approximate understanding of the connectivity is needed. Efficiently approximating connectivity will be one of the goals of the following chapter.

Chapter 8

Incremental elastic roadmaps for unknown environments

A successful motion generation system for mobile manipulation in unstructured, dynamic environments must—in addition to generating global, task-consistent motion—also address issues of perception and uncertainty. Each of these problems has received significant attention in the robotics community, however, little work in mobile manipulation attempts to address all three at the same time in real-world scenarios. This is the goal of this section.

We will start from three simple assumptions, namely, that *(a)* the world changes in a continuous fashion, *(b)* significant connectivity changes occur relatively infrequently, and that *(c)* information about the environment is obtainable exclusively through on-board sensors.

As we will see, these assumptions enable us to divide the overall motion generation problem into three sub-problems, each of which can then be solved by the most appropriate method. First, to respond to small, continuous changes in the world, we employ online trajectory modification to adapt the existing edges of the roadmap. Second, to update local connectivity changes or to reflect significant changes to our world model, we employ an efficient motion planner to add new milestones and edges to the roadmap. Third, to ensure the global motion goal is attained efficiently, even when the connectivity is uncertain, we reason about the uncertain environment by solving an expected-shortest-path problem (Briggs et al., 2004) on our incrementally maintained roadmap. We call this novel motion generation framework **Expected Shortest Path Elastic Roadmap** (ESPER).

We evaluate this approach to motion generation in real-world mobile manipulation scenarios (also shown in Figure 8.1), demonstrating that it satisfies the following three requirements. First, the motion satisfies task constraints: in our experimental evaluation, the robot maintains constraints on end-effector orientation and position. Second, the robot deals with unpredictable dynamic obstacles: the robot handles significant connectivity change as

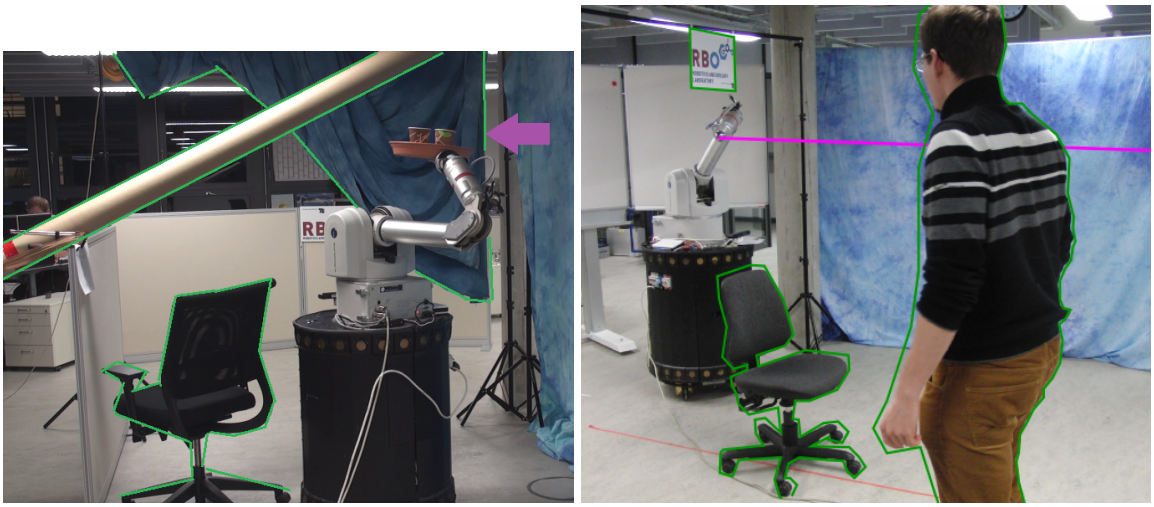


Figure 8.1 Two example motion generation tasks for a mobile manipulator. *left:* The WAM+XR4000 maintains the orientation of a tray while avoiding unstructured obstacles (green). The robot uses only on-board RGB-D sensors and has no prior model of this environment. *right:* The robot has to move the end-effector on a virtual pink line. Additionally, it has to perceive and avoid dynamic obstacles: a rolling chair, a sign hanging from the ceiling, and a moving person

well as fast dynamic obstacles. Third, the robot relies only on on-board sensors: throughout the experiment, the mobile manipulator acts truly autonomous and does not rely on external sensors or pre-built maps of the world. It can generate task-consistent collision-free motion in a world that is completely unknown initially.

This chapter is an aggregation of two publications (Sieverling et al., 2014; Lehner et al., 2015). I was second and first author, respectively. The publications are based on a motion generation system developed by Nicolas Kuhnen (NK), Peter Lehner (PL), and me. Section 8.2 describes the elastic roadmap framework which was conceived by Yang and Brock (2010). The text is written by me and an original contribution. Section 8.3 was developed by PL with my assistance. We both contributed equally to writing. Section 8.4 describes the Expected Shortest Path algorithm which I conceived and integrated into the elastic roadmap framework. I am main contributor to writing that section. Section 8.5 describes a perception system developed by me with assistance from Roberto Martín-Martín. The text is my own contribution. The experimental evaluation in 8.6 is based on an implementation of the elastic roadmap by NK and PL. I ported the implementation to the real robot system and performed the experimental evaluation together with the two authors.

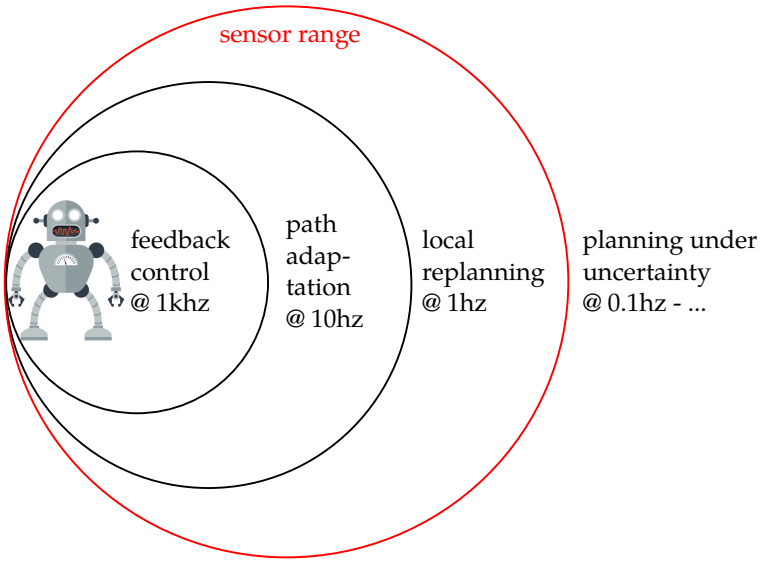


Figure 8.2 The incremental elastic roadmap decomposes motion generation in four parallel processes. Within the sensing range, feedback control, path adaptation, and local planning maintain an elastic roadmap. Outside the sensing range, the planner reasons continuously about the uncertainty of the environment.

8.1 Expected shortest path elastic roadmap

Above, we stated three assumptions for incremental, sensor-based planning: *(a)* most changes in real world problems result from small, continuous object motions, *(b)* significant changes to the connectivity occur rarely, and *(c)* changes in the environment can only influence the plan if they are perceived by the robot's on-board sensors. The goal of this section is to present a roadmap-based planning method based on these assumptions.

These three assumptions motivate a division of the problem into three sub-problems. The first component adapts the roadmap within the sensing range to reflect the continuous changes in the environment *(a)*. The second component detects the rarely occurring significant changes *(b)* that can not be handled by the first component. The component addresses these changes and treats them by solving local motion planning problems. This component is efficient because it only plans within the robots sensing range. Beyond the sensing range, we will reason efficiently about the uncertainty of the world *(c)*.

All these components run continuously at different update rates. Components with higher update rate effect immediate decisions. Components with lower update rate effect long-term behavior. An overview of the architecture that shows the interplay of the components is given in Fig. 8.2.

We will now give a high-level overview about the three components and their interconnections. In the following sections, we will give the necessary implementation details.

Handling continuous change: the elastic roadmap

To handle continuous change we need to adapt multiple possible paths in a roadmap in response to continuous changes. Trajectory optimization methods address continuous change in paths efficiently, however not many of these methods are able to deal with optimizing a graph structures. The elastic roadmap framework is an efficient method to adapt a roadmap in response to continuous changes under task constraints (Yang and Brock, 2010). The elastic roadmap framework uses a roadmap, i.e. a graph, as the underlying representation. Edges in the elastic roadmap are not specific paths in configuration space but instead controllers. A vertex or milestone represents a configuration of the robot. Two vertices are connected by an edge in the roadmap if an associated controller is able to move the robot between them. The elastic roadmap can be visualized as a cover of the configuration space with funnels (Fig. 8.3). Each milestone is also associated with a multi-priority task-space controller that maintains task constraints while staying away from obstacles. Now vertices can also move around. The result is an elastic network of moving milestones reacting to dynamic changes in the environment. Robust motion plans can be generated by sequencing the controllers along the edges so as to connect the start and the goal milestone in the roadmap. Another view on the elastic roadmap is understanding it as continuous optimization of a graph structure represented by control points. We assume that the initial, non-adapted graph structure captures the connectivity well. Therefore the optimization objective is similarity to the original, unoptimized graph. The optimization is constrained by an obstacle avoidance term.

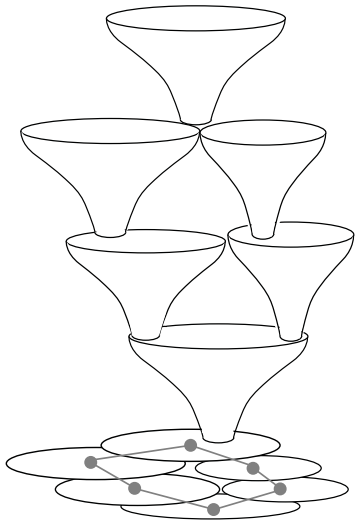


Figure 8.3 The elastic roadmap covers the space with funnels, connected in a graph structure

Handling connectivity changes: workspace connectivity graph

The second component of our method addresses significant, rarely occurring changes in the environment. When adapting an existing elastic roadmap to a dynamic environment certain changes can not be handled sufficiently only using local adaptation. We subsume all of these changes as changes of *connectivity*. Reasons for changes in connectivity could be the appearance of an obstacle - in this case the roadmap would need to capture paths around the obstacle. It could also be the disappearance of an obstacle, i.e. a door opening up a new

possible shortcut in this case the roadmap would need to include new edges passing through the door.

To react to significant changes we need an efficient way to capture connectivity. Reasoning about the connectivity of the high-dimensional configuration space is a hard problem. We will present the *workspace connectivity graph*, a method that approximates configuration space connectivity with workspace connectivity. This graph can be constructed very efficiently at high rates. Using the workspace connectivity we check if the connectivity represented in the roadmap differs from the current state of the environment. If there is a difference, we modify the roadmap locally by inserting or removing milestones and edges. The removal of milestones is simple. We create and insert milestones by using sampling-based motion planners. Configuration space search becomes feasible when we guide the planners with the found workspace connectivity.

Global search: expected shortest path

The remaining planning problem is extracting a global plan that leads the mobile manipulator from its current position to a goal. However, this is not just a simple graph search. In a dynamic environment, passages might be crowded with people, increasing the expected time of the robot to traverse them safely. Doors might be randomly open or closed, requiring the robot to plan for alternatives. To take these possibilities into account, the robot must reason explicitly about the uncertainty of the world.

We will introduce an efficient stochastic graph search based on the expected shortest path method (Briggs et al., 2004). This algorithm does not suffer from the complexity of planning under uncertainty because it makes simplifying assumptions that directly relate to the strategies for efficient belief-space planning we discussed in Sec. 4.8: First, it only plans on those states that are nodes in the elastic roadmap, which is generated from samples ($\rightarrow$ *sampling*). Second, it ignores many sources of uncertainty as state and motion uncertainty are not taken into account. The underlying elastic roadmap is based on controllers that continuously adapt to the environment. As every motion is always executed with feedback, we assume that state and action uncertainty is small enough to ignore it at planning time ($\rightarrow$ *ignoring uncertainty*). The last remaining uncertainty source is the uncertainty of the world. Here we use an optimistic policy that assumes that dynamic obstacles might block passages for some time but will also unblock them relatively quickly. This transforms the POMDP into a variant of an MDP which solvable in polynomial time ($\rightarrow$ *hindsight optimization*). Dynamic obstacles that block passages for a long time are simply incorporated into the world model and lead to a local adaptation of the plan ($\rightarrow$ *replanning*).

In the next three sections we will detail all three components. The elastic roadmap framework is not a novel contribution and has been published before (Yang and Brock, 2010). As this method is of fundamental importance for our system we will introduce it now. The second and third components are novel and will be explained in detail in Sections 8.3 and 8.4.

8.2 The elastic roadmap

The elastic roadmap framework (Yang and Brock, 2010) is an efficient motion generation approach that generates reactive, task-constrained, whole-body motion for mobile manipulation tasks. It captures the connectivity of the workspace in a roadmap that is incrementally modified in response to perceived changes in the environment. As these changes in the environment move nodes and edges of the roadmap, it appears to be "elastic", hence the name.

An elastic roadmap is a graph $G = (V,E)$, consisting of a set of milestones (vertices) $V = \{v_1, v_2, \ldots\}$ and edges $e_{ij} = (v_i, v_j) \in E$. An edge e_{ij} indicates that the milestone v_i is in the region of attraction of the controller associated with v_j, i.e. invoking v_j's controller while the robot is in state v_i will cause the robot to move to v_j. An elastic roadmap corresponds to a dual hybrid automaton: every edge (v_i, v_j) in the roadmap is a control mode in a hybrid automaton that moves the robot from milestone v_i to milestone v_j. The controllers used in this hybrid automaton are multi-priority task-space controllers that maintain task-constraints. A typical controller in the elastic roadmap is given by:

$$\Phi_{\text{pos}} \triangleleft \Phi_{\text{obs}} \triangleleft \Phi_{\text{att}} \triangleleft \Phi_{\text{task}} \tag{8.1}$$

Here Φ_{task} is a task-space constraint such as keeping the end-effector in a certain orientation. Φ_{att} is a potential function that attracts the end-effector to the next milestone v_j, Φ_{obs} is reactive obstacle avoidance behavior. The posture behavior Φ_{pos} is taken directly from milestone v_j.

To achieve the elastic effect, every milestone v_i is continuously avoiding obstacles, just like the elastic strips method. We also implement this using multi-priority task-space control in the following hierarchy:

$$\Phi_{\text{pos'}} \triangleleft \Phi_{\text{obs}} \triangleleft \Phi_{\text{task}} \tag{8.2}$$

Here $\Phi_{\text{pos'}}$ is a default posture that keeps the robot at from joint limits and singularities. As a result, each milestone represents a task-consistent via point in configuration space. When obstacles move, nearby milestones will move with them while maintaining task constraints.

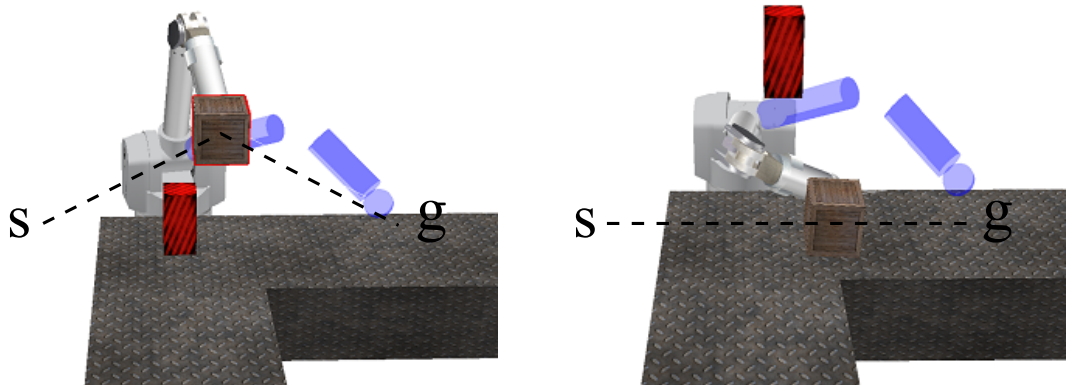


Figure 8.4 The elastic roadmap framework generating motion for a stationary manipulator moving a wooden box among dynamic red obstacles. the robot has to move the box from its initial position s to the final position g; the milestone at the goal configuration is shown in blue; The elastic roadmap generates different motions in response to changes in the environment; the task constraint is maintained throughout the entire motion.

