Motion Generation With Contact-Based Environmental Constraints

vorgelegt von
M. Sc.
Előd Páll
ORCID: 0000-0002-8975-1171

an der Fakultät IV – Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
–Dr.-Ing.–

genehmigte Dissertation

Promotionsausschuss:

Vorsitzender:	Prof. Dr.-Ing. Guillermo Gallego
Gutachter:	Prof. Dr. Oliver Brock
Gutachter:	Prof. Dr. Marc Toussaint
Gutachter:	Prof. Dr.-Ing. Tamim Asfour

Tag der wissenschaftlichen Aussprache: 09. Juni 2023

Berlin 2023

Prepublication and Statement of Contribution

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

(A) **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 ©2018 IEEE.

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

(B) **Előd Páll** and Oliver Brock. Analysis of Open-Loop Grasping From Piles. In *IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China, 2021 ©2021 IEEE.

I (EP) was the sole first author and main contributor to implementation, paper writing, and experimental evaluation. OB gave scientific advice and contributed to paper writing.

Chapters 1, 2, 3, 5, and 7 are original to this thesis. Chapter 4 contains excerpts from (A), but the thesis extends the main evaluations with an additional baseline and a novel planner that combines proposed planning approaches from Chapter 3 and 4. Chapter 6 contains excerpts from (B), but the thesis extends the explorative empirical study with new hypothesis-driven experiments and proposes a simplified grasp planer based on the empirical feedings.

Acknowledgements

First, I thank Prof. Dr. Oliver Brock, my adviser and mentor. His vast knowledge and expertise in robotics allow him to see the big picture. Because of that, his feedback was always constructive, aimed for progress, and provided new perspectives. I greatly appreciate your commitment and invested time to get the best out of our research talks and publications. Besides scientific guidance, he mentored me to become an independent researcher by reminding me to ask relevant questions, build on my strength, and overcome my weaknesses.

I was part of an amazing research team at RBO, where we supported each other and created a friendly atmosphere. Thanks to the older generations for handing over a reliable infrastructure to Arne, Clemens, Mahmoud, Raphael, Rico, Roberto, and Sebastian. Thanks to Adrian, Apoor, Aravind, Can, Divyam, Furkan, Jessica, Pia, Stefan, Steffen, Oussama, Vito, and Xing. Thank you, Vincent, for being my paper and kicker buddy, Manuel, for the stimulating discussions about life and fiction, and Kolja, for your help during the difficult times. Thanks to Alexander, Janika, Lizzy, Oscar, Thomas, and Xenia for their immense support. Moreover, thanks to all the other RBO members, visitors, and friends not mentioned.

Thanks to Marc Toussaint and Tamim Asfour for being on my thesis committee and providing valuable feedback.

I'm grateful that I participated in fantastic research projects where I could collaborate with brilliant people. Thanks to the members of the SOMA and SCIoI projects and the founding of the European Commission and the German Research Foundation.

Thanks to my family, Tünde and Szilárd, for your support during this long time. Thanks to Zsófia, Lucian, and Levente, my friends and former colleagues, for helping to pursue research and Ph.D.

My final, most enormous thanks go to Enikő. Thank you for listening, encouraging, and motivating. We start and finish journeys together, and there is no better partner to walk this road together. *"There is no one as you are here for me."*[1]

[1] Kispál és a Borz, *Jutka*, 1994.

Abstract

Motion planning is a computational problem of finding robot motions between start and goal states so that a robot avoids collision with obstacles. This is a non-trivial problem because of the high dimensionality of the search space. The problem becomes even more challenging under realistic assumptions: Real-life robots' sensing and motion capabilities are inherently inaccurate. The accumulation of inaccuracies can lead to failure during the execution of a motion plan. Therefore, a planner must reason about uncertainty to avoid execution failure. It can search for actions unaffected by uncertainty, which are frequently a needle in a haystack, or it can augment uncertain states with sensor measurements, which requires reasoning about sensing.

The thesis contributes conceptual and algorithmic approaches to efficiently reduce state uncertainties in motion planning problems resulting from perception and motion inaccuracies. We reduce uncertainty with "collision-exploiting" motions that conflict with traditional "collision-free" planning. This conflict raises several challenges, such as collision-exploiting motion requires frequent collision checking, which is an expensive computational operation, and reasoning about contact states or contact sensing further increases an already high-dimensional problem space. We tackle these and other related challenges by representing the state space using the Environmental Constraint (EC) concept.

The Environmental Constraint concept was introduced in the context of human grasping experiments. Deimel et al. (2016) have proven that humans deliberately increase contact with their environment when their vision is experimentally blurred to achieve robust grasping. Roboticists have shown that contact with the environment reduces uncertainty and simplifies grasping (Odhner et al., 2013; Eppner and Brock, 2015; Hang et al., 2019). This thesis builds on these findings and further advances the EC-based manipulation and motion planning field.

In the first part, we introduce sampling-based motion planning and apply EC exploitation in high-dimensional planning problems by devising multiple contact-based configuration-space motion planning algorithms. One planner leverages an EC-based

decomposition of the workspace to avoid exploring task-irrelevant regions. Another planner detects deviations using contact events to reduce large amounts of state uncertainty. We evaluated all planners in simulation and one planner with real-robot experiments. The results show that our algorithms generalize to high-dimensional problems and handle increased state uncertainty.

In the second part, we characterize a new environmental constraint manifesting in piles of objects and show that the novel EC provides similar beneficial effects as static parts of the environment, simplifying grasping from piles of objects. We empirically study human-like grasping from piles of objects with a robotic hand. The interaction between the hand, objects, and the environment produces patterns in interaction forces. These force patterns constrain objects' motion and reduce uncertainty mechanically. We show that these force patterns are consistent in regions of the workspace, and changes in the force patterns produce detectable contact events. Our real-robot experiments show that grasping and planning become substantially simpler when using these effects.

The third and final part discusses the practical application of EC exploitation in a robotic system. We present the Soft Manipulation System devised for integrating and evaluating concepts and technologies developed by a consortium of academics and industrial partners. Our tightly integrated but modular system design enabled EC exploitation with various hardware components. We conclude the thesis by proposing an EC-based problem factorization for manipulation and motion planning problems by blurring the boundaries between control, perception, and planning; and shifting manipulation responsibilities from these components to the environment.

Zusammenfassung

Die Bewegungsplanung ist ein Berechnungsproblem, bei dem es darum geht, Roboterbewegungen zwischen Start- und Zielzustand zu finden, sodass ein Roboter Kollisionen mit Hindernissen vermeidet. Dies ist ein nicht-triviales Problem aufgrund der hohen Dimensionalität des Suchraums. Unter realistischen Annahmen wird das Problem noch schwieriger: Die Wahrnehmungs- und Bewegungsfähigkeiten von Robotern im realen Leben sind von Natur aus ungenau. Diese Anhäufung von Ungenauigkeiten kann zu Fehlern bei der Ausführung eines Bewegungsplans führen. Daher muss ein Planer Unsicherheit in seinen Planungsprozess miteinbeziehen, um Ausführungsfehler zu vermeiden. Er kann nach Aktionen suchen, die nicht von der Ungewissheit betroffen sind, was häufig eine Nadel im Heuhaufen ist, oder er kann unsichere Zustände mit Sensormessungen ergänzen, was jedoch ebenfalls erfordert, dass Messvorgänge im Planungsprozess miteinbezogen werden.

In dieser Arbeit werden konzeptionelle und algorithmische Ansätze zur effizienten Reduzierung von Zustandsunsicherheiten in Bewegungsplanungsproblemen vorgestellt, die aus Wahrnehmungs- und Bewegungsungenauigkeiten resultieren. Wir reduzieren die Unsicherheit mit "kollisionsausnutzenden" Bewegungen, die mit der traditionellen "kollisionsfreien" Planung in Konflikt stehen. Dieser Konflikt wirft mehrere Herausforderungen auf, wie z.B. die Tatsache, dass kollisionsausnutzende Bewegungen häufige Kollisionsprüfungen erfordern, was eine aufwändige Rechenoperation ist, und dass die Schlussfolgerungen über Kontak- tzustände oder Kontaktwahrnehmung einen bereits hochdimensionalen Problemraum weiter vergrößern. Wir gehen diese und andere damit verbundene Herausforderungen an, indem wir den Zustandsraum mit Hilfe des Environmental Constraint (EC) Konzepts darstellen.

Das Environmental Constraint-Konzept wurde im Zusammenhang mit Experimenten zum menschlichen Greifen eingeführt. Deimel et al. (2016) haben bewiesen, dass Menschen bewusst mehr Kontakt mit ihrer Umgebung erzeugen, wenn ihre Sicht experimentell unscharf gemacht wird, um ein robustes Greifen zu erreichen. Robotiker haben gezeigt, dass der Kontakt mit der Umgebung die Unsicherheit reduziert und das

Greifen vereinfacht (Odhner et al., 2013; Eppner and Brock, 2015; Hang et al., 2019). Die vorliegende Arbeit baut auf diesen Erkenntnissen auf und bringt das Gebiet der EC-basierten Manipulation und Bewegungsplanung weiter voran.

Im ersten Teil stellen wir stochastische Bewegungsplanung vor und wenden die EC-Nutzung, indem wir zwei kontaktbasierte Konfigurationsraum-Bewegungsplanungsalgorithmen entwickeln. Ein Planer nutzt eine EC-basierte Dekomposition des Arbeitsraums, um die Erkundung aufgabenirrelevanter Regionen zu vermeiden. Ein anderer Planer erkennt Abweichungen anhand von Kontaktereignissen, um große Mengen an Zustandsunsicherheit zu reduzieren. Wir haben beide Planer in Simulationen und einen Planer auch in Experimenten mit realen Robotern evaluiert. Die Ergebnisse zeigen, dass unsere Algorithmen auf hochdimensionale Probleme verallgemeinert werden können und mit erhöhter Zustandsunsicherheit umgehen können.

Im zweiten Teil identifizieren wir die vorteilhaften Auswirkungen der Ausnutzung von Environmental Constraints auf das Greifen und zeigen, dass diese Effekte das Greifen von Objekten aus Objekthaufen vereinfachen. Wir untersuchen empirisch das menschenähnliche Greifen aus Objekthaufen mit einer Roboterhand. Die Interaktion zwischen der Hand, den Objekten und der Umgebung erzeugt Muster in den Interaktionskräften. Diese Kraftmuster schränken die Bewegung von Objekten ein und reduzieren Unsicherheit auf mechanische Art. Wir zeigen, dass diese Kraftmuster in bestimmten Bereichen des Arbeitsraums konsistent sind und dass änderungen in den Kraftmustern zu erkennbaren Kontaktereignissen führen. Unsere Experimente mit realen Robotern zeigen, dass das Greifen wesentlich einfacher wird, wenn diese Effekte genutzt werden.

Der dritte und letzte Teil befasst sich mit der praktischen Anwendung der EC-Ausnutzung in einem Robotersystem. Wir stellen den Soft Manipulation Planner vor, der für die Integration und Bewertung von Konzepten und Technologien entwickelt wurde, die von einem Konsortium aus Wissenschaftlern und Industriepartnern entwickelt wurden. Unser stark integriertes, aber modulares Systemdesign ermöglicht die Nutzung von EC mit verschiedenen Hardwarekomponenten. Zum Abschluss der Arbeit schlagen wir eine neu EC-basierte Problemfaktorisierung für Manipulations- und Bewegungsplanungsprobleme vor, indem sie die Grenzen zwischen Steuerung, Wahrnehmung und Planung verwischt und die Verantwortung für die Manipulation von diesen Komponenten auf die Umgebung verlagert.

Contents

List of Figures

List of Tables

1
Introduction

Robotics is an interdisciplinary field that combines computer science and engineering to create machines that interact with their environment. A robotic system is composed of hardware and software components. The hardware components include actuators that move parts of a robot's body or the whole robot and sensors that detect or measure physical properties of the robot or the environment. Software components generally include perception, control, and planning. The perception component collects and interprets sensory information. The planner makes decisions based on prior and collected information, and a control component computes and sends input commands to the actuators. The synergistic combination of these components enables a robot to safely and purposefully interact with its environment, other robots, or humans.

Robots have a great potential to help humankind by assisting elderly or disabled people and replacing manual labor in dangerous, hazardous, or strenuous jobs. However, most robots are used in industry to solve simple, pre-defined, and repetitive tasks, such as polishing automotive parts or patrolling and inspecting equipment. In our daily life, robots also have limited use for now, for example, vacuum cleaning or lawn mowing.

Robots will reach their full potential when they can autonomously make decisions to change their environment and interact with humans safely and purposefully. Interestingly, high-level reasoning, such as task scheduling, requires less computation than lower-level reasoning, such as motion generation.

The difference between computational complexity for high- and low-level reasoning was formulated in the 1980s as Moravec's paradox. Moravec's paradox states that higher-level reasoning requires little computation, but motion and perception skills require enormous amounts of computation. The paradox became more apparent in 1997 when Chess Champion Garry Kasparov lost to the Deep Blue computer program (Hsu et al., 1995). Then in 2017, Go Champion Ke Jie lost to AlphaGo (Silver et al., 2016). In 2019, world champion Team OG was defeated by OpenAI Five (OpenAI

et al., 2019) in the Dota 2 computer game. In the latter case, OpenAI Five's success was impressive because the multiplayer computer game requires reasoning about collaborative gameplay strategies. Logically, OpenAI Five computer program played Dota 2 without a mechanical embodiment. However, with Chess and Go, it is surprising that both human champions were not defeated by a robot but by the hands of another human who moved the chess pieces or go-stones based on the instructions given by a computer program.

The thesis focuses on a complex lower-level reasoning problem, namely motion planning. Motion planning is a computational problem of finding a continuous transition of configurations from A to B, where a configuration describes the state of a body. For example, the configuration of a key turning in a keyhole can be represented by a single number, the angle relative to when it was inserted. However, if the key is not yet inserted, we need three values to represent its position and another three values to represent its orientation. As another example, the configuration of a robot arm can be represented by its joint angles. Most robot arms have at least six or more joints. The increased number of values that uniquely define the state of a three-dimensional body or robot makes the configuration space *high-dimensional*. Reasoning in high-dimensional spaces makes motion planning a computationally complex problem.

Another challenge arises when decisions are made under *uncertainty*. Uncertainty results from various sources of inaccuracies. Real robots are inherently inaccurate in their sensing, motion, and knowledge about their environment and physics. Such inaccuracies accumulate over time, making a robot uncertain about its state, motion outcomes, and observations about its own or the world's state. When inaccuracies accumulate beyond a given task's tolerance, it leads to execution failure. We can avoid such execution failures if we eliminate the source of inaccuracies, such as by developing high-accuracy sensors or high-precision actuators. However, not all sources of inaccuracies can be eliminated. For example, it is impractical to assume that a robot could have an exact model of apples since there are no identical apples on an apple tree. Therefore, unresolved uncertainty must be dealt with during planning, which further increases the computational complexity of motion planning.

Motion planning is a computationally complex problem because the state space is high-dimensional and knowledge about past, current, and future states is uncertain.

To solve motion planning problems, we need to reduce their computational complexity using realistic assumptions about a given task and a robot's capabilities. On the one hand, we can use assumptions to tackle the high dimensionality by omitting

less relevant dimensions, discretizing continuous valued dimensions, or using different approximations. On the other hand, we can reduce complexity by reducing the scope of a motion planning problem. By reducing the scope of a problem, we can use more task-specific assumptions to simplify planning. There exist numerous sub-problems in motion planning. For example, when computing a collision-free path between two configurations, motion planning considers velocity and acceleration constraints. In contrast, path planning is a purely geometric problem of finding a collision-free path, disregarding dynamics, duration of motions, or limitations on motions or control inputs to a robot. As another example, manipulation planning focuses on changing the state of the environment, and grasping is a manipulation sub-problem dealing with stably seizing and holding an object against external disturbances.

To compute robust motion trajectories under uncertainty, we can tackle uncertainty by eliminating some sources of inaccuracy and by reasoning about the remaining uncertainty during planning. From a planning perspective, we can differentiate between three approaches for handling uncertainty depending on how much uncertainty a planner has to reason about.

First, if uncertainty is below a given task's tolerance, the planner can *ignore uncertainty*, i.e., it is not required to be modeled. If uncertainty is more significant than the task-tolerance, it needs to be modeled to predict its effect on motion outcomes.

Secondly, a planner searches for *robust actions* unaffected by uncertainty, i.e., avoiding execution failure and leading to an expected outcome. However, for a large amount of uncertainty, robust actions may not exist, or it may be challenging to find them, like searching for a needle in a haystack.

Thirdly, we can *integrate sensing* into planning to handle a large amount of uncertainty. With sensing, a planner can anticipate deviations that may occur during execution and compute appropriate reactions. However, each deviation creates a new planning problem of computing appropriate reactions, and inaccurate sensing requires additional reasoning, which increases the computational complexity of a given problem.

1.1 Goal of the Thesis

The thesis explores all three approaches for motion planning under uncertainty, and it overcomes challenges related to uncertainty and high dimensionality by integrating contact-based information into planning. Contact-based information is gained from deliberately moving into contact with environmental constraints. Thus, the thesis investigates two research questions:

1. How to use contact-based information to reduce uncertainty?
2. How to use contact-based information to simplify a given planning problem?

The thesis tackles both questions simultaneously by integrating contact-exploiting motion, contact sensing, and contact-based priors as assumptions into motion planning to solve real-world motion trajectory generation, object localization, and grasping problems. Since our planners solve real-world problems, the proposed planners can only rely on realistic assumptions about perception, control, model of the environment, and model of the interaction physics.

The thesis builds on the premise that a well-structured environment, designed and built by humans, holds task-relevant information. We[1] argue that this information can be obtained and leveraged by deliberately establishing contact with the environment, i.e., using contact-exploiting motions.

Prior research has shown that contact-exploiting motion and contact sensing reduce robot state uncertainty. As an example, Will and Grossman (1975) introduced guarded moves, a linear motion ending in an easily detectable contact event with the environment. Guarded moves reduced an object's position uncertainty because a state was constrained to the contacted surface, and the associated contact event indicated when the motion terminated. In another example, Lozano-Pérez et al. (1984) combined guarded moves and sliding motion in contact to generate complex motion behaviors. The resulting motion behaviors reduced state uncertainty and succeeded despite significant motion inaccuracies. Furthermore, Hsiao et al. (2007) combined contact-exploiting motion with contact sensing to discriminate between possible motion outcomes. The computed motion plan had branches based on the possible contact events a robot could sense during execution. In this example, contact sensing enabled the localization of a target object with an initially unknown location.

[1]For the sake of consistency, I will use the first-person plural narrative "we" throughout the thesis because a large part of the work presented in Part I and III is collaborative.

Motion planning with collision-exploiting motions conflicts with traditional collision-free planning. This conflict raises several challenges, such as collision-exploiting motion requires frequent collision checking during planning, which is an expensive computational operation, and reasoning about contact states or contact sensing further increases an already high-dimensional problem space.

We think that a recently formulated manipulation concept, *environmental constraint exploitation* (ECE) (Deimel et al., 2013) can be applied in motion planning under uncertainty to overcome the mentioned challenges. This concept uses contact with the environment to generate robust manipulation behaviors that simplify aspects of planning, perception, and control. The thesis aims to apply the concept in motion planning under uncertainty.

1.2 Environmental Constraint Exploitation

Moravec's paradox may seem logically unacceptable to the reader. How can it be that devising a winning game strategy for a chess game is an easier computational problem than physically moving the pieces on the board? Intuition is deceptive in this case. While Chess can be traced back almost 1500 years, humanity had a few million years to perfect their perception and motor skills. Hence, it is reasonable to think that humans have found a way to simplify manipulation that made them so handy with tools. This is why analyzing human manipulation skills can provide insights to simplify robotic manipulation.

The concept of environmental constraint exploitation emerged from a human grasping study (Deimel et al., 2013). The authors explained how humans simplify grasping using deliberate contact with the environment, argued that robust grasping requires deliberate contact, and showed that their insights are transferable to robotic grasping. Even though the concept is relatively new, we must note that contact-based motion and manipulation research has already discovered multiple benefits of contact-exploiting motion and sensing central for environmental constraint exploitation.

Environmental constraint exploitation is a collective term referring to different ways in which the environment can be used. Deliberate interaction between a robot and the environment can generate robust motion behaviors against uncertainty and simplify a manipulation task, such as grasping. An environmental constraint is a recurring structure or property of the environment that restricts the motion of a body, such as a hand's or an object's motion. The interaction between the body and the environment results in motion regularities that can be used to simplify computational aspects of

Figure 1.1 Humans touch the environment, even for picking up a pencil. Deliberately contacting the table simplifies grasping because it signals when the fingers reach the table and constrains the fingers' motion at an appropriate height relative to the pencil.

perception, control, and planning to manipulate the body. Since the relevant property of an environmental constraint is recurring, the associated environmental constraint exploitation is applicable in a region of the environment.

As a thought experiment, consider picking up a pencil on a tabletop at a reachable distance by looking at it once before grasping it but not while reaching to grasp it. First, we approach the pencil somewhat aligned with our fingers and with a significantly wider distance between fingers and thumb than the width of the pencil. When we think our palm is above the pencil, we lower our hand until we feel the table surface with our fingertips. Then, we close our fingers while sliding on the table to grasp the pencil. In this example, the horizontal surface constrained the pencil's position onto a plane. Using our sense of touch, we used this constraint to signal when the fingers reached the appropriate height and followed the constraint with our fingers to maintain the appropriate height. For a robot, this grasping strategy would allow omitting to compute the vertical position of the pencil from a camera image, omit to control the vertical position of the fingers while closing them, or omit to plan precisely the points where the fingers should touch the pencil. This way, the hand's interaction with the environment replaces computational aspects of perception, control, and planning.

Environmental constraint exploitation simplifies perception, control, and planning by replacing some computational aspects of sub-problems solved by these components. Below, we explain four ways to leverage environmental constraints central to the thesis:

- reducing uncertainty with *manipulation funnels*
- reducing uncertainty with *contact sensing*
- composing complex motion behaviors with *contact forces*
- reducing search space with *task-relevant manipulation funnels*

Uncertainty Reduction With Manipulation Funnels

Contact with the environment implicitly reduces state uncertainty. Uncertainty is implicitly reduced because when a body (robot or object) is in contact, its possible configurations are restricted by the environment. Consequently, uncertainty about its configuration is also constrained. For example, a body's three-dimensional position is constrained to a two-dimensional plane when contacting a surface, to a one-dimensional line when contacting an edge, and to a single point when contacting a corner.

In the context of robotic manipulation, Mason (1985) referred to such uncertainty reduction as *manipulation funnels*. The thesis uses the funnel analogy in later parts because it captures the core effect: Motion in contact reduces a large set of possible states (the entrance of a physical funnel) to a smaller set of possible states (the exit of a funnel) mechanically (the walls of a funnel).

With human grasping, Deimel et al. (2013) showed that humans use contact with the environment for robust grasping. They blurred the vision of humans with frosted goggles and observed that the participants increased their interaction with the environment. Kazemi et al. (2014) confirmed that humans extensively use their environment for robust grasping with another grasping study. Later, Deimel et al. (2016) extended their study showing robust robotic grasping under uncertainty concerning a constraint's placement and an object's size, shape, and position with grasp strategies exploiting flat horizontal and vertical surfaces, concave and convex edges along these surfaces.

One use of environmental constraint exploitation is reducing uncertainty mechanically with a manipulation funnel that simplifies perception, control, and planning. With planning, it can be used implicitly or explicitly by ignoring uncertainty (Eppner and Brock, 2015) or modeling uncertainty to search for actions that reduce it (Sieverling et al., 2017b), respectively. With control, manipulation funnels can be used implicitly with structurally compliant hardware or explicitly if compliant motions are provided via control. Moreover, some controllers are not required to accurately (or at all) regulate variables that are subject to implicit uncertainty reduction. Perception is implicitly simplified because it needs less accurate information about the environment or state variables when a manipulation funnel reduces uncertainty about these properties or variables.

Uncertainty Reduction With Contact Sensing

Another use of environmental constraint exploitation is reducing uncertainty with contact events. Contact between a body and the environment produces interaction

forces, such as the contact normal vector. Interaction forces change discretely when a new contact is made or an existing one is lost. Such *contact events* are easily detected with appropriately-placed tactile or force sensors on a robot. Since an environmental constraint has recurring properties (e.g., geometry), contact events are helpful for sequence manipulation funnels (Erdmann, 1986; Eppner and Brock, 2015; Sieverling et al., 2017a). Moreover, contact events can discriminate between possible configurations of a body in contact reducing state uncertainty (Hsiao et al., 2007).

With the funnel analogy, interaction forces indicate when a funnel's wall is reached or left, the entrance or exit of the funnel, respectively. Moreover, one can distinguish between funnels when the respective interaction forces are perceivably different.

Reasoning about contact events increases the computational complexity of decision-making because a planner must consider possible sensor measurements and plan appropriate reactions. However, it can also simplify planning when searching for robust actions. After a contact event reduces state uncertainty, more actions become robust; consequently, a planner needs less searching to find a robust action. Similarly to planning, control and perception can be simplified. Some state variables are optional to be accurately regulated or measured if uncertainty about their values is eliminated after a contact event. Perception must collect and interpret contact-based measurements, increasing its computational complexity.

Composing Motion Behaviors With Contact Forces

A different use of environmental constraint exploitation is composing motion behaviors with interaction forces. A body in contact with the environment is subject to interaction forces, such as contact normals and friction. When a body in contact with the environment is actuated with external forces, its motion behavior is the sum of all external and interaction forces acting on it. In the context of compliant motion generation, Lozano-Pérez et al. (1984) devised such composed motion behaviors using hybrid position and force control, sometimes in contact with the environment, and called the resulting behavior fine-motion strategies. With grasping, Chavan-Dafle and Rodriguez (2015) used interaction forces with the environment to reposition a grasped object and called it external dexterity. Environmental constraints can also be thought of as a virtual finger that may hold an object for repositioning and help in pivoting, sliding, or stabilizing its motion (Chang et al., 2008; Odhner et al., 2013; Chavan-Dafle and Rodriguez, 2015; Deimel et al., 2016).

The additional actuation provided by interaction forces increases a robot's dexterity without increasing its hardware or control complexity. Planning also becomes simpler

when a simple actuation produces complex behaviors without needing to plan a series of multiple actuations.

Environmental Constraints Structure State-Action Space

Environmental constraint exploitation structures a robot's workspace into regions with consistent geometrical properties. Also, it structures the actions that a robot may take in those regions for a given task.

On the one hand, an environment's geometrical properties and connectivity implicitly structure a robot's workspace into regions of environmental constraints. When these regions are known, a planner can search for a sequence of regions between a start and goal configuration and then search for the actual motion plan through the selected regions. Limiting the state space search to a subset of regions simplifies the planning problem since fewer states must be considered.

On the other hand, environmental constraints restrict possible motion, and thus, it restricts actions that a robot can take. Limiting the action space simplifies planning similarly to limiting the state space because a planner can reason about a subset of actions in a state space region without considering all possible actions.

When environmental constraint exploitation regions are not known prior to planning, these must be computed from the geometrical model of the environment or sensor measurements. For the former case, Lozano-Pérez et al. (1984) proposed to compute these regions exactly, and they called them pre-images. For the latter, these regions can be approximated by detecting affordances of environmental constraint exploitation for robust manipulation (Kaiser et al., 2014; Eppner and Brock, 2015)

A further use of environmental constraint exploitation is representing structural context to reduce the search for manipulation funnels and contact events. A planner has to explicitly reason about these regions in a given environment for a given task, and perception has to detect these regions.

In summary, environmental constraint exploitation can be viewed as a set of "tools" to reduce uncertainty and simplify planning, control, and perception. One challenge is combining these tools appropriately for a given task. Another challenge arises from the fact that all robotic system components are involved in environmental constraint exploitation. Therefore, environmental constraint exploitation requires a synergistic combination of hardware, control, perception, and planning.

1.3 Content and Contributions

The thesis is split into three parts. The first part generalizes environmental constraint exploitation by applying its effects, observed mostly in manipulation research, to motion planning under uncertainty. The second part aims to expand our understanding of environmental constraint exploitation by investigating the benefits of a new environmental constraint in a specific grasping scenario. The third part focuses on the implications of environmental constraint exploitation on a robotic system as a hole by integrating algorithmic solutions presented in Part I and II into a grasping system designed to solve real-world industrial applications. Note that we split background material into two chapters: First, Chapter 2 introduces relevant concepts and planning approaches for Part I. Secondly, Chapter 5 introduces classical grasp planning approaches to highlight what computational aspects are simplified in Part II.

Below, we present each part's content and provide a list of the main contributions. A detailed outline and list of contributions are provided at the beginning of each part.

In *Part I*, we apply environmental constraint exploitation in motion planning under uncertainty. Chapter 2 provides background on motion planning. We assume to know the environment's geometry, and we model motion and perception uncertainty and its effect on a robot's state. We differentiate between planning problems under low uncertainty and high uncertainty. For low uncertainty, Chapter 3 presents a planning method that searches for robust contact-exploiting and free-space motions to reduce state uncertainty when needed. The method approximates task-relevant environmental constraint regions and uses these regions to limit the state and action space exploration for *robust actions*. For high uncertainty, Chapter 4 *integrates contact sensing* into planning to discriminate between outcomes of contact-exploiting and free-space motions to reduce increased state uncertainty. We evaluate our methods in simulation and show that the respective motion planners scale to high-dimensional problem spaces, complex environments, and increased uncertainty.

In *Part II*, we solve a particular grasping problem as simply as possible by exploiting environmental constraints. By focusing on simplicity, we use all the beneficial effects of environmental constraint exploitation. Chapter 5 provides background on grasping, and Chapter 6 tackles grasping from piles of nearly identical objects, sometimes next to static walls. We assume that uncertainty originates from inaccurate sensing, motion, and modeling of movable objects' interactions. First, we present a new environmental constraint emerging from the complex interaction of movable objects in a pile and show robust grasping with a task-tailored sequence of manipulation funnels. We characterize the novel environmental constraint empirically to derive realistic assumptions for grasp

planning. Then, we simplify the planning problem and devise a grasp planner that ignores uncertainty because our task-tailored sequence of manipulation funnels enabled robust grasping under uncertainty. Finally, we evaluate the novel environmental constraint in simulation, with real-world experiments, and in a bin-picking application.

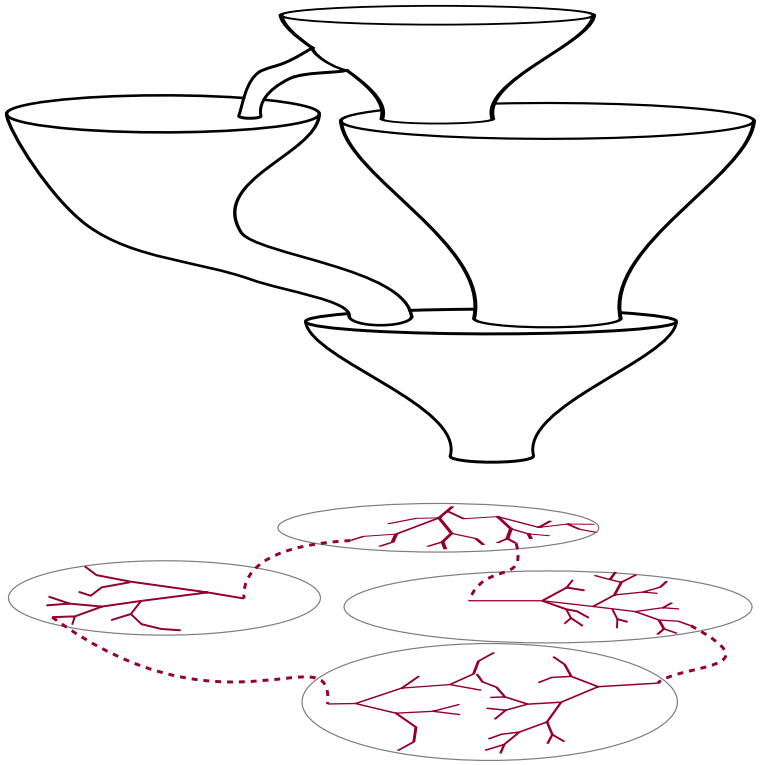
In *Part III*, we analyze the implications of environmental constraint exploitation on robotic system designs. Chapter 7 presents a complex robotic system that combines algorithmic contributions presented in Part I and II to solve real-world industrial applications. We explain the practical implications of environmental constraint exploitation on perception, planning, control, hardware, and environment design and propose general system-building practices beneficial for robotic systems exploiting environmental constraints. On the next page, Table 1.1 lists the thesis's main scientific and technical contributions.

Topic	Chapter	Sec.	Contribution
Motion planning under uncertainty	3	3.2.1	We propose an approximation of environmental constraint exploitation regions in high-dimensional configuration spaces that is computable efficiently for task-relevant regions.
		3.2.4	We devise a motion planning method that efficiently finds a sequence of motions from A to B under uncertainty. Using the approximated environmental constraint regions, the planner guides searching for free-space motions unaffected by uncertainty or contact-exploiting motions that reduce uncertainty.
	4	4.2.1	We propose an effective uncertainty reduction technique by moving into contact and observing the resulting contact measurements to rule out possible outcomes.
		4.2.3	We devise a planning method that handles uncertainty by efficiently planning contingencies for anticipated deviations. It integrates contact sensing as discrete feedback to anticipate deviation and reuses solved contingencies to extend the goal region.
Grasping from piles	6	6.1	We discover and empirical study a new environmental constraint emerging from complex and dynamic interactions of movable objects in a pile. We instantiate the novel environmental constraint with a simple grasping strategy using open-loop controllers without detecting individual objects in a pile.
		6.2	We devise a grasp planning method that sequences environmental constraints for grasping from piles of nearly identical objects. We simplify the representation of the planning problem using the results from our empirical study of the novel environmental constraint.
Robotic system design	7	7.3	We expose practical implications of environmental constraint exploitation on perception, planning, control, and hardware by devising a complex robotic system for two real-world tasks using planning methods presented in previous parts of the thesis.
		7.5	We propose hypotheses of beneficial design choices for general robotic system-building when using environmental constraint exploitation.

Table 1.1 List of scientific and technical contributions of the thesis.

Part I

Motion Planning With Environmental Constraints



"All human plans [are] subject to ruthless revision by Nature, or Fate, or whatever one preferred to call the powers behind the Universe."

Arthur C. Clarke, 2010: Odyssey Two

Motivation

Motion planning is a computational process of finding a collision-free motion trajectory between two robot configurations. Motion planning considers the whole robot that can be composed of a mobile base, robot arm, and an end-effector. A collision-free path means that the robot avoids colliding with the environment because an undesired or uncontrolled collision might damage the environment or the robot leading to failure.

Finding a collision-free path becomes increasingly difficult if we consider inaccurate perception, motion, or model of the environment. Such inaccuracies make a robot state uncertain. If uncertainty is handled poorly, the solution becomes brittle leading to execution failure. On the other hand, if a planner anticipates all possible deviations and plans contingency for each deviation, it needs enormous computation time to solve all contingencies.

Fortunately, environmental constraint exploitation reduces state uncertainty implicitly using manipulation funnels and explicitly by reasoning about contact events. With ECE, we can redefine the motion planning problem from finding a collision-free path to combining collision-free and contact-exploiting motions. A planner combining collision-free and contact-exploiting motions still needs tremendous computation time.

This part of the thesis applies ECE in motion planning to reduce uncertainty and to simplify planning by leveraging the benefits of environmental constraint exploitation.

Contributions

The contributions of the first part of this thesis are the following:

- We propose an approximation of environmental constraint exploitation regions in high-dimensional configuration spaces computable for task-relevant regions efficiently.

- We devise a motion planning method that finds a sequence of motions from A to B under uncertainty. It assumes to know the geometrical model of the environment. However, motion and proprioception are inaccurate. The planner approximates environmental constraint regions to guide the exploration of configurations, free-space motions unaffected by uncertainty, and contact-exploiting motions that reduce uncertainty.

- We devise another motion planner that finds a sequence of motions from A to B under uncertainty. It uses the same assumptions and EC region approximation as stated above but only reduces the configurations space for exploration.

- We propose to reduce uncertainty explicitly by moving into contact and observing the resulting contact measurements to rule out possible outcomes and plan appropriate reactions.
- We devise a planning method that finds alternative motion trajectories from A to B, leveraging contact events to anticipate deviations. It assumes a noisy motion model, inaccurate proprioception, a known geometrical model of the environment, and a fully observable contact sensing. As a result, the solution is a contingency plan that reacts appropriately to deviations during execution.
- We devise a further motion planning method that combines guided configuration space exploration with efficient contingent planning under the same assumptions as stated for the previous method.

Outline

This part presents motion planning methods leveraging geometrical features of a given environment as presented in Section 1.2. We explicitly model the effect of motion and perception uncertainty on a robot and handle uncertainty by searching for robust actions immune to it or augmenting uncertain states with contact measurements.

Chapter 2 introduces fundamental concepts and motion planning algorithms that are relevant to the contributions presented in later chapters.

Chapter 3 presents a belief space planner that searches for robust actions in task-relevant ECE regions. Section 3.1 formally defines the planning problem. Section 3.2 explains the key ideas to identify task-relevant regions in a robot's workspace, map these regions to a combined configuration and action space, and describe two motion planners, the Contact Exploring Exploiting Tree planner and the Contact Exploring Tree planner. The former leverages both configuration and action space reduction, while the latter only uses configuration space reduction for **guided exploration**. Section 3.3 experimentally evaluates the described motion planners showing that our approach scales to high-dimensional configuration space planning and large workspace with numerous surfaces. Finally, Section 3.4 discusses related planning approaches, and Section 3.5 concludes the chapter with a discussion about limitations and future considerations.

Chapter 4 presents a motion planner that integrates contact measurements into planning to handle increased amounts of uncertainty by reasoning about contingencies. Section 4.1 defines the **contingent motion planning** problem in a combined configuration and contact spaces under motion uncertainty. Section 4.2 explains the core ideas

about anticipating detectable deviations with contact events to reduce uncertainty and planning appropriate contingency for each deviation efficiently. Moreover, we describe the Contingent Contact Exploiting RRT motion planner. Section 4.3 evaluates the proposed motion planner showing increased robustness against a large amount of uncertainty, scalability to high-dimensional configuration space planning, and its application for tactile object localization with a real-world experiment. Similar to the previous chapter, we conclude this chapter by discussing related work, the limitation of our approach, and further considerations for contingent motion planning.

2

Background on Motion Planning

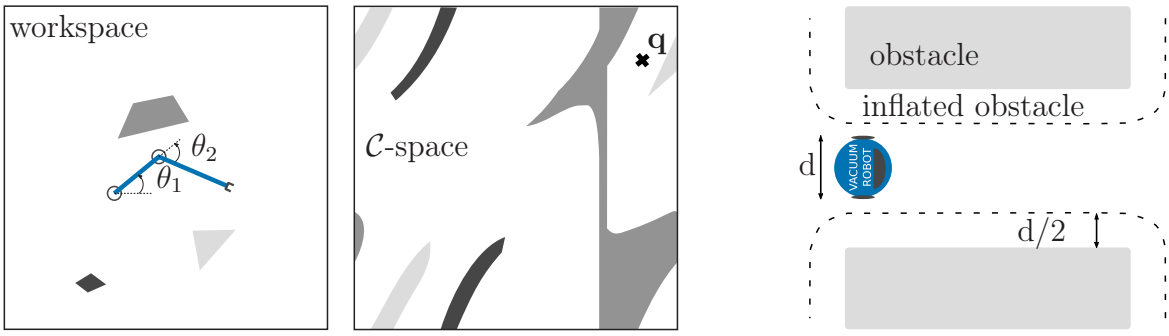
This chapter introduces the basic motion planning concepts and algorithmic approaches relevant to our contributions presented in the later chapters. The content of this chapter is unique to this thesis, and for more details, we refer to planning textbooks (LaValle, 2006; Latombe, 2012; Lynch and Park, 2017). Note that we provide background on grasping in Chapter 5 at the beginning of Part II since the second part focuses on grasping and this part only focuses on motion planning.

First, Section 2.1 defines the representation of a robot's configuration, and then, Section 2.2 presents three different configuration space planners. We discuss how these algorithms use prior task information because we also integrate ECE-based task-relevant information into motion planning. Then, Section 2.3 discusses how to generate motion on task-space constraints because a contact surface is a task-space constraint, and we use it as a geometrical EC to reduce state uncertainty. Uncertainty handling during planning is central to this part of the thesis, so Section 2.4 presents planning approaches to handle it. Motion planning under uncertainty is a complex computational problem, so, Section 2.5 discusses common assumptions and strategies to improve a planner's computational efficiency.

2.1 Configuration Space Obstacles

Motion planning is the process of finding a robot's motion between a start and a goal configuration such that the motion plan respects the kinematic and dynamic limitations of the robot and avoids collision with obstacles.

The most fundamental concept for motion planning is *configuration space*, or $\mathcal{C}$-space. In $\mathcal{C}$-space, a point represents a unique robot configuration $\mathbf{q} \in \mathbb{R}^n$, where n is the number of joints of the robot. In classical motion planning, the environment is an



a) obstacles in workspace and configuration space b) workspace obstacle inflation

Figure 2.1 Examples of configuration space obstacles. *Left and middle:* workspace obstacles (gray) even with simple geometries have a highly non-linear shape in $\mathcal{C}$-space visualized for a robot arm with two rotational joints at angles θ_1 and θ_2 for configuration $\mathbf{q}$. *Right:* when the workspace and $\mathcal{C}$-space are identical, obstacles (gray) are inflated, in this example, by the radius $d/2$ of a circular robot (blue), to represent the robot with a point and account for its shape for collision avoidance.

obstacle, and a robot must avoid collision with the environment. This view results in a division of $\mathcal{C}$-space into two parts: $\mathcal{C} = \mathcal{C}_{\text{obstacle}} \cup \mathcal{C}_{\text{free}}$. The free configuration space $\mathcal{C}_{\text{free}}$ defines the space where a robot is not in a collision or penetrating an obstacle and is within its joint limits. Moreover, the configuration space occupied by obstacles is $\mathcal{C}_{\text{obstacle}}$, where a robot cannot go. Since ECE is a central theme in this thesis, we define the boundary of an obstacle as $\delta\mathcal{C}_{\text{free}}$. $\delta\mathcal{C}_{\text{free}}$ becomes relevant for contact-exploiting motions, but for now, we will only consider $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{\text{obstacle}}$.

With $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{\text{obstacle}}$, the motion planning problem is finding a motion trajectory of a point robot in the free space under kinematic and dynamic constraints between $\mathbf{q}_{\text{start}}$ and $\mathbf{q}_{\text{goal}}$. However, obstacles can crate separated components of $\mathcal{C}_{\text{free}}$. If the start and goal configurations are in separated components of $\mathcal{C}_{\text{free}}$, there is no collision-free solution to the planning problem.

It is challenging to represent $\mathcal{C}_{\text{obstacle}}$ mathematically. The shape of $\mathcal{C}_{\text{obstacle}}$ is very complex even when obstacles have simple geometrical shapes, as visualized on the left two sketches in Figure 2.1. Even if a robot's configuration space is identical to the workspace, an exact $\mathcal{C}_{\text{obstacle}}$ representation is impractical. For example, consider a circular mobile robot that can only translate without rotation. The obstacles are "inflated" by the robot's radius to represent the robot with a point, as shown on the rightmost sketch in Figure 2.1. An obstacle's exact representation becomes quite complex if a robot can rotate and it has an irregular shape. Thus, $\mathcal{C}_{\text{obstacle}}$ is rarely described with an exact mathematical model.

To overcome modeling difficulties of $\mathcal{C}_{\text{obstacle}}$, we can approximate the connectivity of $\mathcal{C}_{\text{free}}$ by sampling configurations and using collision checking, such as the Gilbert-Johnson-Keerthi algorithm (Gilbert et al., 1988). Collision-checking algorithms efficiently approximate distances between two convex bodies. This distance is used to verify if a robot is in a collision or not for a given configuration. We should point out that collision checking is an expensive computational operation. Next, we present two planning approaches that use collision checking but approximate $\mathcal{C}_{\text{free}}$ differently.

2.2 Motion Planning Approaches

This section presents three motion planning approaches relevant to our motion planners in the following chapters. The first two planners use sampling to approximate $\mathcal{C}_{\text{free}}$ connectivity, and the third uses virtual potential fields.

2.2.1 Grid-Based Planning

We can approximate $\mathcal{C}_{\text{free}}$ by discretizing $\mathcal{C}$ into a grid, where the configuration space is discretized with k grid points. With this discretization, $\mathcal{C}_{\text{free}}$ can be approximated with a graph where nodes are grid points in $\mathcal{C}_{\text{free}}$ and edges are connections between neighboring nodes where motion along grid lines is collision-free. Graph search algorithms, such as $A*$ (Hart et al., 1968), can generate a motion trajectory by adding $\mathbf{q}_{\text{start}}$ and $\mathbf{q}_{\text{goal}}$ as nodes and connecting them to the closest node with a collision-free edge.

If the environment and $\mathbf{q}_{\text{goal}}$ are the same for different planning problems, it is worth preprocessing the grid by assigning a distance value to each grid point relative to the goal with a *wavefront* planner.

A wavefront planner updates distance values of grid points similarly to how a wave propagates from a splash in a pond, where the splash is the goal grid point, and the pond is the grid itself, as illustrated in Figure 2.2. In our example, the green cell with the goal was initialized with a value of 0. Using breath-first traversal of adjacent cells

9	8	7	6	5	∞	∞	∞	∞	10
8	7	6	5	4	∞	∞	∞	∞	9
9	∞	∞	4	3	4	5	∞	∞	8
10	∞	∞	3	2	3	4	∞	∞	7
11	∞	∞	2	1	2	3	∞	∞	6
12	∞	∞	1	0	1	2	3	4	5
11	10	∞	2	1	2	3	4	5	6
10	9	∞	3	2	∞	∞	∞	∞	7
9	8	∞	4	3	∞	∞	∞	∞	8
8	7	6	5	4	∞	∞	∞	∞	9
9	8	7	6	5	6	7	8	9	10

Figure 2.2 Solution of a wavefront planner on a 2D grid with obstacles (gray), a goal cell (green), and cell labels showing the distance to the goal.

starting from the goal, we increased the distance value of a cell by one for each newly visited depth. The distance values of cells with obstacles were set to ∞, since we want to avoid obstacles. After computing each cell's value, the planning problem is trivial. To reach the goal, a robot can follow the gradient of distance values starting from any free cell.

Although grid-based planners return an optimal solution (with a k resolution) or failure (if start and goal are disconnected) after a limited amount of time, it is only applicable on low-dimensional $\mathcal{C}$-space problems. As the number of dimensions increases, the number of grid points increases exponentially because a k resolution grid needs k^n points. As a result, the problem's computational complexity increases exponentially as well, and this computational complexity problem is referred to as the *curse of dimensionality*.

One approach to overcome the curse of dimensionality is to use multi-scale grids to refine the representation of $\mathcal{C}_{\text{free}}$ near obstacles. Another approach is to sample configurations randomly.

2.2.2 Sampling-Based Planning

Sampling-based planners overcome the curse of dimensionality by drawing random samples from $\mathcal{C}$-space, determining the closest previously drawn sample, and moving from the closest sample toward the random sample. If this motion is collision-free, the respective configuration is added to a graph or tree. The resulting graph or tree approximates $\mathcal{C}_{\text{free}}$. Such a graph-based planner is the Probabilistic Roadmap planner (Kavraki et al., 1996), and a tree-based example is Rapidly-Exploring Random Tree (RRT) planner (LaValle, 1998).

Since our algorithmic contributions in later chapters build on the RRT planner, we describe this algorithm in detail, including relevant variations.

The basic RRT motion planner (Algorithm 1) assumes to know the environment's exact geometry and a robot's kinematics and/or dynamics. The planner builds a tree from $\mathbf{q}_{\text{start}}$ until $\mathbf{q}_{\text{goal}}$ is reached with an ϵ_{goal} precision by adding configurations as nodes and edges between nodes. At each tree expansion, the planner *samples* a random configuration $\mathbf{q}_{\text{rand}}$, searches for the *nearest neighbor* $\mathbf{q}_{\text{near}}$ that is the closest node in the three to $\mathbf{q}_{\text{rand}}$, and *extends* $\mathbf{q}_{\text{near}}$ toward $\mathbf{q}_{\text{rand}}$ with a small distance d. Since d is small, a *local planner* can simulate a simple straight-line motion. If the motion is collision-free and within kinematic and dynamic limitations, the reached configuration $\mathbf{q}_{\text{new}}$ is added to the tree with an edge from $\mathbf{q}_{\text{near}}$ to $\mathbf{q}_{\text{new}}$.

Algorithm 1 Basic RRT planner

Input: $\mathbf{q}_{\text{start}}, \mathbf{q}_{\text{goal}}$
Output: $T = (V, E)$
Output: $\mathbf{q}_{\text{new}} = \mathbf{q}_{\text{start}}$
1: $V \leftarrow \{\mathbf{q}_{\text{start}}\}$ *init tree with start configuration*
2: **while** $\|\mathbf{q}_{\text{new}} - \mathbf{q}_{\text{goal}}\| \geq \epsilon_{\text{goal}}$ **do** *search until goal reached*
3: $\quad \mathbf{q}_{\text{rand}} \leftarrow \text{SAMPLE}()$
4: $\quad \mathbf{q}_{\text{near}} \leftarrow \text{NEAREST\_NEIGHBOUR}(V, \mathbf{q}_{\text{rand}})$ *the closest neighbors of* $\mathbf{q}_{\text{rand}}$ *from* V
5: $\quad \mathbf{q}_{\text{new}} \leftarrow \text{EXTEND}(\mathbf{q}_{\text{near}}, \mathbf{q}_{\text{rand}})$ *local planner*
6: $\quad$ **if** $\text{IS\_VALID}(\mathbf{q}_{\text{new}})$ **then**
7: $\quad\quad V \leftarrow V \cup \{\mathbf{q}_{\text{new}}\}$
8: $\quad\quad E \leftarrow E \cup \{(\mathbf{q}_{\text{near}}, \mathbf{q}_{\text{new}})\}$
9: **return** T

The RRT planner is a single-query planner, i.e., it needs to re-plan for any new $\mathbf{q}_{\text{start}}$ or $\mathbf{q}_{\text{goal}}$. However, the tree rapidly grows into the search space and later fills in the gaps. The planner achieves this behavior by extending the nearest node from a random sample. If random samples are drawn uniformly, the probability of extending a node is proportional to its Voronoi region. The Voronoi region of a node consists of all the configurations closer to the node than any other node. This exploration behavior is called Voronoi bias.

The RRT planner has numerous variations because its simple components can be easily extended to achieve a task-tailored $\mathcal{C}$-space exploration. For example, the sampler can be biased toward the goal by drawing $\mathbf{q}_{\text{goal}}$ with a given probability. As a result, the tree grows faster toward the goal. The local planner can simulate the entire motion from $\mathbf{q}_{\text{near}}$ to $\mathbf{q}_{\text{rand}}$, and thus, connect $\mathbf{q}_{\text{near}}$ with $\mathbf{q}_{\text{rand}}$ using the *connect* local planner. The connect local planner was used in the RRT-Connect motion planner (Kuffner and LaValle, 2000) that grew two trees: one from the start and one from the goal. This planner used a greedy connect-action to connect one tree with the other; when that action failed due to collision, the respective tree grew using random sampling.

A rather different variant, the RRT* planner, is in the family of optimal sampling-based planners (Karaman and Frazzoli, 2011). The goal of the RRT* planner is to find an optimal solution for a cost function associated with the path, for example, path length. It has two key differences compared to the basic RRT planner. First, a new configuration is not added to the tree with an edge to the nearest node. However, it is connected with a neighboring node with the lowest cost from the start node, i.e., cost to come. Secondly, after the new node is added to the tree, the planner rewires neighboring nodes to reach them from the new node if their cost to come is reduced.

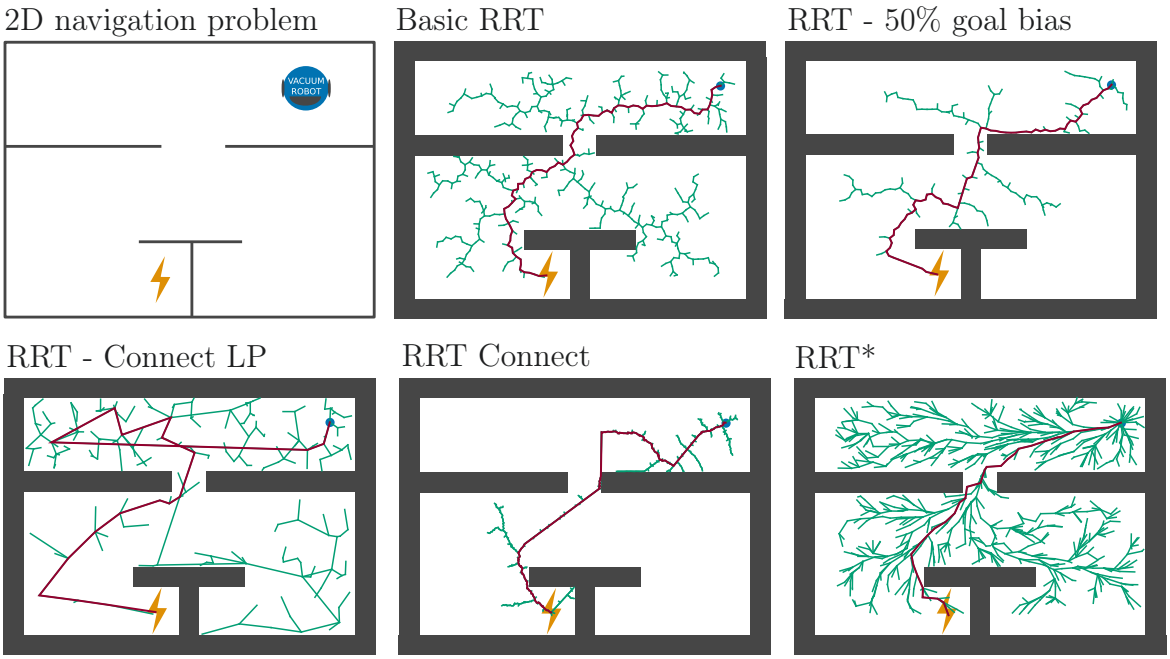


Figure 2.3 Variations of the RRT motion planner show different exploration behavior on the same planning problem, where a circular vacuum robot (blue) that can only translate needs to navigate to its charging station (yellow bolt) shown in the first sketch. The tree is visualized with green lines, and the solution path is red. Using priors about the task reduces the tree size to reach the goal and, thus, makes planning more efficient, where LP stands for *local planner*. However, searching for an optimal solution with RRT* requires more exploration, a larger tree, and more computation.

In Figure 2.3, we show $\mathcal{C}$-space exploration behaviors of different RRT-based planners to illustrate how simple modifications change the exploration behavior and the planning efficiency for the same planning problem. The planning problem is finding a motion trajectory of a robot that can only translate from the top right corner of a room to the charging station in the bottom left room. One can observe a stark difference in how each planner explores the configuration space. The required number of nodes to reach the charging station is less than the basic RRT for each modified planner except RRT*. Having fewer nodes means that a planner is more efficient because it needs less $\mathcal{C}$-space exploration to find a solution.

First, let us discuss the modifications that increased planning efficiency in the previous example. The first three modifications exploited different prior information about the planning problem. With goal bias, the planner exploited knowledge about the location of the charging station. With the connect local-planner, the respective motion planner exploited knowledge about the robot's motion controller that allowed driving the robot in straight lines. With two tree expansions from start and goal, the

RRT-Connect planner exploited the fact that a path from the robot's current location to the charger is interchangeable with a path from the charger to the robot.

Secondly, let us discuss why the RRT* planner generated a significantly denser tree than the other planners indicating a lower efficiency in finding a solution. RRT* searched for the shortest path by storing the path length from the start to each node and exploiting this information to rewire the tree. Since the planner does not stop when the goal is reached but further optimizes the tree structure, it grows further in $\mathcal{C}_{\text{free}}$ and fills in more gaps than the other four planners. Therefore, searching for an optimal solution requires more computation than finding any solution.

2.2.3 Virtual Potential Field Based Planning

Virtual potential field-based methods are inspired by naturally occurring potential fields such as magnetic or gravitational fields. As gravity induces a force on an object and can move it from a high altitude to a valley, we can define a virtual potential field over $\mathcal{C}$-space to induce a force that moves a robot from a high- to a low-potential. Suppose the goal region has a very low virtual potential and all configurations in $\mathcal{C}_{\text{obstacle}}$ have a very high virtual potential. In that case, the negative gradient of the respective virtual potential field provides a force that drives a robot toward the goal while avoiding collision with obstacles.

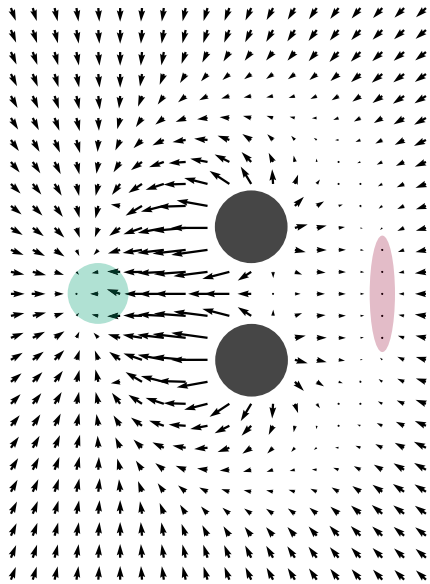


Figure 2.4 A virtual potential field with a goal region (green), two obstacles (grey), arrows showing the induced forces, and a local minimum region (red).

The virtual potential field approach differs from the previously presented grid- and sampling-based approaches. It is easy to compute the potential field, and a robot can avoid moving or unseen obstacles with appropriate sensorization. However, the basic method has a severe drawback. When the environment has multiple obstacles, the computed potential field can have local minima away from the goal, as illustrated in Figure 2.4. In such local minima, a robot gets stuck even if a collision-free path exists to the goal. We can overcome the local minimum problem using a wavefront planner because a wavefront planner computes a local-minimum-free potential function.

2.3 Task Constrained Planning

We explained in the introduction (Section 1.2) that the environment provides beneficial constraints for manipulation. We want to use such constraints in motion planning as well. Therefore, we discuss task-constrained planning concepts.

When a robot moves in contact with the environment or on a task constraint, its configuration is constrained to a lower dimensional manifold. We define this m-dimensional manifold as function $F : \mathcal{C} \to \mathbb{R}^m$ such that $F(\mathbf{q}) = 0$. All configuration that fulfill this equation are on the manifold $\mathcal{M}_F = \{\mathbf{q} \in \mathcal{C}, F(\mathbf{q}) = 0\}$.

Using this definition, we can formulate a task-constrained path planning problem as follows: Given a constraint function F, find a motion trajectory that brings a robot from a start configuration $\mathbf{q}_{\text{start}} \in \mathcal{M}_F$ to a goal state $\mathbf{q}_{\text{goal}} \in \mathcal{M}_F$ such that the trajectory is on the manifold.

Our pencil grasp strategy presented in Section 1.2 used two types of constrained motions: *pose constrained motion* and *contact constrained motion*.

With *pose constrained motion*, our hand approached a pencil from free space with its palm facing down and moving straight down. We can define such a pose constraint as:

$$F(\mathbf{q}) = 0 \text{ if } f(\mathbf{q}) \in \mathcal{M}_F \subset \mathcal{W},$$

where f is the forward kinematic model of the robot mapping a configuration $\mathbf{q} \in \mathcal{C}$ to an end-effector pose (e.g., hand pose) $\mathbf{x} \in \mathcal{W} \subset SE(3)$.

With *contact constrained motion*, our fingers slid on a tabletop toward a pencil.We can define such a contact constraint for a robot as follows:

$$F(\mathbf{q}) = 0 \text{ if } \mathbf{q} \in \mathcal{M}_F \subset \delta\mathcal{C}_{\text{free}}.$$

It is difficult to randomly sample a configuration from $\mathcal{C}$-space so that it is on a pose- or contact-manifold because these manifolds are lower-dimensional than the state space, and they are a relatively small region compared to the whole state space. Nevertheless, we can borrow the principles of potential fields to drag a sample onto a manifold and on the manifold. Suppose we have a parametric representation of a manifold. In that case, we can move on a manifold using *direct sampling* (Shkolnik and Tedrake, 2009), otherwise using *task projection* (Stilman, 2007), both illustrated in Figure 2.5 and explained below.

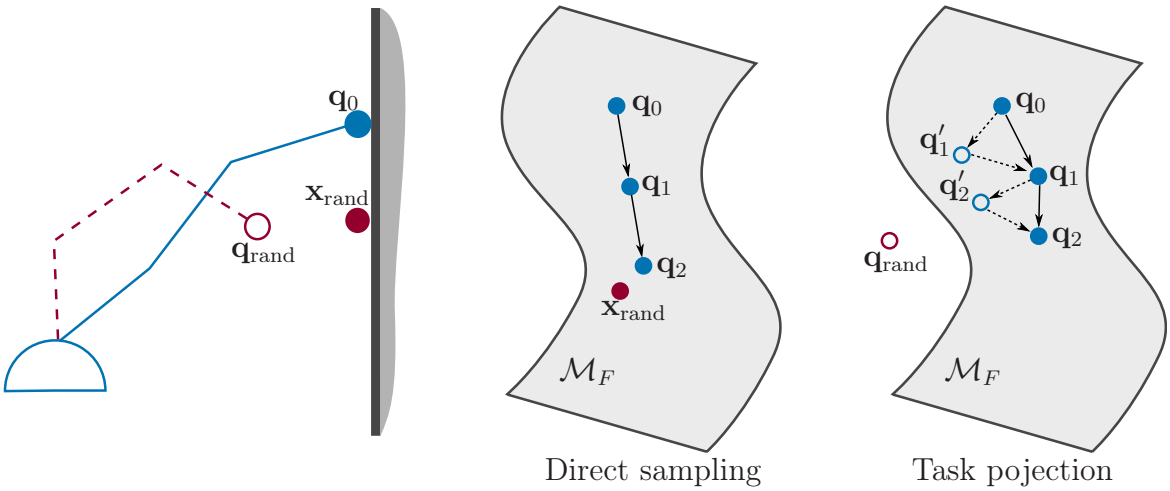


Figure 2.5 The sketch illustrates *direct sampling* and *task projection* using a robot arm with three rotational joints (left) with initial $\mathbf{q}_0$ in contact with a wall (gray). With direct sampling, a random point $\mathbf{x}_{\text{rand}}$ is sampled on the contact surface, and the end-effector is dragged step-by-step toward this point along the contact manifold $\mathcal{M}_F$. With *task projection*, a random configuration $\mathbf{q}_{\text{rand}}$ is sampled away from the surface, and then, $\mathbf{q}_0$ is dragged toward $\mathbf{q}_{\text{rand}}$. If the resulting configuration $\mathbf{q}'_1$ leaves $\mathcal{M}_F$, it is dragged back onto the manifold.

With direct sampling, a random sample is drawn on the manifold $\mathbf{x}_{\text{rand}} \in \mathcal{M}_F$, then, a configuration $\mathbf{q}$ on the manifold is dragged toward the random sample using the transpose or pseudoinverse of the Jacobian at each simulation step:

$$\mathbf{q}_{n+1} = \mathbf{q}_n + \delta \mathbf{J}^+_{\mathcal{M}} \left(f(\mathbf{q}_n) - \mathbf{x}_{\text{rand}} \right).$$

Since direct sampling is parameterized, we can draw uniform samples on the manifold, and the method is scaleable to very high-dimensional configuration spaces. However, it is not probabilistic complete.

With task projection, a random configuration $\mathbf{q}_{\text{rand}}$ is sampled from $\mathcal{C}$-space. The sample can be outside the manifold because a configuration $\mathbf{q}$ initially on the manifold is first moved toward the random sample. Then, the resulting configuration $\mathbf{q}'$ is projected back onto the manifold:

$$\begin{aligned} \mathbf{q}'_{n+1} &= \mathbf{q}_n + \delta \left(\mathbf{q}_{\text{rand}} - \mathbf{q}_n \right) \\ \mathbf{q}_{n+1} &= \mathbf{q}'_{n+1} + \mathbf{J}^+_{\mathcal{M}} \left(\Delta \mathbf{x}_{\text{distance}} \right), \end{aligned}$$

where $\Delta \mathbf{x}_{\text{distance}}$ is the vector between the manifold and the closest point on a robot. The task projection method requires more computation than direct sampling due to

the two steps, but Stilman (2010) showed probabilistic completeness and observed an additional benefit. When computing the Jacobian during planning, $\det(\mathbf{J}\mathbf{J}^{\mathrm{T}})^{1/2}$ can indicate the manipulability of sampled configurations.

2.4 Strategies for Uncertainty Handling

Since we want to handle uncertainty with ECE in motion planning, we present three relevant strategies. First, we can *ignore uncertainty* during planning. Second, we can search for *robust actions* immune to uncertainty. Third, we can *integrate sensing* into planning to augment uncertain robot states with information from sensor measurements. Next, we present approaches for the three strategies and discuss their effect on planning.

2.4.1 Ignoring Uncertainty

Even though it is unrealistic to assume that a robot can perfectly execute motion or sense its environment, uncertainty can be ignored during planning if inaccuracies do not accumulate beyond the tolerance of a given task. In this case, uncertainty handling becomes the responsibility of perception, control, or even of ECE, making the respective planner more efficient than the one that needs to handle it.

We will show in Section 6.1 that using a new EC with a soft hand eliminates uncertainty completely for grasping round objects from a pile. Hence, we will omit to model state uncertainty, so our grasp planner (see Section 6.2.2) can ignore uncertainty.

Another way to ignore uncertainty during planning is by *re-planning*. First, a robot executes a short motion plan (without necessarily reaching the goal). Then, the robot updates its state based on sensor measurements and re-plans with the updated state. This way, the robot eliminates some possible action outcomes through execution. However, perception has greater responsibility for uncertainty handling than control because the planner needs an accurate estimation of its state before re-planning.

2.4.2 Searching for Actions Under Motion Inaccuracies

Another approach is searching for reactive actions under stochastic action outcomes. Then, a planner assumes that a robot's motion is inaccurate, but it can identify its state during execution. Since the robot knows its state during execution, from state memory or sensing, the planner can compute a mapping from states to actions. Such a mapping is called a policy π. As opposed to a motion plan which is a sequence of states, the policy provides an appropriate action for a given state.

We can define a probabilistic motion model for executing an action u at a configuration $\mathbf{q}$ to account for motion inaccuracies. Similarly to grid-based planners, we can define a discrete set of states and actions, which problem formalism leads to the *Markov Decision Process* (MDP) model.

An MDP provides decision-making by modeling the problem with a discrete set of states $\mathbf{q} \in \mathcal{C}$ and of actions $u \in \mathcal{U}$, a probabilistic transition function $P(\mathbf{q}'|\mathbf{q}, u)$, and a reward function $R(\mathbf{q})$. The transition function must have the Markov property, i.e., the probability of reaching $\mathbf{q}'$ depends only on the action u and state $\mathbf{q}$, and probability must be independent of past actions and states. The reward function $R : \mathcal{C} \to \mathbb{R}_{\neq \pm\infty}$ maps a state to a real value that is not infinite. It is used as an immediate reward after reaching a new state. Unlike virtual potential fields, the goal region has a high positive value, and regions to be avoided have a negative value.

An MDP's solution is an optimal policy $\pi^* : \mathcal{C} \to \mathcal{U}$ mapping states to actions. We can obtain an optimal solution because an MDP-based planner's objective is to maximize the cumulative discounted reward for a sequence of actions:

$$\sum_{t=0}^{\infty} \gamma^t R(\mathbf{q}_t),$$

where $\gamma \in [0, 1]$ is a discount factor modifying the planner's preference for early or later rewards. By choosing a low value for γ, the distant future is insignificant, and a planner chooses actions greedily. When the value is closer to one (which is more common), early and later rewards are weighted similarly, allowing the planner to explore the future better.

An MDP planner searches for the optimal policy that maximizes the expected cumulative discounted reward. To estimate the expected cumulative reward of a state, i.e., the current and future rewards, we can use the Bellman equation that computes a value function $V : \mathcal{C} \to \mathbb{R}$:

$$V(\mathbf{q}) = R(\mathbf{q}) + \gamma \cdot \max_u \int P(\mathbf{q}'|, \mathbf{q}, u) V(\mathbf{q}') d\mathbf{q}'.$$

Observe that this is a recursive function, which is non-linear due to the *max* operator. Therefore, we calculate this function with a value iteration algorithm to obtain the optimal policy. To compute the optimal policy π^*, we choose actions that maximize the utility of a state:

$$\pi^*(\mathbf{q}) = \operatorname*{argmax}_u \int P(\mathbf{q}'|, \mathbf{q}, u) V(\mathbf{q}') d\mathbf{q}'.$$

While an MDP-based approach can handle motion inaccuracies well, it is subject to the *curse of dimensionality*, similar to grid-based planners. Therefore, some MDP planners use a sampling-based approach to approximate the state space. For example, Alterovitz et al. (2007) presented an MDP-based PRM planner, and Melchior and Simmons (2007) presented an MDP-based RRT planner. These two planners represented the approximated configuration space with a graph or a tree, respectively, and both computed the transition probabilities between nodes with Monte-Carlo simulation (Thrun, 1999).

2.4.3 Integrating Sensing

Previously, we assumed a robot could accurately observe its current state during execution, so an MDP's policy handled motion inaccuracies by searching for robust actions. However, sensing is also subject to inaccuracies. When sensing inaccuracy is significant, we must consider possible sensor measurements for the same action during planning. We can assume that the robot partially observes its state, leading to the *Partially Observer Markov Decision Process* (POMDP) model.

The POMDP model extends the MDP model by reasoning about sensing inaccuracies. In addition to two discrete sets of states and of actions, a probabilistic motion model, and a reward function (inherited from MDP), we also define a discrete set of observations $o \in \mathcal{O}$ and a probabilistic sensor model $P(o|\mathbf{q})$.

Now that a robot does not know its true state, it needs to reason about the possible states it might be in at a given time. To represent the possible states in which a robot can be, we introduce the *belief state* concept. A belief state $b(\mathbf{q}) \in \mathcal{B}$ is a probability distribution assigned to the true state $\mathbf{q}$.

A belief state holds the observation history. Hence, a POMDP planner does not need to reason about observation history explicitly. To integrate the observation history into a belief state, we update the current b after taking action u as follows:

$$b'(\mathbf{q}') = \alpha P(o|\mathbf{p}') \int_{\mathbf{q}} P(\mathbf{q}'|\mathbf{q}, u) b(\mathbf{q}), \tag{2.1}$$

where α is a normalization constant such that the $\int b'(\mathbf{q}') = 1$.

With POMDPs, the optimal policy $\pi^* : \mathcal{B} \rightarrow \mathcal{U}$ is a mapping from belief states to actions, and it does not depend on the actual state.

A POMDP planner has three general steps. It simulates an action for a given belief, observes the measurement o, and computes the new belief state b' with Equation 2.1. Because the belief state has the Markov property, we can write the respective transition

function as $P(b'|b, u)$. Moreover, we can define the reward function for a belief state as:

$$R(b) = \sum_{\mathbf{q}} b(\mathbf{q}) R(\mathbf{q})$$

Having $P(b'|b, u)$, $R(b)$, and that a belief is fully observable, we cast the problem as an MDP over the space of belief states. Now, we can solve the MDP on the belief-state spaces, and the resulting optimal policy π^* is also optimal for the original POMDP. A belief space is continuous, which makes value iteration very inefficient. Nevertheless, the optimal value function is piecewise linear and convex, which allows the construction of the value function incrementally (Kaelbling et al., 1998).

In summary, we can integrate information from sensing into a belief state to augment state uncertainty caused by sensing and motion inaccuracies. POMDP planners can handle both sources of inaccuracies and provide an optimal solution. Similarly to MDPs, POMDPs are also under the curse of dimensionality because the dimension of $\mathcal{B}$ equals the dimension of $\mathcal{C}$. Moreover, the number of discrete observations further increases the complexity of POMDP planning problems compared to MDPs. For practical problems, the optimal solution of a POMDP is computationally intractable because the problem is PSPACE-complete (Papadimitriou and Tsitsiklis, 1987). There exist reasonable approximations that reduce the problem complexity and increase a POMDP planner's computational efficiency, which we discuss in the next section.

2.5 Strategies for Efficient Belief-Space Planning

Our goal is to handle uncertainty efficiently during motion planning using the beneficial effects of ECE. Previously, we have shown that a planner can handle uncertainty from motion and sensing inaccuracies by representing uncertain states with a belief state. However, searching for a robust solution is a computationally complex problem.

Now, we discuss different techniques to reduce the computational complexity of belief space planning because we also use some of these techniques in later chapters. First, we explain how different uncertainty handling methods increase planning efficiency, then discuss assumptions that simplify the representation and planning. Finally, we discuss how to balance information usage to augment uncertain states.

2.5.1 Uncertainty Handling

The simplest way to handle uncertainty during planning is to ignore it, making planning more efficient. We can use different heuristics to ignore uncertainty.

One heuristic is to ignore uncertainty from certain sources. Ong et al. (2010) considered some dimensions of the state space fully observable and other dimensions partially observable, which made the planning problem a Mixed Observable MDP (MOMDP). By combining fully and partially observable states, they represented a high-dimensional $\mathcal{B}$ as the union of lower-dimensional subspace. This reduction of dimensions improved the planner's computational efficiency.

Another heuristic is to consider uncertainty unchangeable for a state. Yoon et al. (2008) used hindsight optimization with a pessimistic approach by searching for a policy that succeeds without gathering new information. In contrast, Littman et al. (1995) introduces QMDP, where they assumed that any uncertainty of a belief state disappears after the next action. This is an optimistic approach to finding a solution. In both cases, the POMDP was transformed into an MDP, and the problem's computational complexity was reduced while giving up optimality guaranties (Arora et al., 2018).

A further approach is to reduce uncertainty with contact-exploiting actions actively. Will and Grossman (1975) introduced the guarded motion, which action moved a robot into contact with the environment. By doing so, the robot's state uncertainty was constrained to the contact manifold. Since contact manifolds are lower dimensional than the configuration space, they reduce state uncertainty along the "*lost*" dimension. Lozano-Pérez et al. (1984) combined guarded moves with sliding motions in contact to generate complex motion behavior that is also robust against uncertainty. The uncertainty reduction effect of guarded motions and siding are illustrated in Figure 2.6. These contact-exploiting actions were used in an RRT-like motion planner (Sieverling et al., 2017b) to reduce state uncertainty while searching for a solution. We also use such actions to reduce state uncertainty in our planners.

A previously mentioned heuristic is re-planning by interleaving planning and execution on a short time horizon until the goal is reached. This approach tightly links planning and perception because after executing a short sequence of actions, the robot needs to perceive its state to restart planning. It can outperform complex reasoning approaches because it cuts off unlikely branches through real-life execution. However, this approach requires an efficient planner for time-sensitive tasks so that a robot is not blocked by planning during execution (Kaelbling and Lozano-Perez, 2013; Buşoniu et al., 2018).

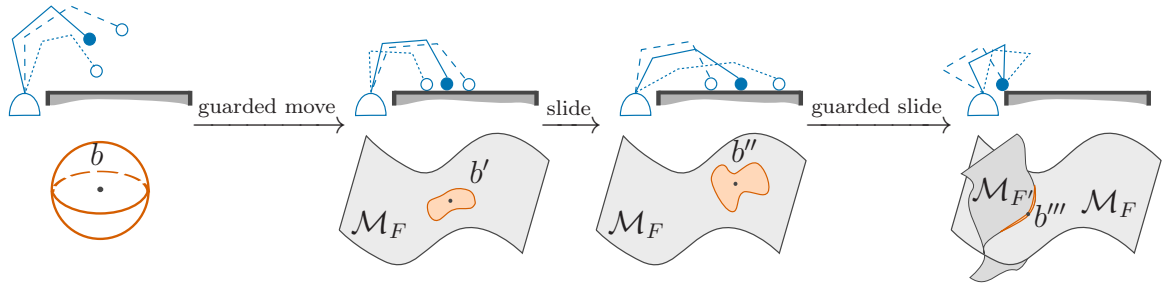


Figure 2.6 All four sketches show a robot arm (blue) with three rotational joints in front of a tabletop, where its state uncertainty is illustrated with the three configurations (solid and dashed lines), and the respective beliefs and contact manifolds are illustrated below each sketch. When the robot performs a *guarded move*, the round end-effector moves into contact. Consequently, the initial belief b, illustrated as a ball, is projected onto the contact manifold $\mathcal{M}_F$, and b' becomes smaller than b. If the arm *slides* on the surface, the state uncertainty only increases along the manifold. A *guarded slide* moves the end-effector to the edge of the tabletop, reducing state uncertainty significantly because b'' collapses into a line at the intersection of the two contact manifolds $\mathcal{M}_F$ and $\mathcal{M}_{F'}$.

2.5.2 Simplifying the Representation

To reduce computational complexity, one can simplify the problem representation. The following simplifications of the problem representation relax the optimality requirement to reduce computational complexity. For example, a Gaussian representation of a belief state or a linear motion or sensor model greatly simplifies the computation of a new belief. Furthermore, a quadratic cost term as a reward can make the planning problem a *Linear Quadratic Regulator* (LQR), and LQRs have a closed-form solution, which can be computed with the Ricchatti equation even for high-dimensional problems.

When belief states are non-Gaussian, we can represent them with particles. A particle-based belief state can be updated similarly to the particle filter algorithm (Thrun et al., 2005), and the state transition function can be approximated using Monte-Carlo simulation (Thrun, 1999). Our planners also use a particle-based representation of beliefs because contact-exploiting actions make beliefs non-Gaussian.

Another approach is to sample only the reachable state space and apply value iteration on this sub-space with a point-based POMDP planner (Kurniawati et al., 2008). One of our planners reduces the search space based on reachability and relevance for a given task.

2.5.3 Balancing Information Usage

Previously, we have shown that various modifications of the basic RRT planner increased the resulting planner's computational efficiency because the modifications used information about a given task. We can use task-based information in belief space planning to explore a belief space efficiently. However, partial information, like in the case of virtual potential fields, may lead to local minima, and information gathered from inaccurate sensing increases the problem's computational complexity. To avoid such issues, we can balance the use of information during planning.

Regarding information balancing, we can place motion planners on a spectrum of how much information they use. One end of the spectrum is *exploitation*, where planners compute a plan based on the available information, such as virtual potential field methods. On the other end of the spectrum is *exploration*, where planners use no information about a given task but sample the space to approximate its connectivity, such as the basic RRT planner. Between the two ends of this spectrum is *guided exploration*. With guided exploration, planners use some information to select a region of the space and explore that region for a solution.

It is important to balance exploration and exploitation. Rickert et al. (2008) proposed to use information about workspace decomposition that guides the exploration of the configuration space between a given start and goal states. To avoid local minima, they adaptively balanced the used workspace information achieving 13 times faster planning than RRT-Connect. Yang (2013) adapted the exploitation of goal bias sampling when the tree approached a goal region to find better samples around the goal for a non-holonomic robot. In the next chapter, we use adaptive exploitation and exploration, where our planner leverages ECs' workspace structuring property to explore only task-relevant configurations.

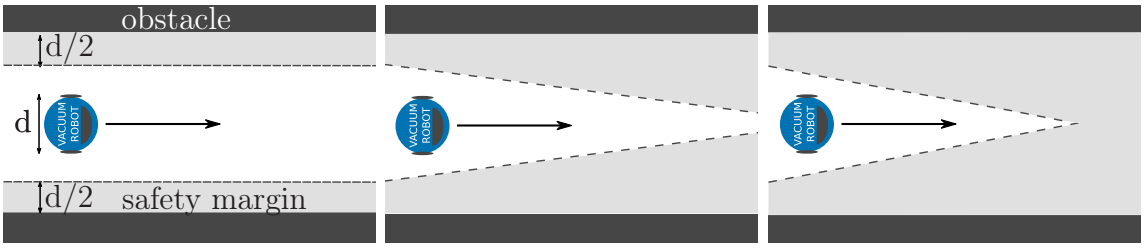
3

Guiding Configuration Space Exploration With Environmental Constraints

This chapter applies environmental constraint exploitation in motion planning under uncertainty. We leverage two of the four ECE portieres discussed in Section 1.2 to simplify motion planning while handling uncertainty. First, we use the structural context of the environment to divide the workspace into ECE regions and reason about these regions' relevance for a given task to guide configuration space exploration. Secondly, we reduce uncertainty with manipulation funnels by contact-exploiting motions along task-relevant parts of the environment. The content of this chapter is unique to this thesis and has not been published before.

While workspace connectivity can be derived from the environment's geometry, it is difficult to compute an exact representation of configuration space connectivity. Configuration space is generally higher dimensional than workspace, and the environment in this representation has a very complex shape (see Figure 2.1). Therefore, we approximate configuration space connectivity using a sampling-based approach. Even though a sampling-based approach can overcome the curse of dimensionality, another challenge arises when planning under uncertainty.

Configuration space connectivity depends on a robot's kinematics, its geometry, and the environment's geometry. When a robot's perception and motion are inaccurate, we are uncertain about the space occupied by the robot, making motions closer to obstacles more probable to lead to undesired collisions and failure. Hence, the feasibility of a motion depends not only on configuration space connectivity but also on state uncertainty which we illustrate with a narrow passage that closes under high motion



(a) zero motion uncertainty maintains passage width

(b) low motion uncertainty reduces passage width

(c) high motion uncertainty closes passage

Figure 3.1 The three sketches illustrate a robot (blue circle) moving through a corridor (arrow) for different motion errors. The safety margin (light gray), where collision can be avoided for a forward motion, increases as motion error accumulates and state uncertainty increases. The narrow passage closes when state uncertainty is so large that the robot might collide with the corridor walls.

error in Figure 3.1. Since motion feasibility also depends on state uncertainty, we model uncertainty and reduce it when needed.

To reduce uncertainty, our planner uses contact-exploiting actions, such as guarded moves, sliding, or guarded sliding, introduced in Section 2.5.1 and illustrated in Figure 2.6. Our planner uses free-space connect actions when uncertainty can be ignored, described in Section 2.2.2.

Since our planner can choose from four actions to explore a configuration space and approximate its connectivity, the planning problem's complexity is further increased. The planner must pick from possible actions and reason about probabilistic outcomes due to uncertainty. For the former, we leverage information provided by ECs to guide action selection and configuration space exploration. For the latter, we search for robust actions leading to an expected outcome, including actions that reduce uncertainty.

Configuration space exploration can be guided with task-relevant workspace information. ECs implicitly structure the workspace into contact manifolds and free space regions from where a manifold is reachable. We can reduce the number of contact and free space regions considered for planning by reasoning about their usefulness for a given task. By ignoring less useful regions, a planner becomes computationally more efficient because it can avoid exploring the entire search space.

In our view, the environment is an essential friend, and we deliberately establish contact with it where and when needed!

We evaluated our planner in simulation and reached up to two orders of magnitude increase in planning performance when using workspace information for guided configuration space exploration in high-dimensional spaces under motion uncertainty.

3.1 Belief-Space Motion Planning Problem Definition

Before presenting our algorithm, we describe the planning problem formally. We plan in an n-dimensional configuration space $\mathcal{C} \in \mathbb{R}^n$, and define $\mathcal{C}_{\text{valid}}$ the *valid* configuration where a robot is within its joint limits and does not collide with the environment with its parts that do not sense contact. We decompose $\mathcal{C}_{\text{valid}}$ into $\mathcal{C}_{\text{free}}$ free space and $\partial\mathcal{C}_{\text{free}}$ configurations in contact at the boundary of free space. We assume to have perfect knowledge about the environment and the robot's kinematic model.

Moreover, we define the task-relevant configuration space as a subset of the valid space $\mathcal{C}_{\text{task}} \subset \mathcal{C}_{\text{valid}}$ where a robot is within its joint limits, does not collide with the environment with its parts that do not sense contact. The subspace comprises manipulation funnels (free- and contact spaces) directly relevant to a given task. A funnel is task-relevant if its wall reduces uncertainty so that robust actions with an expected outcome are available to progress toward a given goal or the goal is reachable with ϵ_{goal} precision.

The planner can simulate an action u, which is either a straight line joint space motion in free space, a guarded move (a free space motion ending in contact), a slide along a surface, or a guarded slide (slide until a new contact is gained or the existing one is lost). All motions have an uncertain outcome, and the robot can not fully perceive its configuration but must estimate it from noisy sensor measurements. Hence, we use a motion model with independent joint noise $\delta\hat{\mathbf{q}} = \delta\mathbf{q} + \mathcal{N}(0, \sqrt{|\delta\mathbf{q}|}\boldsymbol{\sigma}_{\text{motion}})$. Moreover, the initial configuration is not known accurately and we model the initial state uncertainty with a Gaussian distribution $\mathcal{N}(\mathbf{q}_0, \boldsymbol{\sigma}_{\text{init}})$ around a configuration $\mathbf{q}_0$ and variance $\boldsymbol{\sigma}_{\text{init}}$.

Due to uncertainty, 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 configurations. Each belief state is associated with a fully observable contact state $C = \{c_1, \ldots, c_k\}$ and a contact state $c = (\text{surface}_{\text{robot}}, \text{surface}_{\text{EC}}, \hat{\boldsymbol{n}})$ is a pair of robot and environment surfaces and the EC's contact normal. A belief b is in a task-relevant space if it lies mostly in the task-relevant configuration space:

$$\int_{\mathbf{q} \in \mathcal{C}_{\text{task}}} b(\mathbf{q}) d\mathbf{q} > 1 - \epsilon, \tag{3.1}$$

and $\mathcal{B}_{\text{task}}$ is the space of all task beliefs.

The planning problem is now the following: given a start and goal belief $b_0, b_g \in \mathcal{B}_{\text{task}}$, approximate the task-relevant configuration space $\widetilde{\mathcal{C}}_{\text{task}} \subset \mathcal{C}_{\text{task}}$ and search for a policy $\pi : \widetilde{\mathcal{B}}_{\text{task}} \rightarrow \mathcal{U}$ that brings the robot from the start to the goal belief state.

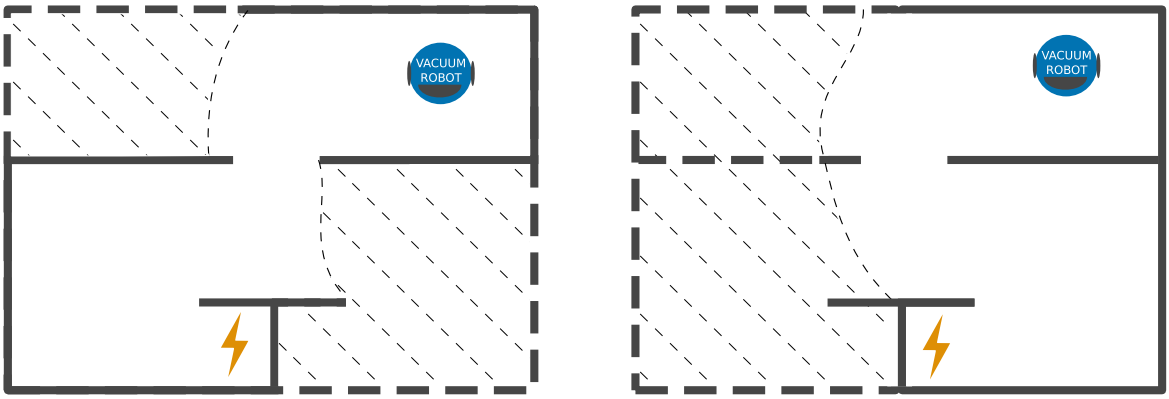


Figure 3.2 The two sketches illustrate the difference between task-relevant regions for different goals. In both sketches, a mobile robot must go to its charging station. Depending on the charger's location, some parts of the workspace are less relevant to visit while moving toward the charger.

3.2 Contact Exploring Exploiting Tree Planner

Following the previous problem definition, we present the Contact Exploring Exploiting Tree (CEET) planner, a sampling-based motion planner that finds a robust motion trajectory against perception and motion uncertainties. Before presenting the algorithm, we first explain two essential insights for understanding our planner.

3.2.1 Limiting the Search Space to Task-Relevant Regions

The first insight is related to the fourth property of ECs explained in Section 1.2, namely that ECs implicitly structure the workspace into regions. Some regions are more relevant for a given task, while other regions are less relevant. Thus, we leverage a workspace structure to extract task-relevant regions for a given task. We reduce the search space by searching for a motion plan in task-relevant regions. As a result, planning becomes more efficient because the planner avoids exploring less useful parts of the environment.

We illustrate the difference between task-relevant and less useful regions in Figure 3.2 with two instances of a 2D planning problem where a robot needs to go to its charging station. Intuitively, we can split the workspace into a task-relevant space where the charger can be directly reached and a space that the robot should not enter if its battery is low. By dashing the less relevant parts of the workspace (free space and surfaces), it is visible that the remaining task-relevant region depends on the charger's

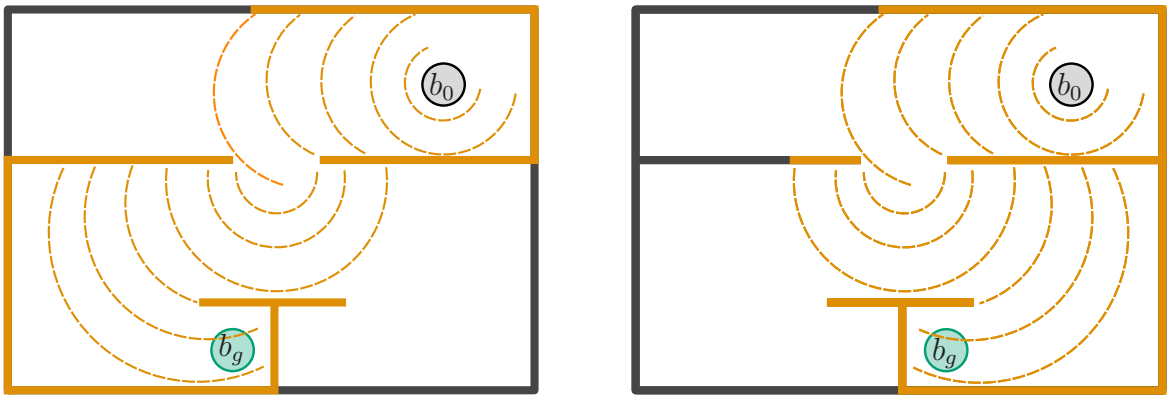


Figure 3.3 Both sketches illustrate how wavefront expansion can identify task-relevant free and contact regions in a two-dimensional navigation problem where a robot moves from b_0 (gray) to b_g(green).

location. If the robot's initial position changes, the task-relevant region also changes. The question is how to identify task-relevant free- and contact-space for planning.