As milestones move, the planner updates the connectivity between milestones continuously, maintaining the roadmap.

The Elastic Roadmap planner covers large parts of the configuration space with the regions of attractions of milestones, using workspace information to identify task-relevant configuration space regions. The roadmap G defines a feedback plan over the state space.

Figure 8.4 shows an execution of the elastic roadmap method for a 7-dof manipulator in a dynamic environment. Here the whole kinematic chain avoids the dynamic obstacles. When the obstacles block the lower path, the global plan guides the arm above the obstacles to avoid collisions. When the obstacles move to the upper level, the motion instantly adapts. Now the robot reactively avoids colliding with the boxes with its elbow using all of its seven degrees of freedom to execute the task. Still, during the complete motion, the end-effector orientation remains constant.

8.3 Local motion planning using workspace connectivity

In this Section we will present details on our method to augment elastic roadmaps to deal with connectivity changes. Fig. 8.5 serves as an introductory example for this process. It shows how our method constructs a roadmap iteratively from scratch.

The incremental construction is based on two components. The first component approximates the connectivity of the configuration space using workspace information. It then proposes parts of the workspace that are not yet covered by the roadmap. The second

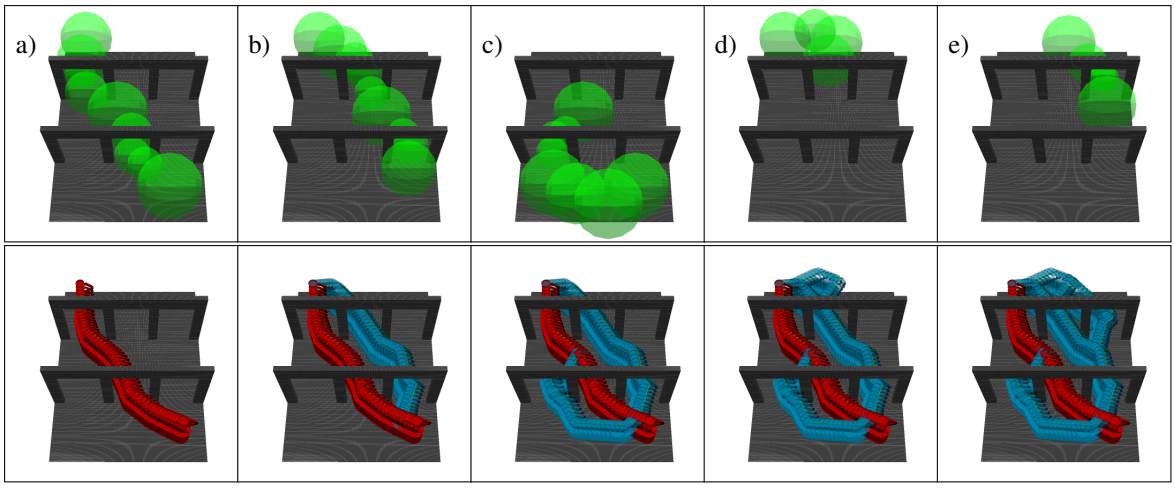


Figure 8.5 Illustration of the incremental roadmap generation process. The representation of the environment is shown in grey. The upper row shows five tunnels (green) through the free workspace. These tunnels are the result of a workspace analysis that detects parts of the workspace where the current roadmap does not represent the connectivity sufficiently. The lower row shows the outcome of a local planner generating paths guided by the workspace tunnels. The lower right image shows how the final roadmap approximates the connectivity of the environment well.

component, is a sampling-based motion planner that generates constraint-satisfying paths guided by the workspace analysis. We will now detail both components.

Approximating workspace connectivity

The **workspace connectivity graph** is an efficient representation of the connectivity of the environment. Its creation is based on a sphere-based wavefront algorithm similar to the one used in decomposition-based planning (Brock and Kavraki, 2001). The algorithm floods the free workspace with spheres (see Fig. 8.6b). It tries to minimize the amount of spheres by greedily preferring larger spheres over small ones. In a second step we construct an undirected graph where nodes correspond to the sphere centers. We add edges between nodes if two sphere overlap significantly (see Fig. 8.6c). Each path through this graph is a sequence of overlapping spheres that can be seen as a tunnel through the free workspace.

Generating paths in workspace

The incremental planner uses the workspace connectivity graph to extract tunnels through the free workspace, These tunnels then guide our underlying sampling-based planner. We now

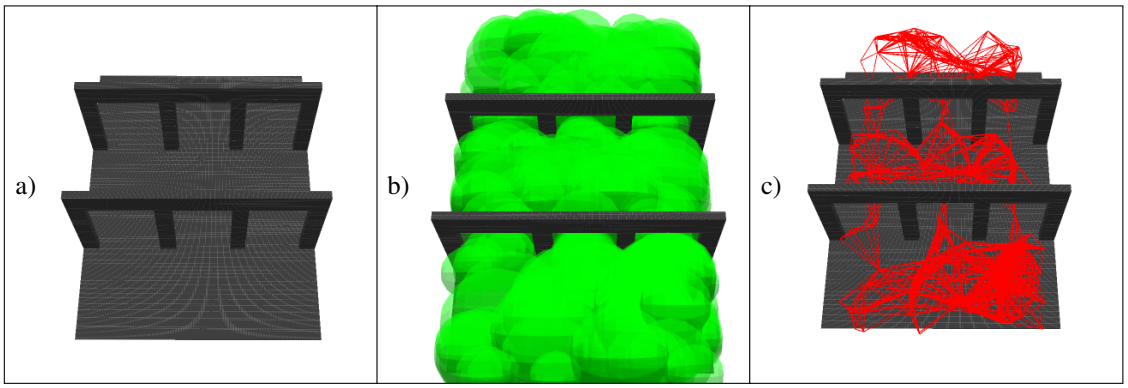


Figure 8.6 Illustration of the workspace connectivity graph: a) shows a sample environment with six passages. b) shows the decomposition of the workspace with spheres. c) shows the resulting workspace connectivity graph.

need to extract tunnels from the workspace connectivity graph *(1)* leading from start to goal and *(2)* are not yet covered by the existing roadmap. To do so, we compute a shortest path from start to goal, but add a penalty term for edges that are already covered by the current roadmap. We define the cost of an edge $e_{i,j}$ as

$$c(e_{i,j}) = \begin{cases} d(v_i, v_j) + k_{\text{covered}} & \text{if there is a milestone intersecting with } v_i \text{ or } v_j \\ d(v_i, v_j) & \text{else} \end{cases} \tag{8.3}$$

where $d(v_i, v_j)$ is the Euclidean distance between the sphere centers at nodes v_i and v_j, and k_{covered} is a constant penalty term with $k_{\text{covered}} > d_{i,j}$ for all edges $e_{i,j}$ in the graph. This cost function has the effect that edges that are already covered by the current roadmap will only be considered if there is no other possibility. We summarize the construction of the workspace connectivity graph and the cost function in Algorithm 8. After the graph construction we apply Dijkstra's algorithm to compute a minimal cost path. The minimal cost path is a tunnel of overlapping spheres that connect start and goal. The tunnel leads through unoccupied workspace and is as short as possible. We then use several of these tunnels to guide sampling-based motion planners locally, create milestones, and add them to the roadmap.

Locally guided motion planning to find connectivity

Generally, all sampling based motion planners can be used in our method provided they satisfy the following two requirements: first, the planner needs to generate paths that satisfy

Algorithm 8 CREATE_WGRAPH($p_{\text{start}}, V_{\text{roadmap}}$)

Input: $G = (V, E)$; Priority Queue $Q \leftarrow \emptyset$; Sphere s_{start}

```
1: s_start.center ← p_start
2: s_start.radius ← DISTANCE(p_start)
3: Q.PUSH(s_start, s_start.radius)
4: repeat
5:    s_top ← Q.POP()
6:    G.ADD_VERTEX(s_top)
7:    for all s in V do
8:       if OVERLAP(s, s_top) ≥ r_min then
9:          cost ← DISTANCE(s, s_top)
10:         for all v in V_roadmap do
11:            if s.COVERS_MILESTONE(v) then
12:               cost ← cost + k_covered
13:         G.ADD_EDGE(s, s_top, cost)
14:   for i = 1 to N · s_top.radius do
15:      s_new.center ← SAMPLE_POINT_ON_SURFACE(s_top)
16:      s_new.radius ← DISTANCE(s_top.center)
17:      if s_new.radius ≥ r_min AND ¬COVERED(s_new, V) then
18:         Q.PUSH(s_new, s_new.radius)
19: until Q = ∅
20: return G
```

task constraints. Second, the planner should be able to use workspace information to guide sampling.

In our implementation we integrate the exploring / exploiting tree (EET) planner introduced in Section 4.5. The EET satisfies both stated conditions. The EET accommodates task-space constraints easily using direct sampling (see Sec. 4.6): The EET samples task frames and then grows a search tree in configuration space. By restricting the sampling of task frames to those satisfying task constraints the EET only searches paths that lie completely on the task manifold. The EET is also always guided by workspace information. Once the EET finds a new path the algorithm places milestones at an regular interval Δq into the elastic roadmap. Each milestone becomes an initial configuration for a controller that reacts to changes in the environment.

8.4 Efficient planning in uncertain environments

In this section, we describe the graph search method that extract policies from the Elastic Roadmap in uncertain environments. Planning in uncertain environments is a hard problem. To solve it optimally, a belief-space would be needed that captures all possible states of

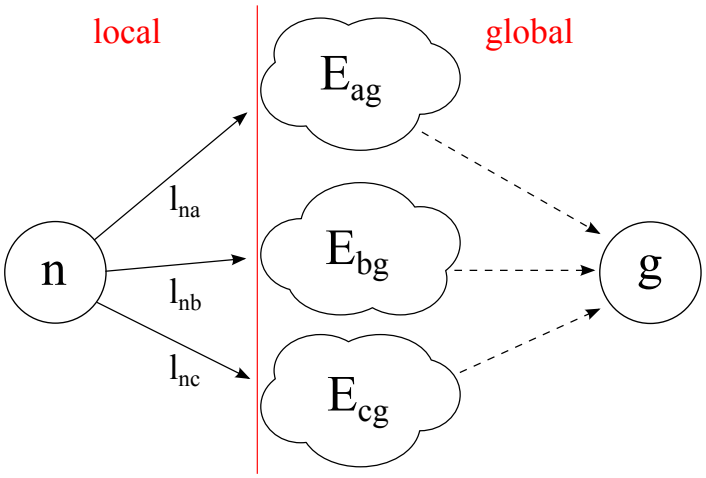


Figure 8.7 Local decisions for a problem with three alternatives. The robot is at node n. The clouds represent the multitude of paths to the goal, after reaching a, b, or c. Knowledge about the expected outcome of future actions is reflected in E_{ig}.

the world. Such a state would quickly lead to a combinatorial explosion, making naive approaches computationally infeasible. We reduce the complexity of the problem by making a simple, appropriate, and justified assumption about the sensor capabilities of the robot: we assume that a mobile manipulator can only obtain information about the world through its onboard sensors and that a all static obstacles are continuously added to a map by the local motion planning component discussed in Sec. 8.3.

Based on these two assumptions, we decompose planning under uncertainty in two sub-problems: *1)* reasoning about uncertainty within the local, perceivable region, and *2)* reasoning about uncertainty in the remainder of the space. The local region around the robot is bounded by a "horizon of relevance", influenced by the sensor range, acceleration capabilities, and the environment. In this local region, we rely on sensor feedback to address uncertainty implicitly. In the global region, we explicitly reason about environment uncertainty, as will be explained in this section.

Local region In the local region, the robot maintains and continuously adapts its plan using the two previous components. We assume that within the sensor range, the robot has enough feedback to eliminate uncertainty of obstacle positions completely. We also assume the controllers are robust enough to eliminate state and action uncertainty.

Global region (Time-dependent Expected Shortest Path) What does it mean to reason about uncertainty in the global region? We have a global map of the static part of the environment. Within this map, we have an elastic roadmap, capturing the connectivity of

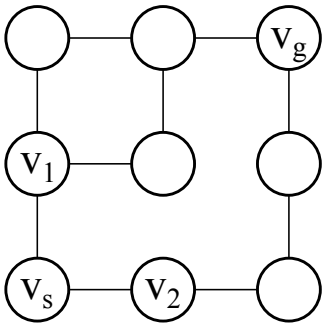


Figure 8.8 Effect of graph structure on expected costs: The path from v_s to v_g via v_1 provides two alternative paths to g with equal length, while the path via v_2 provides only one. The expected cost of the path via v_1 is lower because the robot can avoid one possibly blocked edge following v_1.

the space. Uncertainty stems from the possibility of any of the edges of the roadmap being blocked by unobserved, moving obstacles. We associate with each edge in the roadmap information about its "blocking characteristics" (defined in detail below).

Reasoning about uncertainty during the planning process can now be viewed as follows (please refer to Figure 8.7): In the local region around milestone n, we use sensor information to make local decisions. The cost of going to milestones a, b, and c is known, as the costs l_{ni} between milestones n and i can be determined from sensor data. However, the overall expected cost to the goal, which our algorithm seeks to minimize, also depends on the remainder of the path to the goal milestone g. The key now is to reason about the expected cost E_{ig} to travel through the roadmap from each of the three milestone a, b, c to the goal g. The parts of the elastic roadmap that have to be traversed after the local decision has been made, are shown as clouds in Figure 8.7).

The expected cost E_{ig} depends on the graph structure of the roadmap and the blocking probabilities of the edges contained in it. It is easy to see how blocking probabilities affect the cost, as blocked edges cannot be traversed and may lead to detours. Figure 8.8 illustrates the effect of the graph structure on the expected cost. Assuming equal blocking probabilities for all edges, the route from v_s to v_g through v_1 has lower expected cost than the route through v_2, as the probability of both routes following v_1 being blocked is lower than the probability of the edge following v_2 being blocked.

In our setting, we use time as the cost, seeking to minimize the time to reach the goal. To estimate the expected travel duration along the entire path, we derive a model for the blocking probability of edges from sensor observations. We model the unknown state of an unobserved edge as a continuous-time Markov Chain with two states: *blocked* and *free*. Three parameters characterize every edge $e = (i, j)$:

- l_{ij}: the expected time it takes to traverse e,
- t_{ij}^B: the expected duration the edge is *blocked*, and
- t_{ij}^F: the expected duration the edge is *free*.

In this setting, determining the expected cost requires to reason about all possible blocked/unblocked states of all edges based on the robots observation history. This formulation can be solved using a POMDP formulation (Marthi, 2012), but too computationally complex to be computed continuously in dynamic environments. However, our assumption that the static obstacles are captured in an incrementally maintained map allows us to devise a more effective solution. We assume that edges are never blocked for time periods much longer than the robot requires to move along the edge, i.e. $t_{ik}^B < l_{ij}$ for all pairs of edges $(i,j),(i,k)$. And we remove edges that are blocked for very long time periods from the roadmap. Additionally, we remove edges which are blocked so frequently that they can not be traversed safely ($t_{ij}^F < l_{ij}$). Also, once the robot observes the state of an edge, it is not considered to change again. Based on these assumptions, it suffices to reason about the currently known state of the world, as opposed to all possible states. This is the key step to addressing environment uncertainty efficiently.