Task-relevant and free space regions can be efficiently computed by decomposing a robot's workspace using a wavefront planner (Rickert et al., 2014). Task-relevant regions are computationally less expensive to obtain from workspace decomposition than from $\mathcal{C}$-space because the workspace is generally lower dimensional than $\mathcal{C}$-space. Moreover, ECs have a simpler representation in the workspace than in $\mathcal{C}$-space. Hence, we can efficiently compute task-relevant contact regions as well. Suppose the waves are only expended toward the goal. In that case, the waves cover a task-relevant free space region and touch task-relevant surfaces, as illustrated in Figure 3.3. Such workspace information can guide $\mathcal{C}$-space exploration.

Workspace information can guide $\mathcal{C}$-space exploration because a joint-space configuration can be moved efficiently toward a workspace pose when the robot's kinematic model is known. With a robot's forward kinematic model, we can efficiently compute its Jacobian to move a configuration toward a pose. Moreover, we can also efficiently compute a robot's pose from its configuration to check if a configuration is in a task-relevant region of the workspace.

We use a sphere-based wavefront expansion in workspace (Brock and Kavraki, 2001) to approximate the task-relevant free space region and surfaces. First, we compute a tree of workspace spheres in free space from a given start until the goal. Then, we extract a sequence of spheres from start to goal and save its neighboring surfaces and contact normals for each sphere, as shown in Figure 3.4. The extracted *spheres* approximate a task-relevant sub-space in the robot's workspace and configurations within these spheres approximate $\widetilde{\mathcal{C}}_{\text{task}}$ as well. With the *neigh-*

boring surfaces of these spheres, $\widetilde{\delta\mathcal{C}}_{\text{task}}$ can be approximated by projecting $\widetilde{\mathcal{C}}_{\text{task}}$ on the respective surfaces. By knowing the *sequence* of spheres between the start and goal, we also know the direction in which a robot should move within a sphere.

We approximate $\widetilde{\mathcal{C}}_{\text{task}}$ and $\widetilde{\delta\mathcal{C}}_{\text{task}}$ by defining two potential fields. One potential field drags a robot from one sphere to the next one. The other field drags the robot into contact with a neighboring surface of a sphere. If the robot is already in contact with a surface, we combine the two potential fields to slide toward the next sphere while maintaining contact. However, purely exploiting potential fields might not always work due to a local minima or kinematic limitations of a robot.

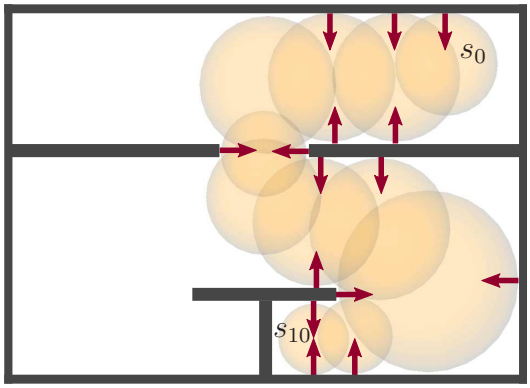


Figure 3.4 A sequence of ten spheres approximating the task-relevant free space region and surface normals (arrows) indicating nearby surfaces of the spheres.

When pure exploitation of workspace information fails, we are going to shift from exploitation to exploration of the configuration space. Next, we explain our second insight about the relation between finding robust actions and state uncertainty.

3.2.2 Configuration Space Connectivity Depends on Accumulated Uncertainty

The second key insight concerns how state uncertainty affects finding robust actions. With motion planning under uncertainty, robust actions become spares (or nonexistent) depending on state uncertainty and locally imposed constraints. Uncertainty affects actions when it is larger than the maximum amount of uncertainty allowed for which the action avoids undesired collisions. The maximum uncertainty depends on locally imposed constraints by the environment, and state uncertainty results from error accumulation over time while the robot moves.

Figure 3.5 illustrates uncertainty's temporal and local nature. When motion error accumulates slowly, a belief state's uncertainty is low enough to pass the first narrow passage and only hinders progress before reaching the goal. In contrast, when motion error accumulates rapidly, the belief state's uncertainty is larger than the first narrow passage. In both cases, uncertainty can be reduced with contact-exploiting actions near the respective narrow passages.

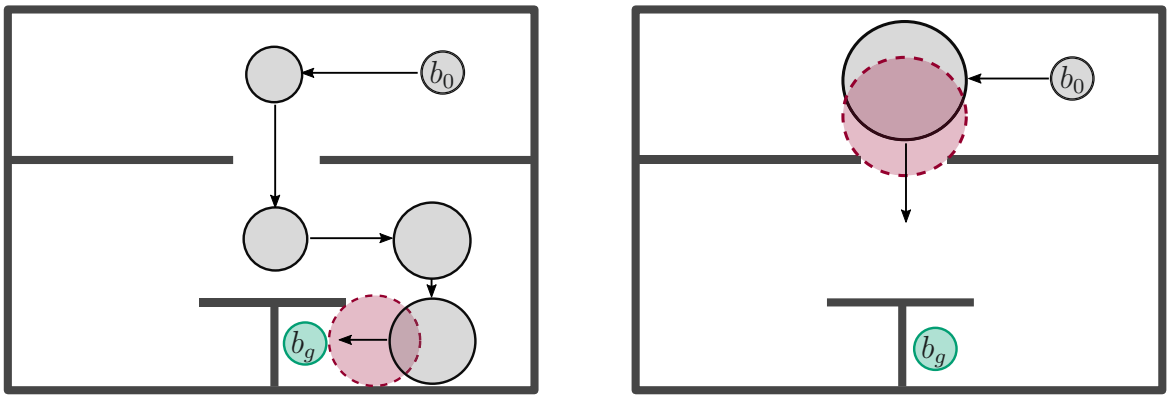


Figure 3.5 The two sketches illustrate how different amounts of motion error and resulting state uncertainty close narrow passages at a different location in a two-dimensional navigation problem. A point mass robot goes from b_0 toward b_g while its state uncertainty increases (gray circles) until its belief state (red circle) does not fit through a narrow passage.

We want to reduce belief state uncertainty only when needed because no free lunch with contact-exploiting motions exists. On the one hand, contact-exploiting motions reduce uncertainty, which can improve finding robust actions. On the other hand, contact-exploiting actions require more frequent collision checking than motion in free space because contact must be maintained without penetration. To reduce collision checking and improve planning efficiency, we start by exploring $\widetilde{\mathcal{C}}_{\text{task}}$ and shift to $\widetilde{\delta\mathcal{C}}_{\text{task}}$ when progress is hindered.

3.2.3 Adaptive Guided Exploration Behavior

We balance exploitation and exploration, and we balance $\widetilde{\mathcal{C}}_{\text{task}}$ and $\widetilde{\delta\mathcal{C}}_{\text{task}}$ exploration similarly. The balancing strategy is similar because we *shift* from exploitation to exploration and also *shift* from exploring $\widetilde{\mathcal{C}}_{\text{task}}$ to $\widetilde{\delta\mathcal{C}}_{\text{task}}$. We initially exploit workspace information for the former to quickly reach the next sphere. We start exploring an alternative path to the next sphere if pure exploitation fails. For the latter, we initially explore $\widetilde{\mathcal{C}}_{\text{task}}$ because it requires less computation than $\widetilde{\delta\mathcal{C}}_{\text{task}}$. If we cannot find a path to the next sphere through $\widetilde{\mathcal{C}}_{\text{task}}$, we start exploring $\widetilde{\delta\mathcal{C}}_{\text{task}}$ as well to reduce uncertainty.

Observe that we start shifting from exploitation to exploration and from exploring $\widetilde{\mathcal{C}}_{\text{task}}$ to $\widetilde{\delta\mathcal{C}}_{\text{task}}$ for the same reason, i.e., when progress toward the next sphere is hindered. Therefore, we measure the lack of progress with a single variable $\beta \in [0,\ 1]$ and use its value to balance both exploitation and exploration, and exploring $\widetilde{\mathcal{C}}_{\text{task}}$ and $\widetilde{\delta\mathcal{C}}_{\text{task}}$. We illustrate the use of β in Figure 3.6 and explain it below.

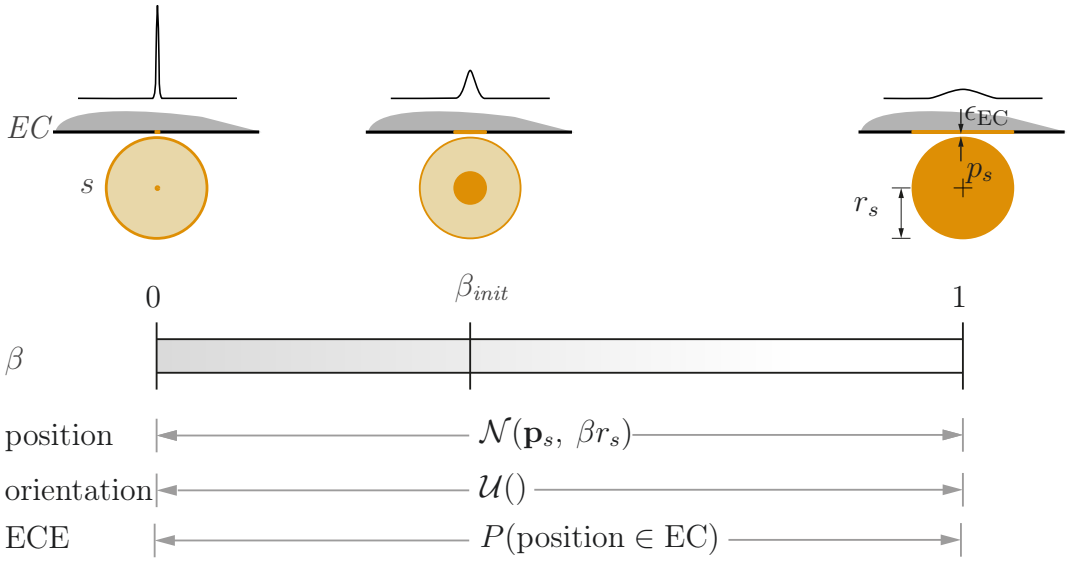


Figure 3.6 The value of β balances a) exploitation and exploration of workspace information and b) free and contact space exploration. a) to balance exploitation and exploration, a pose sample is drawn from a sphere s (orange circles) where the position is from a normal distribution with variance proportional to β and the orientation is from a uniform distribution. b) to balance free and contact space exploration, the previously drawn sample is projected with β probability to a surface within ϵ_{EC} distance away from s.

With $\beta = 0$, the planner purely exploits workspace information and behaves like a potential field planner using a global navigation function approximated with the sequence of spheres S. Since the planner drags a robot toward the next sphere's center $\mathbf{p}_s$ that is in free space, we can consider it as an "exploration" of $\widetilde{\mathcal{C}}_{\text{task}}$, even though in this edge case the planner does exploitation and not exploration.

When pure exploitation fails, β increases, and exploration starts. Exploration is realized by sampling a pose close to the sphere's center s. To draw a sample, we sample an orientation from a uniform distribution and sample a position from a Gaussian distribution $\mathcal{N}(\mathbf{p}_s, \beta r_s)$ around the centers of a sphere $\mathbf{p}_{s \in S}$ with variance proportional to β. Hence, β indicates how closely the planner follows (probabilistically) the potential field defined by task-relevant workspace information. At the same time, β balances $\widetilde{\mathcal{C}}_{\text{task}}$ and $\widetilde{\delta\mathcal{C}}_{\text{task}}$ exploration by projecting the previously drawn sample onto a randomly selected neighboring surface with β probability.

When $\beta = 1$, samples are drawn with high probability anywhere inside a sphere and projected on a neighboring surface. If β is further increased, more samples would be drawn outside the sphere, indicating that the planner could not reach the next sphere and it is stuck.

We propose two strategies to unstuck the planner, backtracking to the previous sphere or randomly exploring $\mathcal{C}_{\text{valid}}$ until a new sphere is reached, and explain them in more detail in the next section.

3.2.4 Algorithm Outline

The CEET planner presented in Algorithm 2 is an RRT-based planner that builds a belief space tree in $\mathcal{C}$-space. Every node of the tree T is a belief b. A belief b is approximated with a set of particles $b = \mathcal{Q} = \{\mathbf{q}_1, ..., \mathbf{q}_N\}$, where each particle $\mathbf{q}_{i=1..N}$ is a configuration. We use a particle-based representation because a belief state becomes a non-Gaussian distribution of configurations when projected on a contact manifold. We denote the mean configuration $\boldsymbol{\mu}_b$, the covariance $\boldsymbol{\Sigma}_b$, and the workspace tool frame $\mathbf{T}_b$ for $\boldsymbol{\mu}_b$. The tool frame is the end-effector frame in the case of articulated robots or an arbitrary point and orientation in the case of rigid-body robots.

The algorithm initializes the tree with b_0 as the root and computes the sequence of spheres S and associated EC. Then, in every iteration, it expands a tree node toward the next sphere s_{next} (line 7). If the resulting belief b_{new} is valid, it is added to the tree (lines 9 and 10). A belief is valid if all particles have the same contact state, respect joint limits, and avoid collisions with robot parts that do not sense contact. Next, the algorithm tries to connect b_{new} and b_{g}, and upon success, returns the tree (lines 11 and 12). Otherwise, it checks if a new sphere (e.g., the next sphere or one even closer to the goal) is reached. When a new sphere is reached, it updates s, resets $\beta = \beta_{\text{init}}$, and initialize the stuck counter k_s with zero (lines 14-16). However, if b_{new} is not valid or a new sphere is not reached, we increase β linearly (line 18) or exponentially (line 20), respectively, to shift toward exploration and to use more contact-exploiting actions. We increase β more when expansion fails because it is a stronger indication that uncertainty hinders progress.

If $\beta > 1$ at the end of the while loop, the planner reaches the exploration limit. Hence, it resets β and increases the stuck counter k_s (lines 22 and 23). The number of times we explore the same sphere depends on the size of the current sphere relative to the next one. Moving from a smaller sphere toward a larger one is like exiting a narrow passage. In this case, we hope to have more feasible paths than entering a narrow passage. Thus we restart the exploration of the same sphere multiple times. In contrast, we restart exploration fewer times when we go from a larger sphere to a smaller one (entreating a narrow passage). When k_s reaches this adaptive threshold, the planner backtracks to the parent sphere (line 25).

Backtracking is one way to escape a local minimum. Another alternative is to explore $\mathcal{C}_{\text{valid}}$ randomly until a new sphere is reached. We are going to evaluate both backtracking with the CEET_b planner and random exploration with CEET_e in Section 3.3.

Algorithm 2 CEET motion planner

Input: b_0, b_g
Output: $T = (V, E)$

1: $V \leftarrow \{b_0\}$ *initialize tree*
2: $S, EC \leftarrow \text{WAVEFRONT}(\mu_{b_0}, \mu_{b_\text{g}})$ *compute sphere tree and associated ECs*
3: $s \leftarrow S_\text{begin}$ *take first sphere*
4: $\beta \leftarrow \beta_\text{init}$ *initialize exploration-exploitation balance*
5: $k_s \leftarrow 0$ *initialize sphere exploration counter*
6: **while** true **do**
7: $\quad b_\text{near}, b_\text{new} \leftarrow \text{EXPAND}(V, s_\text{next}, EC_{s_\text{next}} \cup EC_s, \beta)$ *expend one node in the tree*
8: $\quad$ **if** IS_VALID(b_new) **then**
9: $\quad\quad V \leftarrow V \cup \{b_\text{new}\}$ *add new node to the tree*
10: $\quad\quad E \leftarrow E \cup \{(b_\text{near}, b_\text{new})\}$ *add new edge to the tree*
11: $\quad\quad$ **if** GOAL_CONNECT(b_new, b_g) **then**
12: $\quad\quad\quad$ **return** T *the goal is reached from the new belief*
13: $\quad\quad$ **if** $\mathbf{T}_{b_\text{new}} \in s_\text{unvisited} \ \vee \ s_\text{unvisited} \in S$ **then** *new belief reached an new sphere*
14: $\quad\quad\quad s \leftarrow s_\text{unvisited}$ *update current sphere*
15: $\quad\quad\quad \beta \leftarrow \beta_\text{init}$ *reset exploration-exploitation balance*
16: $\quad\quad\quad k_s \leftarrow 0$ *reset sphere exploration counter*
17: $\quad\quad$ **else**
18: $\quad\quad\quad \beta \leftarrow \beta(1+\alpha)$ *increase exploration when sphere is not reached*
19: $\quad$ **else**
20: $\quad\quad \beta \leftarrow \beta(1+\alpha)^2$ *increase exploration when node expansion failed*
21: $\quad$ **if** $\beta > 1$ **then** *exploration limit reached*
22: $\quad\quad \beta \leftarrow \beta_\text{init}$ *reset exploration-exploitation balance*
23: $\quad\quad k_s \leftarrow k_s + 1$ *increase the stuck counter for s*
24: $\quad\quad$ **if** $r_\text{next} + r_s \geq r_\text{next} * (1+\alpha)^{k_s}$ **then**
25: $\quad\quad\quad s \leftarrow s_\text{parent}$ *backtrack to parent sphere*
26: $\quad\quad\quad k_s \leftarrow 0$ *reset sphere exploration counter for parent sphere*

Guided exploration is realized in the EXPAND method, which has steps similar to the basic RRT planner, but these steps are radically different. Below, we give implementation details for each step, where the numbers in parentheses refer to lines in Algorithm 3.

Algorithm 3 EXPAND

Input: V, s, EC, β
Output: $b_{\text{near}}, b_{\text{new}}$
1: $\mathbf{T}_{\text{sample}} \leftarrow \text{SAMPLE}(s, r_s, \beta)$ *sample target pose from s and its ECs*
2: $b_{\text{near}} \leftarrow \text{NEAREST\_NEIGBOUR}(V, \mathbf{T}_{\text{sample}})$ *find nearest node to sampled pose*
3: $\mathbf{q}_{\text{target}},\ u \leftarrow \text{STEER}(\boldsymbol{\mu}_{b_{\text{near}}}, \mathbf{T}_{\text{sample}}, \hat{\boldsymbol{n}}_{EC}, \beta)$ *pull robot toward sampled pose*
4: $b_{\text{new}} \leftarrow \text{SIMULATE}(b_{\text{near}}, \mathbf{q}_{\text{target}}, u)$ *forward propagate b_{near} toward q_{target} with action u*
5: **return** $b_{\text{near}},\ b_{\text{new}}$

Sampling expansion direction (line 1): We sample a frame $\mathbf{T}$ inside a sphere s:

$$\mathbf{p}_{\text{sample}} = \mathbf{p}_s + \mathcal{N}(0, \beta r_s),$$
$$R_{\text{sample}} = \mathcal{U}(),$$

where the position $\mathbf{p}_{\text{sample}}$ is a sample from a Gaussian distribution with center at the sphere $\mathbf{p}_s$ and βr_s variance, and the orientation R is sampled from a uniform distribution $\mathcal{U}()$. We project the sample on a randomly selected EC with β probability. Thus, a large value of β increases a sphere's exploration and the use of contact-exploiting action.

Selecting nearest neighbor (line 2): To find the nearest neighbor to a sample, we use a combined distance metric:

$$b_{near} = \underset{b}{argmin}\,[\gamma d_{\mathbf{\Sigma}}(b) + (1-\gamma) d_{\mathbf{T}}(b)]$$

where $d_{\mathbf{T}}(b) = \|\mathbf{T}_b - \mathbf{T}_{\text{sample}}\|_2$ is the spatial distance between the sampled frame and the end-effector frame of a belief b, and $d_{\mathbf{\Sigma}}(b) = \sqrt{\text{tr}(\mathbf{\Sigma}_b)}$ is the square-root of the trace of the belief's covariance metrics, and γ balances the cost of state uncertainty. If γ is closer to zero, the metric penalizes more for a larger spatial distance than state uncertainty. If it is closer to one, beliefs with increased uncertainty are more penalized than those with a larger spatial distance. In other words, a large value of γ enforces the planner to expand only low uncertainty nodes in the three.

Steering toward the next sphere (line 3): The STEER algorithm (Algorithm 4) drags a predefined frame on the robot toward the previously sampled frame $\mathbf{T}_{\text{sample}}$. First, we choose between a CONNECT or SLIDE action based on the initial contact state of b_{near} and whether the potential field breaks contact or allows sliding. We simulate an action by moving the robot's frame in a straight line using the robot Jaco-

Algorithm 4 STEER

Input: $\mathbf{q}_{\text{near}}, \mathbf{T}_{\text{sample}}, \hat{\boldsymbol{n}}$
Output: $\mathbf{q}_{\text{sample}}, u$

```
if q_near ∈ C_free ∨ |n̂ × POTENTIAL(T_sample, f(q_near))| < ε_angle then
    u ← CONNET   nearest is in free space or potential field breaks contact
else
    u ← SLIDE   nearest is in contact and potential field maintains contact initially
q_new ← q_near
while |p_new − p_sample| < ε_position do
    Δx ← T_near − T_sample
    Δq ← J*(q_near)Δx
    q′ ← q_new + Δq
    if SINGULARITY(q′) then
        q′ ← RAND()
        q′ ← INTERPOLATE(q_new, q′, |Δq|)
    if u = CONNET ∧ q′ ∈ C_obstacle then   motion in free space ends in contact
        return q_new, GUARDED-MOVE   the action becomes a guarded move
    if u = SLIDE ∧ C_q′ ≠ C_q_sample then   sliding motion ends in a different contact state
        q″ ← BACK-PROJECT(q′, n̂)   back project to the target surface
        if C_q″ ≠ C_q_sample then   back projection failed or new contact gained
            return q″, GUARDED-SLIDE   the action becomes a guarded slide
        else
            q′ ← q″
    q_new ← q′
return q_new, u
```

bian. With sliding actions, we maintain contact with a surface using task projections, as presented in Section 2.3. If the contact state changes after a simulation step, we update u with the action needed to reach that state. A CONNECT action changes to GUARDED-MOVE for a new contact, and SLIDE changes to GUARDED-SLIDE if the initial contact state of b_{near} changes. The algorithm returns the reached configuration and the associated action if the action changed during simulation or if the target is reached with $\epsilon_{\text{position}}$ precision.

Simulating belief propagation (4): The final step forward propagates b_{near} toward the target configuration $\mathbf{q}_{\text{target}}$ by simulating action u with a noisy motion model for each particle in the belief .

In summary, our planner guides exploration using workspace information to improve planning efficiency. On the one hand, the planner reduced the searched configuration space by approximating $\widetilde{\mathcal{C}}_{\text{task}}$ and $\widetilde{\delta\mathcal{C}}_{\text{task}}$ using the spheres in the SAMPLE method.

On the other hand, it reduced the action space by using spheres connectivity in the STEER method. Both methods removed less useful $\mathcal{C}$-space regions and actions for a motion planning problem. Consequently, these reductions increased the planner's efficiency in finding a solution. To analyze the benefits of these reductions, we present two baselines that search the entire action space and/or reduce the configuration space.

3.2.5 Baseline Algorithms

Before evaluating the CEET planner, we present two baseline algorithms: the CERRT planner (Sieverling et al., 2017b) and a new Contact Exploring Tree (CET) planner. The two baselines differ from CEET by the amount of workspace information used during planning. CERRT uses goal bias with a 10% sampling probability without using other workspace information. In contrast, CET uses 10% goal bias and workspace information to reduce the configuration space to task-relevant regions but without reducing the action space. Thus, CERRT uses the least amount of information, CET uses workspace information to reduce $\mathcal{C}$-space, and CEET uses the most information to reduce both $\mathcal{C}$-space and the action space.

Both baseline planners are RRT-based and build a belief tree from a start belief to a goal region. We provide a single pseudo code for the two baselines because a small modification of the CERRT planner results in the novel CET planner.

With Algorithm 5, we can obtain the CERRT planner for $\mathcal{W}_{\text{explore}} = \mathcal{W}$ as input and the CET planner for $\mathcal{W}_{\text{explore}} = S$ as input, where $\mathcal{W}$ denotes the entire workspace and S is the workspace volume covered by a sequence of spheres pre-computed as done in the CEET planner. The sampler (line 3) returns a configuration for which a robot's tool frame is inside $\mathcal{W}_{\text{explore}}$. Hence, it reduces the search space to $\widetilde{\mathcal{W}}_{\text{task}}$ when the workspace is limited to S. The nearest neighbor selection (line 4) is similar to the one in Algorithm 3 but uses the euclidean join distance between configurations rather than the spacial distance between frames. Actions are sampled (line 5) based on the contact state of b_{near} and with γ probability to be a contact-exploiting action. Then, b_{near} is forward projected (line 6) as done in Algorithm 3. Finally, if the resulting $b_{\text{near}} \in \mathcal{C}_{\text{valid}}$, the planner greedily tries to connect to the goal, and upon success, it returns the tree, otherwise continues the search. For more details about this CERRT algorithm, we refer to the respective publication (Sieverling et al., 2017b).

Algorithm 5 CERRT or CET planner depending on $\mathcal{W}_{\text{explore}}$ input parameter

Input: $b_0, b_g, \mathcal{W}_{\text{explore}}$
Output: $T = (V, E)$
1: $V \leftarrow \{b_0\}$ *initialize tree with start belief*
2: **while** true **do** *search until goal reached*
3: $\mathbf{q}_{\text{rand}} \leftarrow \text{SAMPLE}(\mathcal{W}_{\text{explore}})$ *sample a random configuration in a workspace volume $\mathcal{W}_{explore}$*
4: $b_{\text{near}} \leftarrow \text{NEAREST\_NEIGHBOUR}(V, \mathbf{q}_{\text{rand}}, \gamma)$ *the closest belief to the sample in the tree*
5: $u \leftarrow \text{SELECT\_ACTION}(b_{\text{near}}, \gamma)$ *sample an action*
6: $b_{\text{new}} \leftarrow \text{SIMULATE}(b_{\text{near}}, \mathbf{q}_{\text{rand}}, u)$ *a local planner forward projects belief*
7: **if** $\text{IS\_VALID}(\text{b}_{\text{new}})$ **then**
8: $V \leftarrow V \cup \{b_{\text{new}}\}$
9: $E \leftarrow E \cup \{(b_{\text{near}}, b_{\text{new}})\}$
10: **if** $\text{GOAL\_CONNECT}(b_{\text{new}}, b_g)$ **then**
11: **return** T

3.3 Evaluation of the CEET Planner

We evaluated the CEET planner in simulation to show that a) reducing the configuration and the action spaces to task-relevant regions significantly improves planning efficiency and b) our planner scales to high-dimensional configuration space planning problems. Moreover, we analyzed the proposed heuristics to unstuck the planner and the balancing parameters for different amounts of motion errors.

We implemented all experiments using the Robotics Library (Rickert and Gaschler, 2017) and carried out the experiments on a computational cluster. All experiments in Section 3.3.1 ran on Intel(R) Xeon(R) E5-2630 v2 @ 2.60GHz CPUs, and the rest of the experiments ran on Intel(R) Xeon(R) E5540 @ 2.53GHz CPUs. We ran ten experiments for each data sample and listed the planning parameters for each experiment in Table 3.1.

3.3.1 Configuration Space Reduction Increase Planning Efficiency for Large 2D Spaces

We want to show that leveraging task-relevant regions to guide exploration improves a planner's computational efficiency. Therefore, we compared the planning duration of CERRT, CET, CEET_e and CEET_b on three 2D planning problems with increasing $\mathcal{C}$-space volumes.

In all three planning problems (top row in Figure 3.7), a 2-DOF gripper had to move from $\mathbf{q}_0$ to $\mathbf{q}_g$ using only translational motion under initial position and motion

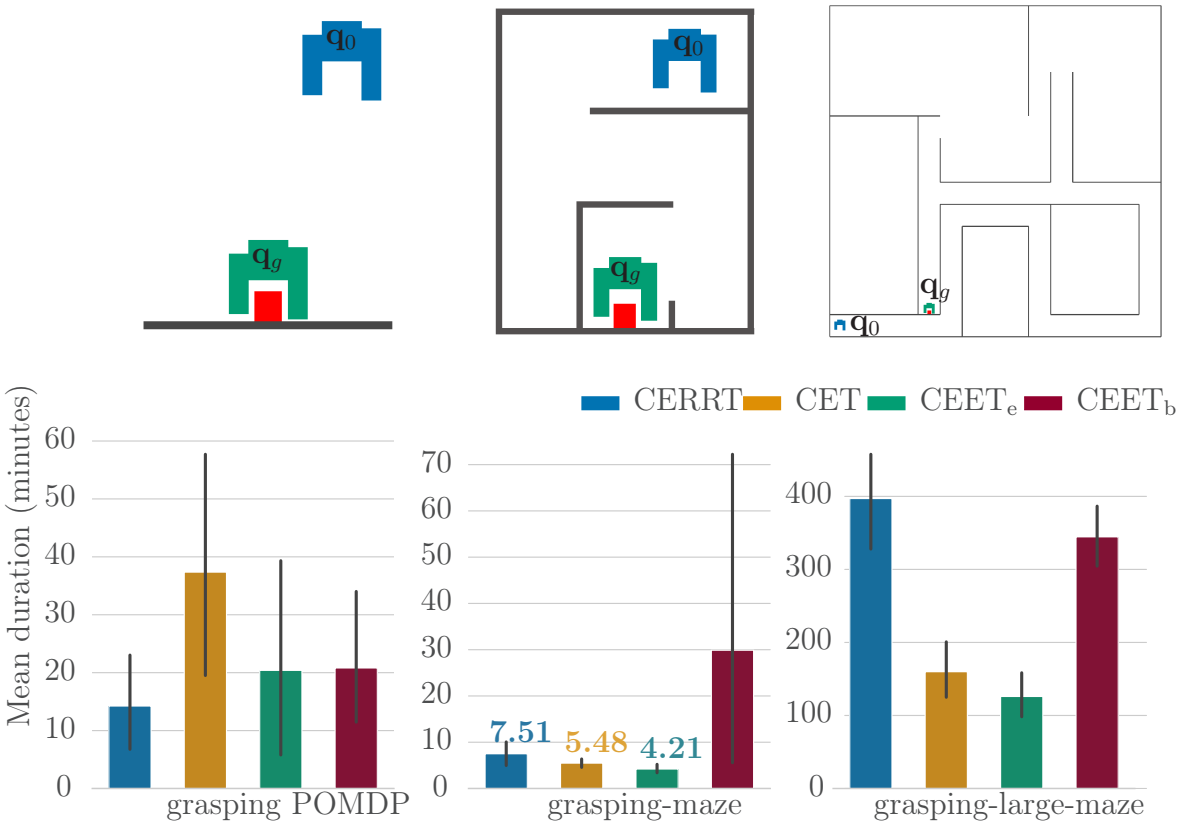


Figure 3.7 Planning efficiency improves significantly when workspace information is used in problems with large free and contact spaces that are less relevant for a task. *Top row:* three planning problems where a 2-DOF gripper had to move from $\mathbf{q}_0$ to $\mathbf{q}_g$ under initial position and motion uncertainty. The problem complexity increases from left to right by increasing the $\mathcal{C}$-space volume and the complexity of the environment. *Bottom row:* comparing mean planning duration of four planners with increasing use of workspace information: CERRT randomly explores $\mathcal{C}_{\text{valid}}$, CET randomly explores $(\widetilde{\mathcal{C}}_{\text{task}} \cup \widetilde{\delta\mathcal{C}}_{\text{task}}) \subset \mathcal{C}_{\text{valid}}$, CEET$_e$, CEET$_b$ is restricted to explore $\widetilde{\mathcal{C}}_{\text{task}} \cup \widetilde{\delta\mathcal{C}}_{\text{task}}$ by backtracking when it is stuck, while CEET$_e$ can explore beyond the spheres because it performs random exploration when stuck. By comparing CEET$_b$ and CEET$_e$, our task-relevant region approximation appears too restrictive for the maze problems.

Param.	**Description**	**2-DOF gripper**	**7-DOF WAM**	**10-DOF Uman**
t [min]	time budget	500	1000	1500
N	number of particles	20	20	20
δ_{step}	simulation step size	0.05	1.0	1.0
ϵgoal	goal region	0.1	0.05	0.05
γ	weighting factor in the composite distance metric	0.3	0.3	0.5
β_{init}	initial exploration and exploitation balance	0.1	0.3	0.1
α	rate of β increase	0.1	0.01	0.1
$\boldsymbol{\sigma}_{\text{init}}$	initial uncertainty	$[0.2, 0.2]$	$0.02 * \mathbf{1}^7$	$0.02 * \mathbf{1}^10$
$\boldsymbol{\sigma}_{\text{motion}}$	motion uncertainty	$[0.2, 0.2]$	$[0.02 * \mathbf{1}^6, 0.0]$	$[0.02 * \mathbf{1}^9, 0.0]$

Table 3.1 Planning parameters, where $\mathbf{1}^n$ is an n dimensional vector with ones and $[\mathbf{1}^n, x]$ is the concatenation of $\mathbf{1}^n$ with a scalar x resulting in an $n+1$ dimensional vector.

uncertainty. The first planning problem is the *grasping POMDP* proposed by Hsiao et al. (2007), and the other two problems, grasping-maz, and grasping-large-maze, are variations with increasing complexity. First, we increased the volume of less relevant regions from 0% to 13% and 81% for the grasping POMDP, grasping-maze, and grasping-large-maze problems, respectively. Secondly, we increased the environment's geometrical complexity by increasing the number of less relevant surfaces and the total number of surfaces from 0/5 to 3/21 and 32/51.

We expect that leveraging task-relevant workspace regions decreases planning duration as the $\mathcal{C}$-space volume and the number of surfaces increase.

The bottom row in Figure 3.7 shows the mean planning duration and the 95% confidence interval on each bar. The results in the two maze problems show that $\mathcal{C}$-space reduction significantly improved planning efficiency, but action space reduction only provided minor improvement. However, the grasping POMDP problem results show that random exploration was better when the whole workspace was task-relevant. This is because the other three planners approximated $\widetilde{\mathcal{C}}_{\text{task}}$ too restrictively so that the reduced search spaces contained fewer solutions making planning more difficult.

The planners found a solution for all three problems under the given time budget, except the CERRT planner for the large maze. CERRT failed two times out of ten in the large maze because it spent too much time exploring the two large openings at the top. In contrast, CET and CEET$_{\text{b}}$ only explored the corridors directly leading to the goal region without ever exploring the top regions of the maze. Even though

tough CEET_e was not restricted to searching less relevant regions when escaping a local minimum, the planner surprisingly found a solution faster than CET and CEET_b in the maze problems.

In the two maze problems, CEET_e outperformed CEET_b significantly. Note that the two planners differ in the local-minimum escaping heuristics, where CEET_e used random exploration and CEET_b backtracked to the parent sphere. The random exploration heuristic was better than backtracking because the sphere-based task-relevant region approximation was too restrictive for 2-DOF problems. In the next section, we further analyzed the two heuristics for higher dimensional $\mathcal{C}$-spaces.

3.3.2 The CEET Planner Scales to High-Dimensional Problems

We want to show that the CEET planner scales to high-dimensional configuration space planning problems. Hence, we compared the planning duration of CERRT, CET, CEET_b, and CEET_e planners on two planning problems with increasing $\mathcal{C}$-space dimensions: using a 7-DOF or 10-DOF robot, as illustrated at the top in Figure 3.8. With 7-DOF, the Barrat WAM arm had to move its end-effector from free space into a deepening on a wall. With 10-DOF, the Uman robot, composed of a holonomic mobile-based and the WAM arm, had to remove its arm from a window on a wall and to insert it into another window on the same wall.

We expect significant planning efficiency improvement when leveraging workspace information to reduce the configuration and action spaces compared to planners that explore beyond task-relevant regions.

The bottom row in Figure 3.8 shows the mean planning duration and the 95% confidence interval on each bar. The results show that CEET scaled to high-dimensional problems when using the proper escaping heuristic. With the backtracking heuristic, CEET_b was the fastest in the 7-DOF problem, and it was the only one that solved the 10-DOF problem with an 80% success rate, while the other three planners could not find a solution within the given time budget. This indicates that backtracking was a better heuristic to unstuck the planner in high-dimensional $\mathcal{C}$-space.

Interestingly, CEET_b was better performing in high-dimensional $\mathcal{C}$-spaces than CEET_e, and the other way around in low-dimensional spaces. We think this is because backtracking reduces the search space compared to random exploration, which opens it up fully. In high-dimensional search spaces, our approximation of $\widetilde{\mathcal{C}}_\text{task} \cup \widetilde{\delta\mathcal{C}}_\text{task}$ potentially contained multiple valid paths. By backtracking in a high-dimensional

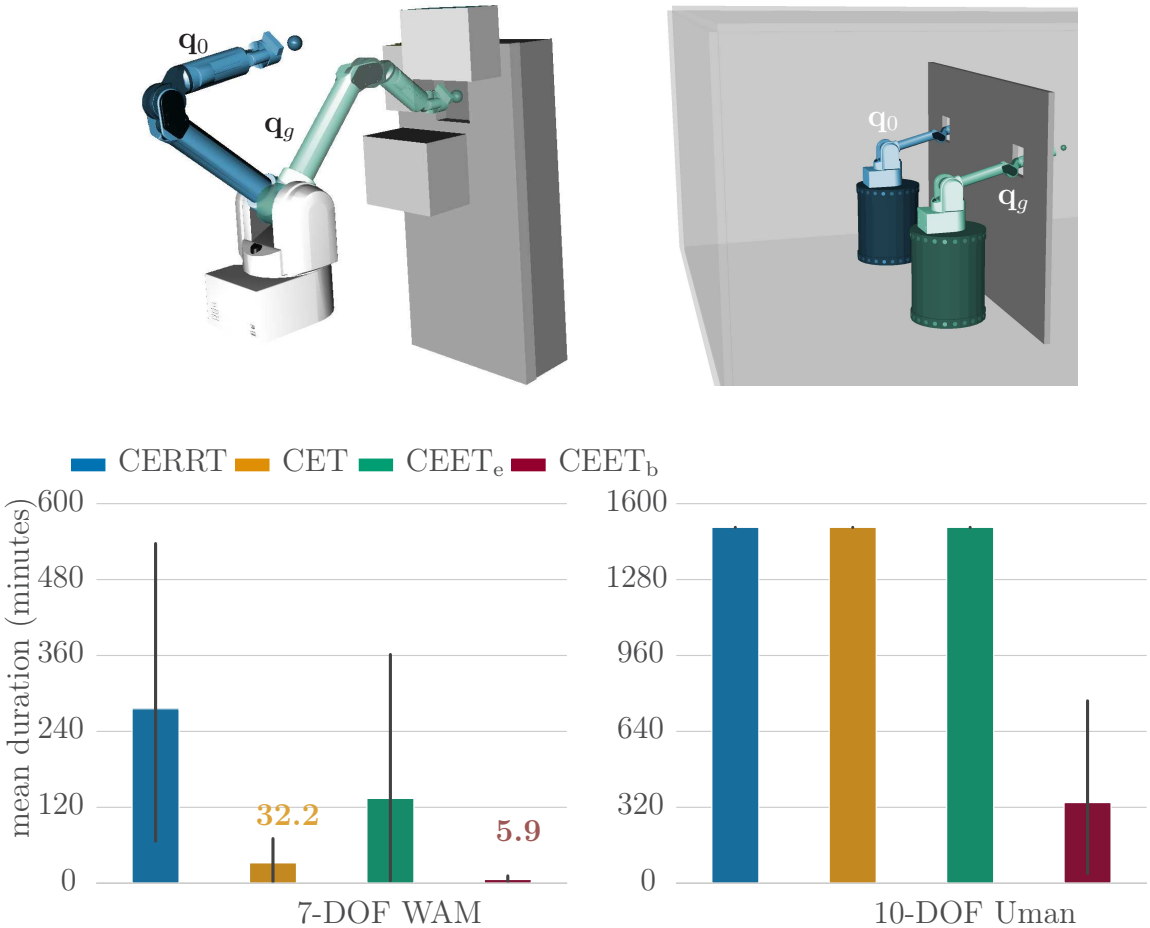


Figure 3.8 CEET generalizes to high-dimensional configuration spaces, and the backtracking heuristic is better than random exploration to *unstuck* the CEET planner. *Top row:* two planning problems with increasing $\mathcal{C}$-space dimensionality: a 7-DOF WAM arm is in front of a wall or a 10-DOF Uman mobile robot in front of another wall. The tasks are similar, reaching $\mathbf{q}_g$ from $\mathbf{q}_0$ under initial position and motion uncertainty. *Bottom row:* comparing planning duration of the four belief space motion planners on the two planning problems. Note that only CEET$_b$ was able to solve the 10-DOF problem with an 80% success rate.

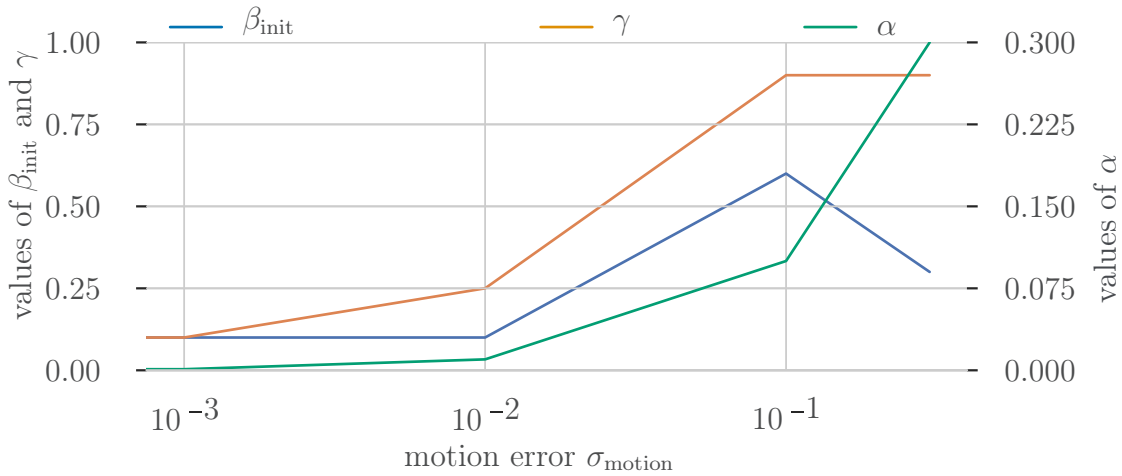


Figure 3.9 Best β_{init}, γ, and α parameters in function of the motion error σ_{motion} for the CEET_{b} planner on the 7-DOF WAM problem (top left image in Figure 3.8). As motion error increases, the parameters increased, indicating that the planner handled increased state uncertainty by shifting faster to exploration, using contact-exploiting actions (α and β_{init}) and by expanding less uncertain nodes (γ).

search space, the planner kept searching for these solutions in $\widetilde{\mathcal{C}}_{\text{task}} \cup \widetilde{\delta\mathcal{C}}_{\text{task}}$. It was like combing through a haystack multiple times to find a needle. This is not the case for low dimensional space, where our approximation of $\widetilde{\mathcal{C}}_{\text{task}} \cup \widetilde{\delta\mathcal{C}}_{\text{task}}$ was seemingly too restrictive. The reduced search space probably contained very few valid paths, while numerous solutions were outside this reduced search space. By using random exploration to unstuck the planner, we searched beyond the task-relevant regions and found those solutions. It was like searching for a needle in the grass beside the haystack.

3.3.3 Planner Parameters for Different Motion Error

We want to analyze the sensible values of α, β_{init}, and γ with respect to motion uncertainty because these parameters balance the exploitation and exploration of workspace information, and these also balance the exploration of $\widetilde{\mathcal{C}}_{\text{task}}$ and $\widetilde{\delta\mathcal{C}}_{\text{task}}$. Therefore, we ran a grid search of various parametrizations of the CEET_{b} on the 7-DOF WAM planning problem (first image in the first row in Figure 3.8), where $\beta_{\text{init}} \in \{0.1, 0.3, 0.6, 0.9\}$, $\alpha \in \{0, 0.001, 0.01, 0.1, 0.3\}$, $\gamma \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$, and the motion error was $\sigma_{\text{motion}} \in \{0, 0.001, 0.01, 0.1, 0.3\}$.

In Figure 3.9, we show the best parameter combination of α, β_{init}, and γ for which the planner needed the least amount of time to find a solution under different motion errors. Note that the x-axis has a logarithmic scale to better visualize the parameters for low motion error. The results show that as motion error increased, the values of the three parameters increased except β_{init} for the largest motion uncertainty $\sigma_{\text{motion}} = 0.3$. However, for the same σ_{motion}, the second-best parameter combination was $\alpha_{\text{motion}} = 0.3$ and $\beta_{\text{init}} = \gamma = 0.9$, where β_{init} increased compared to the case of $\sigma_{\text{motion}} = 0.1$. Moreover, the mean planning duration was less than 7% slower with the second-best parameter combination compared to the best parameterization for $\sigma_{\text{motion}} = 0.3$. Thus, we assume a positive correlation between σ_{motion} and α, β_{init},γ.