Given this assumption, the probability p_{ij} that an edge is traversable is given by the stationary distribution of the Markov chain:

$$p_{ij} = \frac{t_{ij}^F}{t_{ij}^F + t_{ij}^B}. \tag{8.4}$$

Now p_{ij} and l_{ij} are known, and we can apply the polynomial-time Expected Shortest Path (ESP) algorithm, which is based on dynamic programming, to compute the expected cost E_{ig} for every node i (Briggs et al., 2004).

We can add an additional source of information to improve the policies of the Expected Shortest Path algorithm: in our time-dependent model, the robot can estimate the time each blocked edge (i,j) takes to become unblocked by t_{ij}^B. This time estimate enables an additional motion alternative: wait for a promising edge to become unblocked. For each edge the robot has two options:

- If the edge is *free*, the robot can move along it right away. This action is possible with probability p_{ij} and takes time l_{ij}.

- If the edge is *blocked*, the robot can wait until it becomes unblocked and then move along it.[1] This is possible with probability 1 and we estimate the time by $l_{ij} + t_{ij}^{B}$.

The optimal policy follows immediately: The robot executes the controller associated with milestone k, where

$$k = \text{argmin}_{i \in V} \begin{cases} l_{ni} + E_{ig} & \text{and } (n,i) \text{ is } \textit{free} \\ l_{ni} + t_{ni}^{B} + E_{ig} & \text{and } (n,i) \text{ is } \textit{blocked} \end{cases} \tag{8.5}$$

In the second case, the robot will wait until the edge state changes from blocked to free.

The ESP without the new waiting actions can be solved in $O(|E| \cdot \log(|V|))$ with a label-correcting algorithm (Bar-Noy and Schieber, 1991). Adding the waiting actions doubles the amount of edges, but does not increase the amortized runtime.

8.5 Perception

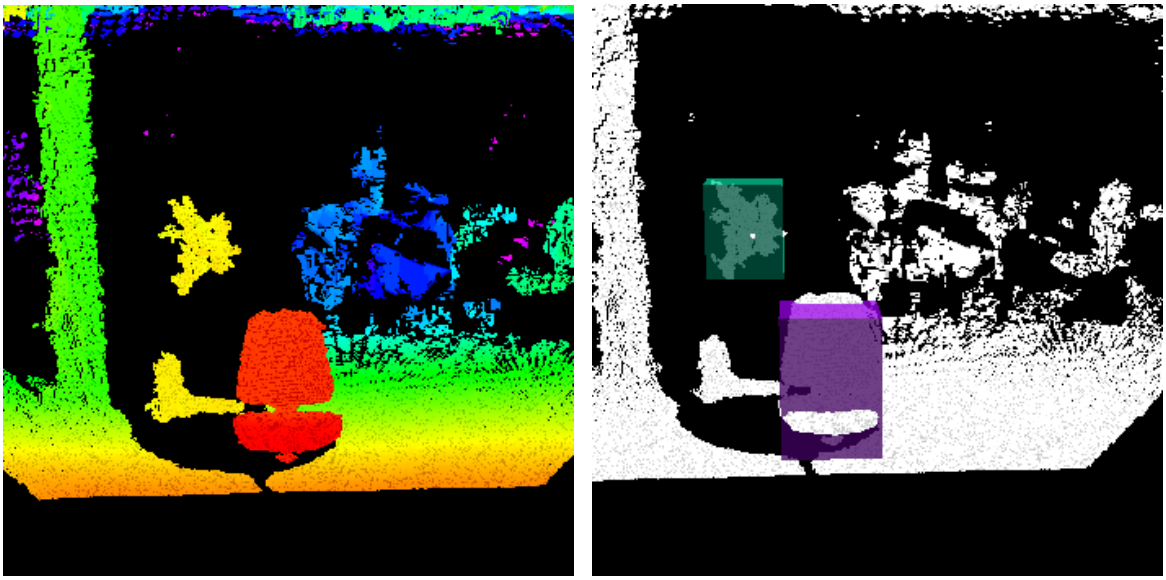
An important part of the proposed framework is the integration with real-world sensing. To be suitable for mobile manipulation applications, a motion generation method should only rely on onboard sensor information during execution.

Our framework has two requirements for perception: First, it needs to maintain a map of the static part of the environment. This map must allow for efficient collision queries as it is used by the local motion planner. Second, to adapt motion efficiently, the vision component needs to be able to track dynamic obstacles with high frequencies so that milestone controllers can update accordingly.

8.5.1 Static environment representation

Our representation of the static world is a hierarchical voxel grid called Octomap (Hornung et al., 2013). This data structure allows to integrate point clouds from RGBD sensors efficiently. The Octomap does not directly support efficient collision queries. To provide efficient collision checking, we also maintain a signed distance function. This function is also updated incrementally in constant time with a brushfire algorithm (Lau et al., 2013). We implement a collision check on this data structure by approximating the robots volume with bounding spheres and evaluating the signed distance function at the center of each sphere.

[1] These actions replace the waiting edges (i,i) of the original expected shortest path (ESP) algorithm. They capture a new option for the robot that is not possible with the original ESP formulation: it might be useful to wait a short time for an edge to become unblocked instead of taking the next best free edge.



(a) (b)

Figure 8.9 (a) Point cloud obtained with the robot's on-board sensor, colored by depth (b) two bounding boxes returned by the segmentation algorithm around dynamic obstacles not contained in the map

8.5.2 Dynamic obstacle tracking

To track dynamic obstacles, the vision component first filters all static obstacles as defined by the static map from the pointcloud. It then clusters all remaining points from the point cloud into point cloud regions (see Fig. 8.9). Each of these point clusters is considered to be an obstacle.

The algorithm tracks dynamic obstacles by comparing the centroids of the point cloud regions to the position of regions in the previous frame. The segmentation processes 10 frames per second and is able to track obstacles moving at up to 1 m/s. The ESPER planner computes an axis-aligned bounding-box around the point cloud region and places milestones at the corners of each box. Using these milestones, we can then estimate the edge parameters t^B_{mn} and t^F_{mn}. To estimate if an edge is traversable, we observe the amount of free workspace between two milestones. In our world representation, we cast rays between sampled points on the milestone configurations and check for collisions with the bounding boxes. We perform these tests continuously for each edge whenever it is inside the sensor range and store the duration that each edge is consecutively blocked or free. We compute the expected durations t^B_{mn} and t^F_{mn} by averaging over all of these time measurements.

8.6 Experimental evaluation of the ESPER planner

In two real-world experiment we show the incremental elastic roadmap is able to satisfy our three initially stated requirements for motion generation. We show the method is able to

- satisfy *task constraints*. The robots carries a tray while avoiding obstacles. During motion it has to solve a whole-body planning task of lowering the tray to avoid collisions.
- deal with *unpredictable dynamics*. The robot reacts to appearing as well as disappearing connectivity in the environment. The robot handles appearing obstacles in the environment but also the removal of previously seen obstacles. In the first experiment we show the incremental roadmap construction using local planning. In the second an third experiment we will focus on the system reasoning about uncertainty.
- rely only *on-board sensors*. In both experiments the robot uses only the sensor data from one RGB-D camera mounted on-board. The robot has no prior information about the structure of the environment.

In the next section we will discuss the setup of the experiment and afterwards evaluate the results.

For the experiments we implemented the proposed method on a mobile manipulator consisting of a Nomadic XR4000 base and a Barrett WAM arm, resulting in a robot with ten degrees of freedom. Mounted on the base is an Asus Xtion Pro RGB-D sensor which obtains depth information in front of the robot. All experiments are based on a C++ implementation of the elastic roadmap planner on a standard Windows desktop PC. We use our mobile manipulation platform, consisting of holonomic base (Nomadic XR4000) and a seven degree-of-freedom manipulator (Barrett WAM). In all experiments the planner continuously plans at 1 Hz. The majority of runtime is spent on milestone maintenance, the time for computing the Expected Shortest Path never exceeds 100 ms.

We perform milestone maintenance using velocity-based multi-objective control (Sec. 2.4.5). We implement two different task constraints in the experiments: in two experiments the robot must hold a tray level. This is implemented using a potential on the end-effector orientation. In the third experiment, the robot must keep its end-effector on a line. We implement this with a 2D potential acting on the end-effector. Obstacle avoidance is realized with repulsive potentials acting on base and elbow of the robot. All controllers and the dynamic simulation are implemented in RoboticsLab, a robotic tool kit by SimLab.

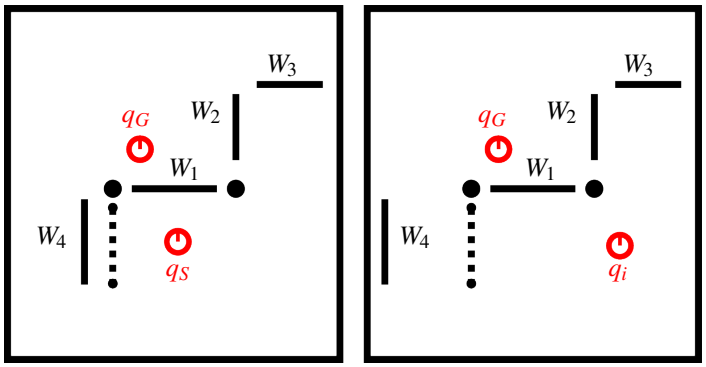


Figure 8.10 Sketch of the experiment with walls $W_1 - W_4$ labeled for explanation. start (q_S), goal (q_G) and intermediate position (q_i) are shown in red. W_4 is moved when the robot reaches P.

8.6.1 Incremental roadmap maintenance in an unknown environment

In this real-world experiment we show the incremental elastic roadmap is able to maintain task-consistent roadmap under significant connectivity changes. The task in experiment is orientation constrained pick and place: The mobile manipulator moves a tray upright to a goal location. The mobile manipulator starts with no information about the environment. During the experiment a person moves a wall which opens up new connectivity. The robot has no access to global localization during motion.

Figure 8.10 shows the setup of the experiment. The robot is standing in front of wall W_1 and oriented towards it. The goal of the experiment is to bring the mounted tablet to the goal position behind wall W_1. The robot has a limited sensing range and only observes wall W_1 in the beginning. Due to the setup the robot will first choose a path to the right of wall W_1. Once it reaches a position (q_i) where it can observe this path is blocked a person moves wall W_4 to the border of the scene. This frees the passage below a curtain. The robot proceeds on a path around the left of W_1 to the goal.

In total the experiment contains three different sources of unpredictable dynamics: First, as the robot starts without prior knowledge, it integrates new measurements continuously in its world model. Every detection of a new obstacle represents a change to the robots perceived environment and invalidates previously planned paths. Second, the manual removal of a wall opens up connectivity that was blocked before. The robot has to notice the change of connectivity and react by planning a new alternative locally. Third, there are additional dynamics present because of the robots odometry error. Without global localization, the world is moving constantly relative to the robot. The roadmap also has to adapt to this motion.

Evaluation

The experiment is depicted in Figure 8.11. After 9 s the planner finds the two paths which describe the connectivity of the scene at the beginning. After an initial period of switching between these two paths to map out the initial scene the robot starts to move on a path to the right of wall W_1 and detects walls W_2 and W_3. The planner reacts to the change by removing all paths around the right of W_1 and chooses a path around W_4 at 152 s. A person moves W_4 to the border of the scene and enables a new disjoint path alternative. The robot continues on its path and observes that W_4 is missing. The planner adapts the current path to pass below the curtain at 187 s and generates a shortcut through the freed space at 216 s. The robot passes below the curtain at 296 s and retracts the arm to avoid collision. It continues on the path and reaches the goal at 405 s.

The experiment shows that the method continuously finds new connectivity at responsive rates. It approximates the connectivity of the scene throughout all stages of the experiment. At the beginning the planner is able to find the two initial path possibilities in quick succession. Both significant changes in connectivity are incorporated into the roadmap:

- The restriction of the connectivity to one possibility after the detection of wall W_2 and W_3.
- The appearing connectivity after wall W_4 is removed.

The experiment also highlights the necessity of local feedback to preserve the connectivity of the roadmap. The initial paths are adapted to the continuous detection of W_1 and W_4. During the experiment the base of the mobile manipulator is subject to significant odometry error. The roadmap is able to compensate this position uncertainty with feedback as well. During the experiment the mobile manipulator never has to 'stop and think' except for a short time of initial roadmap construction. After initialization the roadmap always supplies alternative paths and the robot fulfills the task in one continuous motion.

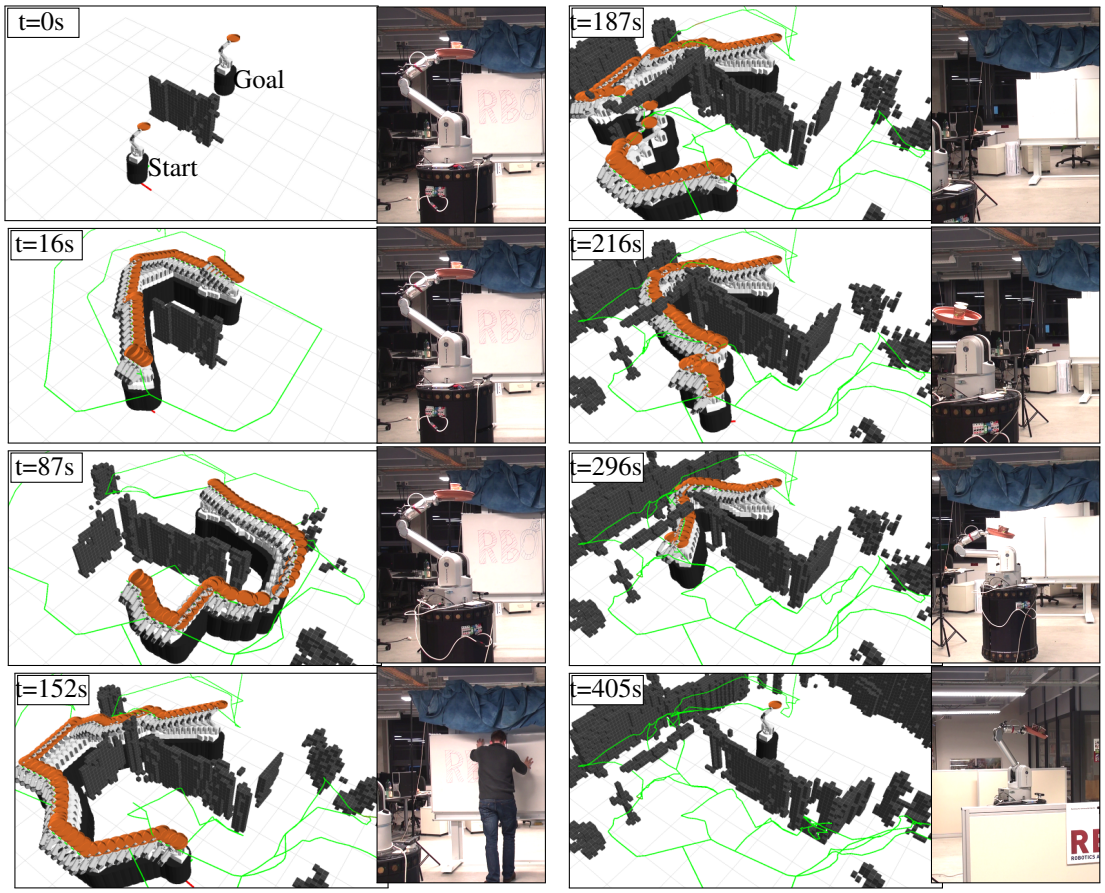
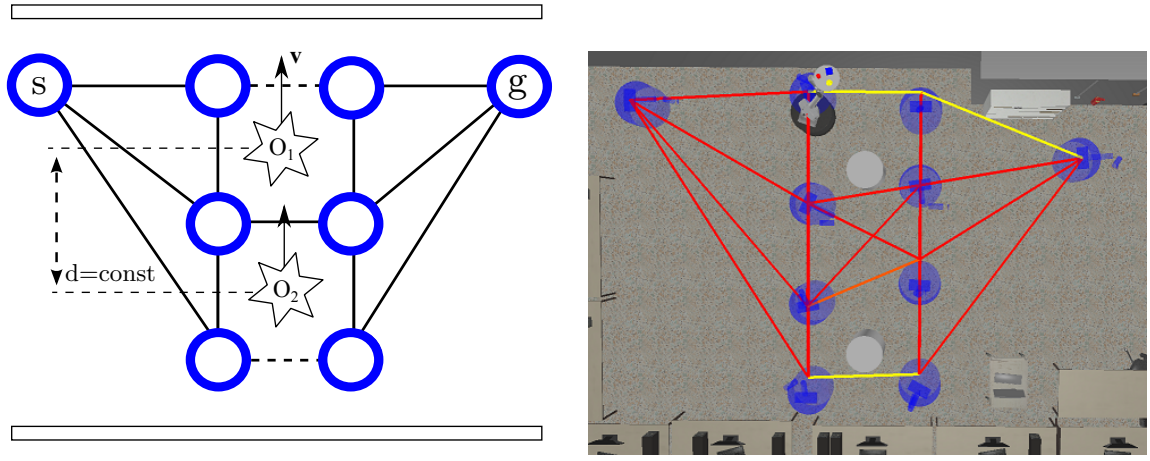