The positive correlation between the parameters and the motion error was expected because a larger motion error increases state uncertainty faster. So, the planner needs to reduce it more frequently. For larger α and β_{init} the planner shifted faster from exploring $\widetilde{\mathcal{C}}_{\text{task}}$ to $\widetilde{\delta\mathcal{C}}_{\text{task}}$. Consequently, it reduced state uncertainty faster. For larger γ, the planner weighted more state uncertainty than spatial distance when searching for the nearest neighbor. As a result, it preferred to extend three branches with less uncertainty.

Adaptive Distance Metric Evaluation

The previously presented results suggest that β_{init} and γ could have the same value for certain σ_{motion}. We want to further analyze the role of γ and whether it should be dynamically adjusted during planning similarly to β. Thus, we defined four heuristics for γ and labeled the CEET planner as follows:

CEET$^{\beta_{\text{init}}}$: $\gamma = \beta_{\text{init}}$, γ is constant during planning but it is adjusted to the motion error using the results from above.

CEET$^{\beta}$: $\gamma = \beta$, it is increasing as exploitation shifts to exploration during planning and resets when β resets. Since β increases if uncertainty hinders progress, γ becomes uncertainty dependent implicitly.

CEET^{k_s}: $\gamma = \beta_{\text{init}}(1 + \alpha)^{k_s}$, it is increasing as the stuck counter increase for a given sphere. This heuristic weighted uncertainty increasingly when the planner restarted the exploration of a sphere. Thus, γ is uncertainty dependent implicitly but increases slower than for CEET$^{\beta}$.

CEET^{r_s}: $\gamma = 1 - [r_{s_{\text{next}}}/\max(r_s|\forall s \in S)]$, it changes with respect to the relative size of the next sphere. With smaller spheres, uncertainty reduction is weighted more

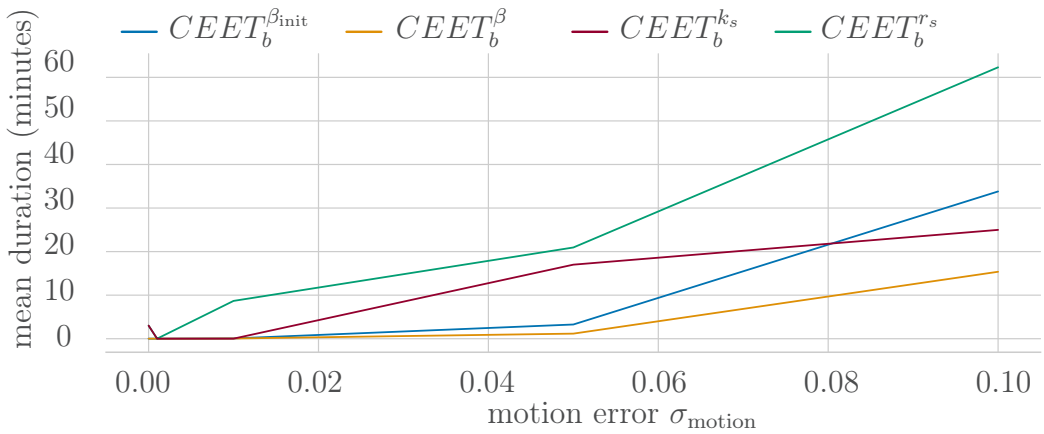


Figure 3.10 Comparing different distance metrics with a constant and three adaptive weighting of state uncertainty for the 7-DOF WAM planning problem with the CEET_b planner.

than for larger spheres because the smaller sphere could indicate a narrow passage. This heuristic assumes that entering a passage is more difficult than exiting one. In contrast to the previous heuristics, this only considers the environment's geometry represented by the spheres.

We evaluated the four heuristics on the 7-DOF WAM planning problem for a set of motion uncertainties $\sigma_{\text{motion}} \in \{0,\ 0.001,\ 0.005,\ 0.01,\ 0.1\}$ with the CEET_b planner. Figure 3.10 shows the mean planning duration for the four heuristics as motion error increases. All heuristics performed similarly for small motion errors. However, for large motion errors, heuristics that change in function of uncertainty (explicitly or implicitly) outperform the heuristic that only considers the environment's geometry.

3.3.4 Evolution of Balancing Parameters

Finally, we want to show the evolution of the β and k_s when executing the CEET_b planner on the grasping-maze 2-DOF planning problem to illustrate how these variables can indicate regions that are difficult to traverse. The left side in Figure 3.11 shows the sequence of spheres and the solution path for each particle. The top-right plot in the same figure shows when spheres were explored and when the planner performed backtracking, and the bottom-right plot zooms on the first two consecutive backtrackings to illustrate the evolution of β and k_s for $s \in \{12, 13\}$.

The planner triggered backtracking the first time while trying to reach the goal in s_{13}, and then it triggered backtracking from s_{12} as well. The top plot clearly shows that the motion planning problem has an extremely difficult region in the last few

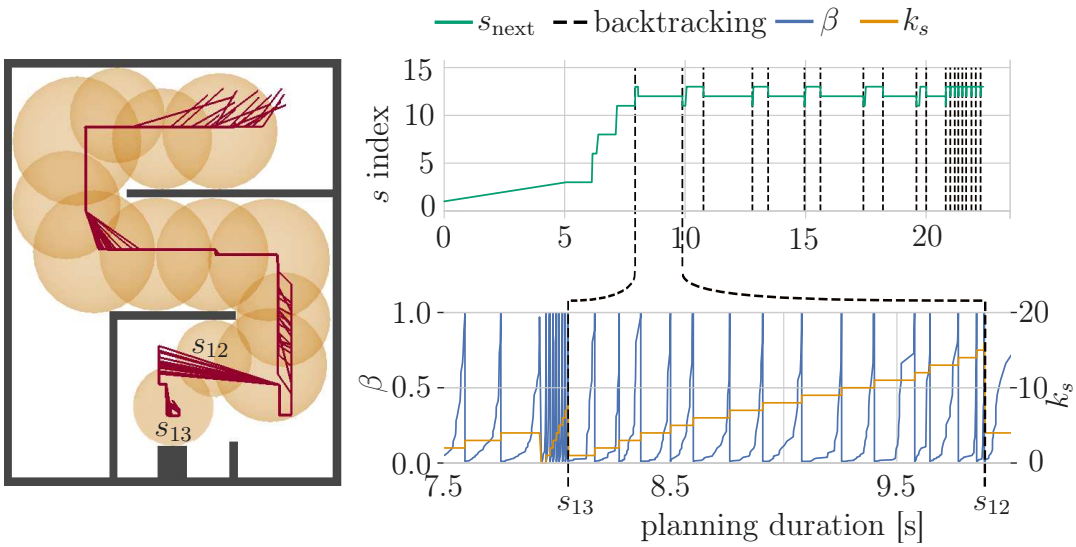


Figure 3.11 *Left:* A sequence of spheres (orange) and the solution path for each particle (red) is visualized for the 2-DOF grasping-maze problem, and the last two spheres are labeled s_{12} and s_{13}. *Right:* Two plots showing the evolution of k_s, β, s_{next} variables and the event of backtracking (dashed lines). In the top plot, it is visible that progress is hindered in the last spheres, where multiple backtracking happened. The bottom plot zoomed in when the first two backtracking events happened. The zoomed-in plot shows that the backtracking is not dependent on a fixed value for k_s. However, it depends on sphere sizes, and the plot also shows how β initially increases linearly and then exponentially.

spheres. On the bottom plot, one can observe that the planner detected being stuck for different k_s values due to differences in sphere sizes. Moreover, it is also visible that β initially increased linearly for each new value of k_s, indicating that the planner found actions with valid outcomes. As β increased, the spheres' perimeter was increasingly explored. Consequently, more actions resulted in inconsistent contact states making β to increase exponentially.

3.4 Related Planning Approaches

We discuss related planning approaches divided into two topics. First, we discuss contact-exploiting motion planners because our planners (CEET and CET) reduce uncertainty by exploiting contact with the environment. Second, we discuss how other motion planners use information because our planners use workspace information to guide exploration.

3.4.1 Contact-Exploiting Belief Space Planners

Geometry-based workspace decomposition was first introduced by Lozano-Pérez et al. (1984). They defined a *pre-image* as a region in the free and contact space from where all states can reach a goal with a single compliant motion. They combined guarded and sliding motions to sequence compliant motions in contact that reduce state uncertainty and produce robust motion plans. Canny (1989) showed that pre-image-based planning is a PSPACE-hard problem; therefore, it is difficult to apply to complex manipulation problems. Pre-images are challenging to compute because they represent task-relevant regions exactly. We overcome their computational complexity by approximating task-relevant regions using wavefront expansion for workspace decomposition. Moreover, task-relevant $\mathcal{C}$-space regions are represented implicitly by guiding $\mathcal{C}$-space exploration with workspace information.

An environment's geometry provides a discrete structure. Thus, contact-based planning can be formulated as a multi-modal problem, where the continuous state space is split into contact manifold or *modes*. Modes generally have a parametric representation rather than an exact one, such as pre-images. A multi-modal planner explores modes and searches for transitions between them to find a path between a start and a goal. The set of modes can be incrementally built by first searching for valid samples in transitional regions (Bretl, 2006; Hauser and Latombe, 2010), similar modes can be bundled into folaitions (Kingston et al., 2020; Orthey et al., 2020; Morgan et al., 2022), or modes can be enumerated instantaneously using a combination of random sampling and projection approach (Cheng et al., 2021). CEET can be considered a multi-modal planner because it leverages the discrete structure of the workspace. It explores modes (contact manifolds) obtained from workspace decomposition and finds transitions between modes either directly at mode intersections or through free space by leaving the mode.

Our planner is closely related to particle-RRT (Melchior and Simmons, 2007) that approximated belief state with particles. Particles can represent state uncertainty, and by forward simulating each particle with a noisy model, they can also represent motion uncertainty. A particle-based approximation of a belief is suitable for planners that use contact-exploiting motions because when a belief is projected on a contact manifold, it becomes non-Gaussian (Sieverling et al., 2017b; Wirnshofer et al., 2018; Phillips-Grafflin and Berenson, 2020).

Contact-exploiting compliant motion can be used to increase positioning error tolerance during planning, simplify sampling on contact manifolds, or transition between manifolds (Hang et al., 2019; Morgan et al., 2022). While this beneficial

effect of contact-exploiting compliant motion was not used in our planner, one could evaluate a sensible increase in ϵ_{EC} and $\epsilon_{\text{position}}$ for surface extraction during wavefront exploration and for selecting a sample and an action during steering, respectively.

Contact-exploiting actions can be combined with contact sensing to further reduce state uncertainty by discriminating between possible outcomes. POMDP-based planners reason about contact observations, and such approaches were successfully applied to low-dimensional manipulation and object localization problems (Hsiao et al., 2007; Bai et al., 2010; Vien and Toussaint, 2015; Koval et al., 2016). Integrating sensing is not in the scope of this chapter. However, it is the focus of the next chapter where we present another sampling-based planner using contact-exploiting actions and contact sensing to reduce increased amounts of state uncertainty.

3.4.2 Levels of Information Usage During Planning

Since we use workspace information to guide configuration and contact space exploration, we discuss approaches to integrate information into planning. We divided these approaches into three categories based on the levels of information usage: *exploitation*, *exploration*, and *guided exploration*.

Exploitation-based planners compute a plan relying solely on prior information without considering any information gained during planning. Such approaches are based on gradient descent and exploit the information encoded in a potential function. For example, virtual potential fields or navigation functions can guide a robot to avoid obstacles and progress toward the goal. These approaches were discussed in detail in Sections 2.2 and 2.5.3.

Exploration-based planners use no information but randomly sample configurations to gain understating of the configuration space. Such sampling-based motion planners represent the configuration space connectivity with a graph or tree data structure. With a graph data structure, the Probabilistic Roadmap (PRM) based planners uniformly draw samples using no information. However, when these planners check the connectivity between samples, they use guided exploration. With a tree data structure, Rapidly-exploring Random Tree (RRT) based planners grow a tree from a start configuration toward the largest Voronoi region associated with the existing samples in the tree. The Voronoi bias enables quick exploration of the configuration space, but such a planner requires a goal bias to find a path to the solution faster.

Goal bias is an informed sampling technique that guides exploration. It can be achieved by sampling the goal configuration instead of a random sample or by trying to connect new nodes to the goal. Such techniques are widely used in particle-RRT-based

planners as well (Melchior and Simmons, 2007; Sieverling et al., 2017b; Phillips-Grafflin and Berenson, 2020; Cheng et al., 2021) and our planner also uses greedy goal connect action beside workspace information to guide $\mathcal{C}$-space exploration.

There are a variety of informed sampling techniques that use a fixed or adaptive heuristic (Burns and Brock, 2007; Hauser and Latombe, 2010; Kingston et al., 2020). Such techniques exploit information gained during exploration to efficiently explore difficult regions of the configuration space, such as narrow passages with bridge sampling. Other heuristics were developed to solve issues related to the Voronoi bias of RRT planners to escape bug traps. For example, a planner can stop to extend nodes for which expansions have failed multiple times or use a bi-directional search with multiple trees, making a forest and even growing multiple forests (Hang et al., 2019).

Valuable information can be obtained from a robot's workspace to guide exploration and consequently increase a planner's efficiency in finding a solution. The most straightforward use of workspace information is to divide the space info free and contact regions and use different sampling techniques or local planners tailored for the respective region (Koval et al., 2016; Guan et al., 2018). Another approach is to leverage the environment's geometry to discretize the state space into regions, sequence the decomposed regions, and search for a motion plan through these regions (Erdmann, 1986; Lozano-Pérez et al., 1984; Goldberg, 1993; Bhatia et al., 2010). A workspace decomposition can be learned (Chamzas et al., 2021) or approximated (Rickert et al., 2014; Rajendran et al., 2019; Liu et al., 2020) and used to bias exploration toward task-relevant regions of the workspace.

The CEET planner extends the EET palnner (Rickert et al., 2014) to solve motion planning problems under uncertainty. We used the same approximation technique to decompose the workspace and identify task-relevant regions in free space. The key difference between the EET and CEET planners is that we considered motion and perception uncertainty. To handle uncertainty, our planner exploits motion in contact. Therefore, we extended the workspace information to include contact regions as well.

3.5 Conclusion and Further Considerations

The chapter aimed to efficiently plan robust motions under uncertainty by exploiting environmental constraints. To achieve that, we leveraged mechanical uncertainty reduction and a geometry-based workspace decomposition. The combination of these two ECE effects allowed efficient belief space planning. First, our planner searched for robust actions unaffected by uncertainty and contact-exploiting actions that reduce

uncertainty. Consequently, the resulting motion plan was robust against perception and motion inaccuracies. Secondly, we reduced the exploration of configurations and actions using task-relevant workspace information, improving planning efficiency. Task-relevant information was obtained by decomposing the workspace into task-relevant ECE regions and sequencing these regions between a given start and goal. We have shown that workspace information is precious when a configuration space has large regions less relevant for a given task. Our planner scaled for high-dimensional configuration spaces as well. Therefore, we achieved our goal of efficiently planning robust motions against uncertainty by sequencing task-relevant manipulation funnels and searching for a motion through these funnels.

While this chapter combined two ECE properties discussed in Section 1.2, the next chapter provides a different combination of properties by integrating contact sensing into planning when a robot moves into contact. We close this chapter with a discussion of the limitations of the CEET planner.

3.5.1 Limitations of the CEET planner

Since the CEET motion planner uses the connectivity of task-relevant ECE regions as a virtual potential field, the planner is subject to local minima where the planner can get stuck. To overcome this, a planner must detect when it is stuck and escape from the local minimum.

We detected when the planner got stuck by virtually increasing a sphere until it fully covered its parent sphere. We showed that this detection works. However, there might be more efficient ways to identify when the planner is stuck. One could estimate the maximum state uncertainty of a region from failed expansions and compare it with neighboring nodes.

We evaluated two escaping strategies that are on the opposite ends of the spectrum: random exploration and backtracking using exploration bias. We showed that the appropriate escaping strategy depends on configuration space dimensionality, and further strategies should be explored for a more efficient resolution of local minima.

Another area for improvement of the CEET planner is the need for reasoning about path optimality. Since the path is searched in a task-relevant subspace, the solution's path length is generally smaller than a path generated by randomly exploring the whole search space. One could further optimize the path generated by CEET by using our planner's solution to initialize an optimizer that reasons about contact (Posa et al., 2014; Toussaint et al., 2014; Posa et al., 2016; Toussaint et al., 2022).

A further limitation arises from the assumption that the environment is static. Despite this assumption, we think that the planner could be extended to reason about movable objects. However, reasoning about movable objects and their state uncertainty would increase the problem's complexity enormously. To overcome this issue, we are going to use a hierarchical planning scheme (Asfour et al., 2006; Jain and Niekum, 2018) in Chapter 7: the CEET motion planner searches for valid motion trajectories, and at a higher abstraction level, an ECE-based grasp planner (presented in Section 6.2.2) reasons about movable objects in a pile.

4

Integrating Contact Events Into Motion Planning

This chapter takes another approach to uncertainty handling than the previous chapter by integrating contact-based sensing into planning. In Chapter 3, we modeled uncertainty to find and sequence task-relevant manipulation funnels. Then, we searched for *robust actions* unaffected by uncertainty moving in free space inside the selected funnels and moving in contact on funnel walls. Contact sensing was required to execute a policy by indicating when entering a desired manipulation funnel. This chapter also models uncertainty and uses manipulation funnels to reduce it, but *integrates contact sensing* into motion planning to distinguish between funnel entrances because an action can lead to different funnels under a large amount of uncertainty. The content of this chapter, to a large extent, has been published before in (Páll et al., 2018) ©2018 IEEE. I was the first author of that paper. I conceived, implemented, and evaluated the proposed ConCERRT algorithm. The other two authors gave scientific advice and helped with writing. The second author and I contributed equally to the writing.

We integrate contact information from sensory measurements into motion planning to distinguish between action outcomes and reduce state uncertainty. Fortunately, ECE provides easily detectable contact events that indicate when an EC is reached or the transitioning between ECEs (Section 1.2). We leverage these contact events to distinguish between action outcomes. Since we obtain contact events from an ECE using contact-exploiting actions, such actions reduce uncertainty mechanically as well, as used in Chapter 3. Thus, this chapter combines mechanical uncertainty reduction and contact events to handle a large amount of uncertainty during motion planning.

When a robot's state uncertainty is too large, motions can lead a single action to have several possible outcomes. Different outcomes need different reactions, so a

motion planner must in advance determine suitable reactions. A motion plan can capture reactive motion behavior as alternative branches. During the execution of such a plan, a robot uses sensor data to select the one matching the current situation among the branches. 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).

To find contingency plans, a planner can model the robot's uncertainty 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 Figure 4.1. These problems require the robot to adapt its motion based on sensor information from contact sensors. The key to our planner is the assumption that contact sensing is uncertainty-free. This assumption allows for ruling out a large part of the robot's state space, given a contact measurement. This simplifies planning and leads to a low number of contingencies.

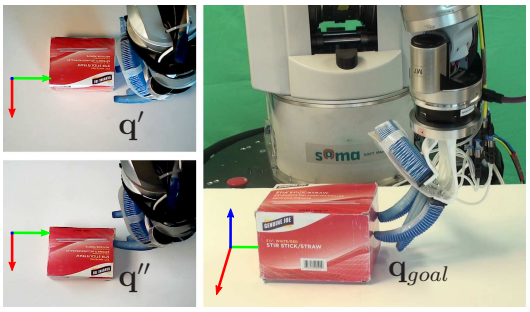


Figure 4.1 A robot performs tactile localization of an object using contact sensors on two fingers of a soft hand. By moving into contact and measuring which finger makes contact first, the robot can estimate the position of the box relative to itself (e.g., box configuration $\boldsymbol{q}'$ or $\boldsymbol{q}''$) and then adapt its action to reach $\mathbf{q}_{\text{goal}}$. ©2018 IEEE

In our view, only uncertainty-free sensing should be integrated into motion planning to reduce state uncertainty and to keep the problem computationally tractable.

Compared to related Partially Observable Markov Decision Process (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. Our experiments show that our planner consistently finds successful contingency plans in practical applications. We show solutions for problems with high uncertainty where

conformant (or non-contingent) planners fail. In real-world experiments, we show that our assumptions about uncertainty hold in realistic scenarios.

4.1 Contingent Motion Planning Problem Definition

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

Similarly to the CEET planner presented in Chapter 3, ConCERRT plans in an 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, 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 $b_0 := \mathcal{N}(\mathbf{q}_0, \boldsymbol{\sigma}_0)$ and use a motion model with independent joint noise:

$$\delta\hat{\mathbf{q}} = \delta\mathbf{q} + \mathcal{N}(0, \sqrt{|\delta\mathbf{q}|}\boldsymbol{\sigma}_\delta).$$

In contrast to the CEET planner, now, we assume the robot has access to reliable contact sensors to observe the active contact(s) at a given configuration $\mathbf{q}$. A key assumption for efficient planning is that contact sensing is fully observable. i.e., we assume that no contact is ever detected wrongly. We use two different contact sensor models:

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

2) *Force sensor model* assumes the robot can detect the contact normal: $\mathcal{O}_{\text{force}}(\mathbf{q}) = \{o_1, \ldots, o_k\}$. Each observation $o_i = (c_i, \hat{\boldsymbol{n}}_i)$ is a pair of a sensor patch and surface normal $\hat{\boldsymbol{n}}_i$. Similarly to Equation (3.1), a belief b is valid if it lies mostly in the valid configuration space:

$$\int_{\mathbf{q} \in \mathcal{C}_{\text{valid}}} b(\mathbf{q}) d\mathbf{q} > 1 - \epsilon,$$

where $\mathcal{B}_{\text{valid}}$ 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. This chapter only cares about finding feasible policies and does 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.

4.2 Continget Contact Exploiting RRT Planner

This section presents 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 two crucial insights for understanding our planner.

4.2.1 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 (Figure 4.2) with an uncertain state moves toward a wall until it touches the wall with its left or right finger. By visualizing this in belief space, uncertainty reduction becomes obvious: the contact event partitions the belief into 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 Figure 4.3. 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. Like before, the result of this action is partitioning 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.

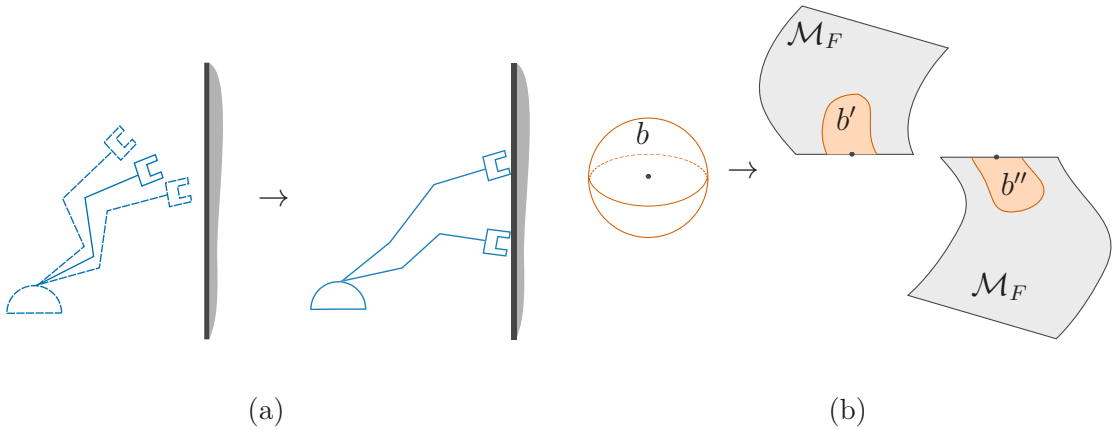


Figure 4.2 Belief state partitioning with binary contact sensing; (a) A 3-DOF robot with a two-finger hand moves from free space into contact. The initial position uncertainty results in two different contact states. (b) This action partitions the belief b into two belief states b' in contact with the left and b'' with the right finger. ©2018 IEEE

4.2.2 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 Figure 4.4. 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 to $\mathcal{B}_{\text{open}}$ and creates a new tree for each of them.

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

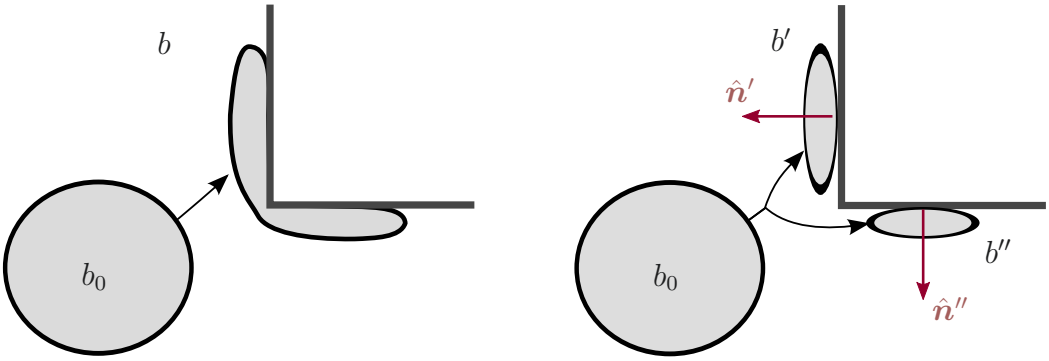
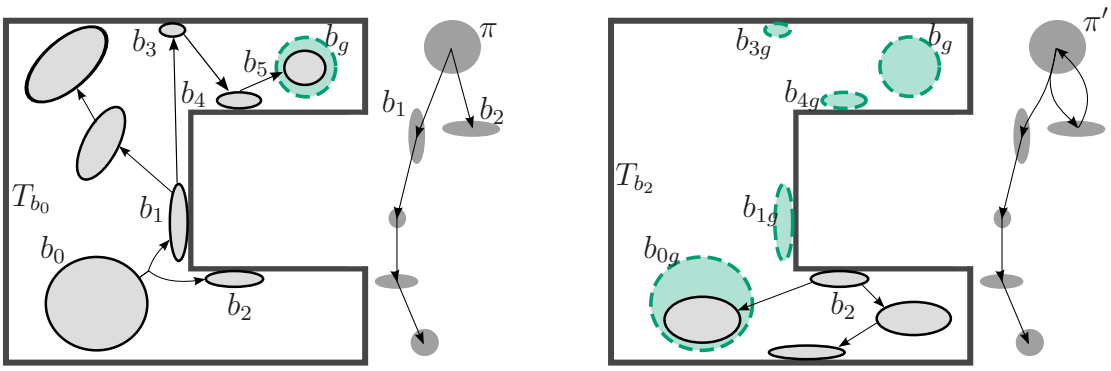


Figure 4.3 *Left:* a robot with state uncertainty moves from state b_0 towards a corner in the world, which projects the uncertainty on the surfaces resulting in state b. *Right:* measuring the contact normal ($\hat{\boldsymbol{n}}'$ and $\hat{\boldsymbol{n}}''$) allows the robot to partition its belief state b into lower uncertainty states b' and b''. ©2018 IEEE



a) first iteration b) second iteration

Figure 4.4 Two iterations of the ConCERRT planner: *a) first iteration*: the initial search tree T_{b_0} connecting start and goal beliefs b_0 and b_g. On the right is the resulting policy π, which consists of one path from start to goal but also contains one unconnected partition b_2. *b) second iteration*: 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_{0g}. On the right, this path is added to the final policy π'. ©2018 IEEE

Algorithm 6 ConCERRT motion planner

Input: b_0, b_g
Output: π

1: $\mathcal{B}_{\text{open}} \leftarrow b_0$ *initialize open contingencies with* b_0
2: $\mathcal{B}_{\text{connected}} \leftarrow b_g$ *initialize goal region with* b_g
3: $T_{b_0} \leftarrow b_0$ *initialize tree*
4: $\pi \leftarrow \emptyset$ *initialize policy*
5: **while** $P(\pi) < 1$ **do** *until all contingencies are solved*
6: **for all** $b \in \mathcal{B}_{\text{open}}$ **do** *expand all open contingencies*
7: $T_b \leftarrow T_b.\text{EXPAND}(\mathcal{B}_{\text{connected}})$ *see Algorithm 7*
8: $\pi \leftarrow \pi.\text{UPDATE}(T_b)$ *the policy is updated if a contingency is solved*
9: $\mathcal{B}_{\text{open}} \leftarrow \mathcal{B}_{\text{open}}.\text{UPDATE}(T_b)$ *add new contingencies and remove the solved one*
10: $\mathcal{B}_{\text{connected}} \leftarrow \mathcal{B}_{\text{connected}}.\text{UPDATE}(T_b)$ *save beliefs in a solved contingency as goal*
11: **return** π

moving from exploration to exploitation (Rickert et al., 2014). Initially, the algorithm must explore most of the space as $\mathcal{B}_{\text{connected}}$ contains only one element. However, 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.

4.2.3 Algorithm Outline

Based on the two previous insights, we can now give the full description of the ConCERRT planner (Algorithm 6). To plan efficiently with the non-Gaussian belief states, we represent the belief with a set of particles $b = \mathcal{Q} = \{\mathbf{q}_1, \ldots, \mathbf{q}_N\}$ where each particle $\mathbf{q}$ is a configuration. We denote the sample mean and covariance of the configurations in a belief b with $\boldsymbol{\mu}_b$ and $\boldsymbol{\Sigma}_b$, respectively.

ConCERRT initially samples a fixed number of particles from the initial belief b_0 and adds them as root to the initial search tree. Then, in every iteration, ConCERRT cycles through all elements of $\mathcal{B}_{\text{open}}$ and expands the respective tree. The expansion works similarly to an RRT planner. It samples a random configuration, finds the nearest neighbor in the current search tree, chooses an action, simulates the effects of that action and 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 Algorithm 7.

Sampling (line 1) The planner samples a random configuration to extend the current tree. The tree growth is randomly biased towards the goal by choosing $\boldsymbol{\mu}_{b_g}$ instead of a random sample with a fixed probability $p(b_\mathrm{g})$.

Nearest neighbor search (line 2) The nearest neighbor selection computes a norm that balances uncertainty and Euclidean distance. As a distance term over uncertainty, we use the trace norm $d_{\boldsymbol{\Sigma}}(b) := \sqrt{\mathrm{tr}(\boldsymbol{\Sigma}_b)}$. As spatial distance to the random sample $\boldsymbol{q}_\mathrm{rand}$ we use $d_{\boldsymbol{\mu}}(b) := \|\boldsymbol{\mu}_b - \boldsymbol{q}_\mathrm{rand}\|_2$. The best neighbor is chosen as:

$$b_{near} = \underset{b}{argmin} \left[\gamma \left(d_{\boldsymbol{\Sigma}}(b) + \sum_{b' \in Sib(b)} d_{\boldsymbol{\Sigma}}(b') \right) + (1-\gamma) d_{\boldsymbol{\mu}}(b) \right],$$

where $Sib(b)$ denotes the set of all siblings of b, i.e., the partitions that were reached from the same action. Similarly to the CEET algorithm presented in Section 3.2.4, the value of $\gamma \in [0..1]$ balances free and contact space exploration. Small values favor free space exploration, and large values result in more contact space exploration. For further details about the influence of the γ-parameter, we refer to the CERRT planner (Sieverling et al., 2017b).

Action selection (line 3) The planner chooses an action u randomly. The possible actions are: 1) *connect*, which moves particles towards the random sample on a straight line; 2) *guarded move*, which moves particles toward the random sample and stops when contact is gained; 3) *guarded slide*, which moves them in contact along a surface, maintaining contact until the contact state changes. These actions are identical to the ones used by CEET (Section 3.2.4) and CERRT planners (Sieverling et al., 2017b).

Simulation (line 4) Similarly to the CEET planner, we compute the resulting belief state b' from applying action u in belief b_near by simulating the execution of u for every particle in b_near with the noisy motion model. We check b' for joint limits or collisions with links that do not sense contact in **line 5**.

Belief state partitioning (line 6) If b' is valid, we apply the contact sensor model to find potential partitions of the belief state. For each particle $\boldsymbol{q}' \in b'$ we compute $\mathcal{O}(\boldsymbol{q}') = \{o_0, \ldots, o_k\}$. 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: We cluster based on the sensor patches in contact for the *tactile* sensor model. 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 (line 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 $d_M(\boldsymbol{q}) < \epsilon_M = 2$ for all $\boldsymbol{q} \in b''$, where $d_M(\boldsymbol{q})$ is the Mahalanobis distance between $\boldsymbol{q}$ 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. Moreover, it also updates $\mathcal{B}_{\text{open}}$ with all new partitions that are not yet connected to the goal, as described in Section 4.2.2.

Algorithm 7 T.EXPAND method

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

```
1: q_rand ← RANDOM_CONFIG()  sampling a random configuration
2: b_near ← NEAREST_NEIGHBOR(q_rand, T)  closest node to sample
3: u ← SELECT_ACTION(q_rand, b_near)  γ biased action selection
4: b' ← SIMULATE(q_rand, b_near, u)  forward propagate all particles in b_near
5: if IS_VALID(b') then
6:     B_contingencies ← BELIEF_PARTITIONING(b')  identify contact-based deviations
7:     for all b'' ∈ B_contingencies do
8:         T ← T.ADD_BELIEF(b'')
9:         T ← T.GOAL_CONNECT(b'', B_connected)
10: return T
```

The ConCERRT planner can be changed to Contingent Contact Exploring Tree (ConCET) planner to reduce the exploration space from $\mathcal{C}_{\text{valid}}$ to $\widetilde{\mathcal{C}}_{\text{task}} \cup \widetilde{\delta\mathcal{C}}_{\text{task}}$ similarly how we changed CERRT into CET in Section 3.2.5. The ConCET planner leverages both contact events and structural context obtained from ECE-based workspace decomposition. We include both contingent planners in our evaluations in the next section.

4.3 Evaluation of Our Contingent Panner

We evaluate ConCERRT and ConCET planners in simulation and one planner in a real-world experiment to show that a) our planning approach scales up to high-dimensional problems compared to other belief space motion planners. b) the contingency branches of ConCERRT and ConCET allow solving problems with significantly higher uncertainty

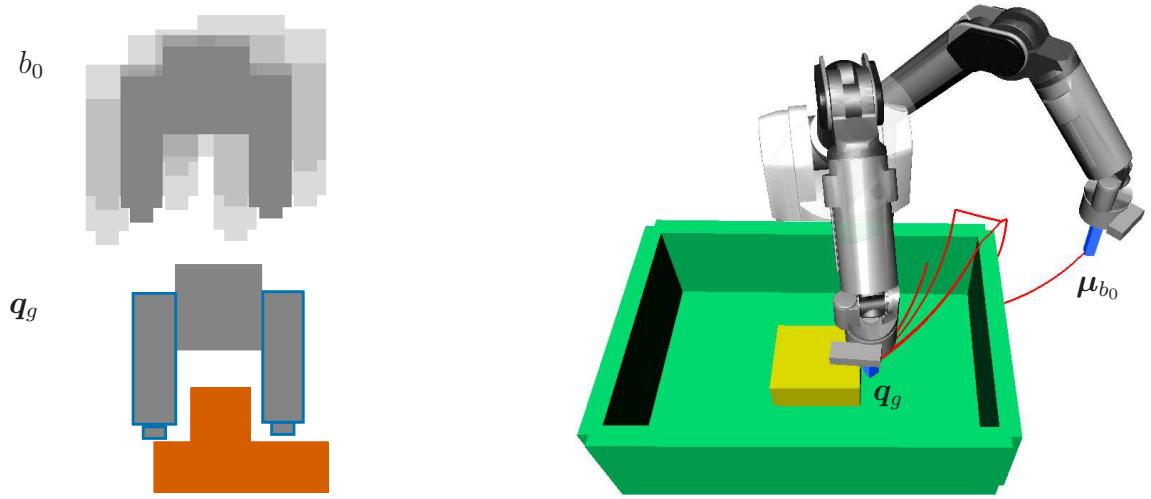


Figure 4.5 *Left*: 2-DOF gripper problem with initial distribution b_0 and goal configuration $\boldsymbol{q}_g$. The fingers and their tips on the gripper (blue) can sense contact with the object (red). *Right*: 7-DOF problem. The right configuration shows the mean of the initial belief $\boldsymbol{\mu}_{b_0}$. The goal configuration $\boldsymbol{q}_g$ 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. ©2018 IEEE

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 4.1 gives the values of the planner's parameters for all experiments.

4.3.1 Our Approach scales to high-dimensional problems

Most belief space planners rely on pre-defined discretization, which lets them fail in complex environments. We now validate in simulation that our approach scales to high-dimensional state space with complex contact states. The first experiment (Figure 4.5 left) modeled a gripper with tactile sensors on the fingers and the fingertips, similar to the setup in (Koval et al., 2020). The gripper could translate into two dimensions. Compared to a similar problem from the POMDP literature (Hsiao et al., 2007), no outer walls limited the workspace, increasing the problem's difficulty. In the second experiment (Figure 4.5 right), a Barrett WAM 7-DOF arm had to reach into a rectangular container and touch a yellow obstacle. To reduce uncertainty, it could measure the contact normal of the surfaces with a stick-shaped end-effector.

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
$\boldsymbol{\sigma}init$	initial uncertainty	$[\sigma, \sigma]$	$\mathbf{0}$	$[2, 2, 2, 3, 3, 3, 3] \times 10^{-2}$
$\boldsymbol{\sigma}_{motion}$	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 4.1 ConCERRT planning parameters during experimental evaluation. ©2018 IEEE

The results for the second problem (Figure 4.7) prove that ConCERRT and ConCET were efficient enough to compute policies directly in a 7-dimensional configuration space. The planners found policies under significant motion uncertainty that slid 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 became 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., 2016) which become intractable under the more significant number of DOF and complex contact manifolds. ConCERRT and ConCET also handled a significant amount of motion uncertainty. This relaxes the assumptions of related approaches (Koval et al., 2020), which require the inverse kinematics of the robot and fully observable joint states. Thus, we can solve a larger set of problems. Our approach also relaxes the assumptions of Particle-RRT motion planners (Phillips-Grafflin and Berenson, 2020) as it does not require reversible actions and fully observable joint states.

4.3.2 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

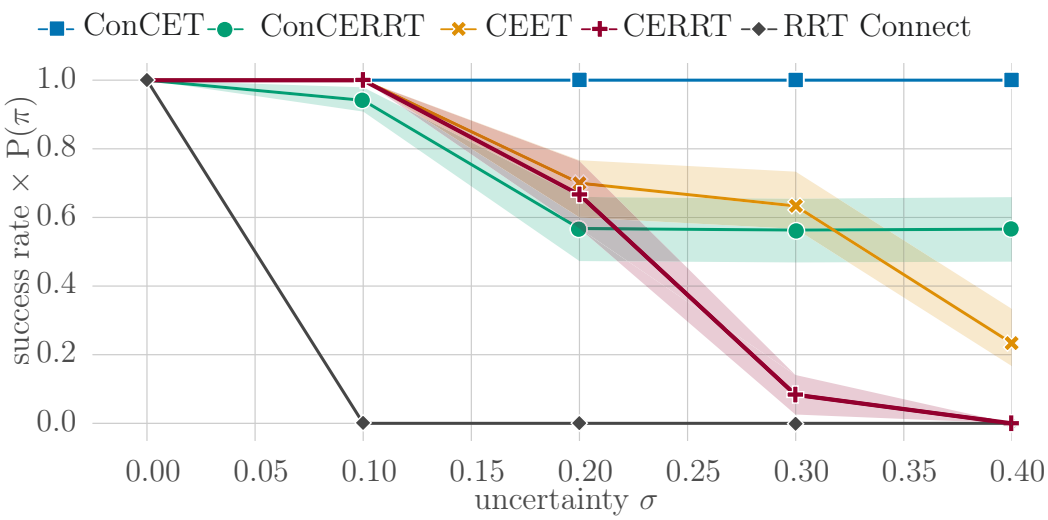


Figure 4.6 Success rate of ConCET, ConCERRT, CEET, CERRT, and RRTCon on the 2-DOF gripper problem as a function of the position uncertainty. ConCET always finds a solution, and ConCERRT still finds solutions for $\sigma > 0.3$ where the conformant planners start failing.

validate this, we ran the ConCERRT and ConCET planners on the simulation scenarios (see Section 4.3.1), varying the amount of uncertainty.

We compared our planners against three baselines. The first baseline was an uncertainty-unaware RRT-Connect with goal bias (RRTCon) (Kuffner and LaValle, 2000). The second baseline was the conformant CERRT planner (Sieverling et al., 2017b), and the third was the non-contingent CEET planner from Section 3.2. To compare contingent and non-contingent planners, we had to define a suitable scoring function considering both the planning time and the plan's quality. Therefore, we computed the score as $P_{\text{success}} = P(\pi) \cdot \frac{N_{\text{succ}}(t)}{N}$, where $P(\pi)$ was the success probability of the policy (equal to 1 for CERRT and CEET) and $\frac{N_{\text{succ}}(t)}{N}$ was the ratio of found solutions. We ran 30 experiments per setup for the 2-DOF and ten for the 7-DOF problems.

In Figure 4.6, we show results for the 2-DOF problem. The RRT Connect always returned a trajectory that failed even with little uncertainty ($N_{\text{succ}} = 0$ for $\sigma \geq 0.05$). The solution quality of the CERRT planner dropped to 0% while ConCERRT solutions' quality stayed over 50%, even for the highest uncertainty. CEET was comparably as good as ConCERRT until uncertainty reached the maximum value making CEET perform more than twice worse than ConCERRT. However, the ConCET planner outperformed all baselines and ConCERRT because it anticipated deviations and

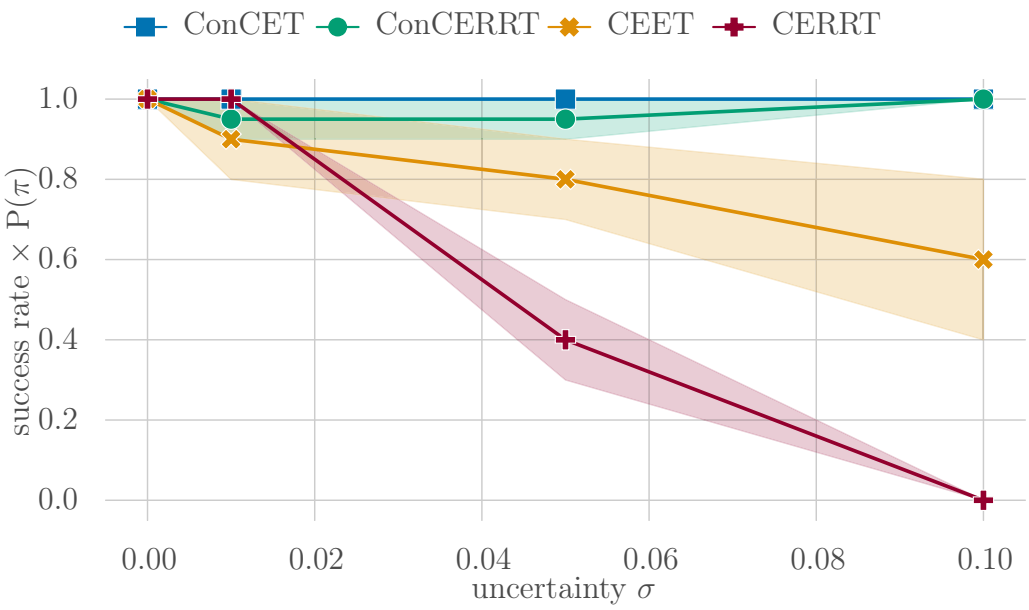


Figure 4.7 Success rate comparison of ConCET, ConCERRT, CEET, and CERRT for a 7-DOF manipulator with force sensing. Our contingent planners can handle up to ten times more uncertainty than conformant planners.

planned contingencies, but it explored only task-relevant regions of the configuration space.

The result in Figure 4.7 shows that ConCET and ConCERRT substantially outperformed the baselines CERRT and CEET in terms of robustness to uncertainty if the dimensionality increases.

In Figure 4.8, we show how the solution quality improves over planning time for the 7-DOF problem with different uncertainty values for the ConCERRT planner. Interestingly, low uncertainties did 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, which leads to one failed plan for $\sigma = 0.01$ and one for 0.5. The planner found policies that succeed in 50% of the runs in under one minute. The policies improved as time went on, approaching 100%. This shows the anytime property of ConCERRT.

4.3.3 Real robot experiment

To validate the policies generated by ConCERRT in a real-world application, we applied one policy on a 7-DOF Barrett WAM robot arm with the RBO Hand 2 (Deimel and Brock, 2016) as an end-effector. The experiment was inspired by the problem in Koval

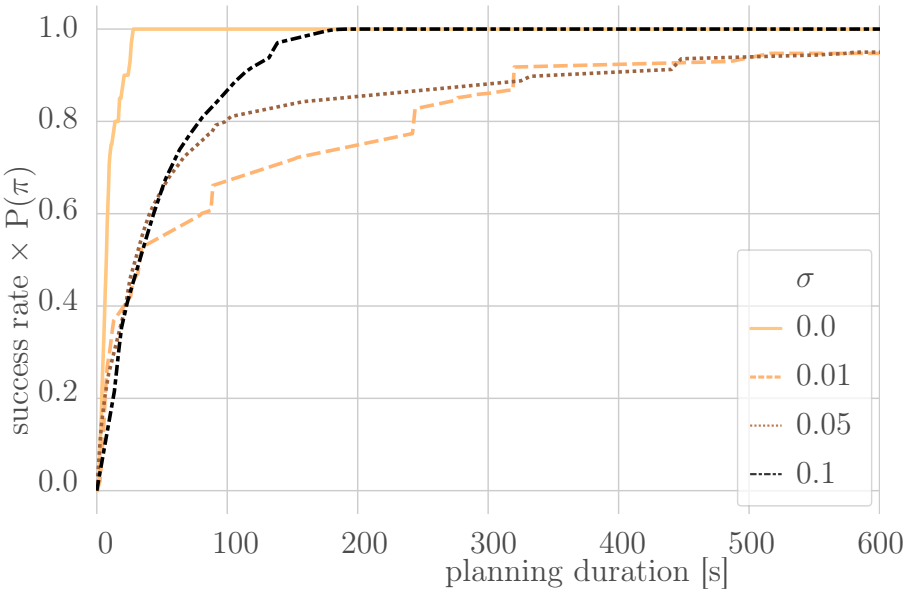


Figure 4.8 The success probability for different ConCERRT policies with increased computation time. The solution quality of the policy increases over time until all possible contact events are covered.

et al. (Koval et al., 2016), where a robot arm with a contact-sensing hand localized an object on a table surface. Similarly, our task was to sequence contact motions that reduce uncertainty enough such that the hand stops centered in front of the box. Figure 4.1 shows the experimental setup. The fingers of a soft hand deform when they come into contact with the environment, resulting in a measurable pressure change. We used large changes in pressure as a proxy for contact sensing. We only used the partially inflated index and little fingers as contact sensors. To find a policy, we ran the ConCERRT planner but excluded the slide action from planning as reliable sliding is hard to implement with a soft manipulator.

ConCERRT consistently found feasible policies in 16.66 minutes. One computed policy π with $P(\pi) = 1.0$ is shown with an exemplary execution of it with two different box displacements in Figure 4.9 and in this video[1]. The policy executes multiple motions in front of the box, expecting no contact. However, the policy also contains contingencies for the contact case. The most likely path through the policy makes 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 moved the box 0, 2, 4, and 6 cm to the left and right relative to the hand's initial position. We executed the policy four times for each displacement while keeping the initial hand position constant.

[1] https://youtu.be/NaRppcg0CtQ

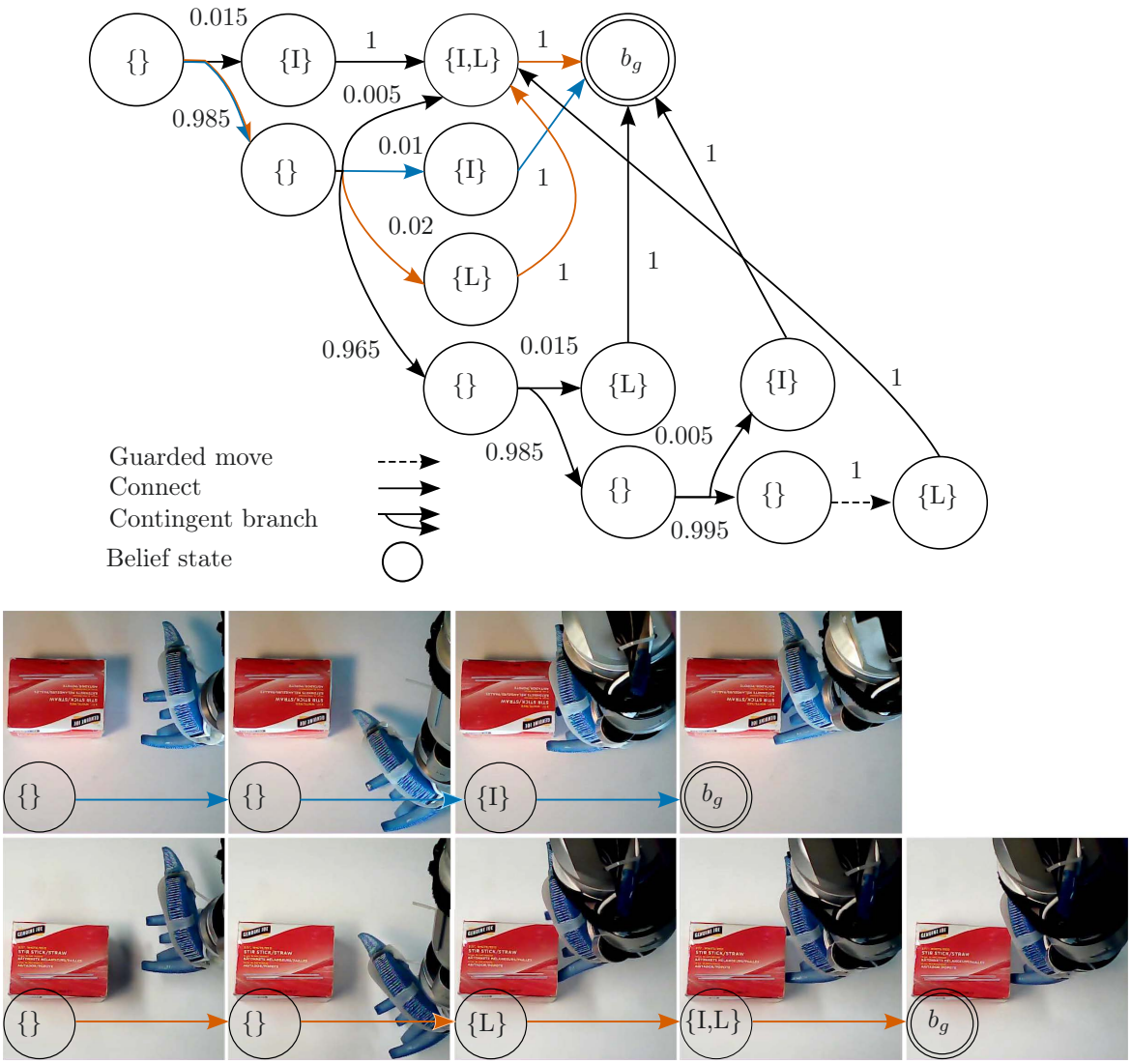


Figure 4.9 *Top:* The ConCERRT policy is executed on the real robot. Circles are belief states with contact state $\{I\}$ (index finger), $\{L\}$ (little finger), $\{I, L\}$ (both fingers), or $\{\}$ (no contact). The edges with the respective probabilities are actions that move the robot to the goal state b_g. The edge coloring shows the path taken by the robot shown below the policy. *Bottom rows*: Snapshots of two executions of the same policy found by the ConCERRT planner. *First row of images with blue arrows*: Box displaced +2 cm—the index finger makes contact first. *Second row with red arrows*: Box displaced −4 cm—the little finger makes contact first. ©2018 IEEE

For the given uncertainty model, CERRT could not find a conformant solution. Thus we compared the execution to an uncertainty-unaware planner such as RRT-Connect (RRT). Here we assumed that the robot executes the RRT trajectory perfectly

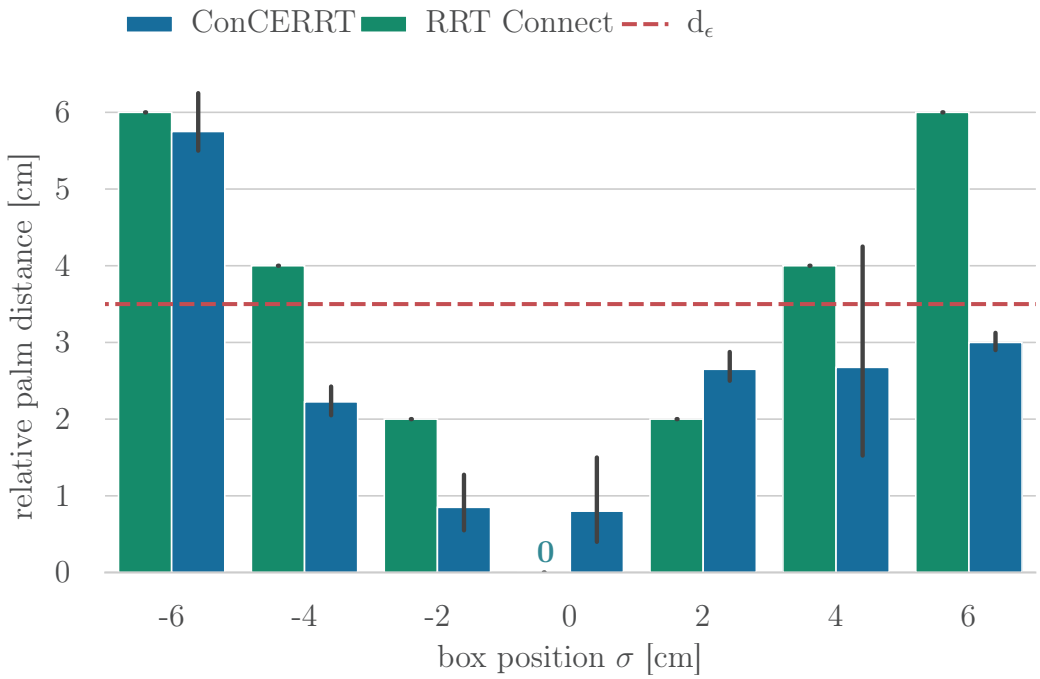


Figure 4.10 Relative position of hand and object after executing the policy shown in Figure 4.9 for different object displacements. The real robot can localize the box until up to 4 [cm] position uncertainty. ©2018 IEEE

but is unaware 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 Figure 4.10 show that the selected ConCERRT policy was robust up to 4 cm uncertainty, and it could handle 6 cm to the right but started to fail when the box was moved further to the left.

Contact sensing via pressure sensors is not fully reliable. A wrong contact event was triggered in six 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. One could mitigate these failures by equipping the soft fingers with tactile sensors (Wall and Brock, 2019, 2022).

4.4 Related Uncertainty Handling Approaches

We integrated contact information into planning to increase execution robustness. Hence, we present planning approaches on the spectrum of how much sensory information is integrated into a motion plan. Too little sensory information can not augment

inaccurate perception and actuation, but too much information can hinder planning efficiency. We need to carefully select the amount and type of information to balance computational tractability with execution robustness.

4.4.1 Sampling-Based Motion Planning

Sampling-Based approaches generate a collision-free path from a geometric model of the environment and robot kinematics by assuming that the world model and execution are perfect. Elbanhawi and Simic (2014) reviewed the classical sampling-based motion planning methods, which methods can efficiently search the high-dimensional configuration space. Such methods usually do not incorporate contact information into the plan or reason about uncertainty, but some replan from scratch or discard portions of the explored space where execution failed. We share the authors' view that planning under uncertainty and in changing environments "represents a next step in robotic research." For that, we need to integrate information accessible during execution into planning.

4.4.2 Conformant Planning

A conformant motion planner generates a fixed sequence of robust actions where the actions are guaranteed to lead to a goal, even under significant uncertainty. Information that is accessible during execution is integrated into the plan to transition between actions. The transitioning events can be proprioception, the duration of motion, or any other sensor events. A classical way is to compute all regions from which compliant actions lead to a goal, so-called pre-images (Lozano-Pérez et al., 1984), and then chain them to a sequence of actions. This approach can bring objects with unknown positions into the desired state without any sensing (Erdmann and Mason, 1988; Goldberg, 1993). In sampling-based frameworks, information types ranging from artificial information-regions to rangefinders and cameras images were used (Bry and Roy, 2011; Platt Jr et al., 2010; Van Den Berg et al., 2011), respectively, to estimate the relative position of a robot from obstacles and transition between actions. The combined configuration in contact and free space using particle-based representation (Phillips-Grafflin and Berenson, 2020; Sieverling et al., 2017b; Guan et al., 2018; Wirnshofer et al., 2018) enabled planning with contact events, as we also have shown in the previous chapter with CEET and CET planners.

Conformant planning needs robust actions leading to one unique goal state, and finding such actions is challenging under large uncertainty. To overcome this issue,

we integrated even more information to react appropriately to actions with multiple outcomes.

4.4.3 Contingent Planning

Pryor and Collins (1996) generated reactive behavior by implementing a decision tree or graph that branches based on observations. Integrating more information into the resulting motion plan becomes a policy that maps observations to actions. Contingent planners generally can solve a broader class of problems than conformant planners, for instance, part orientation for arbitrary shapes and realistic friction models (Amagai and Takase, 2001; Zhou et al., 2017) using visual or contact observation. One way to add contingencies to a motion plan is to reverse and retry motions that do not lead to the desired outcome (Phillips-Grafflin and Berenson, 2020). We tackled non-reversible actions by incrementally constructing a policy. We repeatedly invoked a conformant planner and reconnected outcomes to previously found solutions.

4.4.4 Optimal Planning Under Uncertainty

POMDP solvers are a generic approach to compute globally optimal contingency plans. The solution is a policy that maps from a belief space to actions, and this approach can integrate the most information into the plan. Point-based POMDP solvers can approximate optimal solutions (Kurniawati et al., 2008) for low-dimensional, discrete state-, action-, and observation-spaces. For continuous state space problems (e.g., manipulation and grasping), some methods discretized the lower-dimensional manifold of configurations in contact (Hsiao et al., 2007; Koval et al., 2016, 2020) to use contact as feedback. However, these approaches scale badly in high-dimensional spaces, and the discretization limits the approach to problems with few possible contacts. There exist solvers for continuous state space (Bai et al., 2010) based on sampling and Monte Carlo simulation, for continuous action space (Seiler et al., 2015) relying on numerical optimization, and continuous state-action space by combining sampling and optimization-based techniques (Vien and Toussaint, 2015). However, these approaches can not easily be applied to high-dimensional configuration space or with increased contact manifolds.

Sensory information should be integrated into motion planning to handle increased amounts of uncertainty. However, one should carefully balance the amount and quality of the information to keep the problem computationally tractable and the solution robust under uncertainty.

4.5 Conclusion and Further Considerations

This chapter aimed to handle large amounts of uncertainty during motion planning by integrating contact-based sensing into planning. We leveraged contact-exploiting actions to reduce state uncertainty mechanically and contact sensing to distinguish between the possible outcomes of a noisy motion. Since reasoning about contact sensing increases the computational complexity of a motion planning problem, we assumed that contact sensing is fully observable. Moreover, we increased the goal region with each partial solution making planning easier for unsolved deviations detected via contact sensing. We evaluated our ConCERRT and ConCET motion planners in simulation and showed the former's application for tactile object localization. We achieved our goal by integrating contact events as sensor measurements into motion planning to solve problems with increased uncertainty.

Since this is the last chapter of Part I, we summarize this part's contributions before discussing the limitations of the ConCERRT motion planner. The goal of Part I was to apply environmental constraint exploitation in motion planning under uncertainty. Environmental constraint exploitation provides numerous benefits discussed in Section 1.2. This part showed that ECs structure the workspace into task-relevant manipulation funnels that guide configuration space exploration. Moreover, it showed that contact events associated with ECs reduce uncertainty efficiently. Hence, we successfully applied 1) manipulation funnels to reduce uncertainty mechanically, 2) contact-event-based uncertainty reduction by generating reactive motion behaviors, and 3) leveraged the structural context of the environment to identify task-relevant manipulation funnels. Consequently, motion planning under uncertainty became computationally scalable to high-dimensional configuration spaces, large workspaces, and increased uncertainty. In Part II, we will show that movable objects in a pile offer similar ECE benefits (and more) as the geometrical features of a static environment.

4.5.1 Limitations of the ConCERRT planner

One limitation of the ConCERRT and ConCET planners concerns our assumption about contact sensing. We assumed contact sensing is fully observable, which might seem to be a strong assumption. However, we have shown with a real robot experiment that a robot can localize an object using pressure differences in a soft hand when it deforms by appropriately thresholding pressure measurements. Therefore, we had to couple planning and sensing to uphold our assumption. This implication for ECE-based manipulation is discussed in more detail in Chapter 7.

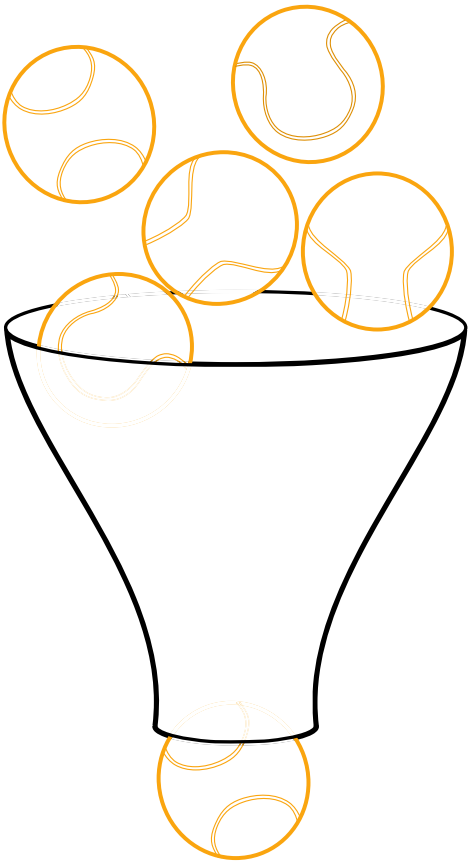
Another limitation is that our planners assumed a static environment without reasoning about movable objects. While handling movable objects is not implemented, in principle, our planners could model the noisy motion of movable objects. However, as we point out in Chapter 3, if the number of movable objects is large, reasoning about all objects' motion increases the problem's computational complexity enormously. We overcome this challenge in a particular grasping problem in the next chapter by using a new EC that allows abstracting away details of individual interactions between objects in a pile.

Similarly to the CEET planner (Section 3.2), the ConCERRT and ConCET motion planners do not minimize the overall path length of the solution. Wirnshofer et al. (2018) have shown that asymptotically optimal planning is feasible for similar problems. Alternatively, a sub-optimal motion plan can be post-processed with an optimizer that reasons about contact as well (Posa et al., 2014; Toussaint et al., 2014; Posa et al., 2016; Toussaint et al., 2022).

Finally, the ConCERRT planner integrated only sensing-based contact information, which increased the computational complexity of the planning problem. We reduced the computational complexity by assuming fully observable contact sensing and by reusing solved contingencies as goal regions. However, one could increase planning efficiency by integrating workspace information about task-relevant free and contact regions. We showed with the ConCET planner that limiting the configuration space exploration increases planning efficiency, and we think that leveraging the structural context provided by workspace decomposition would further improve planning efficiency.

Part II

Grasping With a New Environmental Constraint



"Everything should be made as simple as possible, but not simpler."

Albert Einstein

Motivation

Humans effortlessly grasp a single object on a tabletop or from a pile of objects in a box. They can reach into a box and grasp a single piece of popcorn without looking while watching a movie. As opposed to humans, robots grasp objects seemingly in a more complicated way. Robots look at a scene to choose an object for grasping, approach and position their fingertips accurately, and close their fingers to grasp the chosen object. Why can humans grasp effortlessly? What do humans use to simplify the grasping process? Can we simplify robotic grasping using insights from human grasping? This part of the thesis proposes answers to these questions by studying human-like grasping from piles of objects.

In terms of human grasping, Deimel et al. (2016) have shown that the environment plays a key role in single object grasping, and they have called such use of the environment *environmental constraint exploitation*. When a hand applies forces on an object and the environment, the environment provides additional forces constraining the hand's and the object's motion. Humans use such constraining forces to simplify grasping because these forces help to slide, pivot, flip, or stabilize an object for grasping. Researchers have shown that robots can use static and flat surfaces as environmental constraints to simplify a variety of manipulation tasks (Odhner et al., 2013; Dafle et al., 2014; Eppner and Brock, 2015).

We study complex interactions between a hand and movable objects to identify a new environmental constraint. We use the gained insights to efficiently handle uncertainties caused by inaccuracies in perception, actuation, modeling the environment, and modeling interaction physics when grasping from piles of nearly identical objects.

Contributions

The contributions of the second part are the following:

- We discover and characterize a new environmental constraint and its effect on grasping from piles of objects.
- We describe an empirical procedure to identify and characterize environmental constraints using a real robot and simulated experiments.
- We instantiate the novel environmental constraint with a simple grasping strategy using open-loop controllers without detecting individual objects in a pile.
- We devise a planning method to synthesize grasp strategies leveraging the novel environmental constraint in combination with other environmental constraints.

Outline

This part analyses the implications of environmental constraint exploitation in robotic grasping from piles of nearly identical objects. We explain the gained insights from our empirical study and build on these in Part III.

Chapter 5 presents classical grasping approaches. With the form- and force-closure approaches, we point out the classical assumptions about the considered point contact information and the computation needed to devise a grasp strategy to illustrate what aspects of computation are simplified when using a new environmental constraint.

Chapter 6 analyzes the implications of environmental constraint exploitation on a specific grasping problem. Section 6.1 empirically studies a new environmental constraint emerging in the dynamics of piles of nearly identical objects and characterizes the conditions for simple open-loop grasping. Our grasping strategy is robust even without reasoning about individual contact points between objects or individual objects' motion. Section 6.2 shows how the novel environmental constraint can be used to simplify perception, control, and grasp planning by abstracting away details of the interaction physics of pile dynamics. Finally, Section 6.4 discusses the benefits and limitations of the proposed grasping strategy and future research direction.

5

Background on Classical Grasping

This chapter presents two classical approaches to synthesizing grasp strategies: the *form* and *force closure* approaches rely on analytical interaction models to compute a desired grasp strategy, and these are the most used approaches for grasping. The content of this chapter is unique to this thesis and is based on the robotics textbook by Lynch and Park (2017) that offers a more thorough introduction.

Since our goal is to use the environment to simplify robotic grasping and reduce uncertainties, the following introduction focuses on the accuracy requirements formulated as assumptions and the computation needed for classical grasping. We conclude this chapter with a discussion about these assumptions' realism and the associated limitations of the presented approaches.

5.1 Contact Kinematics

Classical grasping approaches analyze how contacts between a robot's fingers and a movable object constrain the object's motion. Therefore, we define two types of constraints imposed by a single contact between two objects: the *active-* and the *impenetrability-constraint*. For that, we assume to know the contact point, the contact normal, and that two objects remain in contact if the contact point's velocity is constant.

First, we define the active constraint, ensuring that the two objects in contact also maintain contact when they start moving:

$$\hat{\boldsymbol{n}}^{\mathrm{T}}(\dot{\mathbf{p}}_A - \dot{\mathbf{p}}_B) = 0,$$

where $\hat{\boldsymbol{n}} \in \mathbb{R}^3$ is a unit vector aligned with the contact normal expressed in a world fame, $\mathbf{p}_A \in \mathbb{R}^3$ is a contact point on an object A and $\mathbf{p}_B \in \mathbb{R}^3$ is on B, and the

associated contact point velocities are $\dot{\mathbf{p}}_A$ and $\dot{\mathbf{p}}_B$. Even though the contact points $\mathbf{p}_A$ and $\mathbf{p}_B$ are identical, the associated velocities $\dot{\mathbf{p}}_A$ and $\dot{\mathbf{p}}_B$ can be different because the two objects can move in a different direction.

Secondly, we define the impenetrability constraint, ensuring that two objects do not pass through each other when they move into or in contact. For this definition, we choose that the contact normal points into A:

$$\hat{\boldsymbol{n}}^{\mathrm{T}}(\dot{\mathbf{p}}_A - \dot{\mathbf{p}}_B) \geq 0.$$

The active and impenetrability constraints can be expressed in terms of twists $\mathcal{V} = (w, v)$, where $\dot{\mathbf{p}} = v + w \times \mathbf{p} = v + [w]\mathbf{p}$, and a wrench $\mathcal{F} = (\mathbf{p} \times \hat{\boldsymbol{n}}, \hat{\boldsymbol{n}}) = ([\mathbf{p}]\hat{\boldsymbol{n}}, \hat{\boldsymbol{n}})$:

$$\mathcal{F}^{\mathrm{T}}(\mathcal{V}_A - \mathcal{V}_B) = 0 \text{ and} \tag{5.1}$$

$$\mathcal{F}^{\mathrm{T}}(\mathcal{V}_A - \mathcal{V}_B) \geq 0, \tag{5.2}$$

respectively. When object B is fixed, both conditions' left side simplifies to $\mathcal{F}^{\mathrm{T}}\mathcal{V}_A$. After this simplification, the two objects repel each other when $\mathcal{F}^{\mathrm{T}}\mathcal{V}_A > 0$, and they are in an active constraint when $\mathcal{F}^{\mathrm{T}}\mathcal{V}_A = 0$.

When two objects A and B are in contact satisfying the active constraint condition (Equation 5.1), then A and B are in a roll-slide contact. To differentiate between a roll or slide contact, we analyze the relative motion at the contact point. The two objects are in roll contact if there is no relative motion at the contact point:

$$\dot{\mathbf{p}}_A = v_A + [w_A]\mathbf{p}_A = v_B + [w_B]\mathbf{p}_B = \dot{\mathbf{p}}_B. \tag{5.3}$$

Note that this definition considers only rolling and sliding contact. Moreover, rolling contact includes sticking contact because there is no relative motion at the contact point nor relative rotation in a sticking contact. From the definitions, it follows that two objects are in sliding contact if Equation (5.1) is satisfied but Equations(5.3) is not satisfied.

A stable grasp requires more than one contact point. So, we extend the analysis from single to multiple contacts. Consider that an object A is in contact with m other objects at n contact points, where $m \leq n$. We number the other objects with $j = 1, ..., m$, the contact points with $i = 1, ..., n$, and we refer to a specific contact between object A and j at contact i as $j(i)$. Each contact i creates a constraint on $\mathcal{V}_A$

limiting the six-dimensional twists to a half-space that can be expressed as:

$$\mathcal{F}^{\mathrm{T}}\mathcal{V}_A = \mathcal{F}^{\mathrm{T}}\mathcal{V}_{j(i)}.$$

We compute the feasible twist V in the $\mathcal{V}_A$ space:

$$\mathrm{V} = \{\mathcal{V}_A | \mathcal{F}_i^{\mathrm{T}}(\mathcal{V}_A - \mathcal{V}_{j(i)}) \geq 0 \text{ for all } i\},$$

where $\mathcal{F}_i$ is the ith contact normal pointing into object A as chosen previously and $\mathcal{V}_{j(i)}$ is the twist of the other object at the same contact i.

Let us consider all contact i stationary and denote the respective constraint wrench $\mathcal{F}_i$. Then the feasible twist cone V is

$$\mathrm{V} = \{\mathcal{V}_A | \mathcal{F}_i^{\mathrm{T}}\mathcal{V}_A \geq 0 \text{ for all } i\}. \tag{5.4}$$

If the constraint wrench $\mathcal{F}_i$ positively spans the six-dimensional wrench space, then the stationary contacts immobilize object A entirely, and we have *form closure*. Next, we discuss in more detail from closure.

5.2 Form Closure

Form closure analyzes how rigid objects can move relative to each other while respecting the impenetrability constraint Equation (5.2). The analysis is based exclusively on contact normals and geometric properties of the object. A movable object is in force closure if it is fully immobilized by contacts provided by a set of stationary objects, as illustrated in Figure 5.1.

A stationary contact i provides a half-space twist constraint as presented in Equation (5.4):

$$\mathcal{F}_i^{\mathrm{T}}\mathcal{V} \geq 0.$$

To achieve form closure for a spacial object, we need $6 + 1 = 7$ contacts, and for planar objects, we need $3 + 1 = 4$ contacts. Moreover, to compute

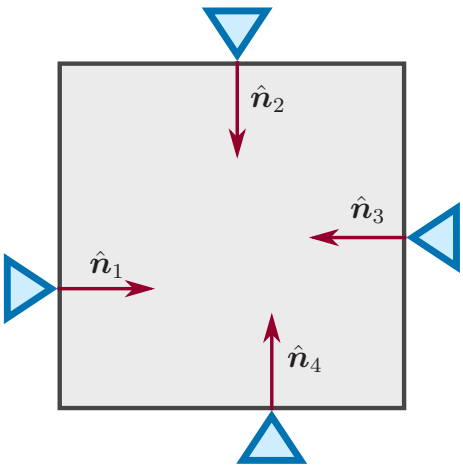


Figure 5.1 Form closure grasp example of a square (gray) with four robot fingertips (blue) and associated contact normals $\hat{\boldsymbol{n}}_{i=1,\ldots,4}$ (red arrows).

a form closure, we need to know the exact geometry of the object and find a set of contact points that constrain all possible external wrenches. We can randomly choose a set of contact points and then solve a linear programming test. For this test, we define the matrix F:

$$F = [\mathcal{F}_1 \, \mathcal{F}_2 \, \cdots \, \mathcal{F}_j] \in \mathbb{R}^{n \times j}, \tag{5.5}$$

where the columns are the j contact wrenches, $n = 6$ for three dimensional bodies or $n = 3$ for planar bodies, and a contact wrench is $\mathcal{F}_i = [m_{iz} \, f_{ix} \, f_{iy}]^{\mathrm{T}}$.

We evaluate a set of contact points, represented with F, using a weighting vector $\mathbf{k} \in \mathbb{R}^j$. Form closure is achieved with a given set of stationary contacts if there exists a weight vector $\mathbf{k}$ such that

$$F\mathbf{k} + \mathcal{F}_{\text{external}} = 0 \text{ for all } \mathcal{F}_{\text{external}} \in \mathbb{R}^n. \tag{5.6}$$

We determine form closure in two steps:

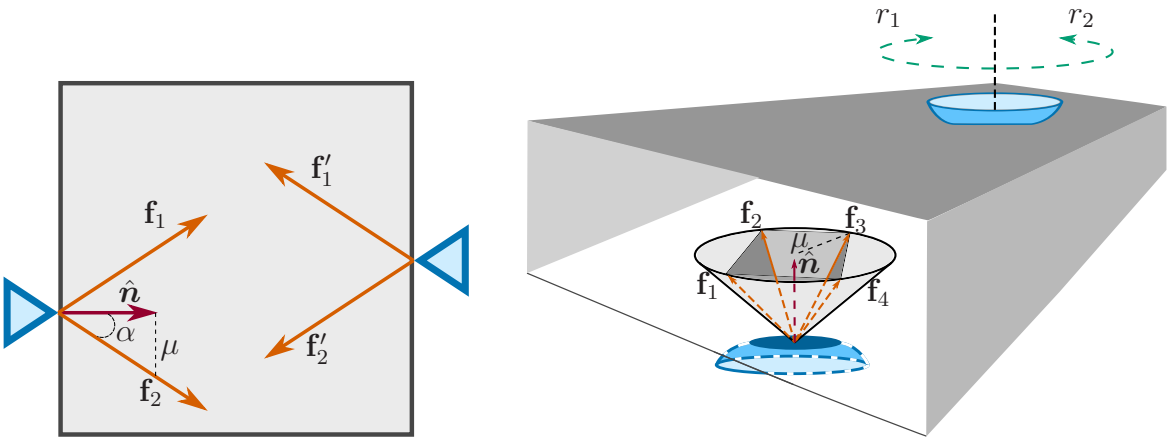
1. compute the rank of F, if $\text{rank}(F) = n$, $n = 3$ for planar problems and $n = 6$ for spacial problems, then

2. find a strictly positive $\mathbf{k}$ with coefficients $k_i > 0$, $i = 1, ..., j$ that satisfies:

$$\underset{\mathbf{k}}{argmin}(\mathbf{1}^{\mathrm{T}}\mathbf{k}) \text{ such that } F\mathbf{k} = 0, \tag{5.7}$$

where $\mathbf{1}$ is a vector of ones of size j, and it is used to make the problem well-posed for some linear program solvers, but it is not required to determine form closure. An object is in form closure if both steps succeed. If any of the two steps fail, the object is not in form closure.

In conclusion, a form closure grasp requires a set of contact points provided by a robot's fingers on a movable object such that the object is fully immobilized. The analysis considers only contact points and normals and assumes that the object's exact geometry is known. The next section also considers friction forces, which allows fewer contact points to achieve *force closure*.



(a) two point contacts with friction (b) two soft contacts with friction

Figure 5.2 *Left:* a force closure grasp example of a square (gray) contacted by two fingertips (blue). At the left fingertip, we visualize the friction cone with two edges $\mathbf{f}_1$ and $\mathbf{f}_2$ (orange arrows) for the point contact with contact normal $\hat{\boldsymbol{n}}$ (red arrow). The two edges are computed using the Coulomb friction model and $\alpha = \tan^{-1}\mu$, where μ is the friction coefficient. *Right:* two soft contacts on a box. The bottom contact patch is approximated as a contact point, and the friction cone, pointing into the box, is approximated with a four-sided polyhedral cone with edges $\mathbf{f}_{i=1,\ldots,4}$. The soft contact on the top illustrates that the contact path can resist rotations in both directions r_1 and r_2 around the contact normal due to torsional friction.

5.3 Force Closure

The difference between *form* and *force closure* is that the latter approach considers friction forces at a contact point. We model frictional contact with the Coulomb friction model $f_\mathrm{t} \leq f_\mathrm{n} \cdot \mu$, where the magnitude of the tangential friction force f_t is proportional to the normal force f_n multiplied by the friction coefficient μ.

Since friction forces are tangential to the contact normal, a contact can provide forces along the plane defined by the contact normal. The forces that can be applied along this plane lie inside a friction cone. The friction cone is characterized by the friction angle $\alpha = \tan^{-1}\mu$. For planar problems, the friction cone is the positive span of the two edges of the cone, as shown on the left in Figure 5.2. The force cone is usually approximated for spacial problems to allow linear formulations of contact mechanics. The approximated cone is discretized into a polyhedral convex cone as illustrated on the bottom right in Figure 5.2.

A friction force f can be expressed as a wrench $\mathcal{F} = ([\mathbf{p}]f, f)$ by choosing a reference frame and then transforming it into a wrench cone. Each edge of the wrench cone provides a wrench $\mathcal{F}_i$, where $i = 1, 2$ for planar problems and $i \geq 3$ for spacial problems.

A movable object is in *force closure* if the wrenches provided by all wrench cones of stationary contacts can balance any external wrench $\mathcal{F}_{\text{external}}$.

Similarly to form closure, we use linear programming to determine if an object is in force closure. For planar problems, the linear programming test provides an exact binary output. However, for spacial problems, it is only an approximation because the friction con is approximated, so the associated wrench cone is also approximated.

We denote the wrenches corresponding to the edges of the friction cones for all the contacts with $\mathcal{F}_i$, where $i = 1, ..., j$. Then, we compose the F matrix using Equation (5.5) and follow the same steps defined for form closure. First, we verify that $\text{rank}(F) = n$, $n = 3$ for planar problems, and $n = 6$ for spacial problems. Second, we evaluate the condition in Equation (5.7) with linear programming. An object is in force closure if both steps succeed. If the matrix rank is less than n or there is no solution to the linear program, the object is not in force closure.

In conclusion, a force closure grasp requires fewer contact points than form closure to restrict an object's motion against an external force by using friction forces and balancing the contact forces. Similarly to form closure, we assume to know the exact geometrical model of the object. However, we also assume to know the friction properties of the object and a robot's fingers.

It is important to note that we can approximate the effect of friction without explicitly modeling the associated forces, and consequently, we can analyze a grasp kinematically. We can consider the following three approximations:

- a **frictionless point contact** considers only the roll-slide contact and enforces only the impenetrability constraint with Equation (5.2),
- a **point contact with friction** differentiates between a roll and slide contact and enforces Equation (5.2) and (5.3),
- a **soft contact** extends the point contact with friction by considering an additional constraint at the contact. The additional constraint forbids spin between the two objects at the contact because a soft contact creates a contact patch that provides torsional friction, as illustrated in Figure 5.2. With planar problems, a point contact with friction and a soft contact is identical because rotation is not allowed around a contact normal but orthogonal to the plane.

5.4 Point Contact Approximation Leads to Inefficient Computation

This section discusses the realism of the assumptions of form and force closure analyses because we need realistic assumptions to handle uncertainty. Moreover, we discuss the computational requirements to compute a stable grasp with the mentioned methods because we want to simplify grasping with environmental constraint exploitation.

The point-contact assumption is a computationally inefficient approximation of complex interactions for robotic manipulation, and ignoring the environment is a missed opportunity to simplify the manipulation problem.

Both form and force closure assume point contacts, which might seem to be a very strong assumption because it is rarely the case in real life. However, a set of point contacts can model a line contact or even a contact patch. A line contact can be represented with two contact points at the end of the line segment, and a contact patch can be represented with a set of points at the vertexes of the patch's convex hull. This simplification allows analyzing the robustness of a grasp with the mentioned approaches. Even though we can simplify a line and a patch contact to a set of point contacts, a stable grasp requires significant computation because we need to search for appropriate contact points on an object's surface that has an infinite number of possible points.

With form closure, the frictionless contact assumption is a reasonable simplification. However, the form closure's requirement to entirely restrict an object's motion is too strong because it is only needed for a narrow set of applications. If we look at human grasping, humans only partially restrain an object because they can anticipate possible external forces that a grasp has to resist. The soft finger pulps always allow some object motion. While some research exists in the context of compliant contact with form closure analysis (Howard and Kumar, 1996; Lin et al., 1997; Ciocarlie and Allen, 2009), it is not applied in real robot experiments, which makes it difficult to assess its realism.

Force closure-based approaches are more popular than form closure because fewer contacts can already stabilize a grasped object, but we need to know the object's friction properties to find good contact points. Similar to form closure, we also need accurate knowledge about the object's geometry. Having accurate information about all objects that humans grasp daily is difficult. Therefore, this assumption is too strong and a potential source of inaccuracies. When we lack this information, we can

use visual perception to identify an object's shape and reason about other physical properties. Nevertheless, it is unreasonable to assume that we can perfectly assess the frictional properties of an object from visual perception.

While the mentioned assumptions require unreasonable knowledge or visual perception accuracy, execution also assumes accurate positioning of a robot's fingertips. However, the robot's motion likewise suffers from inaccuracies that can lead to grasping failure. Even if we fulfill model and motion accuracy requirements, both form and force closure approaches require an enormous amount of computation.

In the context of grasping, re-grasping, and in-hand-manipulation, Chavan-Dafle (2020) showed in his thesis that planning with classical frictional point contacts is 10-1000 slower than reasoning about motion cones. He introduced motion cones to represent a set of instantaneous object velocities using frictional interactions and abstracting away the interaction dynamics between an object and the pusher. The manipulation strategy's computation was quicker using motion cones, but it required a lower bound estimate of friction coefficients. However, Chavan-Dafle has proved that if the environment is appropriately used, the motion cone approach is invariant to the friction coefficient, the object mass, and the grasping force, and so, it became invariant to associated modeling inaccuracies. This is one of the multiple beneficial effects of environmental constraint exploitation, and we will discuss further benefits in the next chapter.

6

Piles Provide Granular Environmental Constraints for Grasping

This chapter analyzes the implications of ECE on a grasping problem and shows that a new EC offers the same benefits as presented in Section 1.2. The main difference to Part I is that the novel *ganular EC* manifests in piles of movable objects and enables robust grasping. Now, the task is to grasp an object from a pile of nearly identical objects, sometimes next to static vertical walls. We assume that state uncertainty results from error accumulation from different sources such as inaccurate perception, actuation, model of the environment, or model of physics. The new granular EC eliminates uncertainty and also simplifies perception, control, and planning. To some extent, the content of this chapter has been published in (Páll and Brock, 2021) © 2021 IEEE. I was the first author of that paper and the main contributor to the writing. I conducted the empirical study and the experiments. The second author gave scientific advice and helped with the writing.

Classically, grasping problems are solved by visually detecting individual objects, computing contact points on objects for stable grasping, and choosing an object based on contact points that promise robust grasping. Then the robot approaches the object and touches it at the computed contact points without disturbing the pile. Alternatively, a robot can separate objects from the pile to reduce the problem of grasping an individual object.

In contrast, we devised a robust grasping strategy without detecting individual objects in a pile, computing contact points, or carefully applying forces on an object. The surprising success of our simple grasping strategy was due to a new environmental constraint, or EC. We discovered the novel EC emerging from the complex interaction

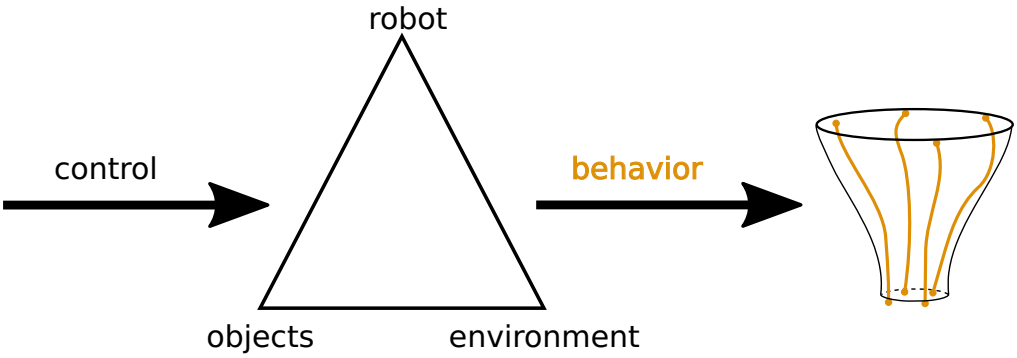


Figure 6.1 The same control signal sent to a robot generates different motions (orange lines) for different starting conditions. However, the manipulation behavior is robust because uncertainty is reduced, just like a physical funnel reduces uncertainty. The manipulation behavior emerges from the interaction between a robot, objects, and the environment. So control, a robot's embodiment, and the environment can reduce uncertainty.

between a robot and a pile of objects, and its exploitation simplified perception, control, and planning.

The first core idea is to generate robust manipulation behaviors from the interaction of three actors: a robot, objects, and the environment, as illustrated in Figure 6.1. An ECE-based grasping behavior is robust against uncertainties because the environment provides manipulation funnels (Mason, 1985), reducing uncertainty mechanically. Uncertainty is reduced mechanically because an object's (or hand's) state is constrained physically by the environment.

We show that complex interactions between movable objects in a pile reduce the position uncertainty of an object relative to the hand when scooping objects from a pile. Hence, pile dynamics provide a manipulation funnel similar to manipulation funnels emerging from the interaction between an object and static parts of an environment (e.g., horizontal or vertical flat surfaces). Since we leveraged the geometrical properties of a static environment, we will refer to this type of EC as *geometrical EC* in the coming parts of the thesis to differentiate it from the granular EC. Until now, we have only considered and leveraged geometrical ECs.

The second core idea is that some effects of ECE on a grasping process are easily predictable, and others are easily observable. For example, mechanical uncertainty reduction is predictable for geometrical ECs since such constraints restrict an object's configuration to a lower dimensional manifold (e.g., plane, edge, or point), and contacting different surfaces is detectable by observing the perceived contact normals. An

appropriate division between predicting and sensing interaction outcomes simplifies the grasping process. While debating the role and importance of sensing in robotics, Mason (1993) formulated this observation citing Brooks (1991) famous quotes:

> *"Let the world be its own model." This homily contains a kernel of truth: "Do not compute that which can be sensed more economically." However, the opposite is equally true: "Do not sense that which can be predicted more economically."*

Since an individual object's motion in a pile is difficult to predict due to complex multi-body dynamics, we studied the novel environmental constraint empirically. We analyzed the emergence of a high-level motion pattern in pile dynamics and the conditions for which the high-level pattern consistently manifests. Based on our findings, we instantiated a simple open-loop grasping strategy that could robustly grasp round objects from a pile without visually detecting individual objects. Finally, we derived realistic assumptions based on our study and devised a grasp planner that can sequence ECs, including the novel environmental constraint.

6.1 An Empirical Study of Granular Environmental Constraint Exploitation

This section characterizes a new environmental constraint and shows how its predictable manipulation behavior and observable manipulation outcomes simplify perception and control while achieving robust grasping. The novel EC emerges from complex interactions of movable objects in a pile, and we devise a grasping strategy to leverage it. We use the outcomes of the following empirical study to simplify the associated grasp planning problem and devise a grasp planner.

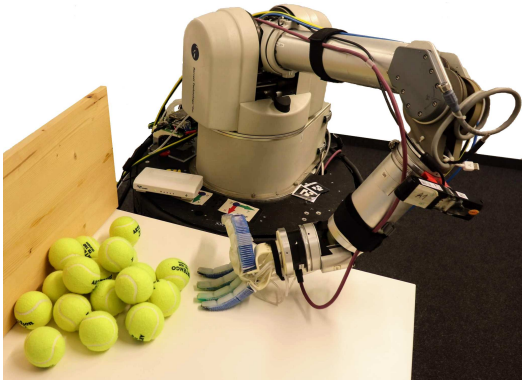


Figure 6.2 Granular environmental constraint exploitation with a soft hand from a pile of tennis balls beside a wall. © 2021 IEEE

We observed the novel EC in piles of nearly identical objects. When a hand on a tabletop slid into a pile, as shown in Figure 6.2, objects in front of the hand *stabilized*, i.e., moved together with the hand, while other objects expanded radially. Based on this observation, we devised a simple grasping strategy.

Surprisingly, the robot achieved a 100% success rate when grasping from piles of tennis balls using open-loop controllers without any visual or tactile perception. The success of this simple strategy can only be explained by the existence of an EC and it provides a plausible explanation for why humans grasp effortlessly from piles of objects, like grasping popcorn or nuts from a container.

To characterize the manipulation funnel provided by the novel EC, we propose a physics-based explanation for why objects stabilize in front of a hand when grasping with our strategy. We consider piles of solid objects to be a granular material (Duran, 2012). Granular materials have two important properties: *penetration resistance* (Stone et al., 2004) and *radial force chains* (Tordesillas et al., 2014). We describe and leverage these properties in our explanation. Even though we can provide an explanation using the physics of granular material, it is difficult to model pile dynamics accurately. Small changes in applied forces on an object produce very different object motions. However, high-level force patterns consistently manifest, and we have named this a *granular environmental constraint*.