Figure 8.11 The roadmap end-effector positions are shown in green, the occupancy grid is shown in dark grey, and the current path is shown as the swept workspace of the configurations. At $t = 0$s the robot perceives only the wall W_1 in front of it. It receives the task of moving to a goal configuration behind the wall. Initially, it chooses a path to the left of W_1 at $t = 16$s. After alternating three times between paths to the left and right of W_1 it chooses a path on the right at $t = 87$s. It follows this path around the corner and discovers the right path is blocked at $t = 152$s. The robot instantly chooses a path to the left of W_4. We manually remove W_4 and the algorithm adapts a path through the hole at $t = 187$s. The algorithm perceives the new motion alternative and finds a path underneath the curtain at $t = 216$s. At $t = 296$s the robot passes below the curtain and retracts the arm to avoid collision. At $t = 405$s the robot safely reaches the goal. See https://www.youtube.com/watch?v=o3XWjU6iuCw for a video of the experiment.



(a) (b)

Figure 8.12 *(a)* Illustration of an elastic roadmap in a scene with two dynamic obstacles (only a subset of roadmap edges is shown): d remains constant throughout the up-and-down motion of obstacles O_1 and O_2; we expect that the dashed have high collision probabilities because the obstacles move near the upper and lower walls; the solid edges should have low collision probability; for high collision probability, the expected shortest path should lie between O_1 and O_2, while the shortest path will be above O_1. *(b)* Screenshot of the execution of the incremental elastic roadmap planner. The two obstacles are grey cylinders in the middle of the room. The milestones are shown in blue. The colouring of the edges corresponds to the estimated collision probability with yellow being high and red low. The values correspond to the expected probabilities in (a)

8.6.2 Reasoning about uncertain environments

In this experiment, we will show how explicit handling of uncertainty reduces execution time and collision rates compared to uncertainty-unaware planning. The mobile manipulator moves in a dynamic environment (Fig. 8.12a), while holding a tray with objects level. Two obstacles, O_1 and O_2, move from wall to wall but always keep a fixed distance d. In this scenario, both the paths above (referring to the figure) O_1 and below O_2 are invalid about half of the time.

ESPER places milestones around the two dynamic obstacles. It observes the moving obstacles for 10 seconds and then computes a higher collision probability for the upper and lower edges shown in yellow. The path between the two obstacles is valid all the time and thus has a lower collision probability (shown by the red lines). Fig. 8.12b shows a snapshot of the roadmap created with ESPER during execution. The edge probabilities qualitatively match the expected behavior from Fig. 8.12a.

We compared ESPER to an uncertainty-unaware implementation of the Elastic Roadmap that only return the shortest path (*ER*). This planner uses the shortest path over the milestones

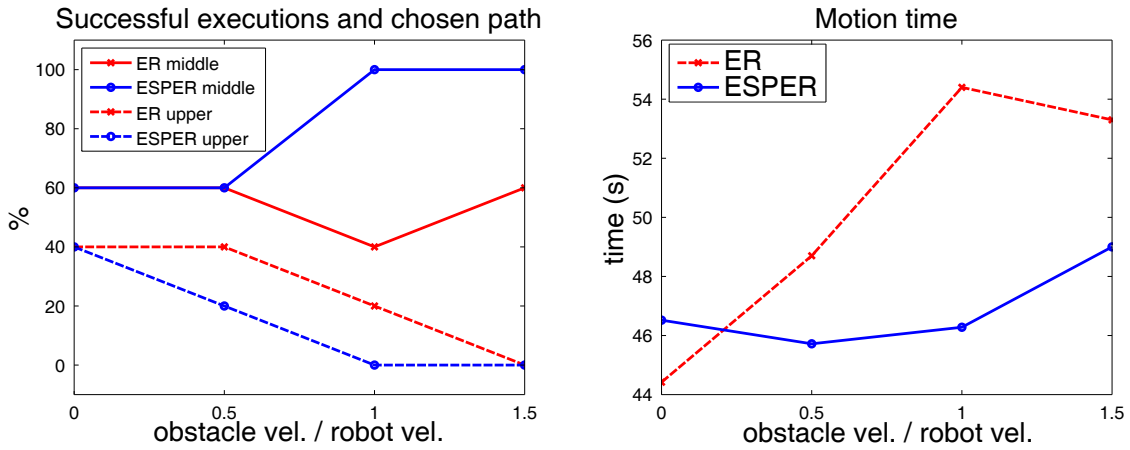
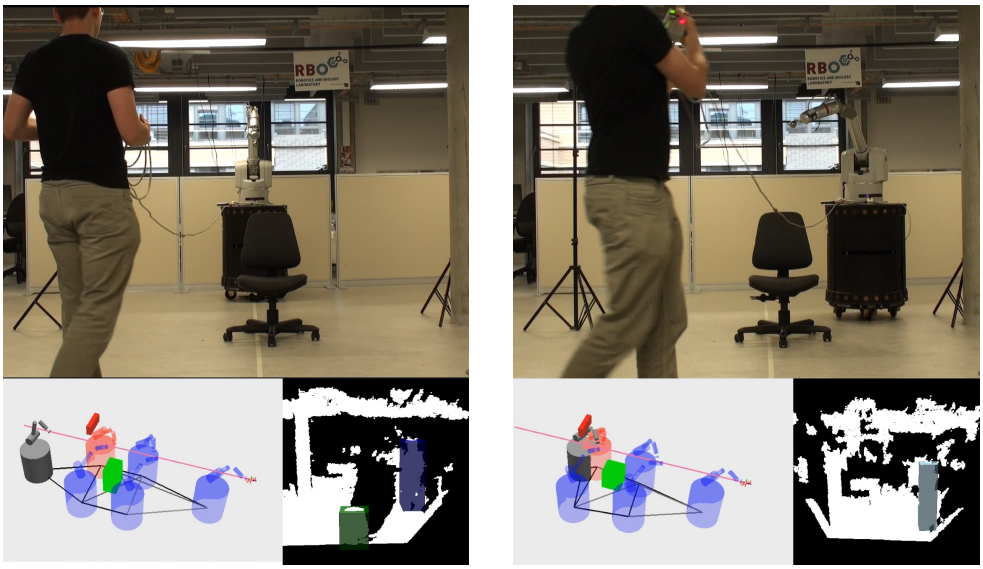


Figure 8.13 In both graphs the x-axis represents the obstacle velocity, relative to robot velocity; the y-axis in (a) represents the success rates for different paths (numbers do not add up to 100% for uncertainty-unaware planning due to collisions) (b) shows the average execution time. The results show that if obstacles move faster, the ESPER planner chooses a safer path between the obstacles more often than the ER planner that does not reason about uncertainty

it believes to be passable at planning time and recomputes a new path every second. We executed the motion in 50 simulation experiments with randomized obstacle positions. For low obstacle velocities, both planners perform roughly equal and both planners choose paths above O_1 and between O_1 and O_2. For higher obstacle velocities, the path above O_1 becomes often blocked. The uncertainty-unaware elastic roadmap planner still chooses both paths, depending on the exact obstacle positions. (Fig. 8.13a). In 20% of the runs with high velocities, the robot gets stuck between the upper wall and O_1, leading to a collision. In contrast, the ESPER planner always chooses the path between the obstacles, which is safe even at high obstacle velocities (no collisions). The overall motion time for the non-colliding runs is also lower for the uncertainty-unaware planner (see Fig. 8.13b). This experiment shows that explicit handling of environment uncertainty reduces motion time and can even help to reduce the probability of collisions.

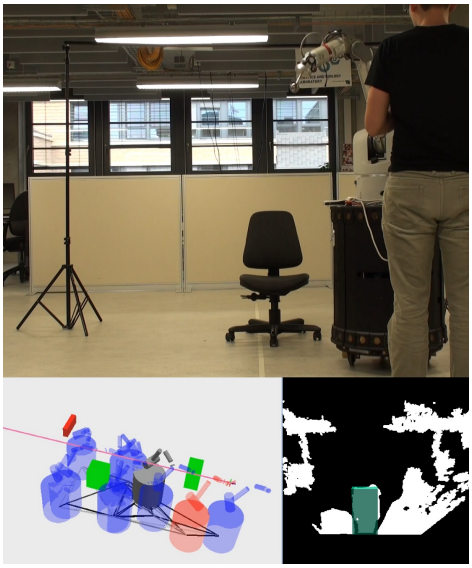
8.6.3 Sensor-based planning and execution

In the last experiment, we will validate the requirement for perceiving the world only with on-board sensors. The robot is placed in the starting configuration shown in Figure 8.14a and receives the task to move 4 meters forwards while keeping its end-effector on a virtual line.

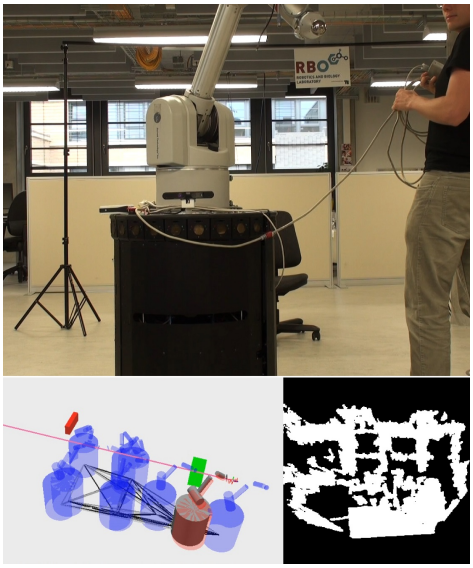


(a)

(b)



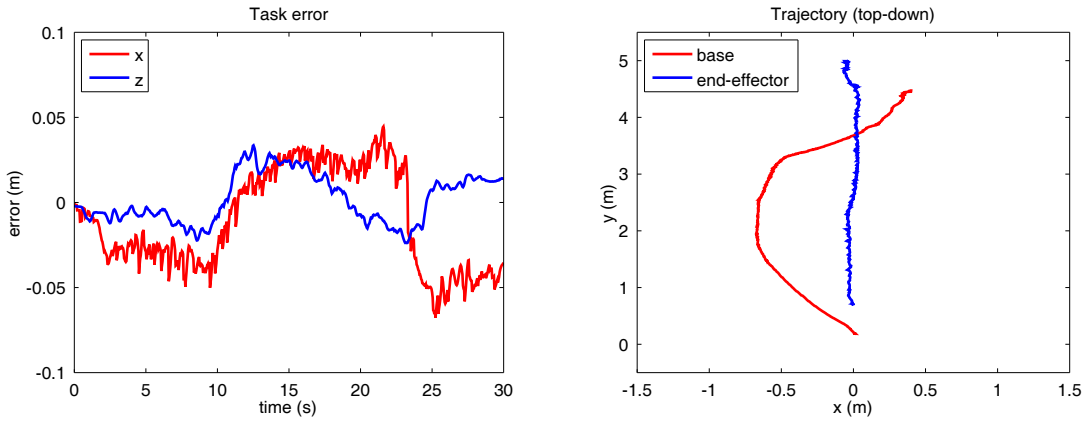
(c)



(d)

Figure 8.14 a) Execution on a mobile manipulation platform. Shown below the screenshots from the execution are on the left: the current state of the ESPER planner. The task is shown as a pink line; The bounding boxes of the observed obstacles are shown in green; The blue robot shapes are all task-consistent milestones; Shown in red is the current milestone with lowest expected cost. On the right we show the matching point cloud view from the robots perspective. The segmented obstacles are shown in color. See https://www.youtube.com/watch?v=netNfOyXjtk for a video of the experiment.

The planner creates the static map from sensor data consisting of the floor, the sign on the ceiling, and the columns for localization and background filtering. Afterwards two unknown, dynamic obstacles enter the scene: a chair and a person. The rolling chair is in front of the robot and static for the rest of the experiment. The ESPER planner generates task-consistent intermediate milestones on both sides of the chair. In front of the chair the person moves continuously on the right side of the line. Initially, the robot observes this scene for 20 seconds. During this time, the robot observes the person. The planner estimates the edge parameters t^F and t^B, which it uses in the computation of the expected shortest path. The resulting roadmap from Figure 8.14a shows a higher value for t^B on the edges that were blocked by the person (visible by the lighter shade). Then we start computing the expected shortest path with the observed parameters. Within the first iteration (under 1 second), it computes a path which guides the robot to the other side of the chair, avoiding the edges that were previously blocked by the person (see Fig. 8.14b). On this path, the intermediate milestone guides the robot to lower its elbow to avoid collision with the sign. This shows the whole-body capabilities of the controllers. After crossing the chair and avoiding the sign, the person unexpectedly crosses the line to the other side (Fig. 8.14c). The robot avoids the collision by moving the milestones, replanning, and adapting its path (Fig. 8.14d). Fig. 8.15b shows the executed trajectory of the robot. During motion the end-effector error with respect to the base localization never exceeds 6 *cm* (see Fig. 8.15a).



(a) (b)

Figure 8.15 Task error and base trajectory for the motion of Fig. 8.14. The end-effector position is constrained to be constant in x and z directions. The task error never exceeds 6 cm.

All requirements R1 to R3 were present in this experiment. The robot motion was subject to task constraints and whole-body motion was required to avoid the sign (R1). The robot reasoned about the uncertain occurrence of the person, which let it move on a path that avoids

the person early on, avoiding collision and reducing the expected motion duration (R2). Finally, the motion was generated based on on-board sensor data using a both an RGB-D sensor and a laser range finder (R3).

8.7 Related work in sensor-based motion generation

We want to briefly discuss alternative approaches to the incremental planning problem. Compared to the more general methods in Chapter 7, these only contain methods that integrate sensing and motion planning on real sensor data in dynamic environments.

Motion planning

Sampling-based motion planners can accommodate dynamic obstacles with known trajectories (Hsu et al., 2002) and they also handle sensor-generated world models (Chitta et al., 2012). However, they are difficult to apply in unpredictable environments as unexpected changes in the environment invalidate large parts of the tree or roadmap.

Online and reactive planning

Several approaches presented in the literature aim to equip global planners with reactive capabilities. We analyze if these methods satisfy the requirements of applications in mobile manipulation.

Replanning

Replanning (Hauser, 2012) is a simple method to adapt a solution path in unpredictable environments and has been applied successfully for small mobile manipulation tasks (Chitta et al., 2012). Replanners observe the feasibility of a planned motion at execution time. The planner plans one trajectory and checks its validity continuously. For invalid trajectories, the planner is simply evoked again. In highly dynamic scenarios, replanning would lead to significant interrupts in the motion of the robot: in the time a new plan is computed, it would be immediately invalidated by new changes in the environment.