We show that granular EC exploitation replaces the control of objects' stability. Objects stabilize in front of the hand due to force patterns in the pile dynamics. When the stabilization force is large enough, one or multiple objects roll on the hand. In the presence of a strong stabilization force, a robot can grasp from a pile without visual or tactile perception and using simple open-loop control. Next, we explain why stabilization occurs, and later, we characterize the condition under which the stabilization forces support grasping with real-world and simulated experiments.

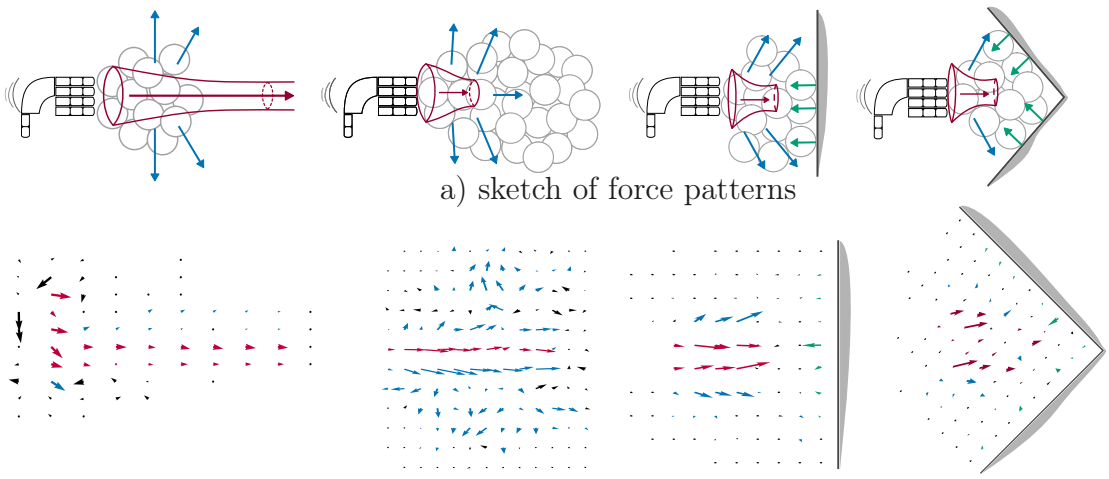
6.1.1 Effects of Granular Environmental Constraint Exploitation on Grasping

To show that pile dynamics provides a manipulation funnel, we follow abduction reasoning:

1. We describe an observed interaction regularity in pile dynamics when a hand pushes into a pile.

2. We propose a possible explanation for this pattern. Based on our explanation, we propose multiple testable hypotheses to show the existence of the granular EC and characterize the conditions for robust grasping with this EC.

3. We validate our hypotheses with real-world and simulated experiments.



a) sketch of force patterns

b) force patterns in simulation

Figure 6.3 All images show an interaction pattern in piles when pushed by an end-effector. *Top row:* the four sketches illustrate the object stabilization and rolling into a hand with red funnels. The red arrows show that objects inside the funnel move with the hand. The blue arrows depict the radial spread of the other objects outside the funnel. The green arrows show the contact normals on static walls in the direction of pushing. For large enough piles or piles supported by walls, interaction forces roll an object on the hand, and the pile expands less. *Bottom row:* we show a vector field depicting the interaction forces averaged throughout multiple simulations. We manually colored some of the arrows indicating similar patterns to the respective sketch. Even though the interactions are complex, a similar pattern to that depicted in the sketches can also be observed in the simulation.

Observed Patterns in Piles

Granular EC results from the interaction between a pusher, objects in a pile, and the environment. The interaction of these three actors creates a high-level force pattern. In the following, we present the observed high-level force pattern, explain the observed pattern using two properties of granular materials to derive testable hypotheses, and define an open-loop grasp strategy to analyze the granular EC empirically.

We observed an interaction regularity when a hand on a tabletop slides into a pile. Some objects in front of the hand were stabilized, i.e., moved together with the hand, while the other objects spread radially. With larger piles, one or multiple objects rolled into the hand, depending on the hand width and object size. With smaller piles, an object rolled into the hand when the environment constrained the pile's expansion.

In the top row of Figure 6.3, we depict the stabilization regularity as a funnel. The funnel entrance illustrates a region of objects that start moving together with the hand. The funnel walls represent a shrinking of the stabilization region. The shrinking of

the stabilization region symbolizes that the number of stabilized objects decreases; sometimes it decreases to a single object that becomes centered on the hand. Finally, the exit of the funnel indicates the point when an object rolls into the hand. Note that the funnel has no ending on the first sketch, indicating that objects move together with the hand and no object rolls into the hand.

At the bottom of Figure 6.3, a similar regularity can be observed where the data was obtained from simulation. In the simulation, a shovel-like end-effector pushed into a pile of spheres until it passed through the pile or an object rolled onto it. Here, we visualize four vector fields of interaction forces inside piles. For each cell in the field, we averaged the forces acting on objects in that cell over the whole simulation duration and repeated averaging for 100 simulations for each of the four cases. Then, we manually colored arrows that indicate object stabilization (red), radial expansion of other objects (blue), and objects' interaction with static vertical walls (green). The black arrows cannot be associated with any of the three cases mentioned before, indicating the complexity of the interaction forces. Similarities and differences between real-world and simulation experiments are discussed in Section 6.1.4.

Based on our observations, we propose three effects of granular EC exploitation concerning the predictability of the grasping process:

1. objects tend to stabilize in front of the hand,
2. an object becomes centered on the hand, and
3. geometrical and granular ECs interact.

With respect to observability, we define a grasping strategy that sequence open-loop controller based on contact events.

We provide a physics-based explanation of granular ECE effects to analyze it empirically. We consider a pile to be a granular material under the assumption that the objects are solid. We use granular materials' properties to explain the proposed effects of granular ECE.

Interestingly, even though it is difficult to predict the individual objects' motion when pushed accurately, a higher-level force pattern arises consistently. Therefore, we refrain from defining mathematical formulas to describe the regularity, but we use the following explanations to find relevant properties of piles, objects, and the environment that affect grasping with granular ECE.

In the following explanations, we consider that the hand and object sizes allow grasping one object at a time. Our explanations should hold when the hand is wider

or the objects are smaller: in these cases, multiple objects stabilize, center, and roll into the hand.

Physics-Based Explanation of Object Stabilization

Granular materials possess the *penetration resistance* (Stone et al., 2004) property. Penetration resistance provides an opposing force on a pusher and, consequently, on a pushed object as well. This opposing force manifests because the objects' kinetic energy dissipates inside granular materials, similar to the effect of the friction force between two objects. Kinetic energy dissipates due to friction between objects and the objects' inertia. Objects can stabilize, i.e., move together with the pusher, if the pusher's front part has a concave or flat shape.

Objects stabilize in front of a hand when pushed due to the penetration resistance. The concavities between the fingers also help this stabilization. When a hand pushes an object into a pile, the loss of kinetic energy generates an opposing force on the pushed object. If the opposing force is large enough, the pushed object rolls into the hand, after which the hand can close its fingers to grasp the object.

We analyze the conditions for grasping when leveraging the object stabilization effect of granular ECs. Since kinetic energy loss affects the amount of opposing force, we want to characterize the relationship between the cardinality of a pile (number of objects in a pile) and grasp success. We propose the following hypothesis:
Hypothesis 1: As the number of objects in a homogeneous pile increases, granular ECE better supports grasping, assuming that the size and mass of the objects relative to the hand allow grasping one object at a time.

Physics-Based Explanation of Object Centering

The second key property of granular materials concerns how an external force propagates through the material. An external pushing force propagates through short and temporal *force chains*, transferring the load. These force chains are parallel or radial for structured- and unstructured granular materials, respectively (Tordesillas et al., 2014). The force chains carry the opposing force, creating compression in front of the pusher.

With unstructured piles, a hand applies forces at multiple locations. Thus, a compression region develops in front of the hand. In this region, objects are pushed toward the hand, and objects roll away outside the region. Since the force chains are temporal, the region's width changes relative to the hand. Therefore, objects can shift laterally in front of the hand, and only a centered object remains stabilized. The changing nature of the compression region causes an object to center in front of the

hand. To analyze the object-centering effect on grasping, we analyze the importance of hand alignment to an object.

Hypothesis 2: A robot can grasp successfully from a homogeneous pile even without initially aligning its hand with an object when leveraging the granular EC, assuming that the size and mass of the objects relative to the hand allow grasping one object at a time.

With structured piles, the object-centering effect does not emerge because the force chains are parallel and constant. Therefore, the compression region is also constant in front of the hand. If the structure breaks, the compression region changes and the object-centering effect manifests. Since we are interested in the object-centering effect of granular ECs, our study focuses on unstructured piles.

Physics-Based Explanation of Granular and Geometrical EC Interactions

Static parts of the environment increase the penetration resistance of the granular material (Stone et al., 2004). The penetration resistance increases near static obstacles because of the contact normals and friction forces between the environment and objects.

When the environment constrains a smaller pile's expansion, the pile propagates the interaction forces between objects and the environment. We can leverage the interaction between a pile and the environment to facilitate grasping from smaller piles.

Hypothesis 3: A robot can grasp with granular ECE from smaller homogeneous piles, which alone provides an insufficient amount of opposing force if the pile is beside static walls assuming that the size and mass of the objects relative to the hand allow grasping one object at a time from large enough piles not supported by geometrical ECs.

The last two sketches in Figure 6.3 illustrate two cases where a small pile is constrained by a wall and constrained even more by a corner. In both cases, the penetration resistance increases when walls are present. A corner further constrains a pile's expansion, which we illustrate by having fewer blue arrows on the leftmost sketch.

In conclusion, granular EC manifests in piles due to kinetic energy loss. Therefore, our analysis of the granular EC focuses on properties of the pile, objects, and environment that influence kinetic energy loss, such as an object's inertia and the number of objects in a pile. Next, we define a grasp strategy that leverages granular EC, sometimes in combination with geometrical ECs, to evaluate the three proposed hypotheses.

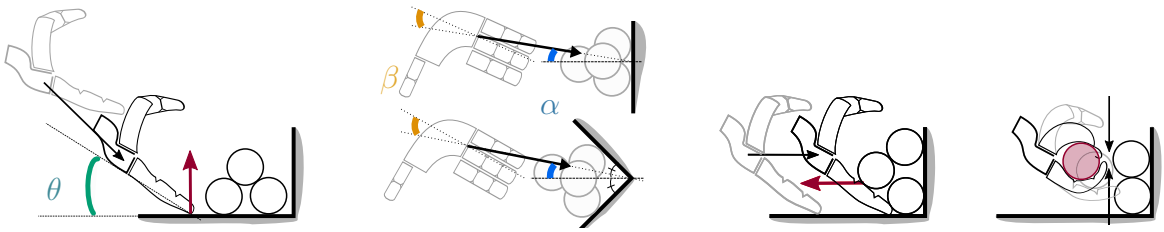


Figure 6.4 2D sketch of the grasp strategy in the order of execution: a hand *approaching* the pile from free space, where θ (green) is the finger's slope, α (blue) and β (yellow) are the relative hand orientation to the pile and a supporting wall or corner, and the hand's motion is visualized with black arrows. The force along the contact normal (vertical red arrow) triggers the *sliding* motion, which terminates when a contact force (horizontal red arrow) is detected, and finally, the hand *grasps* an object that rolled on it. © 2021 IEEE

Figure 6.5 Grasping from a pile of cylinders next to a wall. © 2021 IEEE

6.1.2 Definition of Granular EC Exploiting Grasp Strategy

We define a grasp strategy that uses granular ECE. We assume that granular ECE effectively stabilizes and centers an object. Thus, we omit controlling an object's position and the applied forces. Later, we use this strategy to characterize the effects of the granular ECE.

The strategy consists of three phases, as illustrated in Figure 6.4 and shown in Figure 6.5. First, the hand *approaches* a pile by lowering it to the tabletop with its palm facing up. Second, the hand *slides* on the horizontal support surface and pushes the pile. Finally, *grasping* triggers after the hand moves through the pile or reaches a static wall, and then, the hand stops moving and closes its fingers.

We parameterized the strategy with the hand's orientation (α, β, and θ), velocity $|\boldsymbol{v}_{\text{hand}}|$, sliding distance, and a force threshold F_{grasp}. The hand's orientation is relative to a pile and the environment, where α is the relative hand orientation to the pile. When the pile is next to a wall, α is the angle between the hand and the wall-normal.

With corners, α is the angle between the hand and the interior bisector of the corner angle. β is the offset angle between the fingers and the direction of motion, and θ is the slope on the fingers.

To analyze the object-centering effect, we define two instances of the above-described strategy: the *pile centered* and the *object centered* instances. The two instances differ in the way the hand approaches a pile. With the pile-centered strategy, a robot approaches a pile object agnostic, i.e., the hand is centered on the pile without considering any object's position. The strategy exploits the object centering and stabilization effects without explicitly controlling which object to be pushed on the pile's perimeter, and also, the motion of the pushed object is not controlled. In contrast, the object-centered strategy aligns the hand with an object during the approach phase. This strategy assumes that the centered object is grasped and exploits the object stabilization effect only. Therefore, we shift less control from the robot to granular ECE with the object-centered strategy when compared to the pile-centered one.

For the pile-centered strategy, the hand is lowered to a predefined location on a table in front of a pile. Even if the hand's pose is constant between executions, its relative pose to objects is random because we build random piles described in Section 6.1.3.

Concerning the object-centered strategy, first, an expert chooses an object on the pile's perimeter and then horizontally aligns the hand to the object before the sliding phase starts. For the hand alignment, the expert tuned the relative hand pose using 18 tennis balls in a pile and iteratively testing different hand alignments. A chosen ball's center was aligned with each gap between the fingers and the tip of the index or middle finger. The best alignment was when the index and middle finger gap aligned with the object's center. This outcome was used for piles of any size and for all object types.

6.1.3 Description of the Experimental Protocol

We analyze the granular EC with real-world and simulated experiments. We omit human grasping experiments to avoid biases like human reflexes. Humans can react uncontrollably to visual and tactile stimuli that affect pile dynamics. On the other hand, a robot's motion control can be open-loop by design. All experiments used the same parameters and followed the protocol unless stated otherwise:

1. building a random pile on a table,
2. executing the previously presented grasp strategy, and
3. observing if an object rolled on the hand during siding and after the fingers closed, also observing the number of grasped objects.

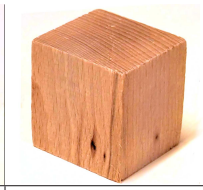
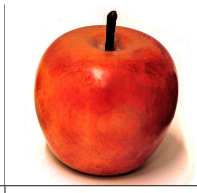
				
tennis ball	cylinder	cube	apple	lime
60 or 180 g	45 g	65 g	168 g	87 g
$r = 27$ mm	$r = 20$ mm $h = 50$ mm	$l = 45$ mm	$r = 70$ mm $h = 64$ mm	$r = 49$ mm $h = 68$ mm

Table 6.1 Objects' mass in grams and size in millimeters, where l, h, r are length, height, and radius, respectively.

Building Random Piles

We built a random pile to have a random hand-object alignment when executing the pile-centered strategy and to sample the grasp success. Since objects' configuration in a pile can affect the pile dynamics, we built unstructured piles with random object configurations by dropping the objects into a cube-like frame.

The frame was placed at a predefined location on a table. When vertical walls constrained a pile, the frame was placed next to a wall or corner. The frame size linearly increased with the pile's cardinality $|\text{Pile}|$ as follows: $0.77 \times |\text{Pile}| + 11.15$. We chose this frame size to build piles with a minimum of two layers of objects. We removed the frame, waited until the pile stabilized, and then executed the strategy.

The real object's properties are given in Table 6.1. All simulated objects had the same friction properties using the tennis ball's properties (Cross, 2002) for comparability between simulated and real-world experiments. The restitution coefficient (COR) of all simulated objects was a linear function of the mass, where $\text{COR}(60\text{g}) = 0.72$ (Cross, 2002), and $\text{COR}(180\text{g}) = 0.37$ using our experimental observation on our sand-filled heavy tennis balls.

Executing the Grasp Strategy

In real-world experiments, we used a Barrett WAM arm with seven degrees of freedom (DOF). We mounted an ATI FTN-Gamma force-torque sensor between the wrist of the robot and the end-effector. As an end-effector, we used a simple and rigid shovel or the compliant and anthropomorphic RBO Hand 2 (Deimel and Brock, 2016). In a simulation, we used the shovel-like end-effector, which was similar to the real one, as shown in Figure 6.6. We used the simple end-effector in the simulation because we expected to observe the granular EC even with this simple shovel. The simulated end-

Figure 6.6 The real shovel-like end-effector approaches a pile of tennis balls next to a wall (left) and similarly in a multi-body dynamics simulation where the pile is in a corner (right). © 2021 IEEE

effector had lateral friction 0.5, spinning-, rolling-friction 0.001, and COR = 0.8. We parameterized the strategy with $\alpha = \beta = 0°$, $\theta = 15°$, $|\mathbf{v}_{\text{hand}}| = 0.1m/s$, $F_{\text{grasp}} = 17N$, and the sliding motion terminated after 30 cm.

Observing Outcomes

We observed the *Roll Success* (RS) and *Grasp Success* (GS) in each experiment. We observed the roll and grasp successes because it is difficult to measure interaction forces inside a pile, and force-torque measurements are noisy. *Roll Success* is true if an object rolls on the hand such that the object's center of mass is above the fingers. *Grasp Success* is true if an object is grasped after the fingers close. We sampled these success rates from 20 observations on the robot and 50 in simulation.

Previously we proposed three hypotheses. We evaluate these hypotheses in the next section. We statistically analyze the results and report the respective p-values. We interpret a p-value as a relative strength of evidence rather than choosing a single threshold value for hypothesis testing. Similarly to Ganesh and Cave (2018), we interpret the ranges of p-values as follows:

- $p_{\text{value}} < 0.001$ indicates very strong evidence that our hypothesis explains the behavior and the null hypothesis can be rejected,
- $p_{\text{value}} < 0.01$ indicates strong evidence,
- $p_{\text{value}} < 0.05$ indicates moderate evidence,
- $p_{\text{value}} < 0.1$ indicates weak evidence, and

- $p_{\text{value}} \geq 0.1$ indicates insufficient evidence.

6.1.4 Experimental Characterization of Granular Environmental Constraints

We want to analyze granular EC in piles of nearly identical objects. We conducted an empirical analysis because piles' dynamics are challenging to model accurately, but force patterns manifest on a high level. We designed hypothesis-driven experiments to analyze the conditions that enable grasping from piles with granular ECE and also conducted explorative experiments to analyze other aspects of open-loop grasping:

- We show that granular ECE provides an opposing force and object centering by evaluating the previously derived Hypotheses 1 and 2.
- We analyze granular and geometrical EC interactions by evaluating Hypothesis 3.
- We analyze the factors that enable granular ECE concerning properties of a pile, objects, and end-effector.
- Finally, we discuss the sim-to-real gap concerning our analysis.

Larger Piles Provide Sufficient Opposing Force For Grasping

We analyzed conditions under which a pile provides enough opposing force on a pushed object for grasping with respect to a pile's cardinality. Previously we established that the opposing force results from kinetic energy loss. Since friction dissipates kinetic energy, if we increase the number of objects, then the number of contacts between objects also increases, so more kinetic energy dissipates. Therefore, if we increase a pile's cardinality, the granular EC better supports grasping. We formally express our first hypothesis as:

Hypothesis 1: There is a positive correlation between grasp success rate and pile cardinality when granular EC enables open-loop grasping.

To test our hypothesis, we observed the *roll* and *grasp* successes for piles with different cardinality. We sampled the success rates from piles with cardinality $|\text{Pile}| \in \{1, 5, 10, 14, 15\}$ of tennis balls of 180 grams on a horizontal surface. Because the object-centering effect of the EC is not in the scope of this experiment, we executed the object-centered strategy and considered an attempt successful if one or more objects were rolled or grasped.

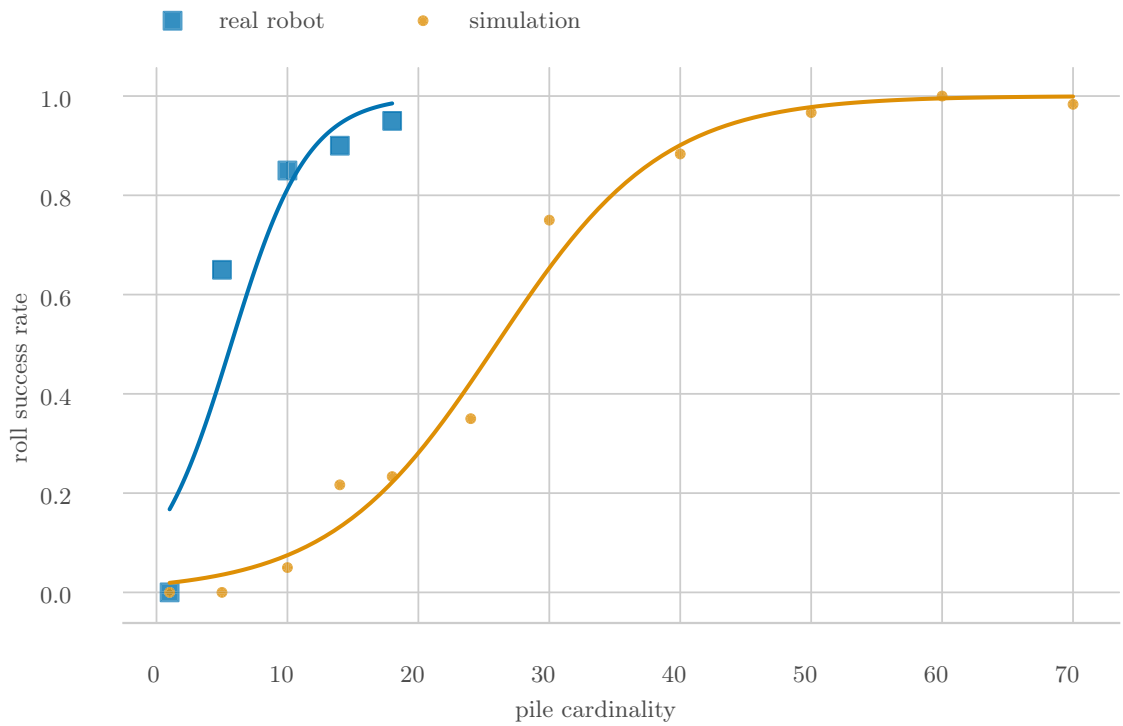


Figure 6.7 Piles with more objects, larger cardinality, enable grasping with granular EC exploitation. The pile's opposing force increases as the pile cardinality increases, which we observed in the increase in roll success rates both in the real-world (blue boxes) and in simulation (yellow dots) and with the fitted logical regression (solid lines) in both cases.

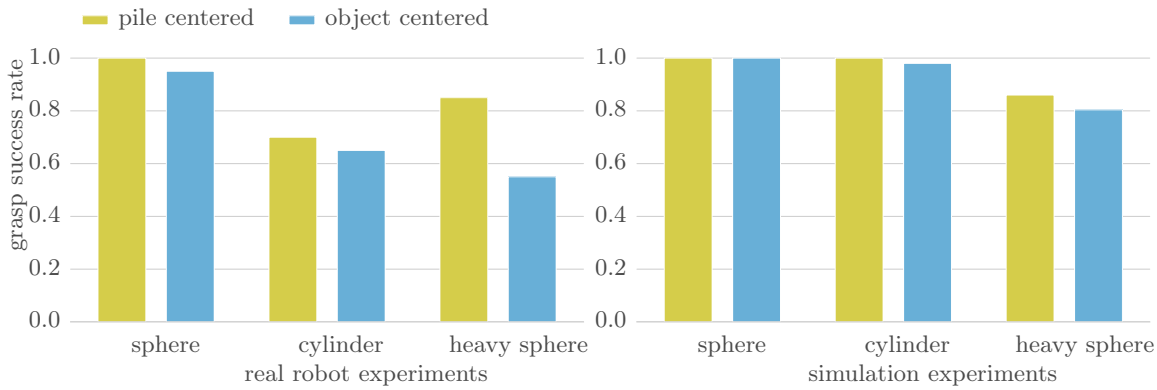


Figure 6.8 Both real robot and simulated experiments strongly indicate that granular ECs can replace controlling object centering. The pile-centered grasp strategy (yellow bars) performs as well as the object-centered strategy (blue bars) ,but it is somewhat better for heavy spheres.

						Spearman coefficient
|Pile|	1	5	10	14	18	N.A.
P(RS)	0	0.65	0.85	0.9	0.95	0.99
P(GS)	0	0.2	0.35	0.65	0.6	0.89

Table 6.2 We observed with the real robot a positive correlation between a pile's cardinality |**Pile**| for roll success (RS) and grasp success (GS) rates strongly indicating that larger piles provide larger opposing forces than smaller piles.

The results are summarized in Table 6.2 and shown in Figure 6.7. The figure shows the roll success rate (markers) for different pile cardinalities and the estimated logistic regression model (solid line). The roll success rate monotonically increased with the pile cardinality. The positive correlation with the Spearman coefficient of 0.99 is a very strong indication that the opposing force increases for larger piles.

The results also indicate a minimum pile cardinality for which the opposing force was large enough to support grasping with a simple straight pushing motion. We observed the same behavior in the simulation, but the minimum pile's cardinality was larger. We account this difference to the simulation gap, which we discuss in more detail in Section 6.1.4.

In Table 6.2, we can observe that the grasp success rate was less than the roll success rate. This is because the fingers closed after the hand stopped sliding. In some cases, the object rolled off due to its inertia before the fingers started closing. However, both observations very strongly support our hypothesis that the grasp success rate increases as the pile becomes larger due to the large positive Spearman correlation coefficients.

Granular ECE Provide Object Centering for Grasping

We want to show the existence of the object-centering effect of the granular EC. So, we compared the pile- and object-centered strategies for various piles. Recall that the pile-centered strategy omits explicitly centering the hand on an object, while the object-centered strategy explicitly does that. Therefore, differences in grasp success rates between the two strategies can indicate if object centering can be replaced (or not) by granular EC exploitation.

We compared the grasp success rates of the two strategies. We considered an attempt successful only if one object is grasped. If an object is not centered initially and not by the granular EC, then the hand grasps more than one object or none. We derive the following null hypothesis:

Null hypothesis: The *object centered* strategy is better than the *pile centered* strategy:

$$\mu(\mathrm{GS}_{\text{object centered}}) > \mu(\mathrm{GS}_{\text{pile centered}}),$$

where $\mu(\mathrm{GS})$ is the mean grasp success. We propose two alternative hypotheses. There is no significant difference between the grasp success rates for the two strategies. Alternatively, the pile-centered strategy performs better than the object-centered one.
Hypothesis 2a: The two strategies are equally good:

$$\mu(\mathrm{GS}_{\text{object centered}}) = \mu(\mathrm{GS}_{\text{pile centered}}).$$

Hypothesis 2b: The pile centered strategy outperforms the object centered one:

$$\mu(\mathrm{GS}_{\text{object centered}}) < \mu(\mathrm{GS}_{\text{pile centered}}).$$

We sampled the grasp success rate for three objects: light and heavy tennis balls and cylinders. The piles had 18 balls or 14 cylinders. Note that the hand can grasp two tennis balls and more than three cylinders. We built the pile beside a wall to increase the strength of the opposing force, which we analyze in detail in Section 6.1.4.

Both real robot and simulation results are shown in Figure 6.8. There are no statistical differences in grasp success rates between pile- or object-centered strategies for light tennis balls and cylinders. These results show that the granular ECE can replace object centering.

Moreover, we found weak evidence indicating that the pile-centered strategy was better than the object-centered one for heavy tennis balls, where the p-value is 0.041 from a single-sided Fisher's exact test. This suggests that it was difficult for the expert to properly model the pile's dynamics and center the hand relative to an object. At the same time, the granular EC provided better object centering.

Geometrical EC Enables Grasping From Small Piles With Granular ECE

We analyze the interaction between geometrical and granular ECs. When a hand pushes into a pile beside vertical walls (i.e., geometrical ECs), objects interact with the environment, and as a result, penetration resistance increases in a pile when the hand is also near a vertical wall.
Hypothesis 3: If a pile's expansion is constrained increasingly in opposition to the hand motion, the grasp success rate increases when leveraging the granular EC.

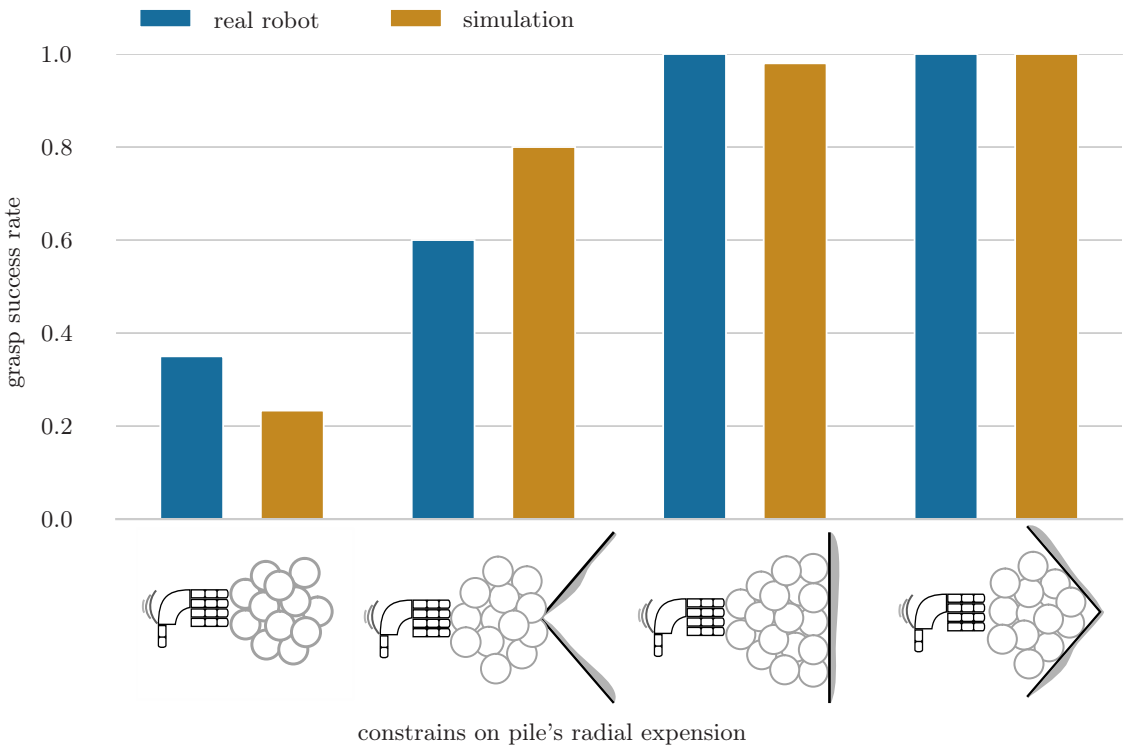


Figure 6.9 Static walls (geometrical ECs) increase the opposing force in small piles and enable grasping with granular EC exploitation.

To analyze our hypothesis, we sampled grasp success rates from increasingly constrained piles to expand. We constrained the pile expansion with two walls. To increase the expansion constraint, we changed the angle between the two walls. When there are no walls, a pile is the least constrained, 270° corner angle is less constrained than 180°, and 90° corner angle is the most constrained case, as shown at the bottom in Figure 6.9. We grasped from piles of 18 light tennis balls with the pile-centered strategy that slides the hand along a corner's bisector on the horizontal surface. We slid it along the bisector so that the wall's normal forces were symmetrical on both sides of the hand. We expect that grasp success increases as the pile's expansion is increasingly constrained.

The results show an increase in grasp success rate as the environment constrains the pile increasingly, as shown in Figure 6.9. Both real-world and simulation results show a strong correlation between grasp success and environment support, with Spearman coefficient > 0.94. Interestingly, a single vertical wall (or two walls with 180°) provided enough additional forces to achieve robust grasping from small piles.

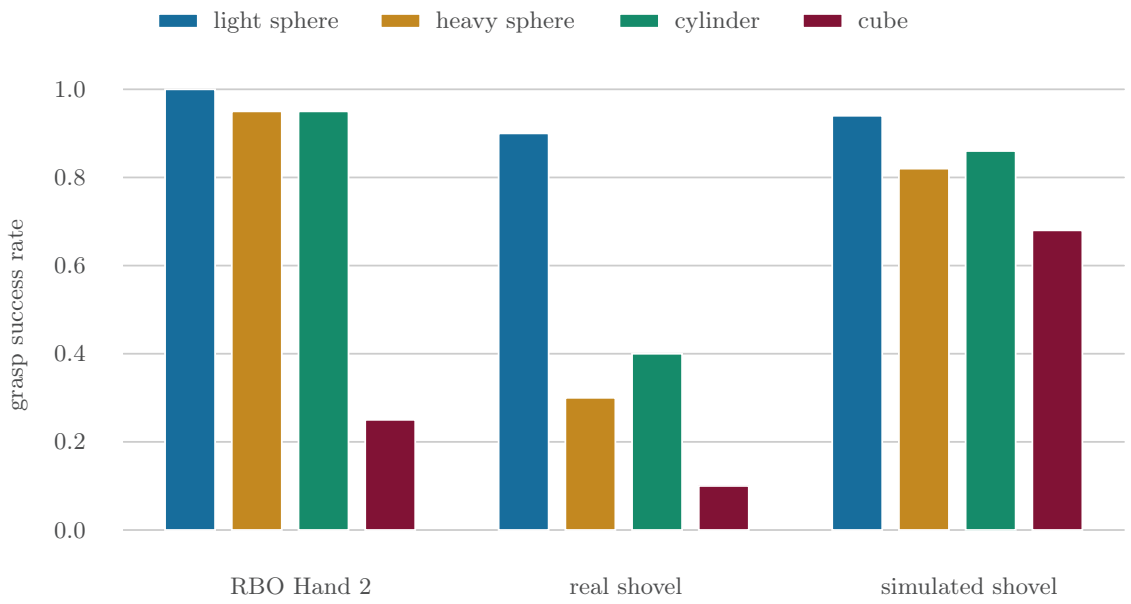


Figure 6.10 Granular EC exploitation is easier for round objects. The real robot performs best with the RBO Hand 2 for all round objects, while the real shovel can only exploit the EC for light tennis balls. © 2021 IEEE

Characterizing Granular ECE Concerning Object Inertia

We want to analyze open-loop grasping when exploiting the granular EC concerning an object's inertia. An object's inertia affects the pile dynamics and the object-hand interaction. The pile dynamics is affected because inertia is an object's property to resist changing its state when an external force is applied, affecting the amount of opposing force provided by the pile. The object-hand interaction is also affected because objects with increased inertia also resist change from stably moving with the hand to rolling (or sliding) on the hand.

To analyze the granular EC for piles of objects with different inertia, we executed the pile-centered strategy for piles of tennis balls of 60 or 180 grams, cylinders, and cubes. A pile's cardinality was 18 for tennis balls and 14 for cylinders and cubes. The piles were beside a wall so the opposing force supports grasping. Since the object-centering effect was not in the scope of the experiments and an object's inertia can hinder rolling an object into the hand, a grasp attempt was considered successful if one or more objects were grasped. We expect that the grasp success rate decreases when the object's inertia increases.

Figure 6.10 shows the grasp success rate for different objects and end-effectors. In all three cases, it is visible that a low-inertia object (a light tennis ball) was the easiest

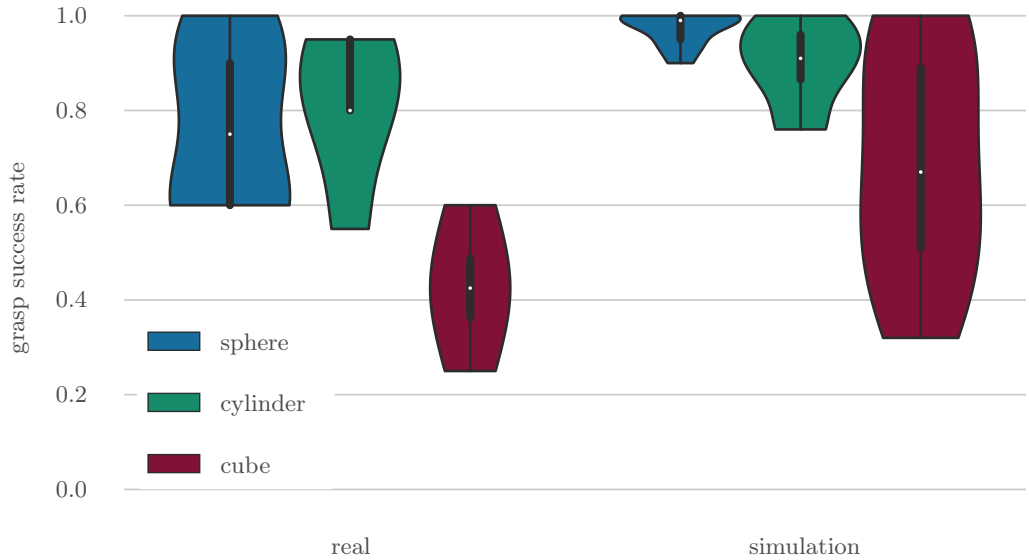


Figure 6.11 We can exploit the granular EC for round objects (blue and green) for a wide range of hand orientations, unlike cubes (red). We can use simulation to identify the EC because the behavior is qualitatively similar to the real world. © 2021 IEEE

to grasp. We can also note that increasing the inertia (increased mass or different shape) affected grasp robustness, but the end-effector's morphology greatly influenced this. For now, we only look at the effects of an object's inertia, and Section 6.1.4 discusses the impact of the end-effector's morphology.

We conducted an exploratory analysis to qualitatively analyze grasp robustness for a wide range of hand orientations relative to a pile next to a wall. We executed the pile-centered strategy from piles of spheres, cylinders, and cubes with various strategy parametrizations $(\alpha,\ \beta) \in \{(0°, 0°), (30°, \text{-}10°), (30°, 20°), (45°, \text{-}20°)\}$. We omitted the heavy 180 grams tennis ball because the other real object's weight is relatively closer to the light tennis balls. We built piles with cardinality $|\text{Pile}| \in \{14, 15, 18, 20\}$. We expect that open-loop grasping succeeds more likely for round objects.

In Figure 6.11, we compare the grasp success rate distributions between different object shapes using a kernel density estimation of the sampled grasp success rates. The violin plots were limited only to the observed data. The mean grasp success rates were $\geq 80\%$ for tennis balls and cylinders. While for cubes, it was $< 50\%$. The results show that granular EC exploitation is more robust for round objects because the distribution shifted down as object roundness decreased.

Similarly to our previous qualitative analysis, we analyzed the influence of an object's mass. We grasped from a variety of piles with different object's mass $m_{\text{object}} \in \{60, 180\}$

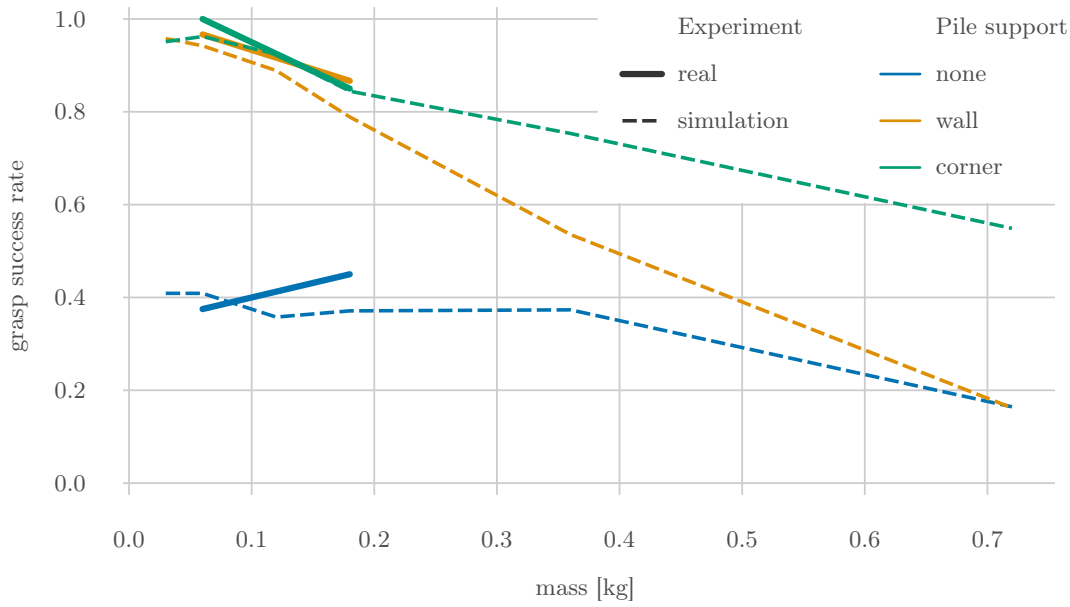


Figure 6.12 Lighter objects are easier to grasp even when the environment constrains a pile's expansion. © 2021 IEEE

in the real world, and in simulation $m_{\text{object}} \in \{30, 60, 120, 180, 360, 720\}$. We grasped from piles with various cardinality, in the real-world $|\text{Pile}| \in \{5, 18\}$, and in simulation $|\text{Pile}| \in \{3, 5, 10, 15, 20, 25, 30, 60, 90\}$, with and without a static wall or corner.

Note that in our previous work (Páll and Brock, 2021), we reported that grasping failed when walls did not support the pile. This failure occurred because the hand only closed if F_{grasp} was reached. However, smaller piles (without static walls) cannot provide a large enough opposing force to trigger grasping. Now, we extended the triggering condition with a maximum sliding distance. We acknowledge that the granular EC, like geometrical ECs, provides an observable discrete contact event. In our case, it indicates if an object rolls onto the fingers or the palm of the hand. However, we used an RBO Hand 2 equipped only with air pressure sensors that were not sensitive enough for our use case. We think that an appropriately sensorized hand can provide even greater benefits than detecting when an object rolls onto the hand, which we discuss in the final section of this chapter. We ran new experiments using the new condition when walls did not support a pile. We expect an increase in grasp success for lighter objects.

Figure 6.12 shows the grasp success rates with segment-wise linear interpolation. The results confirm that lighter objects are easier to grasp when the pile is next to a wall or corner, where the cumulative aligning forces of the granular and geometrical

ECs support grasping. When the environment has no walls, grasping heavier objects has a higher success rate. We think this is because the object's inertia contributes to penetration resistance besides friction. However, there is insufficient evidence to reject that the means are equal due to the 0.32 p-value of a single-sided Fisher's exact test.

We conducted a further experiment to show how to improve heavy object grasping. We grasped 18 heavy tennis balls in a corner because the corner further constrained the objects' motions. We expect that if we increase the hand's velocity and grasping force threshold, grasp robustness increases because heavier objects require more force to induce more interaction in a pile and roll an object on the hand. We increased grasp success from 60% to 85% by increasing the velocity $|v_{hand}|$ from 0.1 to 0.25m/s and force threshold F_{grasp} from 17 to 25N, showing weak evidence with a p-value is 0.07 of a single-sided test.

Characterizing Granular ECE Concerning the End-Effector Morphology

We want to analyze the influence of the end-effector's morphology (shape and structure) on granular EC exploitation. Therefore, we executed the pile-centered strategy with the anthropomorphic, compliant RBO Hand 2, and the shovel-like, rigid end-effector. We grasped from piles of 18 light and heavy tennis balls and 14 cylinders or cubes next to a wall.

Figure 6.10 shows the real and simulated mean grasp success rates. The real robot performed best with the hand. The hand achieved between 95% and 100% grasp success for round objects, while the shovel could scoop up only light tennis balls with 85%. Both end-effectors performed poorly for cubes, indicating a limitation of granular EC exploitation for cuboid objects. We think this drop in grasp success is because cuboid objects require more force to roll onto the hand due to their shape.

Though the shovel can exploit the granular EC, only the soft hand allows open-loop grasping. We needed an impedance controller that uses force feedback to slide the rigid shovel on a surface. As opposed to the rigid shovel, the soft hand provided enough compliance due to its structure that we could slide the hand using a simple operational-space controller.

Grasping with a soft hand had two more benefits. First, the space between fingers created a guiding rail for centering and rolling an object into the palm when the hand was pushed into the pile. Second, the softness of the hand allowed the fingers to adapt passively to the object's shape making the grasp stable.