Online adaption

Some methods plan one (Park et al., 2012; Kappler et al., 2018) or more (Vannoy and Xiao, 2008) initial paths using a global search that subsequently are adapted using trajectory optimization methods. In environments with small changes, these methods lead to satisfactory

results. However, the large scale environments in our experiments require reasoning about the connectivity using planning.

Adaptive roadmaps

Some adaptive roadmap-based methods validate edges in the roadmap at run time, thus adapting the plan to environmental changes (Leven and Hutchinson, 2002; Nakhaei and Lamiraux, 2008). Using hardware parallelization this approach can be scaled to real time performance (Murray et al., 2016). In unpredictable dynamic environments, multi-query roadmaps outperform single-query tree-based planners, as the invalidation of an edge simply leads to the selection of another path in the roadmap. Such roadmap-based methods are an efficient way to reuse previously found connectivity. A similar approach (Yoshida and Kanehiro, 2011) combines an adaptive roadmap with local replanning but does not consider real sensor data and task constraints. Gaussian process motion planning has been applied to optimize roadmaps (Huang et al., 2017), although only in 2D problems using homotopy.

Planning with feedback

Feedback plans map each state of the robot to an action (Burridge et al., 1999; Yang and LaValle, 2004). As a result, feedback plans accommodate action uncertainty: no matter what the outcome of an action, the plan specifies a next step to make progress towards the goal. FIRM (Agha-Mohammadi et al., 2014) generates a feedback motion plan as a roadmap of controllers in the belief space, addressing the robot's state uncertainty. Feedback motion planners rely on known and static environments. Our method instantiates a similar graph-based composition of controllers based on the Elastic Roadmap framework (Yang and Brock, 2010) for whole-body, task-consistent motion in dynamic environments. We do not plan in belief space but we will continuously employ feedback to adapt the plan relative to sensed obstacles.

Planning in uncertain environments

MDPs were used to model planning with environment uncertainty (Missiuro and Roy, 2006; Burns and Brock, 2007). These methods then produce policies that keep safe distance to unknown obstacles. However, these methods rely on a pre-existing map. Approximate solutions to POMDPs were successfully applied to 2D mobile robot navigation (Simmons and Koenig, 1995) and motion planning under sensing, action, and environment uncertainty (Kurniawati et al., 2012).

A similar problem to the Expected Shortest Path problem is the **Canadian Traveller Problem** (Bar-Noy and Schieber, 1991), where edges can be either traversable or blocked but the robot will only know once it is in an incident node. Generalizing this, the **Stochastic Shortest Path Problem with Recourse** (Polychronopoulos and Tsitsiklis, 1996) models the costs of an edge as probability distribution and assumes that the true cost is revealed to the robot once it reaches an incident node. While both problems are of exponential complexity, efficient solution algorithms exist and have been applied to navigation scenarios (Chung et al., 2018).

A related, also exponentially complex, approach models the state of each edge in an MDP as a Markov Chain so as to address environment uncertainty. Policies for 2D navigation were found by planning in the belief space over a reduced graph (Marthi, 2012), or by iterating over discrete time-steps (Loibl et al., 2013).

Pilania and Gupta (2018) present a sensor-based motion generation method similar to ours that also reasoning about the sensor range, performing deliberate sensor placement to improve the world representation. In contrast to our approach, their method decomposes base and arm motion, assuming the base to move in the plane. Our approach does not make any assumptions about the kinematics and could be applied to any platform, including motion on uneven ground or flying mobile manipulators.

Belief space HPN (hierarchical task and motion planning in the now) (Kaelbling and Lozano-Pérez, 2013) is an integrated task and motion planning method under uncertainty. The method handles objects that can be moved by the robot, although it does not consider objects moving independently of the robot. The method plans sensing and motion actions for complex interactions with the environment by planning on symbolic abstractions of the belief space. To keep the reasoning simple, the method interleaves planning with execution, just like our method. Our method is complementary to this and other task and motion planning approaches. Our method currently only supports a single task constraint but we expect it could be extended with more complex constraints coming from a task planner.

8.8 Conclusion and limitations of the ESPER planner

We presented the Expected Shortest Path Elastic Roadmap (ESPER) planner, an approach for the sensor-based generation of task-constrained, whole-body motion under uncertainty. The key contribution of this method consists in the division of the motion generation problem into three sub-problems: addressing continuous changes in the world, addressing connectivity changes, and finally obtaining a global motion policy that takes uncertainty into account. By dividing the problem in this fashion, we can employ the most appropriate and efficient

method for solving each sub-problem. As a result, the method is able to efficiently generate global, task-constrained, and reactive feedback plans. The generation and execution of these plans is based on a world model the robot acquires throughout its motion. We demonstrate the effectiveness of this approach on a real-world manipulator performing a constrained end-effector task in a complex and unpredictably changing environment and also in an environment with unpredictably moving obstacles.

There are two limitations to the ESPER framework. First, our approach assumed unknown space to be free. This only leads to safe motion if the robot's cameras have sufficient field of view so that all obstacles in the direction of motion are visible. The issues can be addressed by reasoning about unknown occupancy (Burns and Brock, 2007; Pilania and Gupta, 2018) which might be a viable extension for the ESPER framework. Second, the task constraints in our method were purely geometric and included no feedback from tactile or visual sensing. Therefore, the ESPER planner is limited in the manipulation skills it can achieve, especially when complex and precise interactions are required. We established the importance of direct sensor feedback in Part I. In the next section we will present first results for a sensor-based approach, but we leave integrating sensor-based control into the planner for future work.

Chapter 9

Towards sensor-based motion generation

In the previous chapters, we presented robust sensor-based methods to generate motion for manipulation. Those methods presented allowed to specify manipulation tasks geometrically by defining constraints in task space. While geometrically constrained motion is a prerequisite for manipulation, it is not always possible to extract geometric constraints from sensor data. Many manipulation tasks can be specified in terms of desired sensor signals, e.g. the position of image features, or the attainment of a specific force signal.

To use rich sensor feedback for manipulation tasks, robots need **forward models** that relate task-relevant features in the sensor stream to actions. These models then aid in solving a particular task by using them for planning and/or control. For some aspects of manipulation like robot dynamics (see eq. 2.7), good forward models are available. For other parts, e.g. the dynamic properties of the environment, we cannot assume forward models to be available. These models must be learned from data of the robot interacting with the world. There is a lot of related work for learning forward models from data. Recent deep learning based methods (Lenz et al., 2015; Levine et al., 2018) learn models for complex interactions when a lot of training data is present, either from simulation, from human demonstrations, or from repeated executions of the task. Obtaining these data incurs substantial costs, as either humans must provide input, or the robot must run for long time spans. These approaches become intractable for manipulation tasks with high variability where robots can not invest significant time into retraining whenever the setting is slightly changed.

In this chapter, we want to approach forward model learning from another perspective and ask: How can we learn forward models while interacting, without any other source of training data? We want the robot to attempt a task and then, in one shot, improve the model and eventually solve the task. Our approach is based on the assumption that the relationship between actions and changes in sensor signals is sufficiently smooth to be continuously approximated linearly. This assumption allows us to continuously approximate a Jacobian-

based forward model using recursive Bayesian estimation from pairs of actions and observed changes.

We will show in simulation experiments how our method can fulfill manipulation tasks with multi-modal feedback, integrating vision and force sensing. The method is completely uncalibrated in a sense that the robot has no a priori knowledge about its perceptual capabilities, but learns during interaction to relate its sensor data to its actions. The contents of this chapter are unique to this thesis and have not been published before.

9.1 Sensor-based feedback control

Vision is a cheap and rich source of data is of fundamental importance for manipulation, for example, as input to our perception system in our Amazon Picking Challenge system from Chapter refcha:apc. The DARPA Robotics Challenge (Zucker et al., 2015) validated that humans are able to solve complex manipulation tasks with a robot just using feedback from a video stream. In this thesis so far, we specified motion constraints for planning and control in terms of geometry. This required users to specifiy tasks as constraints on position and orientation of operational points on the robot. Such constraint definitions pose several challenges for vision, as these constraints require to reconstruct 3D geometry from 2D sensor data. This process is prone to uncertainty which will directly affect the task success. RGB-D sensors provide depth data which simplifies the problem but requires well-calibrated cameras as any error in calibration will directly affect task success.

In this part we propose an alternative, inherently robust approach for control on visual and other sensors' feedback. Instead of specifying manipulation goals in task space, we will specify them in **sensor space** $\mathcal{S} \subset \mathbb{R}^k$. An element $s \in \mathcal{S}$ is an k-dimensional vector of task-relevant features. A task relevant feature is a feature that a) changes if the robot moves and b) obtains a consistent, known value at the goal of a manipulation task. A manipulation task is then specified by providing a feature vector s_d that contains all the desired sensor values that should be attained.[1]

9.1.1 Visual servoing

A well known method for control in sensor space is (image-based) **visual servoing** [2] closes a feedback loop around visual input. The method can be applied either by mounting a

[1] We do not address how task-relevant features can be found automatically and assume they are given by the user.

[2] in contrast to position-based visual servoing which extracts geometry and controls in task space, as mentioned initially

camera on a robot and tracking features on the environment (eye-in-hand) or by letting a static camera view features on the robot and the environment (eye-to-hand). Visual servoing now assumes that some visual processing pipeline can reliably track a list of image features. Commonly used image features are the image coordinates of m tracked point features (u_i, v_i), stacked into a single vector $s = (u_0, v_0, u_1, v_1, \ldots, u_m, v_m)^\intercal$. Image-based visual servoing now requires a target feature vector s_d that describes the position of features at the manipulation goal. The method computes a desired control signal $\dot{x}_d \in \mathbb{R}^6$, which is the 6-dimensional velocity vector of the robot part we can control. The method regulates an error $e_s = s_d - s$ between the measured position of features s and the desired position of features s_d. Just like in task space control, we use a Jacobian to map between two spaces, this time between feature space $\mathcal{S}$ and task space $\mathcal{T}$. The image Jacobian (or interaction matrix) $L \in \mathbb{R}^{k\times 6}$ describes the relationship between robot and feature motion as:

$$\dot{s} = L\dot{x} \tag{9.1}$$

Using a pseudoinverse L^+ and a proportional gain $\lambda \in \mathbb{R}^6$, we can compute a control signal $\dot{x}_d$ that minimizes the error as

$$\dot{x}_d = \lambda L^+ e_s \tag{9.2}$$

We can then execute this task-space velocity on the robot by using the velocity-based task space controller (2.13). The remaining question is how to compute L. For many types of features, L can be computed analytically. For example, the Jacobian for a point feature with image coordinates (u, v), assuming that the task velocity is expressed in the camera frame is given by:

$$L = \begin{pmatrix} -f/Z & 0 & u/Z & uv/f & -(f+u^2/f) & v \\ 0 & -f/Z & v/Z & f+v^2/f & -uv/f & -u \end{pmatrix} \tag{9.3}$$

where $f \in \mathbb{R}$ is the focal length divided by the pixel size and $Z \in \mathbb{R}$ is the Euclidean distance between camera and feature.[3] To control the robot using a number of m different point features, this Jacobian is computed for all features and all of them are stacked into one $(2m \times 6)$ matrix. If the rank of this matrix is greater or equal to six, the pose of the robot's end-effector can be controlled reliably. This is the case when using at least four image features, (Chaumette and Hutchinson, 2006). To actually control the robot, the computed

[3] this distance is usually unknown, but the control law usually converges if Z is set to a constant approximate value.

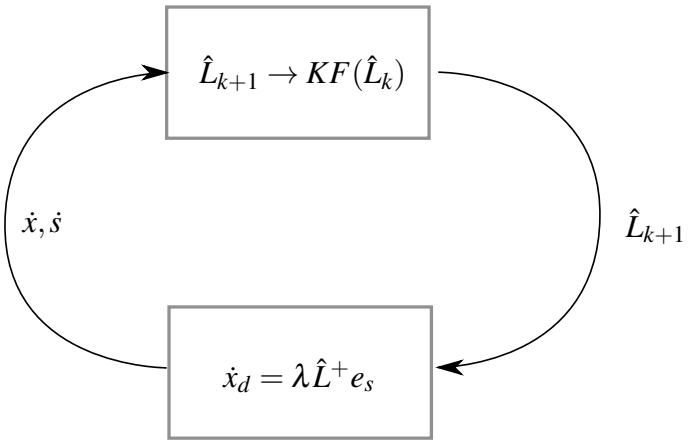


Figure 9.1 Basic functionality of uncalibrated servoing. The Jacobian is continuously updated with measured robot velocity $\dot{x}$ and feature motion $\dot{s}$ using a Kalman filter update. The new Jacobian is then passed to the controller, which computes a new command $\dot{x}_d$ using a Pseudoinverse and sends it to the robot.

velocity $\dot{x}_d$ must be transformed into the frame under control (i.e. the end-effector) using the adjoint transformation between camera and end-effector frame: $\dot{x}_{\text{cmd}} = \text{Adj}(T_{\text{cam}}^{\text{EE}})\dot{x}_{\text{d}}$[4].

9.1.2 Uncalibrated sensor-based control

Using any visual servoing law with analytic Jacobians requires calibrating unknown parameters such as focal length and the mounting of the camera relative to the robot. Especially for eye-to-hand settings, calibration can be tedious , error-prone, and must be repeated every time the camera moves. The idea of uncalibrated servoing is not to use any calibration but to estimate the whole Jacobian while controlling. The robot receives during motion pairs of measured robot motion $\dot{x}$ and feature motion $\dot{s}$. Using this data it then estimates the Jacobian $\hat{L}$ completely based on data from robot motion and feature motion (see Fig. 9.1). Theoretically, this estimation can be done for every feature type where feature motion correlates with robot motion. We will now explain how to do this estimation robustly by taking prior knowledge about the uncertainty of different sensor sources into account. We rewrite equation 9.1, by

[4] see Lynch and Park (2017) for details on velocity vector transformations, including the definition of the adjoint transformation

stacking the rows of the Jacobian L in one column vector $l \in \mathbb{R}^{m \cdot n}$:

$$\dot{s} = L\dot{x} = Hl \tag{9.4}$$

$$l = (L_{1,1}, L_{1,2}, \ldots, L_{1,n}, L_{2,1}, L_{2,2}, \ldots, L_{m,n})^T \tag{9.5}$$

$$H = \begin{bmatrix} \dot{x}^T & & 0 \\ & \ddots & \\ 0 & & \dot{x}^T \end{bmatrix} \tag{9.6}$$

We now assume that the sensor data is influenced by Gaussian observation noise $v \sim \mathcal{N}(0,R)$, so that $\dot{s}_{k+1} = Hl_k + v$. We also assume that the entries of the Jacobian change as if the were affected by Gaussian process noise $w \sim \mathcal{N}(0,Q)$ so that $l_{k+1} = l_k + w$

These assumptions allow to use l as a recursive estimation process that is solved optimally by the Kalman filter (Kalman, 1960). The solution is given by iteratively applying the following update rules:

$$P'_{k+1} = H_k P_k H_k^\intercal + Q \tag{9.7}$$

$$K_{k+1} = P'_{k+1} H_k^\intercal (H_k P_k H_k^\intercal + R) \tag{9.8}$$

$$l_{k+1} = l_k + K_{k+1}(\dot{s}_k - H_k l_k) \tag{9.9}$$

$$P_{k+1} = (I - KH)P'_{k+1} \tag{9.10}$$

To use uncalibrated servoing, we initialize the Jacobian l_0 with random values. Using this random Jacobian in a servoing controller leads to random motion. This random motion provides the robot with observation pairs $(\dot{s},\dot{x})$. The update rule modifies the Jacobian based on this observation. Once the Jacobian is estimated sufficiently well, the controllers starts converging to the goal.

To control the robot we do not directly use the Pseudoinverse 9.2, but take the ongoing estimation into account. One issue with using an estimated Jacobian is that small errors might be mapped to very high robot velocities which are not feasible to execute. To reduce the control signal when $\hat{L}$ is not estimated well, we use an adaptive gain $\lambda_{L_{\max}}$ based on the maximum entry of $\hat{L}^+ e_s$ (Kermorgant and Chaumette, 2014):

$$\lambda_{L_{\max}} = (\lambda_0 - \lambda_\infty) e^{-\frac{\lambda'_0}{\lambda_0 - \lambda_\infty} \|\hat{L}^+ e_s\|_\infty + \lambda_\infty} \tag{9.11}$$

This reduces the commanded velocity when the Jacobian is not estimated well. As a second measure, we compute the end-effector velocity command $\dot{x}_d$ using a weighted Pseudoinverse of $\hat{L}$. The weights w_i shape the error term so that low-weighted features contribute less to the

control signal. We set the weight matrix to the inverse measurement covariance $W = R^{-1}$. This makes unreliable features contribute less to the control signal. The full control law is:

$$\dot{x}_d = \lambda_{L_{\max}}(W\hat{L})^{+}We_s \tag{9.12}$$

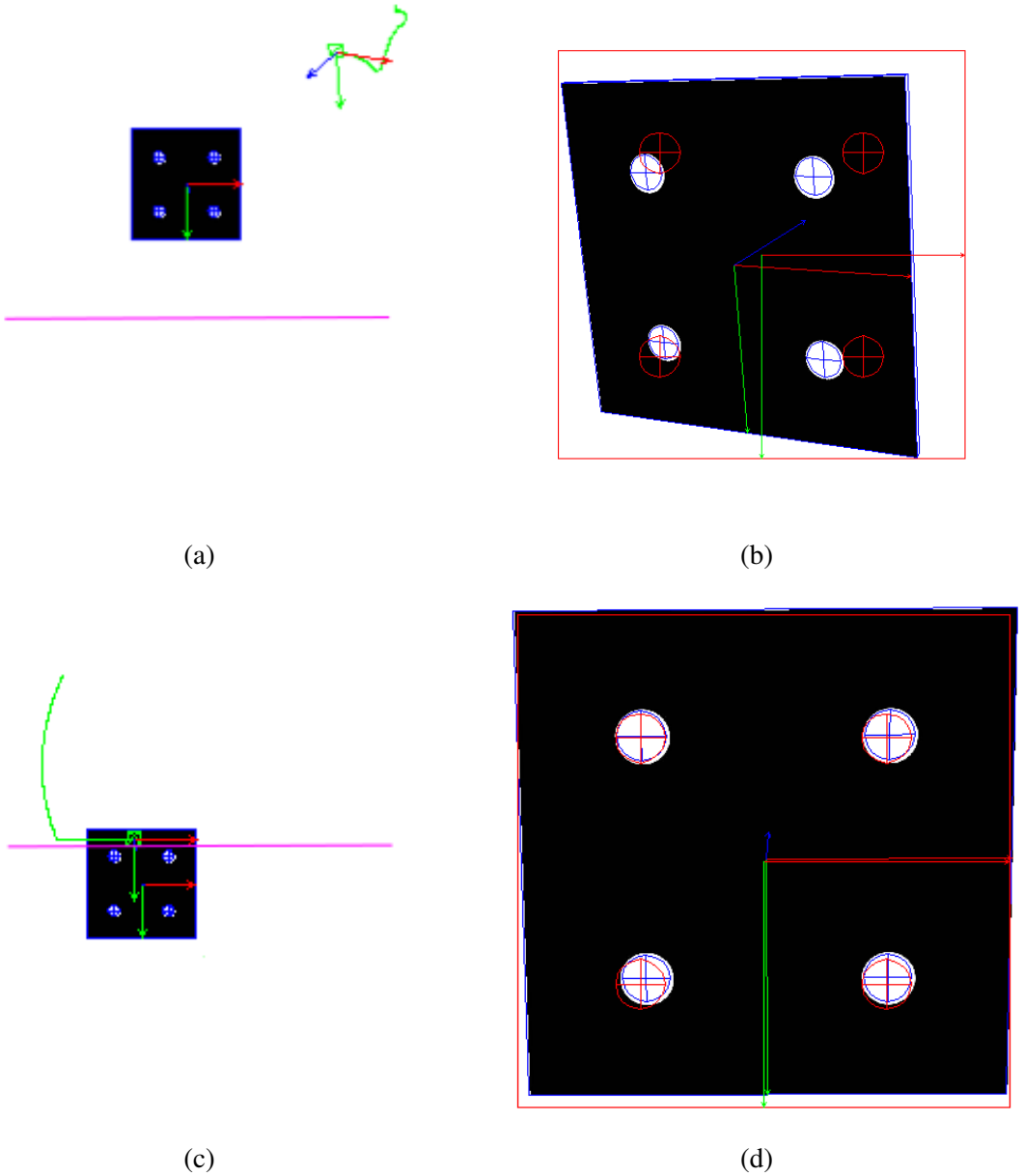
9.2 Experimental evaluation

We evaluate in simulation experiments that our method is able to robustly attain manipulation goals defined in sensor space without any a priori knowledge about the types of sensors. We model a free-flying 6-dof robot that has access to two sensors: A camera and a force sensor. The camera is mounted on the robot and returns the image coordinates (u_i, v_i) of four points on a square, moving plane. Second, a 1-D force sensor returns the force of a linear spring in contact with a ground plane at height d_z. Such a model is a simple approximation for contact with soft actuators, such as soft pneumatic hands (Deimel and Brock, 2016). The force signal F is given by:

$$F = \begin{cases} K_F(d_z - x_z), & \text{if } d_z > x_z \\ 0, & \text{else} \end{cases} \tag{9.13}$$

where $x_z \in \mathbb{R}$ is the robots current height in world reference frame and $K_F \in \mathbb{R}$ is a spring constant. We the define a feature vector as $s = (u_1, v_1, u_2, v_2, u_3, v_3, u_4, v_4, F)^\intercal$. We define the desired feature vector s_d by placing the robot in front of the target and remembering the feature positions (u_i^d, v_i^d). The desired force is set to 0 as we want to avoid contact. Thus, the complete desired feature vector is given by $s_d = (u_1^d, v_1^d, u_2^d, v_2^d, u_3^d, v_3^d, u_4^d, v_4^d, 0)$ We initialize the Jacobian randomly by sampling every entry from $\mathcal{N}(0, 0.01)$. We set the measurement uncertainty R for visual features to $R = 0.05I$ and the process uncertainty to $Q = 0.1I$. The coefficients for the adaptive gain are $\lambda_0 = 0.1$, $\lambda_\infty = 0.99$, and $\lambda_0' = 1$ We control the robot with an update rate of 100 hz and update the Jacobian with a lower rate of 10 hz.

Fig. 9.2 shows the internal and external view of the robot following the frame by visual servoing but also avoiding the bottom plane. In Fig. 9.3 we show the evolution of the error e_s over time for three different scenarios. Fig. 9.3a shows the controller only using visual feedback to converge to a stationary view of the target. In the beginning, the slope of the curve changes often due to the ongoing estimation of the Jacobian. Once the Jacobian is estimated at around $t = 7$s, the error decreases exponentially and converges to zero. Fig. 9.3a shows uncalibrated servoing of a target moving on a circular trajectory. A moving target is a harder task as it requires the robot to adapt the Jacobian continuously. After quick initial convergence, the error can sporadically increase up to 0.05 which is low enough to not loose

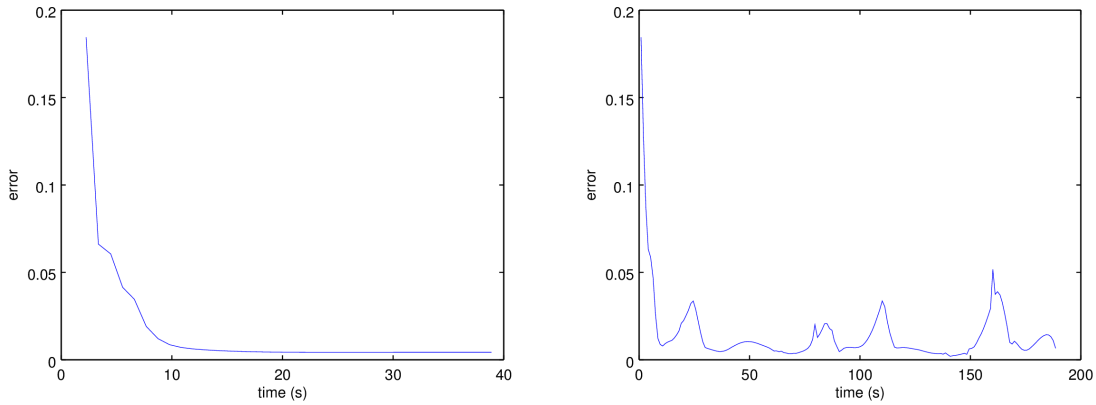


(a) (b)

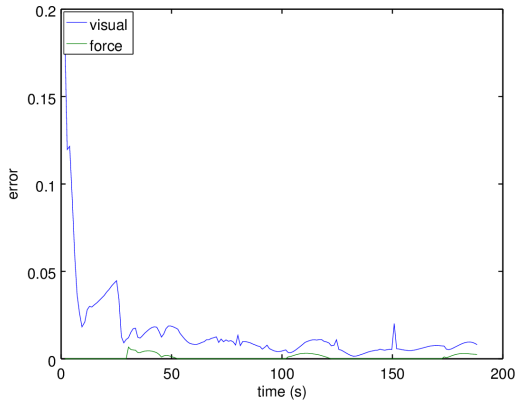
(c) (d)

Figure 9.2 Simulated execution of the uncalibrated servo controller on four point features and one force potential. (a) the robot is a free-flying camera (shown as a coordinate frame) facing a moving, black target. The pink line shows a bottom plane. If the robot moves below this plane, a force sensor measures contact. (b) The robot's view. The black square is the target and the red outline shows the position of the target at the desired view. (c) The uncalibrated servo uses visual feedback to follow a circular trajectory of the target but also avoids contact with the bottom plane. The trajectory of the robot is shown in green. (d) The force feedback term makes the robot leave the circular trajectory. The error in feature space remains low because the error can be reduced by tilting the camera downwards. See https://www.youtube.com/watch?v=p7a9UdR-jE0 for a video of the experiment.

the view of the object. Fig. 9.3c shows the full servo based on force and visual feedback. We add the force feedback term whenever the robot is close to the ground as shown in Fig. 9.2. The added force feedback term does not negatively influence the control and the error stays small enough to track the target reliably. This shows the capability of the method to combine different sensor modalities.



(a) (b)



(c)

Figure 9.3 Absolute value of feature error $\|e_s\|$ over time for three different servoing tasks. (a) convergence to a single, static view given by four image features. The error converges to zero. (b) The robot tracking a moving target. The error stays below 0.05 (c) The robot tracking a moving target and avoiding a ground plane using force sensing (experiment from Fig. 9.2). Both visual and force errors stay low throughout motion

9.3 Related work

Our method is a visual servoing (Chaumette and Hutchinson, 2006) method. Visual servoing techniques were extended to incorporate force sensors (Malis et al., 2001; Baeten et al., 2003; Prats et al., 2009) or tactile sensing (Li et al., 2013).

Uncalibrated visual servoing (Hosoda and Asada, 1994) estimates the image Jacobian while the robot moves using Broyden updates (Jägersand et al., 1997) or recursive least squares (Piepmeier et al., 2004). Uncalibrated servoing has been previously applied to combined force and visual feedback (Hosoda et al., 1998; Pichler and Jägersand, 2000; Yip and Camarillo, 2016). Our method generalizes uncalibrated servoing methods by reasoning explicitly about uncertainty with a Kalman filter. When setting the uncertainty estimates Q and R to zero, our methods performs recursive least squares updates. The Kalman Filtering formulation for Jacobian estimation was introduced first by Qian and Su (2002). A similar approach uses bilinear models to estimate forward models for mobile robots, also resulting in an on-line learning method (Censi and Murray, 2015).

Uncalibrated servoing is an adaptive control method which is closely related to reinforcement learning. Many approaches exist that learn forward models from data. Particularly relevant for manipulation are a number of methods that use neural networks to learn forward models from simulation or real world data. These models were applied to 2D-visual servoing (Lampe and Riedmiller, 2013), learning dynamic properties of robot arms performing insertion tasks (Levine et al., 2016), grasping (Levine et al., 2018), and vegetable cutting (Lenz et al., 2015). An advantage of neural networks is their applicability to raw sensor data, which eliminates the need for feature extraction. Our approach is much simpler than deep learning-based methods as it only learns one state-independent linear function which is modified while the robot acts. The simplicity allows frequent updates which lets our method operate in an on-line setting without needing data from prior executions.

9.4 Conclusion and open issues

In this chapter, we presented a method for multi-modal sensor-based feedback control which allows our robot to fulfill goals defined in sensor space without any a priori knowledge of its sensor capabilities. The method continuously estimates a Jacobian-based forward model of the ongoing manipulation and directly uses it for control. Preliminary simulation experiments validated that our method quickly converges to tasks specified as desired sensor signals and is able to merge feedback of different modalities.

The validation on a real robot system is still open. While the method is taking sensor uncertainty into account, its performance drops significantly when confronted with the non-Gaussian noise of real sensor data. The critical assumption of our method is the smoothness of the sensor features. Contact with hard robots is discontinuous which breaks our assumption and makes our method unsuitable for traditional hard manipulators. However, we believe that our method will be useful in combination with soft robotic end-effectors such as the RBO hand (Deimel and Brock, 2016). Soft materials turn hard contact into elastic contact, thus sensor feedback of soft contact is always smooth but hard to model or simulate. Learning forward models for soft manipulators is possible (Martín-Martín, 2018) but these models do not easily transfer to different tasks which supports the need for online learning of the forward model. Prior results for uncalibrated servoing of soft surgical robots (Yip and Camarillo, 2014, 2016) also lends support that our method might transfer to control for soft manipulation.

Chapter 10

Conclusion: A novel factorization of motion generation

To conclude this thesis, we will now summarize its main contributions. Part I of this thesis discussed the advantages of control-based approaches to mobile manipulation. In Chapter 2, we introduced basics of robot control in joint and task space as a foundation for robust manipulation. Control based on task-relevant features is a highly effective way to reduce uncertainty. Based on the introduced methods, we presented a mobile manipulation system for a real world bin picking task in Chapter 3. The main outcomes of this chapter were 1) an extended evaluation of our and other systems to support the feasibility of feedback-based solutions for complex real world tasks 2) three capabilities that made our system robust, which were: First, the ability to make contact, second, the ability to plan contingencies to make the system resilient against failures and third, the ability to efficiently adapt a plan to new sensor data. While the behavior of the system was mostly hand-coded in the Amazon Picking Challenge, we believe these three capabilities should be requirements for autonomous motion generation systems too.

In Part II we presented a planning framework for motion in contact. We used sampling-based belief-space planning to quantify the uncertainty reduction of contact-based funneling actions. Therefore, in Chapter 4, we discussed sampling-based planning for high-dimensional configuration and belief space. The major contribution of this section was a set of techniques to overcome the intractable complexity of planning under uncertainty. In Chapter 5 we presented a novel sampling-based belief space planner that interleaves search in free space with motion in contact. This planner allowed to efficiently solve POMDP benchmarks and manipulation problems without the need for a priori discretization. We extended this planner in Chapter 6 with a tactile sensor model. The assumption of uncertainty-free contact

allowed to introduce belief space partitioning which are the core of contact-based contingency planning.