We executed over 900 grasp attempts and observed grasps of multiple objects within one attempt. We grasped more than one object 15% for 600 successful grasps with the

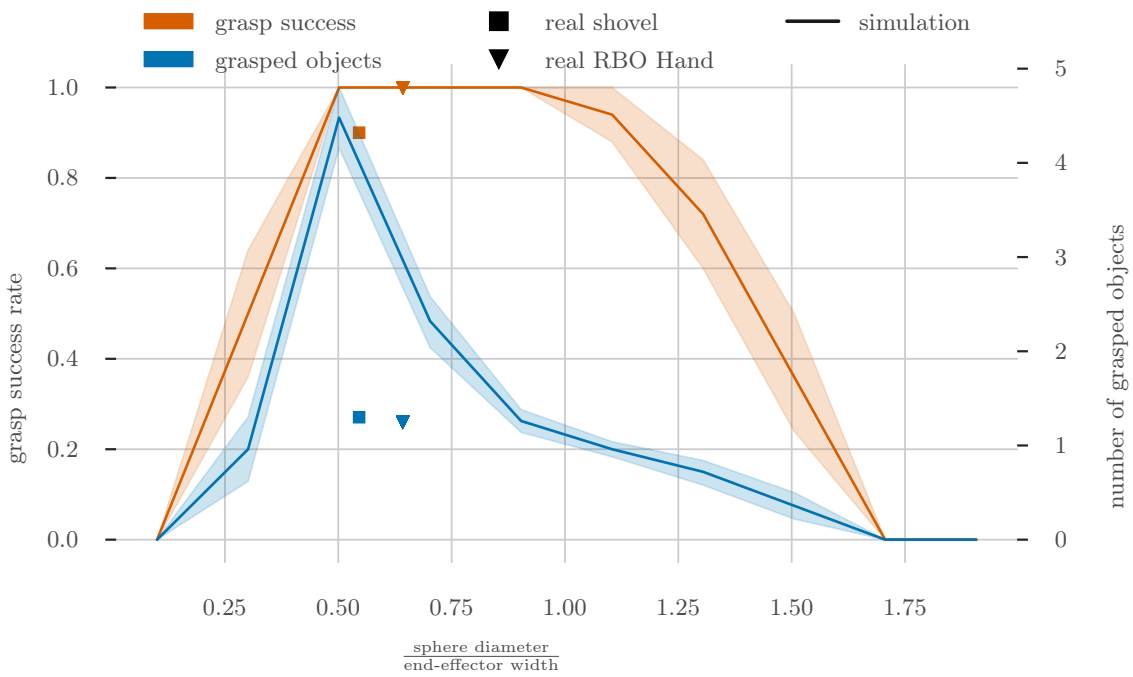


Figure 6.13 The grasp success and number of grasped objects depend on the ratio between the object's and end-effector's size shown for light spheres in a corner. © 2021 IEEE

RBO Hand 2 and 8.75% for 160 successful attempts with the shovel. The number of grasped objects and grasp success depended on the object's and end-effector's size, as shown in Figure 6.13. If we want to grasp a single object, only the hand can solve this with in-hand-manipulation to drop all but one object back into the pile.

Simulation and Real Observations Are Qualitatively Similar

We want to show that simulation is qualitatively similar to real-world open-loop grasping when analyzing the granular EC. Hence, we simulated open-loop grasping likewise to the real robot experiments. Previously, we presented experiments and results that included both real-world and simulation, and now, we interpret the results together.

In Section 6.1.4, we showed that the pile's cardinality and object stabilization by the granular EC are positively correlated. Larger piles provided more penetration resistance to rolling an object into the hand. We observed qualitatively similar behavior in simulation, as shown in Figure 6.7. However, the slope is shallower in simulation compared to the real-world results. We think the simulation gap can be accounted for inaccuracies in kinetic energy loss due to inaccurate friction simulation. We observed

qualitative similarities when analyzing the horizontal aligning forces in Figure 6.8, but simulated grasp success was generally higher than on the real robot.

When analyzing the effects of an object's inertia, we showed that object roundness improves granular EC exploitation in the real world. Figure 6.11 also confirms a qualitative similarity between real-world and simulation results. For both cases, the distributions' mean increased with an object's roundness. We can observe qualitative similarity concerning object roundness in Figure 6.10 as well, where we compared different end-effectors. However, this experiment shows a significant sim-to-real gap between real- and simulated-shovel, similarly between the hand and simulated shovel for cubes. Hence, simulation cannot indicate an end-effector's limitation to exploit granular EC.

The real-world and simulation results indicate (see Figure 6.12) that lighter objects are easier to grasp when supported. In the simulation, we also showed the relationship between grasp success and relative object size to the end-effector (see Figure 6.13).

We analyzed how granular ECs interacted with geometrical ECs and showed that walls enabled robust grasping from small piles, as shown in Figure 6.9. A strong positive correlation was visible for both real robot and simulated experiments, where both experiments had Spearman coefficient > 0.94.

With respect to sim to real gap, the results show that simulation qualitatively captures the high-level force pattern in piles. Thus, we can analyze granular EC in simulation. On the other hand, the quantitative differences show that we must combine simulation with real robot experiments to characterize the EC, choose an appropriate end-effector, and synthesize a grasp strategy for granular EC exploitation.

In summary, we have shown that pile dynamics produce a manipulation funnel because a high-level and consistent interaction force pattern restricts objects' motion in a pile, i.e., granular EC. Granular ECE has all four properties of ECE. It implicitly reduces uncertainty about an object's position in a pile relative to a hand pushing into the pile. It produces observable contact events on the hand, indicating a grasping outcome. It structures the robot's workspace into a region from where the granular EC can be reached (the horizontal support surface under the pile) and a region where it can be leveraged (the pile itself). Granular ECE enabled robust grasping by augmenting a simple straight pushing motion of the hand to separate and stabilize an object. Moreover, its use simplified control and perception such that grasping was robust using open-loop controllers without using visual or tactile perception for object detection.

Next, we define a grasp planner that uses granular and geometrical ECE and explain how our findings from this empirical study simplified our grasp planner.

6.2 Grasp Planning With Environment Constraints

We want to simplify grasp planning from piles by using the gained insights from the previous study about granular ECE. First, we define the granular and geometrical EC-based grasping problem. Since force pattern manifest on a high level, we can abstract away details of pile dynamics from the representation of the planning problem. And as a result, planning becomes simpler. Then, we present an ECE grasp planner that uses the high level nature of the force patterns and that granular and geometrical ECs implicitly structure the workspace into regions. Finally, we apply granular ECE in a real-wold industrial application.

The proposed representation and grasp planner builds on the work of Eppner and Brock (2015) on geometrical ECE-based grasping, and we extend the previous work from single object grasping to grasping from piles of nearly identical objects.

6.2.1 Grasping Problem Definition

Since our open-loop grasp strategy (in Section 6.1.2) leveraged a variety of geometrical and granular ECs, we start with a formal definition of ECE. Eppner and Brock (2015) defined an ECE, Υ, to be a contiguous subset of all possible hand poses, object poses, and the exerted force onto the object by the hand and environment:

$$\Upsilon \subset \mathcal{C}_{\text{hand}} \times \mathcal{C}_{\text{object}} \times \mathcal{F}_{\text{object}},$$

where $\mathcal{C} \in SE(3)$ is a set of configurations and $\mathcal{F}_{\text{object}}$ is the 6D wrench space of the object. To sequence ECEs for grasping, we can evaluate ECE connectivity by computing their intersection. Two ECEs are connected if their intersection is not empty:

$$\Upsilon_i \cap \Upsilon_j \neq \emptyset.$$

This representation captures the interaction force pattern of an ECE as a constraint function $f(\mathcal{F}_{\text{object}}) \geq 0$, and thus, it allows a form closure analysis. However, planning with this representation requires enormous amounts of computation because of the high dimensionality of the state space. If we represent a 3D configuration with a

position vector and three angles $\mathbf{q} = [x, y, z, \alpha, \beta, \theta] \in \mathcal{C}$ for $\mathcal{C}_{\text{hand}}$ and $\mathcal{C}_{\text{object}}$, then the dimensionality of the state space is $\mathbb{R}^6 \times \mathbb{R}^6 \times \mathbb{R}^6 = \mathbb{R}^{18}$.

Because of this high dimensionality, we want to simplify the representation using a series of approximations, and for realism, we based these assumptions on our empirical study of granular ECE.

The first and foremost assumption is that granular ECE supports grasping, i.e., the hand and object properties (such as mass, size, shape, and morphology) are adequate for EC-based grasping with our open-loop strategy as discussed in Section 6.1.4. The following simplifications are the consequence of this first assumption.

We simplified $\mathcal{C}_{\text{hand}}$ because our grasp strategy was very structured within the exploited ECs (a pile, a horizontal surface, and walls) while it achieved robust grasping for a set of hand positions and orientations, as presented in Section 6.1.4. Therefore, we simplified the $\mathcal{C}_{\text{hand}}$ by representing the hand's position with an oriented bounding box *obb* and its orientation with a discrete set of angles R.

We omitted $\mathcal{C}_{\text{object}}$ and $\mathcal{F}_{\text{hand}}$ from the representation. We have shown previously that granular ECE replaces control over an object's position and stabilization. Since our grasp strategy does not use information about the object's position (e.g., not used as feedback in control), we could omit the object configuration $\mathcal{C}_{\text{object}}$ from the representation. Furthermore, since the granular ECE replaces an object's stabilization control, we omitted $\mathcal{F}_{\text{object}}$.

Since we omitted $\mathcal{F}_{\text{object}}$ but observable contact events are key to sequencing ECEs, we added the 6D wrench of the hand $\mathcal{F}_{\text{hand}}$ to represent contact events that are measurable by sensors on the robot. For geometrical ECE, we associated force measurements with contact normals of the respective surfaces. For granular ECs, we can associate a force in the direction of the hand's motion with the high-level opposing force that rolls the object on the hand or an object's weight on the hand.

As a result of these simplifications, the approximated ECE $\widetilde{\Upsilon}$ is a contiguous subset of possible hand poses and the exerted force or torque onto the hand:

$$\widetilde{\Upsilon} \subset \widetilde{\mathcal{C}}_{\text{hand}} \times \mathcal{F}_{\text{hand}},$$

where $\mathbf{q} = \{obb,\ R\} \in \widetilde{\mathcal{C}}_{\text{hand}}$ is an approximated hand configuration and $\mathcal{F}_{\text{hand}} \in \mathbb{R}^6$ is a 6D wrench.

Both granular and geometrical ECEs are manipulation funnels because they reduce uncertainty implicitly and mechanically. Using a physical funnel analogy for granular and geometrical ECEs, the *entrance* of these funnels are any hand pose $\boldsymbol{q} \in \widetilde{\Upsilon}$ where a strategy-specific motion results in $\mathcal{F}_{\text{hand}} \in \widetilde{\Upsilon}$. An object moves through a physical

funnel either constrained by its walls or between the wall. In the case of ECE, we move on the funnel's wall (active constraint exploitation) by leveraging the force pattern, which is represented as:

$$\widetilde{\Upsilon}(\boldsymbol{q}, \mathcal{F}). \tag{6.1}$$

An ECE funnel *exit* is the termination of an ECE usage, and it is indicated by a contact event measured as a discrete force change on the hand $\delta\mathcal{F}_{\text{hand}}$.

Important to note that we omitted to explicitly model uncertainty because we assumed that a sequence of ECE reduces uncertainty below the tolerance for robust grasping. This assumption is realistic for piles of round objects, as we have shown it empirically.

The transition condition between two approximated ECEs $\widetilde{\Upsilon}_i$ and $\widetilde{\Upsilon}_j$ can be defined as follows:

$$\widetilde{\Upsilon}_i \mapsto \widetilde{\Upsilon}_j \overset{\text{iff}}{\iff} \widetilde{\Upsilon}_i \cap \widetilde{\Upsilon}_j \neq \emptyset \begin{cases} obb_i & \cap & obb_j \neq \emptyset \\ & \text{and} & \\ R_i & \cap & R_j \neq \emptyset. \end{cases} \tag{6.2}$$

Based on the transition operator "$\mapsto$", we defined the *planning* problem as a workspace-connectivity search problem. We are going to represent the workspace connectivity with a directed graph, and we will show an example of such a graph in a bin-picking application (see Figure 6.14). The nodes of such a graph are ECEs, and the edges are the possible transitions between ECEs.

Note that we made no assumptions about a pile's cardinality because a single object can be considered a pile with cardinality one. Our simplifications of the representation hold for this edge case as well because our assumptions for geometrical ECs are independent of the pile's cardinality. With respect to granular EC assumptions, even a single object produces an opposing force on the hand due to its inertia, but this force does not necessarily center or roll the object onto the hand. However, a static wall or corner enables grasping a single object with our strategy because our strategy, for $\alpha = \beta = 0°$, is analogues to the wall grasp strategy presented by Eppner and Brock (2015), and they have shown robust grasping of a single round object when supported by a static wall.

Surprisingly, our simplifications of the problem representation lead to the same representation as for single object grasping using geometrical ECE alone (Eppner and Brock, 2015) while we used both geometrical and granular ECE. This is because both types of ECs, geometrical and granular, share the same properties (Section 1.2) when leveraged.

Algorithm 8 Geometrical and Granular ECE Grasp Planner

Input: PCL, c_{hand}
Output: HA
1: $V \leftarrow \emptyset$ *init the list of nodes*
2: $E \leftarrow \emptyset$ *init the list of edges*
3: $V \leftarrow$ DETECT_ECs(PCL) *add detected ECs as nodes to the graph G*
4: **for** $(v_i,\, v_j) \in V | i \neq j$ **do** *find edges between nodes*
5: **if** $v_i.obb \cap v_j.obb \neq \emptyset$ AND $v_i.R \cap v_j.R \neq \emptyset$ **then** *based on Equation 6.2*
6: $E \leftarrow \{(v_i,\, v_j)\}$ *add edge between two connected ECs*
7: $\pi \leftarrow$ PDDL_SOLVER($G(V, E), c_{\text{hand}}$) *find a sequence of ECE for grasping*
8: $HA \leftarrow$ CONVERT(π) *convert grasp strategy to executable hybrid automaton*
9: **return** HA

6.2.2 Granular EC Exploiting Grasp Planner

Based on the previous problem definition, we present the granular ECE grasp planner in Algorithm 8. The planner builds a graph with nodes being ECEs and edges being possible transitions between two ECEs. First, it detects existing ECs from point cloud data obtained from an RGB-D camera. Then, it verifies the connectivity between all pairs of detected ECs and adds the respective edges to the graph. Next, it searches for an ECE sequence leading to a successful grasp. Finally, the planner converts the ECE sequence into an executable plan. Below, we discuss the three main steps: detection of ECs, graph search for a solution, and conversion of the solution into an executable plan. The numbers in parentheses refer to the lines in Algorithm 8:

Detecting ECs (line 3) For geometrical EC detection, we used the method developed by Eppner and Brock (2015) that extracts horizontal support surfaces, vertical walls, concave and convex edges, and an oriented bounding box for a target object. They assumed that only one object is in the scene. Similarly, we assumed that only one pile was in the scene. Therefore, we could detect a pile similarly to single object detection. For pile (or single object) detection, the points associated with geometrical ECs were subtracted from the point cloud data. This assumption implies that the remaining points are from the pile, and one can easily compute an oriented bounding box on the remaining points. Moreover, the number of objects in a pile could be estimated with point cloud volume estimation (Chang et al., 2017). However, we assumed to know this information. The pile cardinality becomes relevant during the graph search step.

Even though EC detection is not in the scope of this thesis, it brings up a very important observation. We need a tight integration between perception and planning

to exploit ECs because we need to plan for an instance of a grasping problem. We will discuss this implication in more detail in Chapter 7.

Creating an ECE-graph (lines 4-6) The planer built an ECE-graph based on the detected ECs and possible transitions. Since we could reason about the ECE regions and their connectivity symbolically, the grasp planning became a symbolic search problem of finding interconnected ECEs that lead to grasping success.

Solving a PDDL (line 7) The symbolic search problem was represented with Planning Domain Definition Language (PDDL) (McDermott et al., 1998) that is a less restricted version of the STRIPS (Fikes and Nilsson, 1971). There exist numerous extensions of PDDL, and we used the PDDL2.1 definition to represent the pile cardinality because this version integrates numerical fluents to model non-binary resources.

A PDDL problem is defined with a domain and a problem description. Our domain consisted of three sets of object types T, predicates P, and actions. Our problem description defined the initial conditions and the goal state. The object types T and predicates P were:

$$T = \left\{\text{hand}, \widetilde{\Upsilon}\right\},$$

$$P = \left\{\begin{array}{c}\text{Connected}(\widetilde{\Upsilon}_i, \widetilde{\Upsilon}_j),\\ \text{Hand}(\widetilde{\Upsilon}),\\ \text{GraspProbability}(\widetilde{\Upsilon}),\end{array}\right\},$$

where `Connected(·,·)` was true if the two given ECs were connected, `Hand(·)` was true if the hand was currently exploiting the specified EC, and `GraspProbability(·)` was the grasp success probability for granular ECE or the combination of a granular and a geometrical ECE. In Section 6.1.4, we have shown that the granular and geometrical ECs interact, and this interaction increased the grasp success probability for piles with low cardinality. So, `GraspingProbability(`$\widetilde{\Upsilon}_{i\in\text{wall, corner}}$`)` $= g(\sphericalangle_{\widetilde{\Upsilon}_i})$ was in function of the angle between two walls, and for a single wall, the angle was 180°. When a granular EC is used alone, `GraspingProbability(`$\widetilde{\Upsilon}_{\text{pile}}$`)` $= h(|pile|)$ was a linear function, and for piles above twenty objects was 100, based on our findings in Section 6.1.4.

In a domain definition, actions have parameters, preconditions, and effects. Actions are universally quantified and apply to any state in which the precondition holds. We defined three actions:

$$\begin{aligned}
&\textit{Action}\Big(\texttt{MoveHand}(\widetilde{\Upsilon}_{from}, \widetilde{\Upsilon}_{to}),\\
&\qquad \textsc{Precond}:\ \texttt{Hand}(\widetilde{\Upsilon}_{from}) \wedge \texttt{Connected}(\widetilde{\Upsilon}_{from}, \widetilde{\Upsilon}_{to})\\
&\qquad \textsc{Effect}:\ \neg\texttt{Hand}(\widetilde{\Upsilon}_{from}) \wedge \texttt{Hand}(\widetilde{\Upsilon}_{to})\Big),
\end{aligned}$$

$$\begin{aligned}
&\textit{Action}\Big(\texttt{ConnectPile}(\widetilde{\Upsilon}_{to}),\\
&\qquad \textsc{Precond}:\ \texttt{Hand}(\widetilde{\Upsilon}_{surface}) \wedge \texttt{Connected}(\widetilde{\Upsilon}_{surface}, \widetilde{\Upsilon}_{pile}) \wedge\\
&\qquad\qquad \texttt{Connected}(\widetilde{\Upsilon}_{surface}, \widetilde{\Upsilon}_{to}) \wedge \neg\texttt{Connected}(\widetilde{\Upsilon}_{pile}, \widetilde{\Upsilon}_{to})\\
&\qquad \textsc{Effect}:\ \texttt{Hand}(\widetilde{\Upsilon}_{pile}) \wedge \texttt{Connected}(\widetilde{\Upsilon}_{pile}, \widetilde{\Upsilon}_{to})\Big),\\
&\textit{Action}\Big(\texttt{GraspObject}(\widetilde{\Upsilon}_{with}),\\
&\qquad \textsc{Precond}:\ \texttt{Hand}(\widetilde{\Upsilon}_{pile}) \wedge \texttt{Connected}(\widetilde{\Upsilon}_{pile}, \widetilde{\Upsilon}_{with}) \wedge\\
&\qquad\qquad \texttt{GraspingProbability}(\widetilde{\Upsilon}_{with}) > 100 - \epsilon\\
&\qquad \textsc{Effect}:\ \texttt{ObjectGrasped}\Big).
\end{aligned}$$

The `MoveHand(·, ·)` action transitioned the hand between two ECE. The second action `ConnectPile( · )` connected the pile with a static wall or corner by moving a portion of a pile next to a geometrical EC. The action created a new edge in the graph, and so it enabled grasping with the respective geometrical EC in combination with the granular EC. While moving the pile was not explicitly analyzed in our empirical study, we have shown in Section 6.1.1 that a small portion of a pile stably moved in front of the hand when pushed. Therefore, a portion of a pile can be moved next to a geometrical EC. The last action `GraspObject( · )` was parameterized by an ECE: $\widetilde{\Upsilon}_{pile}$ was a granular ECE without considering interactions with any geometrical ECs. For logical correctness, we assumed that a granular ECE was connected with itself. With $\widetilde{\Upsilon}_{wall}$ and $\widetilde{\Upsilon}_{corner}$, the granular ECE was used in combination with the chosen vertical support if there was a connection between the two ECEs. Note that the three actions ensured that ECEs could only be sequenced if connected.

Finally, the initial state s_0 and the goal state s_g are represented as a logical formula, which is a conjunction of ground and functionless atoms:

$$
\begin{aligned}
s_0 \Big(& \texttt{Hand}(\widetilde{\Upsilon}_{hand}) \wedge \\
& \bigwedge_{\substack{\widetilde{\Upsilon}_i, \widetilde{\Upsilon}_j \in \widetilde{\Upsilon}, \\ \widetilde{\Upsilon}_i \mapsto \widetilde{\Upsilon}_j}} \texttt{Connected}(\widetilde{\Upsilon}_i, \widetilde{\Upsilon}_j) \wedge \\
& \quad \texttt{GraspingProbability}(\widetilde{\Upsilon}_{pile}) = f(|pile|) \wedge \\
& \quad \texttt{GraspingProbability}(\widetilde{\Upsilon}_{i \in \{wall, cornere\}}) = g(\lhd_{\widetilde{\Upsilon}_i}) \Big), \\
s_g(\texttt{ObjectGrasped}),
\end{aligned}
$$

where $\widetilde{\Upsilon}_{hand}$ was the initially exploited EC by the hand, and $\widetilde{\Upsilon}$ without an index was the set of all detected ECs. We used the closed-world assumption, meaning that all atoms not mentioned in a state are false. For example, `ObjectGrasped` was not part of the initial state, so the hand was empty initially.

Following this definition, the planning problem was reduced to a graph search where the nodes were an EC exploitation, and the edges were the allowed transitions. The search objective was to maximize grasp probability in as few steps as possible. One can find an ECE sequence using a graph search algorithm, such as A* (Hart et al., 1968).

The graph was obtained directly from visual perception, which creates a strong connection between the planner and visual perception. Therefore, we plan for what we see: an instance of a grasping problem.

Generating the executable strategy (line 8) The symbolic ECE sequence was directly convertible into a hybrid automaton (Henzinger, 2000) for execution. A hybrid automaton is a mathematical representation to execute a sequence of continuous processes with discrete switches between these processes. Since an ECE was realized with one (or multiple) continuous controller(s), a node of the graph could be mapped to a set of controllers. Moreover, transitions between two ECEs were contact events, so the switching condition between two sets of controllers was a discrete change in force measurements.

Note that the solution of our grasp planner easily generalizes to different hardware setups because the hybrid automaton provides an interface for different hardware components, which we discuss in detail in Chapter 7.

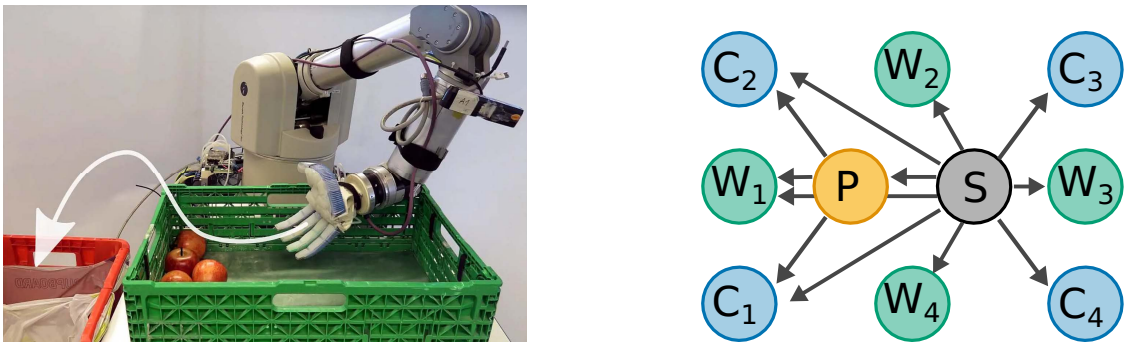


Figure 6.14 *Left:* Granular ECE-based grasping in the Ocado's bin picking use cases (Mnyusiwalla et al., 2020) © 2021 IEEE. *Right:* ECE-graph for the use case on the left, where nodes labeled W stand for a wall, C for a corner, S for a surface, and P for a pile; edges are arrows indicating the possible transitions between ECEs.

In summary, we simplified the general representation of ECE-based grasping using realistic assumptions based on our empirical study about the granular EC and its usage. The predictable manipulation behavior of granular and geometrical ECE allowed abstracting away physics from the representation. The structural context represented by ECE allowed reasoning about regions of the workspace rather than actual states.

The presented grasp planner is extremely simpler than classical approaches, like form or force closure, because some aspects of computation classically done by the robot (e.g., searching for contact points and evaluating these points) are optional (or futile) when using granular and geometrical ECEs. Moreover, the planner ignored uncertainty which contributes to its simplicity as well. Uncertainty was not modeled because the used ECEs reduced positioning and motion inaccuracies, and they made grasping invariant of modeling inaccuracies.

Next, we apply granular ECE-based grasping in an industrial application and show that co-designing the environment for granular ECE further simplifies perceptual requirements.

6.2.3 Application of Granular ECE-Based Grasping

Bin-picking tasks are ideal for leveraging geometrical and granular ECs because a bin has static walls, and the goods are stored in piles in many cases. Mnyusiwalla et al. (2020) have presented such a use case to evaluate robotic hands and grippers for pick and place operations. The use case was inspired by the requirements of Ocado's warehouse fulfillment center, where employees pack customer orders: First, an employee receives a grocery shopping list. Then, a bin arrives at the picking station containing

Objects	**tennis ball 60 grams**	**tennis ball 180 grams**	**cylinder**	**net bag of limes**	**apple**
R/N	4.3/4	5/4	9/4	5.6/4	7.6/4

Table 6.3 We leverage granular ECs in a bin picking application, where we measured R the average grasp attempts to pick and place $N = 4$ objects of the same type.

one type of grocery item from the list. Next, the employee places the given number of items into a delivery bag, and the process repeats itself for another item on the list.

We used Ocado's industrial grocery logistics use case (Mnyusiwalla et al., 2020) to show the applicability of granular ECE-based grasping.

Like in the Ocado use case, our robot had to fulfill an order by picking N items from a bin and placing it into a delivery bag, as shown on the left in Figure 6.14. Only one type of good was stored in the bin. The bin's location was considered known, so geometrical EC detection was not required, and the ECE-graph was pre-defined, as shown on the right in Figure 6.14. We measured the system's efficiency with R the average grasp attempts to fulfill an order of N items of the same type. In our adaption of the use case, the items were apples, tennis balls, cylinders, or net bags of limes (see Table 6.1). The order was of four items of the same type: $N = 4$.

The key difference to Ocado's use case was that we tilted the bin 5°, so the pile remained beside the same wall after each grasp attempt. This way, we could consider a pile's location known, and thus, visual detection of the pile was not required. We justify changing the environment because humans design their tools to maximize their utility. The robot uses the environment as a tool, so it is reasonable and advised to change the environment to increase its utility for granular EC exploitation. Compared to our empirical experiments in Section 6.1.4, we increased the slope of the fingers from $\theta = 15°$ to 30° due to kinematic and environmental limitations.

Before each grasp attempt, an operator decided to leverage the granular EC with a wall or corner and executed the pile-centered strategy. If the robot grasped more items than required, the robot dropped the grasped objects back into the pile. For example, the robot already picked and placed three apples out of four. If the next time it picks two apples in one attempt, it drops both apples back into the pile. Since we allowed the robot to grasp more than one item in one attempt and our hand can grasp two to three objects simultaneously, the ideal R value is between two and four.

We executed four times an order fulfillment request per each object type. The results are shown in Table 6.3 and in this video[1]. On average, the robot needed 6.3

[1] https://youtu.be/BtCBNdkgejU

grasp attempts to pick and place four items indicating that granular ECE is applicable in bin-picking applications. It also indicates that granular ECE generalizes for irregular object shapes like apples and net bags of limes.

We evaluated the symbolic planner by running it 100 times to grasp a light tennis ball from a pile for the described application on a desktop computer with an Intel i5 3.5GHz processor. The average computation time was 6.75 ms to select a grasp strategy. While this is an impressive result, there is no free lunch with the applied simplification, and we discuss related limitations in Section 6.4.1.

6.3 Related Grasp Approaches

We analyzed a novel environmental constraint that replaces some aspects of control, perception, and planning when leveraged for grasping. Thus, we discuss related work concerning how different aspects of control, perception, and planning are distributed between the robot and the environment in various grasping approaches.

6.3.1 Form and Force Closure Grasp Approaches

Form- and force-closure approaches solely rely on the robot to plan and control grasping based on perfect knowledge about the object or perfect perception to identify the physical properties of the object. Such approaches first compute desired contact points and applied forces between fingers and an object, considering physics, kinematics, and dynamics. Then, the robot accurately positions its fingertips on an object and applies the desired forces. A large body of research exists, including optimality analysis (Zheng, 2013; Guay et al., 2014; Lévesque et al., 2018; Jia et al., 2017) where robots control the grasp and even resist external wrench loads. Because the robot models all aspects of the interactions, these methods can analyze the robustness of a grasp strategy and provide mathematical stability guarantees.

Even though these approaches can provide stability guarantees, they are difficult to apply in our context when leveraging pile dynamics for grasping. Since form and force closure approaches need to model the interaction physics accurately, these approaches also need to know the object pose and geometry accurately. The pose of multiple objects is occluded inside a pile, making it difficult to accurately model the interaction in piles. Therefore, we searched for high-level patterns. The high-level pattern manifesting consistently allowed us to abstract away details of the interaction physics, and as a result, we replaced aspects of control from the robot to the environment.

6.3.2 Machine Learning Grasp Approaches

Machine learning grasp approaches can replace aspects of control. Recent research has shown impressive grasp performance (Mahler et al., 2019). In this research, a robot learned from a large amount of training data to grasp from piles of non-identical objects. Researchers collected data from simulated and real-world experiments. When experiments capture beneficial EC exploitation, by chance or design, the learned grasp strategy can include EC exploitation. If EC exploitation is in the learned strategy, the machine learning technique shifts aspects of control and perception from the robot to the environment.

When the data contains EC exploitation by chance, it is difficult to extract which aspects of the control can be shifted from the robot to the environment. In contrast, when we know about the beneficial effects of ECs, we can use these as an inductive bias for learning. This bias should reduce the amount of data required for learning the policy. Our empirical research closes the knowledge gap when grasping from piles by characterizing the granular ECE so researchers can use our insights as an inductive bias.

6.3.3 Environmental Constraint Exploiting Approaches

Researchers investigated how humans leverage contact with the environment when grasping and successfully transferred human-like grasp strategies to robots. These robots leveraged static surfaces and their edges as constraints, i.e., geometrical ECs. Static surfaces and concave edges constrain the motion of an object in the direction of surface normals which was leveraged to stabilize the object for grasping (Deimel et al., 2016). Moreover, a surface provides support to pivot or reorient an object (Odhner et al., 2013; Chavan-Dafle and Rodriguez, 2015), and a concave edge can expose a portion of the object for pinch grasping (Kappler et al., 2010; Hang et al., 2019). Mandery et al. (2015) also observed ECE usage in human's whole-body posture for locomotion and manipulation. In these examples, geometrical ECE replaced stabilization control. Similarly, objects in front of a hand stabilize when a robot uses granular ECE.

Dynamic object properties constrain an object's motion, for example, its inertia. An object's inertia in combination with gravity was used to reorient an object (Mason, 1999; Dafle et al., 2014; Woodruff and Lynch, 2017). Since gravitational and inertial forces constrain the motion of an object, we can consider it a form of ECE. In the given examples, the effect of inertia and gravity was modeled to complement the forces

provided by a robot, and so a robot's dexterity was increased, but the computational complexity of the manipulation problem remained unchanged.

Bhatt et al. (2021) showed that compliant parts of a robot hand provide similar constraints as geometrical ECs for in-hand-manipulation. They used some fingers of a soft hand to move an object on the palm. Other fingers were used as vertical support to constrain the object's motion. Compliant vertical supports, like soft fingers, provide contact force as well. As opposed to rigid surfaces, where contact changes discretely, the soft fingers provide continuously changing contact forces acting as damping that makes an object's motion smoother. Since they exploited this compliant constraint without explicitly modeling the interaction physics, they provided another example where ECE simplifies the representation by allowing to abstract away details of physics.

The granular EC is different from the mentioned ECs because it manifests in a pile of movable objects but shares the four key properties of ECE: It provides 1) a manipulation funnel that implicitly reduces uncertainty, 2) contact events that could be used to explicitly reduce uncertainty, for example, about the number of grasped objects, 3) structural context about the environment, and 4) the interaction forces inside the pile augment the grasping behavior. Consequently, its use reduced uncertainty and simplified perception, control, and planning for grasping from piles of nearly identical round objects.

6.4 Conclusion

The chapter's goal was to show the implications of ECE-based grasping on perception, control, and planning. Thus, we used ECE benefits in a particular grasping problem.

We discovered a complex environmental constraint manifesting in piles of objects when a hand scoops an object from a pile. We characterized the novel granular EC and showed that its use provides similar benefits to geometrical ECE (Section 1.2). We devised a simple grasping strategy that used granular and geometrical ECs to robustly grasp a random object from piles of nearly identical objects. Uncertainty about an object's position and stability was eliminated by the used ECs so that the grasp strategy could be executed with open-loop controllers and without detecting individual objects in a pile.

Besides reducing uncertainty, granular and geometrical ECEs simplified control, perception, and planning by replacing some computation performed by the robot classically. We could replace computational aspects of control, perception, and planning

with ECE because the granular and geometrical ECs 1) provided predictable high-level motion patterns simplifying the representation, 2) also provided easily observable feedback about the manipulation process that enabled reactive behavior generation, and 3) implicitly structured the robot's workspace into regions with predictable behavior and observable outcomes.

We devised a grasp planner to sequence geometrical and granular ECs for grasping from piles. The proposed grasp planner handled uncertainty the most efficiently by ignoring it because it avoided any computation burden to reason about it. The planner could ignore uncertainty because the task-tailored ECE sequence, in our grasp strategy, eliminated uncertainty entirely.

6.4.1 Limitations and Further Considerations

The last section of this chapter discusses the limitations and future research directions about ECE-based manipulation from practical and conceptual views.

For the practical applicability of granular ECE, there is room for improvement. Even though we have conducted an extensive empirical study of open-loop grasping from piles, some properties of the *three actors*: objects, environment, and the robot should be further analyzed. One should further investigate the effect of different friction properties of the three actors because of the motion and force regularity of the granular EC manifest due to kinetic energy loss. While we successfully applied the strategy on a few irregularly shaped objects, like apples and net bags of limes, it is essential to characterize the EC for irregularly-shaped and deformable objects.

A significant limitation arises from the fact that we considered only an end-effector's motion without considering the used robot arm. We used open-loop controllers, assuming that these controllers generate the desired behavior and avoid an undesired collision or kinematic limitation of the respective robot arm. To voided some kinematic and environmental limitations in our bin picking application, we increased the slope of the hand. This solution does not generalize to other robot arms or environments. For that, we need motion planning. Part I tackled this issue by applying ECE in motion planning under uncertainty, and Part III combines our EC-based grasping and motion planning approaches into a complex robotic system to overcome the mentioned limitation.

For future manipulation research, one could apply our insight as an inductive bias to learn an object-dependent parametrization of our grasp strategy or to find sophisticated motions for grasping less round objects as well. Moreover, machine learning techniques

could be used to co-design the hand and the environment to maximize the benefits of granular EC exploitation.

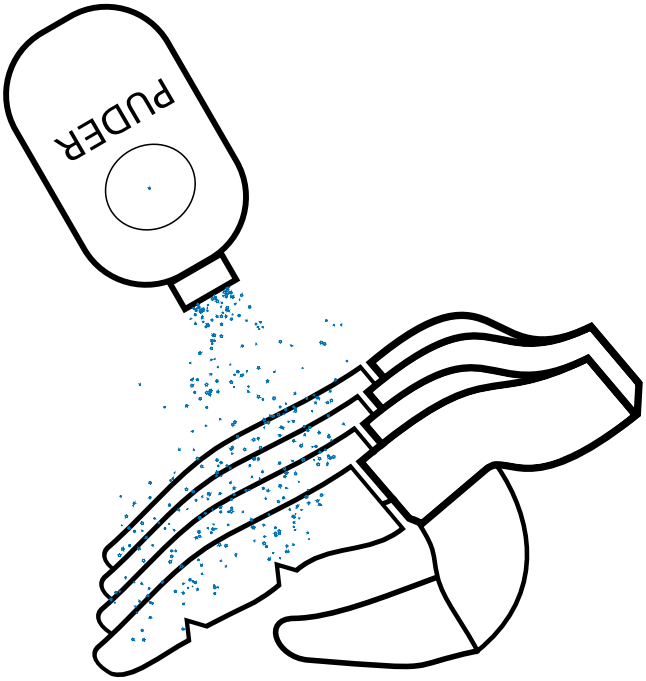
Furthermore, we propose a practical in-hand-manipulation problem associated with grasping more objects than desired. The problme has two challanges. The first challenge is to extract information about the grasped objects, and the second is to manipulate the objects so that only the desired amount of objects remains grasped. Since fingers partially occlude the grasped objects, visual perception can not detect their position. However, the RBO Hand 2 could be equipped with strain or acoustic sensors (Wall and Brock, 2019, 2022) that can measure the hand's deformation. The increased deformation capability of the hand is beneficial for stable grasping since it can passively adapt to an object's shape, and so, its deformation holds information about the number and positions of grasped objects. One could use gravity alone or in combination with finger and thumb actuation to manipulate the grasped objects. With gravity alone, objects could be dropped by opening appropriate fingers. The latter approach is a challenging in-hand-manipulation problem since the hand has to manipulate multiple objects at the same time.

For future ECE research, our study contributes a procedure to identify and characterize new ECs, which helps to find other ECs that will simplify perception, control, and planning. We are fascinated to see novel ECs and their usage and hope to find answers to two questions: Are there other complex ECs with different beneficial effects? Moreover, what other physical interaction regularities will be used as ECE beyond manipulation or motion planning? We think that answering the first question requires examining even more complex interactions than pile dynamics. Regarding the second question, interaction forces manifest from other physical phenomena without establishing contact. Interaction forces manifest on different scales and magnitudes in the context of electromagnetism, gravity, and solar pressure. These forces can be seen as constraints, and their physics shows regularities. Therefore, we are eager to see their use as ECs to ease challenges in different research fields, such as space travel.

For now, let us keep our feet on the ground and continue with Part III, where we combine our previous contributions into a robotics system that uses granular and geometric ECs for two industrial applications.

Part III

Implications of Environmental Constraint Exploitation on Robotic System Designs



"To err is human; to manage error is system."

Kevin Kelly, Out of Control: The New Biology of Machines, Social Systems and the Economic World

Motivation

Robotics is a multidisciplinary field because a robotic system combines a variety of components developed in different fields, such as mechanical engineering, control engineering, or computer science. Experts from different fields choose appropriate technologies for sub-problems of a given task and combine these different components into a robotic system to solve the task efficiently and reliably. The synergy of components from different fields makes robotic system building challenging. Complex tasks often need complex systems where the overall system performance depends not only on individual components' performance but also on the proper combination and cooperation of components.

In order to understand how to build complex robotic systems, we need to analyze existing systems concerning the requirements and applied design choices. Such an analysis can highlight design principles enabling researchers and industry to build complex and robust systems for their needs.

We integrated our motion planning and grasping approaches from Part I and II, respectively, into a system to evaluate the ECE concept's realism and analyze the implications of ECE on complex robotic system building. We have shown that environmental constraint exploitation is beneficial for motion generation (including grasping) under uncertainty by individually evaluating our algorithmic implementations in Part I and II. These benefits are apparent in the overall system and affect how different components are combined for good cooperation.

Contributions

The contributions of the third and final part of the thesis are the following:

- We present a complex grasping system designed to evaluate and demonstrate the realism of environmental constraint exploitation and soft-bodied manipulation.
- We explain the practical implications of environmental constraint exploitation on our robotic system.
- We propose hypotheses of beneficial design choices for building complex robotic systems which use environmental constraint exploitation.

Outline

The third part of the thesis discusses the implications of environmental constraint exploitation on robotic system building.

In Chapter 7, the first section presents two industrial applications where the ECE concept is applied. Then Section 7.2 describes the system, Section 7.3 explains how the ECE concept influenced our system design, and Section 7.4 summarizes the evaluation of the system. Finally, Section 7.5 compares the presented system with three other complex robotic systems designed for similar pick and place tasks. We conclude the chapter by proposing general system-building principles beneficial for environmental constraint exploitation-based manipulation.

In Chapter 8, we conclude the thesis by summarizing the contributions, explaining how our initial objectives were handled using environmental constraint exploitation, and discussing problem factorization in robotics from a *motion-behavior generation* perspective.

7

Analyzing Grasping Systems That Use Environmental Constraints

This chapter focuses on the implications of ECE usage on robotic system design. We developed a robotic grasping system that combines contributions from Part I and II. Since ECE usage affected our grasping and motion generation approaches, we also want to analyze its effect on the overall system. The content of this chapter is unique to the thesis and has not been published before.

A robotic system combines different hardware and software components. Actuators move parts of a robot, and sensors detect or measure the physical properties of a robot or the environment. A perception component collects and interprets sensory information, a planner makes decisions based on sensory information and prior information about an instance of a task, and control executes desired motions.

The performance of individual components of a robotic system can indicate its capabilities and limitation. However, the overall system performance also depends on the *combination* and *cooperation* of the components.

Traditional grasping systems follow the sense-plan-act paradigm, where perception, planning, and control components are connected sequentially. First, a *perception* component processes visual information about a given object. Then, a *planner* uses the processed information to compute a grasp pose and the forces to be applied to the object. Finally, a *control* component executes the plan. The cooperation of the three components relies on accuracy because a planner computes a grasp pose based on the assumption that object properties are accurately known or perceived, and controllers can precisely touch the object and apply the desired forces.

In contrast, our system's components cooperate to generate robust motion behaviors using ECE rather than relying on perception and control accuracy. We showed

in previous chapters that ECE reduces sensing, motion, and modeling inaccuracies that reduce perception and control accuracy requirements. Thus, our components' combination and cooperation are influenced by ECE usage.

We analyze the implications of ECE on a robotic system with a case study. We devised the Soft Manipulation (Soma) system to evaluate and demonstrate the realism of technologies concerning soft robotics and environmental constraint exploitation. These technologies were developed in the *Soft-Bodied Intelligence for Manipulation* H2020-ICT-645599[1] research project. The project partners, the consortium, were composed of five research groups from academic institutions and two from the industry. The developed software and hardware technologies were synergistically integrated into the Soma system and evaluated in two real-world applications.

The purpose of the Soma system was to evaluate and demonstrate the realism of novel technologies in real-world applications. Both applications involved grasping: one focused on *commercial food handling* and the other on *human-robot interactions*.

Below, we present the two real-world applications, system requirements, and the Soma system. Then, we explain the practical implication of ECE concerning the combination and cooperation of our system's components. We summarize evaluations of this system performed by the consortium to show how the project goals were accomplished. Finally, we compare the Soma system to three other complex grasping systems, along with software component integration and general system-building aspects, to provide a broad perspective on the implications of ECE on robotic systems.

The development of the Soma system was a team effort, combining many people's work within a research project. When discussing components concerning my contributions, I will use the first-person plural "we." For contributions from other project partners, I will use the noun "consortium." The lead developer responsible for integration and deployment was the Robotics and Biology Laboratory from Technische Universität Berlin, and I, the author of the thesis, was part of this team. My unique contributions to the system are the design and implementation of various software interfaces and components: the design of grasp feasibility modules, the implementation of the simulated environment, the implementation of a grasp heuristic, the implementation of various communication interfaces, and the augmentation of the grasp planner and the graphical user interface.

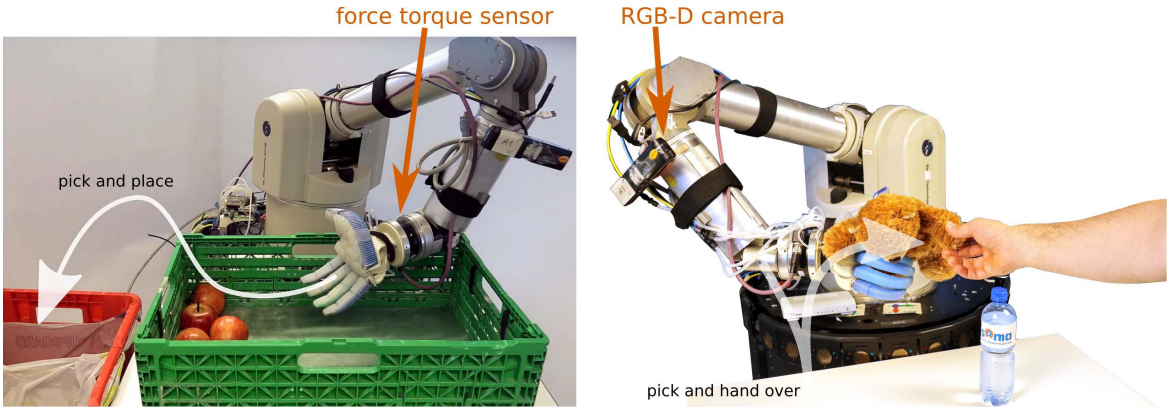


Figure 7.1 Soma system's hardware setup at the Technische Universität Berlin and the two real-world applications. The depicted system used a Barrett WAM 7-DOF arm and an RBO Hand 2 with a wrist-mounted force-torque sensor and a lower-arm-mounted RGB-D camera. *Left:* in the *commercial food handling* application, the robot picks the same fruits or vegetables from a green bin and places them into bags in the red bin © 2021 IEEE. *Right:* in the *human-robot interaction* application, the same robot picks up an object from a table and hands it over to a human or places it back on the table if the human did not take it.