Part III looked at motion generation from a systems perspective. We analyzed different architectures to integrate perception into planning in Chapter 7, and discussed optimization-based methods as an effective tool to adapt plans to sensor data. Based on this background, we introduced the ESPER planner in Chapter 8, a decomposition of the motion generation problem into path adaptation, local replanning, and expected shortest path extraction. This decomposition allowed to move mobile manipulators under task constraints in completely unknown environments with dynamic obstacles. In Chapter 9, we presented preliminary results for motion generation in sensor space, allowing robots to exploit multi-modal feedback by learning forward models while interacting.

10.1 Motion generation as multi-level reasoning

While task-general autonomous motion generation remains unsolved, many methods presented in this thesis are mature, robust solutions to components of the problem. While there is much scientific debate about the implementation of these components, the architectures of the whole motion generation system is discussed rarely. We believe, that many challenges for motion generation could be overcome by a more systematic view on the different aspects of motion generation and their interconnections. Popular motion generation frameworks, like MoveIt, mostly factorize motion generation into vision, planning, and control modules. This factorization forces the user into a sense-plan-act architecture from which it is hard to escape from. As of now, there exist no frameworks that easily let inexperienced users combine sensor-based control with collision-free motion. This section is an attempt to start this discussion and find a better factorization of motion generation systems. In this section, we incorporate all discussed contributions into one unifying motion generation architecture. As we stated initially, motion generation is usually factorized along the spectrum of planning and control. We propose an orthogonal factorization along different aspects of continuous reasoning that are all relevant for motion generation.

We propose to view motion generation as a hierarchy of interconnected reasoning loops as depicted in Fig. 10.1. Each loop continuously integrates sensor data and adapts a subplan that captures a specific aspect of the motion generation problem. There is a hierarchy between the different reasoning loops induced by their horizons of relevance. This horizon is the time window in which each level guarantees a response. The length of this horizon allows to arrange the reasoning loops into different levels. The lowest level provides the quickest reactions at high rate, however its actions will only be feasible for a short time

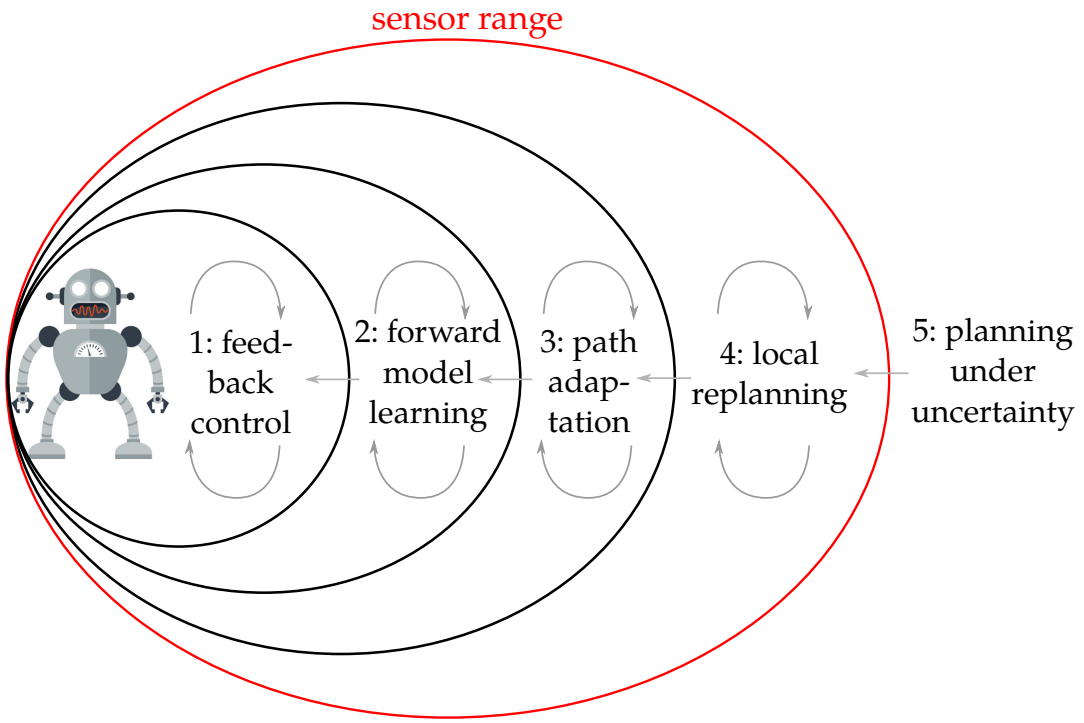


Figure 10.1 A motion generation architecture composed of five interconnected reasoning loops. Each loop continuously adapts a subplan for a certain aspect of the motion generation problem in response to sensor data. The loops are sorted by their horizons of relevance: Loops close to the robot make decisions with immediate impact, while the reasoning on the far end affects future, long-term rewards.

span. The higher levels do not have to react that quickly but find plans that are valid for longer time spans. All loops obtain their goals from the higher level and pass refined goals to the next lower level. Such an architecture of interconnected continuous reasoning has been a successful strategy for interactive perception (Martín-Martín, 2018), implemented as interconnected recursive estimation processes. It is not surprising that such architectures also apply to motion generation tasks, as estimation and control are dual problems.

In this thesis we provided implementations for five different hierarchical reasoning loops, solving fundamentally different parts of the motion generation problem. From highest to lowest level, these levels are: **(1)** Reactive, immediate behavior based on feedback control. This level takes care of all safety-critical aspects of motion are handled at the highest possible rate. The high update rate makes it inherently robust to uncertainty. **(2)** Online estimation of task-dependent forward models. This level allows the robot to predict the outcome of its actions for the close future. This level continuously relates sensor data to task success, allowing the specification of manipulation tasks in sensor space. **(3)** Continuous adaptation of action sequences. This level continuously adapts a sequence of actions to real world sensor

data of the changing world. **(4)** Discrete search over actions. This level maintains a structure that captures the connectivity of the world. It captures the essential discrete steps needed to solve the task, and alternatives, but the execution details are left to lower levels. **(5)** captures all reasoning outside the sensor range. The robot predict possible world states and finds suitable reactions, includes choosing informative or explorative actions. Reasoning on this level is most complex, but not subject to real time constraints.

The methods of this thesis all proved possible implementations for this framework. CERRT and CONCERRT methods planned hybrid control strategies by reasoning over uncertainty (1&5). The ESPER framework combined local control with path adaptation, local replanning, and planning under uncertainty (1,3,4,5). The uncalibrated servoing method described the connection between forward model learning and feedback control (1&2). The integration of all these different contributions in a common framework is left for future work.

These five levels are not a complete description of all relevant processes in a capable motion generation system. Currently, not included are, for example, any form of symbolic planning or dynamic constraints. We invite researchers in other related fields to include missing levels and their connection to other components. We believe that placing different methods into such a structure allows to modularize complex motion generation systems, find synergies between existing methods, and make architectural decisions explicit.

Bibliography

Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M. Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304, 2014.

Ron Alterovitz, Thierry Siméon, and Ken Goldberg. The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty. In *Robotics: Science and Systems*, pages 246–253, 2007.

Nancy M. Amato, O. Burchan Bayazit, Lucia K. Dale, Christopher Jones, and Daniel Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, 1998.

Ronald C. Arkin. *Behavior-based robotics*. MIT Press, Cambridge, MA, USA, 1998.

Sankalp Arora, Sanjiban Choudhury, and Sebastian Scherer. Hindsight is only 50/50: Unsuitability of MDP based approximate POMDP solvers for multi-resolution information gathering. *arXiv preprint arXiv:1804.02573*, 2018.

Johan Baeten, Herman Bruyninckx, and Joris De Schutter. Integrated vision/force robotic servoing in the task frame formalism. *The International Journal of Robotics Research*, 22 (10-11):941–954, 2003.

Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A Ngo. Monte carlo value iteration for continuous-state POMDPs. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, pages 175–191, 2010.

Amotz Bar-Noy and Baruch Schieber. The Canadian traveller problem. In *Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms*, pages 261–270, 1991.

Jérôme Barraquand, Lydia Kavraki, Jean-Claude Latombe, Rajeev Motwani, Tsai-Yen Li, and Prabhakar Raghavan. A random sampling scheme for path planning. *The International Journal of Robotics Research*, 16(6):759–774, 1997.

Dmitry Berenson, Siddhartha S. Srinivasa, Dave Ferguson, and James J. Kuffner. Manipulation planning on constraint manifolds. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 625–632, 2009.

Dimitri P. Bertsekas. *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995.

Nassime Blin, Michel Taïx, Philippe Fillatreau, and Jean-Yves Fourquet. I-RRT-C: Interactive motion planning with contact. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4244–4249, 2016.

Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics*, 33(6):1273–1291, 2017.

Valérie Boor, Mark H. Overmars, and A. Frank van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1018–1023, 1999.

Amy J. Briggs, Carrick Detweiler, Daniel Scharstein, and Alexander Vandenberg-Rodes. Expected shortest paths for landmark-based robot navigation. *The International Journal of Robotics Research*, 23(7-8):717–728, 2004.

Oliver Brock and Lydia E. Kavraki. Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional configuration spaces. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1469–1474, 2001.

Oliver Brock and Oussama Khatib. High-speed navigation using the global dynamic window approach. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 341–346, 1999.

Oliver Brock and Oussama Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, 2002.

Rodney A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation*, 2(1):14–23, 1986.

Rodney A. Brooks. Intelligence without representation. *Artificial intelligence*, 47(1-3): 139–159, 1991.

Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 723–730, 2011.

Brendan Burns and Oliver Brock. Toward optimal configuration space sampling. In *Robotics: Science and Systems*, pages 105–112, 2005.

Brendan Burns and Oliver Brock. Sampling-based motion planning with sensing uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3313–3318, 2007.

Robert R. Burridge, Alfred A. Rizzi, and Daniel E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999.

Eduardo F. Camacho and Carlos Bordons Alba. *Model predictive control*. Springer, London, UK, 2013.

Lewis Carroll. *Alice's Adventures in Wonderland*. Macmillan, 1865.

Andrea Censi and Richard M. Murray. Bootstrapping bilinear models of simple vehicles. *The International Journal of Robotics Research*, 34(8):1087–1113, 2015.

François Chaumette and Seth Hutchinson. Visual servo control. I. basic approaches. *Robotics & Automation Magazine, IEEE*, 13(4):82–90, 2006.

Sachin Chitta, Edward Gil Jones, Matei Ciocarlie, and Kaijen Hsiao. Mobile manipulation in unstructured environments: Perception, planning, and execution. *IEEE Robotics & Automation Magazine*, 19(2):58–71, 2012.

Howie M. Choset, Seth Hutchinson, Kevin M. Lynch, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun. *Principles of robot motion: theory, algorithms, and implementation*. MIT Press, Cambridge, MA, USA, 2005.

Jen Jen Chung, Andrew J. Smith, Ryan Skeele, and Geoffrey A. Hollinger. Risk-aware graph search with dynamic edge cost discovery. *The International Journal of Robotics Research*, 2018. preprint.

Nikolaus Correll, Kostas E Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M Romano, and Peter R Wurman. Analysis and observations from the first Amazon Picking Challenge. *IEEE Transactions on Automation Science and Engineering*, 15(1):172–188, 2018.

John J. Craig. *Introduction to robotics: mechanics and control*. Pearson/Prentice Hall, Upper Saddle River, NJ, USA, 2005.

Raphael Deimel and Oliver Brock. A novel type of compliant and underactuated robotic hand for dexterous grasping. *The International Journal of Robotics Research*, 35(1-3): 161–185, 2016.

Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using Gaussian processes and factor graphs. In *Robotics: Science and Systems*, 2016.

Clemens Eppner and Oliver Brock. Planning grasp strategies that exploit environmental constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4947–4952, 2015.

Clemens Eppner, Raphael Deimel, José Álvarez-Ruiz, Marianne Maertens, and Oliver Brock. Exploitation of environmental constraints in human and robotic grasping. *The International Journal of Robotics Research*, 34(7):1021–1038, 2015.

Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock. Four aspects of building robotic systems: lessons from the Amazon Picking Challenge 2015. *Autonomous Robots*, 32(7):1459–1475, October 2018.

Michael A. Erdmann and Matthew T. Mason. An exploration of sensorless manipulation. *IEEE Journal on Robotics and Automation*, 4(4):369–379, 1988.

Roy Featherstone. An empirical study of the joint space inertia matrix. *The International Journal of Robotics Research*, 23(9):859–871, 2004.

Roy Featherstone. *Rigid body dynamics algorithms*. Springer, Boston, MA, USA, 2014.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

Elmer G. Gilbert, Daniel W. Johnson, and S. Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

Kenneth Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10(2): 201–225, 1993.

Charlie Guan, William Vega-Brown, and Nicholas Roy. Efficient planning for near-optimal compliant manipulation leveraging environmental contact. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 215–222, 2018.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4 (2):100–107, 1968.

Kris Hauser. On responsiveness, safety, and completeness in real-time motion planning. *Autonomous Robots*, 32(1):35–48, 2012.

Kris Hauser and Jean-Claude Latombe. Multi-modal motion planning in non-expansive spaces. *The International Journal of Robotics Research*, 29(7):897–915, 2010.

Thomas A. Henzinger. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, pages 265–292. Springer, Berlin, Germany, 2000.

Carlos Hernandez, Mukunda Bharatheesha, Wilson Ko, Hans Gaiser, Jethro Tan, Kanter van Deurzen, Maarten de Vries, Bas Van Mil, Jeff van Egmond, Ruben Burger, Mihai Morariu, Jihong Ju, Xander Gerrmann, Ronald Ensing, Jan van Frankenhuyzen, and Martijn Wisse. Team Delft's robot winner of the Amazon Picking Challenge 2016. *CoRR*, abs/1610.05514, 2016. URL http://arxiv.org/abs/1610.05514.

Christopher Holleman and Lydia E. Kavraki. A framework for using the workspace medial axis in PRM planners. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1408–1413, 2000.

John Hollerbach, Wisama Khalil, and Maxime Gautier. Model identification. In B. Siciliano and O. Khatib, editors, *Springer Handbook of Robotics*, pages 321–344. Springer, 2008.

Robert Holmberg and Oussama Khatib. Development and control of a holonomic mobile robot for mobile manipulation tasks. *The International Journal of Robotics Research*, 19 (11):1066–1074, 2000.

Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, 2013.

Koh Hosoda and Minoru Asada. Versatile visual servoing without knowledge of true jacobian. In *IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS)*, pages 186–193, 1994.

Koh Hosoda, Katsuji Igarashi, and Minoru Asada. Adaptive hybrid control for visual and force servoing in an unknown environment. *IEEE Robotics & Automation Magazine*, 5(4): 39–43, 1998.

Kaijen Hsiao, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Grasping POMDPs. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4685–4692, 2007.

David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

David Hsu, Tingting Jiang, John Reif, and Zheng Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4420–4426, 2003.

Eric Huang, Mustafa Mukadam, Zhen Liu, and Byron Boots. Motion planning with graph-based trajectories and Gaussian process inference. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5591–5598, 2017.

Manfred Huber and Roderic A. Grupen. A feedback control structure for on-line learning tasks. *Robotics and autonomous systems*, 22(3-4):303–315, 1997.

Martin Jägersand, Olac Fuentes, and Randal Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2874–2880, 1997.

Léonard Jaillet and Thierry Siméon. Path deformation roadmaps: Compact graphs with useful cycles for motion planning. *The International Journal of Robotics Research*, 27 (11-12):1175–1188, 2008.

Shervin Javdani, Matthew Klingensmith, J. Andrew Bagnell, Nancy S. Pollard, and Siddhartha S. Srinivasa. Efficient touch based localization through submodularity. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1828–1835, 2013.

Xuerong Ji and Jing Xiao. Planning motions compliant to complex contact states. *The International Journal of Robotics Research*, 20(6):446–465, 2001.

Rico Jonschkowski, Clemens Eppner, Sebastian Höfer, Roberto Martín-Martín, and Oliver Brock. Probabilistic multi-class segmentation for the Amazon Picking Challenge. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–7, 2016.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32(9-10):1194–1227, 2013.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4569–4574, 2011.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jeannette Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3): 1864–1871, 2018.

Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

Lydia E. Kavraki, Petr Svestka, Jean-Claude Latombe, and Marc H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

Olivier Kermorgant and François Chaumette. Dealing with constraints in sensor-based robot control. *IEEE Transactions on Robotics*, 30(1):244–257, 2014.

Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 500–505, 1985.

Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation*, 3(1):43–53, 1987.

Oussama Khatib. Mobile manipulation: The robotic assistant. *Robotics and Autonomous Systems*, 26(2-3):175–183, 1999.

Ross A. Knepper and Matthew T. Mason. Real-time informed path sampling for motion planning search. *The International Journal of Robotics Research*, 31(11):1231–1250, 2012.

Michael C. Koval, David Hsu, Nancy S. Pollard, and Siddhartha S. Srinivasa. Configuration lattices for planar contact manipulation under uncertainty. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016a.

Michael C. Koval, Nancy S. Pollard, and Siddhartha S. Srinivasa. Pre- and post-contact policy decomposition for planar contact manipulation under uncertainty. *The International Journal of Robotics Research*, 35(1-3):244–264, 2016b.

Danica Kragic and Henrik I. Christensen. Survey on visual servoing for manipulation. *Computational Vision and Active Perception Laboratory, Fiskartorpsv*, 15, 2002.

James J. Kuffner and Steven M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 995–1001, 2000.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Robotics: Science and Systems*, 2008.

Hanna Kurniawati, Tirthankar Bandyopadhyay, and Nicholas M. Patrikalakis. Global motion planning under uncertain motion, sensing, and environment map. *Autonomous Robots*, 33 (3):255–272, 2012.

Thomas Lampe and Martin Riedmiller. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2013.

Jean-Claude Latombe. *Robot motion planning*. Springer, Boston, US, 2012.

Boris Lau, Christoph Sprunk, and Wolfram Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10):1116–1130, 2013.

Jean-Paul Laumond. Kineo CAM: a success story of motion planning algorithms. *IEEE Robotics & Automation Magazine*, 13(2):90–93, 2006.

Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Department of Computer Science, Iowa State University, 1998.

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, USA, 2004.

Peter Lehner, Arne Sieverling, and Oliver Brock. Incremental, sensor-based motion generation for mobile manipulators in unknown, dynamic environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4761–4767, 2015.

Ian Lenz, Ross Knepper, and Ashutosh Saxena. DeepMPC: learning deep latent features for model predictive control. In *Robotics: Science and Systems*, 2015.

Peter Leven and Seth Hutchinson. A framework for real-time path planning in changing environments. *The International Journal of Robotics Research*, 21(12):999–1030, 2002.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The Internatioal Journal of Robotics Research*, 37(4-5):421–436, 2018.

Qiang Li, Carsten Schürmann, Robert Haschke, and Helge Ritter. A control framework for tactile servoing. In *Robotics: Science and Systems*, 2013.

Zakary Littlefield, Dimitri Klimenko, Hanna Kurniawati, and Kostas E. Bekris. The importance of a suitable distance function in belief-space planning. In *Robotics Research*, pages 683–700. Springer, 2018.

Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *Machine Learning Proceedings*, pages 362–370. Elsevier, 1995.

Stefan Loibl, Daniel Meyer-Delius, and Patrick Pfaff. Probabilistic time-dependent models for mobile robot path planning in changing environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5545–5550, 2013.

Tomas Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, pages 108–120, 1983.

Tomás Lozano-Pérez, Matthew T. Mason, and Russell H. Taylor. Automatic synthesis of fine-motion strategies for robots. *The International Journal of Robotics Research*, 3(1): 3–24, 1984.

Kevin M. Lynch and Frank C. Park. *Modern Robotics*. Cambridge University Press, Cambridge, MA, USA, 2017.

Ezio Malis, Guillaume Morel, and François Chaumette. Robot control using disparate multiple sensors. *The International Journal of Robotics Research*, 20(5):364–377, 2001.

Bhaskara Marthi. Robust navigation execution by planning in belief space. In *Robotics: Science and Systems*, July 2012.

Roberto Martín-Martín. *Leveraging problem structure in interactive perception for robot manipulation of constrained mechanisms*. PhD thesis, Technische Universität Berlin, 2018.

Matthew T. Mason. The mechanics of manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 544–548, 1985.

Matthew T. Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1:1–28, 2018.

Nik A. Melchior and Reid Simmons. Particle RRT for path planning with uncertainty. In *IEEE International Conference onRobotics and Automation (ICRA)*, pages 1617 –1624, 2007.

Patricia E. Missiuro and Nicholas Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1261 –1267, 2006.

Mustafa Mukadam, Xinyan Yan, and Byron Boots. Gaussian process motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9–15, 2016.

Sean Murray, Will Floyd-Jones, Ying Qi, Daniel J. Sorin, and George Konidaris. Robot motion planning on a chip. In *Robotics: Science and Systems*, 2016.

Jun Nakanishi, Rick Cory, Michael Mistry, Jan Peters, and Stefan Schaal. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research*, 27(6):737–757, 2008.

Alireza Nakhaei and Florent Lamiraux. Motion planning for humanoid robots in environments modeled by vision. In *IEEE-RAS International Conference on Humanoid Robots*, pages 197–204, 2008.

Balaubramaniam Kausik Natarajan. The complexity of fine motion planning. *The International Journal of Robotics Research*, 7(2):36–42, 1988.

Petter Ogren and Naomi Ehrich Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2):188–195, 2005.

Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *The International Journal of Robotics Research*, 29(8):1053–1068, 2010.

Előd Páll, Arne Sieverling, and Oliver Brock. Contingent contact-based motion planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Chonhyon Park, Jia Pan, and Dinesh Manocha. ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments. In *International Conference on Automated Planning and Scheduling (ICAPS)*, 2012.

Calder Phillips-Grafflin and Dmitry Berenson. Planning and resilient execution of policies for manipulation in contact with actuation uncertainty. In *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2016.

Andreas Pichler and Martin Jägersand. Uncalibrated hybrid force-vision manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1866–1871, 2000.

Jenelle Armstrong Piepmeier, Gary V McMurray, and Harvey Lipkin. Uncalibrated dynamic visual servoing. *IEEE Transactions on Robotics and Automation*, 20(1):143–147, 2004.

Vinay Pilania and Kamal Gupta. Mobile manipulator planning under uncertainty in unknown environments. *The International Journal of Robotics Research*, 37(2-3):316–339, 2018.

Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1025 – 1032, 2003.

Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomás Lozano-Pérez. Belief space planning assuming maximum likelihood observations. In *Robotics: Science and Systems*, 2010.

Robert Platt Jr, Leslie Kaelbling, Tomás Lozano-Pérez, and Russ Tedrake. Simultaneous localization and grasping as a belief space control problem. In *International Symposium on Robotics Research*, volume 2, 2011.

George H. Polychronopoulos and John N. Tsitsiklis. Stochastic shortest path problems with recourse. *Networks: An International Journal*, 27(2):133–143, 1996.

Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1): 69–81, 2014.

Mario Prats, Pedro J. Sanz, and Angel P. Del Pobil. Vision-tactile-force integration and robot physical interaction. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3975–3980, 2009.

Sam Prentice and Nicholas Roy. The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In *Robotics Research*, pages 293–305. Springer, 2010.

Louise Pryor and Gregg Collins. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996.

Jiang Qian and Jianbo Su. Online estimation of image jacobian matrix by Kalman-Bucy filter for uncalibrated stereo vision feedback. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 562–567, 2002.

Sean Quinlan and Oussama Khatib. Elastic bands: Connecting path planning and control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 802–807, 1993.

John H. Reif. Complexity of the mover's problem and generalizations. In *20th Annual Symposium on Foundations of Computer Science*, pages 421–427. IEEE, 1979.

Markus Rickert and Andre Gaschler. Robotics library: An object-oriented approach to robot applications. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 733–740, 2017.

Markus Rickert, Arne Sieverling, and Oliver Brock. Balancing exploration and exploitation in sampling-based motion planning. *IEEE Transactions on Robotics*, 30(6):1305–1317, 2014.

Ludovic Righetti, Mrinal Kalakrishnan, Peter Pastor, Jonathan Binney, Jonathan Kelly, Randolph C. Voorhies, Gaurav S. Sukhatme, and Stefan Schaal. An autonomous manipulation system based on force control and optimization. *Autonomous Robots*, 36(1-2):11–30, 2014.

Elon Rimon and Daniel E. Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Pearson, Upper Saddle River, NJ, USA, 2016.

Shankar Sastry and Marc Bodson. *Adaptive control: stability, convergence and robustness*. Prentice-Hall, Upper Saddle River, NJ, USA, 2011.

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 33(9):1251–1270, 2014.

Max Schwarz, Anton Milan, Christian Lenz, Aura Munoz, Arul Selvam Periyasamy, Michael Schreiber, Sebastian Schüller, and Sven Behnke. NimbRo picking: Versatile part handling for warehouse automation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3032–3039, 2017.

Konstantin M. Seiler, Hanna Kurniawati, and Surya P.N. Singh. An online and approximate solver for POMDPs with continuous action space. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2290–2297, 2015.

Luis Sentis. *Synthesis and control of whole-body behaviors in humanoid systems*. PhD thesis, Stanford University, Stanford, CA, USA, 2007.

Luis Sentis and Oussama Khatib. Synthesis of whole-body behaviors through hierarchical control of behavioral primitives. *International Journal of Humanoid Robotics*, 2(4): 505–518, 2005.

Alexander Shkolnik and Russ Tedrake. Path planning in 1000+ dimensions using a task-space voronoi bias. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2061–2067, 2009.

Bruno Siciliano and Luigi Villani. *Robot force control*. Springer Science & Business Media, New York, NJ, USA, 2012.

Arne Sieverling, Nicolas Kuhnen, and Oliver Brock. Sensor-based, task-constrained motion generation under uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4348–4355, 2014.

Arne Sieverling, Clemens Eppner, Felix Wolff, and Oliver Brock. Interleaving motion in contact and in free space for planning under uncertainty. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4011–4017, 2017.

Thierry Siméon, Jean-Paul Laumond, and Carole Nissoux. Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, 14(6):477–493, 2000.

Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *The International Journal of Robotics Research*, 23(7-8): 729–746, 2004.

Reid Simmons and Sven Koenig. Probabilistic navigation in partially observable environments. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1080–1087, 1995.

Mandyam Srinivasan, Shaowu Zhang, Miriam Lehrer, and Thomas S. Collett. Honeybee navigation en route to the goal: visual flight control and odometry. *Journal of Experimental Biology*, 199(1):237–244, 1996.

Cyrill Stachniss and Wolfram Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 508–513, 2002.

Mike Stilman. Task constrained motion planning in robot joint space. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3074–3081, 2007.

Richard S. Sutton, Andrew G. Barto, and Ronald J. Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*, 12(2):19–22, 1992.

Osamu Takahashi and Robert J Schilling. Motion planning in a plane using generalized voronoi diagrams. *IEEE Transactions on robotics and automation*, 5(2):143–150, 1989.

Russ Tedrake, Ian R. Manchester, Mark Tobenkin, and John W. Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

Sebastian Thrun. Monte carlo POMDPs. In *NIPS*, volume 12, pages 1064–1070, 1999.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, Cambridge, MA, USA, 2005.

Michael Tomasello and Josep Call. *Primate cognition*. Oxford University Press, USA, 1997.

Marc Toussaint. Robot trajectory optimization using approximate inference. In *26th Annual International Conference on Machine Learning*, pages 1049–1056, 2009.

Marc Toussaint. *A Tutorial on Newton Methods for Constrained Trajectory Optimization and Relations to SLAM, Gaussian Process Smoothing, Optimal Control, and Probabilistic Inference*, pages 361–392. Springer International Publishing, Cham, 2017.

Marc Toussaint, Nathan Ratliff, Jeannette Bohg, Ludovic Righetti, Peter Englert, and Stefan Schaal. Dual execution of optimized contact interaction trajectories. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 47–54, 2014.

Jur Van Den Berg, Pieter Abbeel, and Ken Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *The International Journal of Robotics Research*, 30(7):895–913, 2011.

Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *The International Journal of Robotics Research*, 31(11):1263–1278, 2012.

John Vannoy and Jing Xiao. Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes. *IEEE Transactions on Robotics*, 24(5):1199–1212, 2008.

Ngo Anh Vien and Marc Toussaint. Touch based POMDP manipulation via sequential submodular optimization. In *International Conference on Humanoid Robots (Humanoids)*, pages 407–413. IEEE, 2015.

Oskar Von Stryk. Numerical solution of optimal control problems by direct collocation. In *Optimal Control*, pages 129–143. Springer, Berlin, Germany, 1993.

Vincent Wall, Gabriel Zöller, and Oliver Brock. A method for sensorizing soft actuators and its application to the RBO hand 2. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4965–4970, May 2017.

William H. Warren Jr, Bruce A. Kay, Wendy D. Zosh, Andrew P. Duchon, and Stephanie Sahuc. Optic flow is used to control human walking. *Nature neuroscience*, 4(2):213, 2001.

Florian Wirnshofer, Philipp Sebastian Schmitt, Wendelin Feiten, Georg von Wichert, and Wolfram Burgard. Robust, compliant assembly via optimal belief space planning. In *IEEE International Conference on Robotics and Automation (ICRA),*, 2018.

Libo Yang and Steven M. LaValle. The sampling-based neighborhood graph: an approach to computing and executing feedback motion strategies. *IEEE Transactions on Robotics and Automation*, 20(3):419–432, 2004.

Yuandong Yang and Oliver Brock. Adapting the sampling distribution in PRM planners based on an approximated medial axis. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4405–4410, 2004.

Yuandong Yang and Oliver Brock. Elastic roadmaps—motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130, 2010.

Michael C. Yip and David B. Camarillo. Model-less feedback control of continuum manipulators in constrained environments. *IEEE Transactions on Robotics*, 30(4):880–889, 2014.

Michael C. Yip and David B. Camarillo. Model-less hybrid position/force control: A minimalist approach for continuum manipulators in unknown, constrained environments. *IEEE Robotics and Automation Letters*, 1(2):844–851, 2016.

Sung Wook Yoon, Alan Fern, and Robert Givan. FF-Replan: A baseline for probabilistic planning. In *ICAPS*, volume 7, pages 352–359, 2007.

Sung Wook Yoon, Alan Fern, Robert Givan, and Subbarao Kambhampati. Probabilistic planning via determinization in hindsight. In *AAAI*, pages 1010–1016, 2008.

Eiichi Yoshida and Fumio Kanehiro. Reactive robot motion using path replanning and deformation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5456–5462, 2011.

Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2):3–9, 1985.

Kuan-Ting Yu, Nima Fazeli, Nikhil Chavan-Dafle, Orion Taylor, Elliott Donlon, Guillermo Diaz Lankenau, and Alberto Rodriguez. A summary of team MIT's approach to the Amazon Picking Challenge 2015. *ArXiv e-prints*, April 2016.

Jiaji Zhou, Robert Paolini, Aaron M. Johnson, J. Andrew Bagnell, and Matthew T. Mason. A probabilistic planning framework for planar grasping under uncertainty. *IEEE Robotics and Automation Letters*, 2(4):2111–2118, 2017.

Matt Zucker, Nathan Ratliff, Anca D. Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M. Dellin, J. Andrew Bagnell, and Siddhartha S. Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.

Matt Zucker, Sungmoon Joo, Michael X. Grey, Christopher Rasmussen, Eric Huang, Michael Stilman, and Aaron Bobick. A general-purpose system for teleoperation of the DRC-HUBO humanoid robot. *Journal of Field Robotics*, 32(3):336–351, 2015.

Index