7.1 Demonstrator Applications and Requirements

Before we describe the Soma system, we present two real-world applications for which the system was designed. Each industrial project partner provided one application by defining a real-world task and a set of requirements for later evaluations. The two applications involved grasping in well-structured environments, but the domains were different. Hence, the task requirements and evaluation criteria were different as well. The first application, *commercial food handling*, was defined by Ocado, the world's largest dedicated online grocery retailer. The second application, *human-robot interaction*, was defined by The Walt Disney Company, a mass media and multi-industry entertainment company.

Below, we list all task and project requirements, and then we describe each task and explain their requirements:

- R1 modular design,
- R2 reconfigurability,
- R3 reliable execution,
- R4 damage-free food handling,
- R5 reasonable execution speed,
- R6 safe human-robot interaction, and
- R7 realistic and conformable interaction.

[1]project page https://soma-project.eu/

7.1.1 General Requirements

The Soma project defined general system requirements to evaluate the developed technologies and to demonstrate these technologies in relevant applications. Formally, the objective was defined as reaching a Technology Readiness Level (TRL) seven.

The consortium used the TRL, defined by the Horizon 2020 guidelines, as a realism and applicability indicator. Initially, the consortium validated their technologies in well-controlled laboratory environments (TRL 4). Then, these were validated in an application-relevant environment using production line containers and real fruits and vegetables (TRL 5). Next, different technologies were combined into the Soma system, and the system was used as a demonstrator during periodic project evaluations (TRL 6).

Since components were developed and evaluated continuously, their integration and deployment were also continuous. Therefore, the project requirements aimed to ease integration and deployment efforts in two ways:

R1 modular design – Since project partners had to integrate their technologies into the Soma system, the system design was required to support easy integration of new components. The system had to be modular to reduce integration efforts so that partners could integrate technologies as individual components. A modular design affected both software and hardware solutions and supported the validation of different combinations of alternative system components.

R2 reconfigurable design – The Soma system had to be deployed at industrial partners for periodic evaluations and demonstrations. The system had to be reconfigurable for the two applications and for different hardware platforms. Our industrial partners used different robot arms (e.g., Kuka LBR IIWA 14 and P-Rob2 by F&P Robotics), and they also had to evaluate different end-effectors developed by the consortium.

7.1.2 The Commercial Food Handling Application

The food and agriculture industry is a multi-trillion Dollar industry[2], and humans still perform food handling in the lack of relevant robotic technologies. Food handling and packaging is a challenging robotic problem because food packages have irregular shapes and deformable structures, and the goods are easily damaged. The Soma project developed novel technologies capable of addressing the mentioned challenges and tested these technologies within the following application.

[2] https://www.statista.com/outlook/cmo/food/worldwide

We provide a brief description of our commercial food handling application with an emphasis on the requirements because we described the same application in detail when evaluating the use of granular ECE in Section 6.3.3. For more details, we refer to the task and requirement definitions in the public project deliverables D5.1 and D5.2[3].

In the commercial food handling application, a robot had to grasp one delicate and fragile fruit or vegetable from a bin. The bin's location was known and contained only one type of fruit or vegetable. Then, the robot had to transport and drop off the grasped object into a collection bag with a known location. The fruit and vegetable types were: a net bag of limes, mango, cucumber, a pack of three avocados, and a bag of green salad. Before grasping, the type of fruit or vegetable was known, but their exact properties (e.g., shape, mass, size) or location inside the bin was unknown. Note that the objects never filled the bottom of the bin. Figure 7.1 illustrated on the left side the task and one hardware setup.

Since the bin provided static walls and corners, the robot was able to choose between a surface-, wall-, or corner-grasp strategy. The latter two strategies are analogs with the one presented in Section 6.1.2 when $\alpha = \beta = 0°$ and the pile was next to a wall or corner, respectively. With surface-grasping (Eppner and Brock, 2015)[4], a hand approached an object with its palm facing down. Then, the hand was lowered until a contact event was detected with the object or the support surface. Finally, the fingers were closed such that they slid on the support surface until they wrapped around the object. Our industrial partner defined the commercial food handling system requirements as follows:

R3 reliable grasping and transportation of goods – This requirement ensured an object's successful pick and place and affected all components. Our system used a modified version of the grasp planner from Section 6.2 and leveraged granular and geometrical ECE by distributing responsibilities between control, planning, perception, hardware morphology, and the environment to achieve robust grasping. Hence, all software and hardware components needed appropriate cooperation for reliable grasping.

R4 damage-free food handling – This requirement ensures that fruits and vegetables are not damaged (e.g., bruised). It mostly affected end-effector designs, but it also influenced the execution. For example, we limited the applied forces and the drop-off location's height.

[3]D5.1 Requirements Analysis and D5.2 Revised Requirements Analysis are published at https://cordis.europa.eu/project/id/645599/results

[4]Video of the surface-grasp strategy: https://youtu.be/wzWAkwj95hk

R5 reasonable grasping speed – While this requirement had a lower priority than the previous two, it affected mainly software components because a robot's maximum velocity was limited for damage-free and safe execution.

7.1.3 The Human-Robot Interaction Application

Human-robot interaction spans an enormously large application domain because robots and humans should move closer to co-exist and co-work. Therefore, it is vital to address the safety of manipulation systems and humans' willingness to approach or collaborate with robots. The Soma technologies were tested in a handshaking and a handover task. Below, we focus on the handover task because it involves object grasping.

In the handover task, a robot equipped with an anthropomorphic robotic hand picked up an object from a tabletop and handed it over to a human. If the human did not take the object, the robot had to place it back onto the table. The location and geometry of the table and the objects were not known. The object type was known prior to grasp, and the following objects were considered for handover: a plush toy, a headband, an apple, a banana, a bottle of water, and a ticket. For a detailed task and requirement description, we refer to the public project deliverables D6.1 and D6.2[5].

In a tabletop environment, a robot could use the surface- or the edge-grasp strategy. The surface-grasp strategy is identical to the one presented above. The edge-grasp strategy (Kappler et al., 2010)[6] leverages convex edges of a surface to expose a flat object's opposing sides for pinch grasping. A thin object is pushed to the edge of a table so that the opposing surface of the object is free to be grasped. Since no vertical walls or corners were on the table, wall- and corner-grasp strategies were not available for grasping in this application.

While the previous application domain formulated quantifiable requirements (e.g., execution success, damage-free interaction, and execution speed), this application focused on qualitative aspects. This is because the consortium wanted to analyze how humans perceive interactions with a robot:

R6 safe interactions and increasing perceived-safetiness by humans – This requirement influenced both software and hardware components. The developed hardware technologies had distinctive benefits for human safety due to their inherent compliance. It also affected planning and execution because motion behaviors influenced perceived-safetiness as well.

[5] D6.1 HRI Use Cases and D6.2 HRI Use Cases are published at https://cordis.europa.eu/project/id/645599/results

[6] Video of the edge-grasp strategy: https://youtu.be/0LnvVSEINH4

R7 increased realism and comfort perceived by humans – This requirement similarly affected hardware and software components. Compliance was again beneficial from a hardware point of view. With software, an early system evaluation has shown that the reliable grasping (R3) and smooth motion trajectories influenced perceived realism and comfort.

7.2 Software Architecture of the Soma System

This section describes the software architecture of the Soma system and explains how the system fulfills the previously presented requirements. We start with a high-level description of the system and later provide details about each software component.

We implemented a modular software architecture (R1) using the Robot Operation System (ROS). In ROS, we implemented each module as a stand-alone application (node). We used ROS communication protocols to implement various interfaces between nodes. With ROS service calls, we implemented one-to-one synchronous communication between nodes that implement specific functionalities. With ROS topics, we implemented one-to-many asynchronous communication for data logging.

The system is composed of four main software components that were further decomposed into sub-components and implemented as ROS nodes. In the following, we describe each component and its sub-components.

7.2.1 High-Level Task Manager

The *Task manager* implemented the high-level behavior of the system. It synchronized sensing, planning, and control and the data flow between components. Moreover, it provided failure handling, data logging with execution labeling, and a Graphical User Interface (GUI) to configure, monitor, and control task execution.

The Soma system had to pick and place fruits/vegetables or handover objects autonomously. This high-level behavior was implemented with a state machine, as illustrated in Figure 7.2. The *Task manager* implemented the state machine with five states: Visual detection, Grasp planning, Strategy execution, Handover/Drop-off (depending on the application), and Idle mode. The system started in *Idle mode*, and the manager activated the states with a service call. If all objects were picked or a failure occurred, the system transitioned back to *Idle mode*.

In *Idle mode*, the robot stopped moving, and it was either in gravity compensation or in a locked-joint mode increasing the system's safety (R6). This way, the system avoided

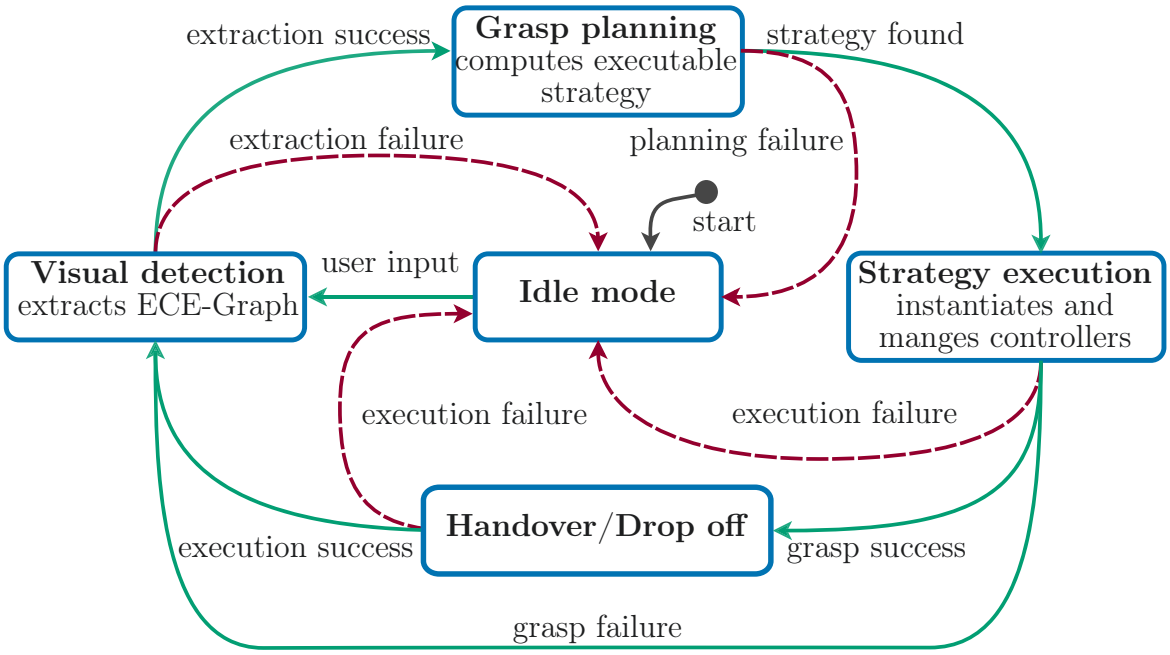


Figure 7.2 The *Task manager* used a state machine with five states (blue boxes), successful, and failed (dashed) transitions, where the starting state is indicated with a gray circle.

potentially dangerous interactions with humans, fragile fruits or vegetables (R4), or the environment. We discuss each state's failure modes in more detail when describing the associated system component. Note that a failed grasp attempt was not considered a system failure.

For the *Idle mode* and *Handover/Drop-off* states, the *Task manager* generated an executable plan and sent it to the ROS node that implements *Strategy execution*.

Data-Flow Between States

The system was designed to be highly modular (R1) using ROS nodes because the consortium developed many (sometimes alternative) software and hardware technologies while researching soft-bodied intelligence. For easy integration, it was essential to implement well-defined communication interfaces. These interfaces were implemented as ROS services, and the *Task manager* was calling the services to perform computation with specific nodes. The service calls returned the requested data (or failure). The data flow between components is illustrated in Figure 7.3 and described below.

The *Visual detector* received point cloud data from an RGB-D camera and computed an ECE-graph, as presented in Section 6.2.2. The graph was sent to the *Grasp planner* that selected a grasp strategy and generated an executable hybrid automa-

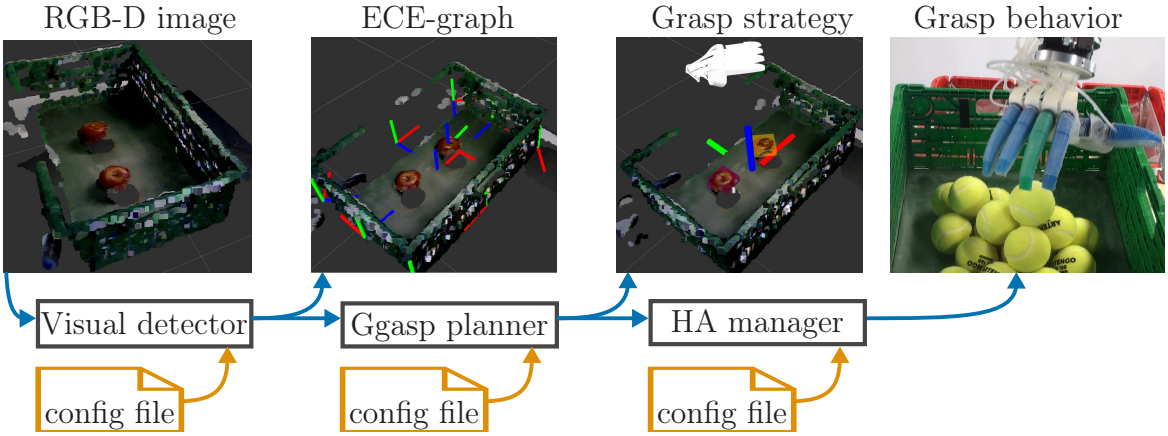


Figure 7.3 Data flow between high-level components of the Soma system. The *Visual detector* received a point cloud of the scene from the lower-arm-mounted RGB-D camera. It computed the ECE-graph where ECE nodes contained a frame attached to ECs and the object. The *Grasp planner* chose an object (yellow bounding box) and a grasp strategy (e.g., surface-grasp) visualized with the pre-approach hand position. Finally, the strategy was sent to the *Hybrid automaton* (HA) *manager* that generated the desired grasp behavior by sequentially executing different controllers. A configuration file, specific for the used hardware and application domain, was an additional input for each system component.

ton (Henzinger, 2000) as discussed in Section 6.2.2. Finally, the *Hybrid automaton manager* instantiated the controllers and sequenced their execution to generate the desired manipulation behavior.

To easily reconfigure the system for different hardware platforms and applications (R2) (e.g., various robot arms, pick-and-place, or handover), software components were parameterized with configuration files. We provide further details about the configuration files for each component in their respective sections.

Graphical User Interface and Data Logging

The GUI was implemented as part of the *Task manager* and served multiple purposes, as shown in Figure 7.4. It enabled reconfiguration of active components at run time (R2), increased execution safety (R6), and decreased the integration effort of new components (R1).

The GUI increased the system's reconfigurability by allowing the user to select between alternative functionality implementations (e.g., motion feasibility checking) or turn on/off optional functionalities (e.g., grasp success estimation) without restarting the system. The alternative and optional functionalities are discussed for each component in later sections.

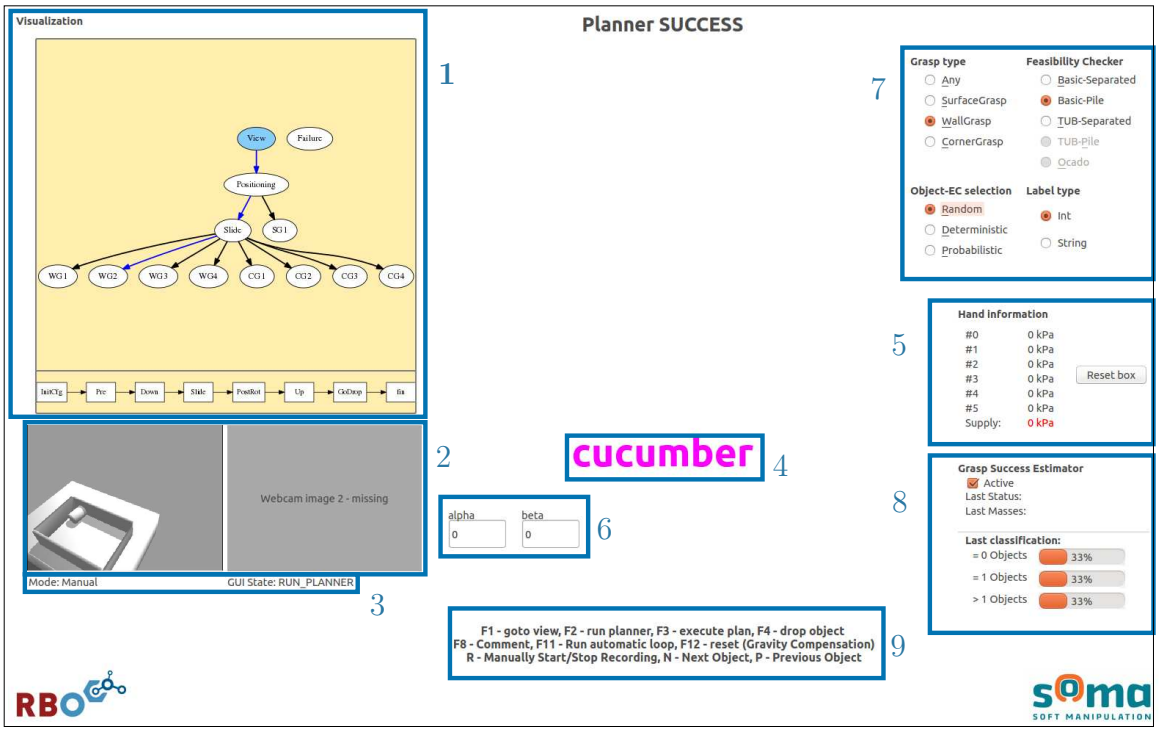


Figure 7.4 Image of the graphical user interface of the Soma system. We visualized the system's current state by showing: 1) the ECE-graph, the selected strategy and its phases, 2) up to two live video streams, 3) the operation mode and current state of the state machine, 4) the selected object type, and 5) the end-effector state. The user could select 6) hand orientation parameters when leveraging granular ECE for grasping, 7) system behavior, and 8) activate grasp success estimation. At the bottom, 9) we listed the hotkeys for the manual operation mode.

The GUI also increases operational safety because a user could trigger failure at anytime. Moreover, a user had to approve transitions between states when the system was in manual operation. For informed approvals, the GUI visualized the system's intermediate states and internal variables, such as the ECE-graph, the selected strategy, and other system parameters.

The GUI decreased integration efforts because it allowed testing a particular state of the system while reusing the outcomes of previous states. For expert users, the GUI provided hotkeys to transition to any state while reusing the last registered outcomes from all other states. For example, an expert could fine-tune controller gains by re-executing a grasp strategy over and over using the same hybrid automaton.

The *Task manager* provided systematic data logging, and the GUI complemented it with labeled execution outcomes. The manager saved sensor measurements such as video feeds, joint states, and force-torque measurements in a ROS Bag file. In

contrast, the ECE-graph, hybrid automation, current system configuration, and labeled execution outcomes were saved in an XML file. Execution outcomes could be labeled manually or automatically. With manual labeling, a user could enter the number of grasped objects or free text to describe the execution process. With automatic labeling, the manager estimated the mass held by the robot after lifting the closed hand and logged the estimated number of grasped objects.

In summary, the *Task manager* implemented a high-level behavior of the system and combined system components following the sense-plan-act paradigm. At lower abstraction levels, we used different component combinations depending on desired motion behaviors. Next, we describe the general components and their sub-components.

7.2.2 Visual Detector

The *Visual detector*[7] computed the ECE-graph from point cloud data of an RGB-D camera. The consortium decomposed the detector into EC extraction and object detection and developed alternative solutions for each sub-component. We present two solutions: A general solution was applicable for both applications using minimal assumptions about the tasks, and it could detect ECE for surface-, wall-, corner-, or edge-grasp strategies. The other solution was designed specifically for commercial food handling using strong task-based assumptions. Both solutions returned an ECE-graph where the nodes contained geometrical properties of the environment (e.g., a frame attached to the detected surface where the z-axis was aligned with the surface normal.) and a list of detected objects (or the entire pile) represented as oriented bounding boxes. If the ECE-graph was empty or no object was detected, the detector returned failure.

The general visual detector was initialized with a configuration file (R2) that could limit the possible ECs to be detected, provided camera-specific parameters for pre-processing, and defined ECE transitioning conditions based on the used end-effector.

With the task-tailored solution, the consortium aimed to reduce ECE-graph computation (R5) by leveraging priors about the commercial food handling task as assumptions. Since the bin's geometry was known and its location was constant, the ECE-graph was predefined. The component computed the EC-relevant data for each node in the graph only when the Soma system was newly set up. It computed the bin's location for a new setup by detecting two perpendicular and vertical planes in

[7] Visual Detector: https://github.com/SoMa-Project/vision

front of the robot, assuming that those are two bin walls, and assigned frames to ECs using the bin's geometry.

7.2.3 Grasp Planner

The *Grasp planner*[8] devised a grasp strategy based on the ECE-graph and generated an executable hybrid automaton. We decomposed grasp planning into two abstraction layers. The symbolic planner from Section 6.2 reasoned about grasping, and on a lower abstraction level, the motion planner from Section 3.2 reasoned about motion trajectories. The *Grasp planner* was initialized with a configuration file defining hardware-specific parameters for grasp strategy selection, motion generation, and hybrid automaton generation.

The symbolic planner searched for a grasp strategy in the received ECE-graph. We defined a heuristic function to estimate grasp success probability using the findings of Eppner and Brock (2017) about single object grasping with geometrical ECE. This function weighted grasping nodes in the ECE-graph. A user could select if the planner should maximize grasp success probability (R3), select a random object-strategy tuple, or sample a tuple based on grasp success probability. Since the grasp plan was symbolic, it did not ensure a collision-free motion with other ECs or that the desired hand motion is followable with the used controllers. Thus, we provided a sub-component to ensure motion feasibility (R3).

Motion feasibility evaluation was an optional component that could be activated in the GUI, and the consortium developed alternative solutions. We present below two alternative solutions. One solution geometrically reasoned about objects' location relative to the bin's walls, and the other used sampling-based motion planning.

With geometrical reasoning, we filtered grasp strategies based on an object's relative location to other ECs that were not used with the respective strategy. For instance, surface-grasp strategies were filtered out for objects within ϵ distance from a wall of the bin. While geometrical reasoning could quickly filter nodes of the ECE-graph (R5) that may cause execution failure, it could not handle a robot arm's kinematic or dynamic limitations. So, we provided a motion planning solution as well.

Our planning-based solution[9] could avoid undesired collision and kinematic or dynamic limitations of a robot arm, as shown in this video[10]. It is a task-tailored

[8]Grasp planner: https://github.com/SoMa-Project/ec_grasp_planner

[9]TUB's motion feasibility module: https://github.com/SoMa-Project/tub_feasibility_check

[10]example of collision avoidance using our feasibility module: https://youtu.be/8h4tCEGkmQs

version of the CEET planner presented in Section 3.2. First, we omitted wavefront exploration because the task and strategies were well-structured but provided the approximated task-relevant regions with cuboids based on the detected ECs and geometrical priors of the bin or table. Second, we limited the planner's action to the guarded and sliding motions for the grasping strategy's approach and slide phases. Moreover, the end-effector orientation was kept constant as defined by the grasp strategy, for example, the palm facing up for wall-grasp or down for surface-grasp. The consortium further extended our planning-based solution to generate jerk-limited trajectories increasing the realism and comfort preconceived by humans (R7). The output of the motion planner was a joint trajectory.

A chosen grasp strategy, including the optional joint trajectories, was converted into a hybrid automaton and returned as an XML file to the *Task manager*. To improve damage-free commercial food handling (R4) and safe execution (R6), the *Grasp planner* included safety switches in the hybrid automaton. Safety switches were conditioned to stop a robot's motion and trigger execution failure transitioning the system into *Idle mode*. Such switches were triggered if joint velocities, force, or torque measurements breached a hardware-specific limitation.

The *Grasp planner* returned failure if it could not find a strategy with a grasp probability score larger than a predefined threshold unless the user selected a random object-strategy selection. It also returned failure if the motion planner could not find a valid trajectory for a given strategy.

7.2.4 Hybrid Automaton Manager

The *Hybrid automaton manager* instantiated the controllers and sequenced them into sets of controllers. A set of controllers could be composed of a single controller, for example, a joint position controller that moved the arm to the drop-off location, or multiple controllers running in parallel. For example, a task-space controller rotated the wrist of the hand, and the hand controller closed fingers during grasping while executing the final phase of a wall grasp strategy. The transitioning between different control sets was based on discrete sensor events, for example, contact events measured with a force sensor when transitioning between two ECEs. The manager also published sensor measurements about the robot's state and its own state for data logging.

We provided the core functionality with the Hybrid Automaton Library[11], developed prior to the Soft Manipulation project. The library defined the communication interface

[11]Hybrid Automaton Library:
https://github.com/SoMa-Project/hybrid-automaton-library

and general functionalities. The *Hybrid automaton manager* implemented low-level communication with hardware components. Hence, partners implemented their manager for their hardware platforms.

7.2.5 Application Domain Simulator

In order to develop an integrated demonstrator and evaluation platform, it is essential to start the integration of research contributions as early as possible.

For early integration, we provided a Gazebo-based simulation. The simulation included a task-relevant environment, a partner's robot arm, an RGB-D camera, a force torque sensor between the arm and the end-effector, and a minimal working example of a *Hybrid automaton manager* for the respective robot. With the simulator, we were able to support early integration and deployment (R1).

Next, we explain the practical implications of ECE on the Soma system building on our detailed system description. Then discuss the achieved project goals and summarize system evaluations. Finally, we compare our system with other systems to derive beneficial system-design practices for ECE-based manipulation.

7.3 Implications of Environmental Constraint Exploitation on the Soma System

ECE-based grasping affected how our system's software and hardware components interacted and cooperated because we used ECE to reduce state uncertainty, generate reactive manipulation behavior, and replace computational aspects traditionally performed by different software components.

Traditionally, a manipulation system was divided into components with well-defined responsibilities: perception provided an accurate world model, a planner computed accurately where and how to apply forces on an object, and controllers accurately executed the computed plan. Such strong assumptions about accuracy are optional when using ECE.

With ECE-based grasping, we did not assume an accurate perception of the world or accurate execution but augmented inaccurate knowledge about the world and robot state with contact-based sensing. ECE provided discrete contact events that were easy to detect, and we successfully integrated such events into the presented motion

and grasp planners in Part I and II. This way, we tightly integrated perception into planning and control.

Furthermore, we reduced uncertainty mechanically. To leverage mechanical uncertainty reduction, we analyzed the conditions under which granular ECE manifests and have shown that it can replace computational aspects of perception, planning, and control. Thus, the Soma system distributed responsibilities between the environment, software, and hardware components by co-design hardware, control, and the environment.

In the following, we explain how planning, perception, control, and hardware design were affected due to the relaxation of accuracy assumption and distribution of responsibilities.

7.3.1 Planning for What We See and Touch

Traditional grasp planners were responsible for providing guarantees for grasp stability. In contrast, Chapter 6 showed that this responsibility could be shifted from the planner to the environment because granular ECE provided reliable object stabilization for grasping from piles of round objects. To shift grasp stability responsibility to the environment, the planner had to know about grasp affordances for an instance of a problem, and this information was obtained with visual perception. Therefore, planning and visual perception became tightly coupled, so the robot planned for what it saw.

Moreover, robust grasping was achieved with a sequence of ECE. The planner created a reactive plan that reasoned when to start and stop an ECE based on contact events. To generate a reactive plan, we anticipated contact events from detected EC affordances and integrated them into the plan. Thus, we tightly integrate contact sensing and planning, so the robot plans for what it will touch.

7.3.2 Integrating Perception Into Planning and Control

Traditional grasp planning assumed perfect knowledge of the manipulandum's geometry or that the geometry can be perceived accurately. In contrast, we relieve the accuracy requirement about objects' geometry for perception. We used simple features, such as an oriented bounding box, indicating which grasping strategy is applicable (Eppner and Brock, 2017). Even though we reduced accuracy requirements for perception, it became tightly coupled with planning, control, and hardware.

Visual perception is not only coupled with planning but also with a robot's embodiment. Visual perception had to detect ECs and their connectivity to compute

an ECE-graph. The graph's connectivity depends on the environment's geometrical connectivity and an end-effector's morphology. Therefore, we needed information about the hardware used for grasping to compute the ECE-graph creating a tight coupling between visual perception and the robot's morphology.

Contact-based sensing became essential for our system because it was tightly integrated into planning and control. Besides visual perception, our system used contact sensing for planning and control. Our planners anticipated detectable contact events indicating an expected outcome or a deviation. Our planners required a sensor model to reason about detectable contact events. We used force torque sensors and defined contact events with threshold values on the continuous force measurements. During execution, the thresholded force measurements triggered controllers' activation and deactivation, generating a reactive motion behavior.

7.3.3 Co-Designing Control, Hardware, and Environment

Traditionally, execution focused on motion generation and applying forces at specific locations of an object while ignoring the environment. With ECE-based grasping, we used the environment for robust grasping and focused on generating manipulation behaviors. Our manipulation behaviors resulted from the interaction between a robot, objects, and the environment depending on the applied control signals. We co-design the applied control, robot hardware, and the environment to achieve the desired manipulation behavior.

We provide co-design examples from the consortium's integration and system evaluation efforts. Since the thesis focuses on motion generation, we present practical examples affecting grasp and motion planning. We encourage the reader to review the following publications resulting from the Soft Manipulation project for a more in-depth discussion on co-design Ghazi-Zahedi et al. (2017) and Abele and Brock (2017).

Hand morphology and control: The consortium developed a variety of compliant end-effectors, where compliance was achieved mechanically with human-like tendon-driven fingers (Catalano et al., 2014), actively controlled (Friedl et al., 2018), or by using soft structure (Deimel and Brock, 2016). A compliant end-effector is beneficial for grasping because it can passively adapt to the shape of an object. Thus, grasping was robust for some amount of variation in an object's shape (such as for fruits and vegetables) and position inside the hand. Moreover, the number and size of contact patches between an end-effector and an object increased due to compliance making the grasp better resist external forces. While compliance is highly beneficial, stiffening of a

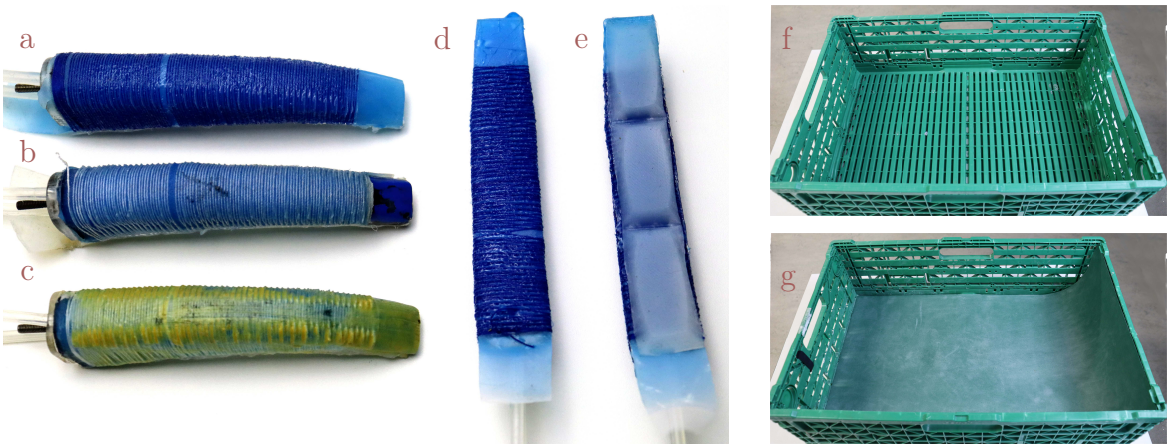


Figure 7.5 Co-design examples of hardware components and the environment: a) a new RBO Hand 2 finger, b) similar finger but with blue fingernails and powdered surface, c) finger with chemically reduced surface friction, d) the flat inner surface of a finger, e) finger with soft pulps, f) original bin, g) modified bin with a smooth and powdered surface to reduce sliding friction, and a rounded edge along the right wall to enable wall grasping less round objects.

finger can improve resistance against an object's mass so that the fingers do not open or bend away when lifting a heavy object.

As an example of co-designing hardware and control, we used the morphological compliance of the RBO Hand 2 to adapt to the shape of an object and compensate lack of stiffness with wrist-hand motion when grasping heavy objects. We pre-inflated the fingers before grasping and pitching the hand while closing the finger. Pre-inflation increased the stiffness of fingers. However, heavy objects (e.g., cucumbers or mangoes) could still bend the fingers away and not around the object. Thus, we used controlled wrist motion to roll a heavy object toward the palm when using the wall- or corner-grasp strategies.

Importance of friction: Friction forces are beneficial to hold and transport an object reliably because such forces provide resistance against slippage due to gravity or other external forces. Yet, friction forces are less desired when a hand slides on a surface because they increase the hand's wear. Moreover, these can trigger contact events identical to contact normals which could prematurely switch the active controller. Therefore, we desired higher friction on the inner side of the fingers than on the outer side.

Inspired by human anatomy, the consortium increased the friction on the inner side of the RBO Hand 2 fingers with pulps made of very soft silicon (see *d* and *e* in

Figure 7.5). These pulps increased friction and the size of contact patches due to their softness, further improving grasp robustness.

For friction reduction, the consortium also experimented with changing the fingers and the environment. With the RBO Hand 2, one approach was adding fingernails inspired by human anatomy (see *b* in Figure 7.5). However, fingernails had no observable or measurable effect on grasping. Another approach was to apply a chemical on the back of the fingers (see *c* in Figure 7.5). While it reduced friction, it also increased production cost and duration significantly. A further approach was to powder the back of the fingers, which was an effective and inexpensive solution. However, friction between a hand and the environment can be reduced by changing the environment, for example, by adding an inlay or powdering the bottom surfaces of the bin.

Modifying the environment: We modified the bin used in the commercial food handling application to simplify perception, planning, and control. To simplify perception and planning, we tilted the bin, so round objects always stayed next to one specific bin wall. Hence, pile (or object) detection was not required to select ECs for grasping.

To simplify control, the consortium applied two changes to the bin. First, we covered the ridges on the bottom of the bin with a sheet, making the sliding motion smooth even with open-loop control. Secondly, the consortium placed an inlay next to a wall making the convex edge rounded. The rounded corner enabled grasping less round objects with the wall grasping strategy (*f* and *g* in Figure 7.5).

The described co-design examples resulted from periodic system evaluations and from close collaboration between researchers developing hardware and software solutions. System evaluation provided feedback for hardware and software development and led to new research questions, for example, the one answered in Chapter 6: Why can a robot reliably grasp from piles of objects using simple open-loop control?

7.4 Soma System Evaluation

The Soma system aimed to evaluate concepts and technologies of soft-bodied intelligence for manipulation and to demonstrate the realism of these technologies in relevant environments. The consortium achieved both goals during the project run-time and published relevant results that we summarize below.

7.4.1 The Soma System Became an Evaluation Platform

The Soma system became an evaluation platform shortly after its initial deployment. First, the consortium used the Soma system to evaluate two compliant hands (Triantafyllou et al., 2019) developed by the consortium. Then, Mnyusiwalla et al. (2020) performed an overall system evaluation using four different compliant end-effectors (three gripers and a hand). Beyond the published results, the Soma system was used with three robotic arms: Barratt WAM, KUKA LBR iiwa 14 R820, P-Rob2 by F&P Robotics, and five end-effectors: Pisa/IIT Hand (Catalano et al., 2014), Pisa Softgripper, the CLASH Hand (Friedl et al., 2018), the RBO Hand 2 (Deimel and Brock, 2016) and RBO gripper. As a result, our system achieved its first goal of becoming an evaluation platform of software and hardware technologies.

Summary of Published Evaluations Concerning System Design

The Soma system distributed computational responsibilities between system components and the environment. Triantafyllou et al. (2019) have shown increased grasp success with below-average visual perception and without using contact sensing as continuous feedback when leveraging ECE with two different compliant hands. This suggests that ECE and the compliance of the hands reduced object position uncertainty. Mnyusiwalla et al. (2020) increased the complexity of the commercial food handling task by increasing the number of objects in the bin. Their results confirmed that ECE-based strategies work in cluttered environments without object singulation prior to grasping and indicated the existence of the granular EC. Therefore, object singulation, as a traditional responsibility of visual detection, planning, and controller, was replaced by the exploitation of ECs.

Mnyusiwalla et al. (2020) also noted that some system failures occurred due to controller failures, for example, joint limits, singularities, or unintended collisions when executing a grasp strategy generated by the symbolic planner. Furthermore, the consortium also observed an increase in similar system failures when using the P-Rob2 arm with only 6-DOF compared to the other used arms with 7-DOF. Both observations motivated the need for a motion planner that can mitigate such issues. With the modified CEET motion planner, we demonstrated robust grasping in real-world demonstrations during the final project review meeting in 2019.

7.4.2 The Soma System Became an Integrated Demonstrator

The consortium continuously integrated newly developed technologies into the Soma system and demonstrated its capabilities during periodic evaluations in July 2018 and May 2019.

During the final evaluation, the consortium demonstrated both applications using similar software components on two different hardware platforms. Both demonstrations have shown that the Soma system integrated a variety of scientific contributions and reliable task execution in both application domains.

Since the consortium demonstrated reliable execution in relevant environments for commercial food handling and human-robot interactions, the system with its software and hardware components reached TRL 6 (Section 7.1.1). This strongly indicates that our system, the ECE concept, and soft-bodied manipulation are applicable for industrial use.

7.5 Implication of Environmental Constraint Exploitation on Grasping Systems

Next, we compare the Soma system with three other complex grasping systems to highlight the implications of ECE on general system design. We compare the four systems with respect to *integration* and *general system building* aspects.

With component integration aspects, Triantafyllou et al. (2021) proposed a methodology for analyzing the integration of complex robotic systems, and they illustrated the use of the methodology within a bi-manual pick and place task. The respective grasping system was developed under similar conditions as the Soma system because five partners from academia and one from industry developed it within the SecondHands research project[12]. We use the proposed methodology and the described system in our analysis to identify how component integration affects general system-building aspects.

With general system building aspects, Eppner et al. (2018) proposed four key dimensions for designing robotic systems and compared their system with multiple other systems developed for the Amazon Picking Challenge in 2015. The authors' system is our second candidate for comparison, and we use their four comparison criteria to identify the implications of ECE on robotic system building.

[12]SecondHands project page: https://secondhands.eu/index.html

First, we present the three systems and their respective grasping tasks and describe comparison criteria. Then we discuss similarities and differences between all four systems for component integration and general system building.

7.5.1 Candidate Robotic Systems for Comparison

The State-of-the-Art Dexterity Network

We included the Dexterity Network, or Dex-Net, in our comparison because it is the state-of-the-art universal picking solution. The respective research project aims to develop highly reliable robotic grasping for various rigid objects. Their latest solution, Dex-Net 4.0 (Mahler et al., 2019), learns ambidextrous robot grasping policies. We will refer to this system as *DN system* in later parts of this chapter.

Their task was to learn ambidextrous robot grasping from 75 different objects, where the system knew 50 of these objects. They used the known objects in a simulation to train a grasp policy on synthetic datasets. In real-world experiments, a robot had to pick objects from a randomly built pile of 75 objects and place it into a tray on the side. Their robot picked and placed objects until all objects were placed into a tray or if the robot failed ten times consecutively.

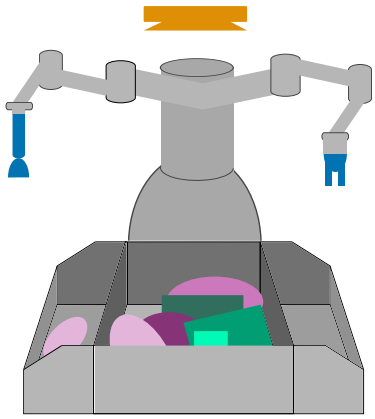


Figure 7.6 Illustration of the DN system with a bi-manipulator, a suction cup, a gripper, and a 3D scanner used for universal packing.

Their hardware setup was composed of the ABB YuMi bi-manual industrial collaborative robot equipped with a custom-built parallel-jaw gripper on one arm. The other arm had a custom-built suction cup gripper. The system used an industrial 3D scanner with a fixed location to perceive objects in a pile.

The software part of the system was composed of a grasp planner, a linear motion planner, and linear motion controllers. The grasp planner used the offline learned policy and the 3D scan of a given scene to choose a grasp pose for a specific object and end-effector. Then, a motion planner computed a trajectory, and the controllers moved the end-effector to the grasp pose and to the drop-off location. We assume that their system used the ROS framework even though it is not disclosed in their publication. We based our assumption on the fact that Dex-Net is integrated into ROS and the ABB YuMi robot that also has planning and control implementations in ROS.

The DN system used no ECE for grasping, and its design is similar to a traditional grasping system. The grasp planner used the policy to compute a grasp pose by assuming that perception provides accurate 3D features. It also assumed that the robot could reach the desired grasp pose accurately. In contrast to traditional grasping approaches, such as form or force closure, the system used a learned grasp policy rather than an analytical contact-based interaction model to compute a grasp pose. Since the DN system used no ECE and had a traditional design, differences to our system should highlight the implications of ECE on robotic system building.

The Bi-Manipulation System of the SecondHands Project

The SecondHands' (Triantafyllou et al., 2021) manipulation task was to pack a toolbox with tools lying around the box on a tabletop. The packed tools had to be uniformly distributed inside the toolbox. The toolbox's location and the objects were known, but the exact location of the objects was unknown. We will refer to this system as *SH system*.

The hardware components included two identical robot arms with two different mechanically compliant hands. For sensing, force-torque sensors were mounted between each arm and hand, and a fixed RGB-D camera was placed above the workspace.

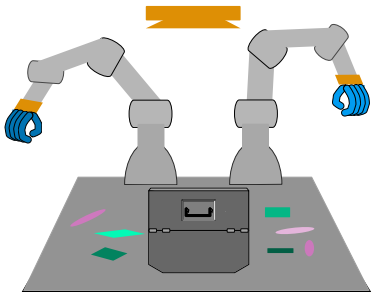


Figure 7.7 Illustration of the SH system with two Kuka arms, wrist-mounted force sensors, two compliant hands, and an RGB-D came used for tool packing.

The software components were implemented in the ROS ecosystem similar to ours. They had a vision component detecting individual tools, a planning component devising a motion trajectory for grasping and placing a tool in the toolbox, and an execution component implementing the drivers of the arms and hands. The SecondHands project coordinator was the same industrial partner, Ocado, who was also part of the Soma project, and the two projects started a the same time in 2016. Therefore, we can observe similarities in grasping techniques (e.g., a similar surface-grasp strategy was used in both projects) and in system design choices (e.g., using compliant hands).

Since the SH system uses both ECE and compliant hands for grasping, we expect that the characteristics of this system should align with our system and indicate ECE implications on the robotic system's design.

The Winner Systems of the Amazon Picking Challenge in 2015

With the Amazon Picking Challenge in 2015, contestants built their robotic system to pick and place autonomously twelve objects out of 25 from a warehouse shelf into a storage container in 20 minutes. The objects were known, and the content of each shelf was also known. However, the exact location of the objects on the shelf was unknown.

We compare our system to the one that won the challenge in 2015, and we will refer to this system as *APC system* (Eppner et al., 2018). The hardware consisted of a holonomic mobile base, Barrett WAM 7-DOF arm using a suction cup end-effector. The robot was equipped with a laser range finder, an RGB-D camera, and a force torque sensor mounted between the arm and the end-effector.

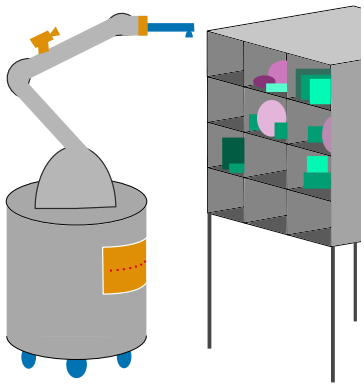


Figure 7.8 Illustration of the APC robot approaching the shelf. The robot was composed of a mobile base with a lidar and the WAM arm with an RGB-D camera, a force sensor, and a suction cup end-effector.

Similarly to the previous systems, the APC system also used a ROS-based ecosystem. It had a vision component identifying objects inside a shelf, a localization component to align the base with the shelf, a planning component devising a grasp strategy and transportation to the storage container, and an execution component implementing the drivers of the arms and the suction cup.

Notably, the APC system used ECE with its slide grasp strategy, where an object was pushed against a side wall of the shelf to align the object with the wall. Hence, the characteristic of this system should also indicate beneficial design choices for ECE-based manipulation.

The APC system was developed by the Robotics and Biology Laboratory, which research group was the coordinator of the Soma project and the lead developer of the Soma system. A few APC system developers were contributing actively to the Soma system as well. Thus, their experience and understanding of robotic system building influenced the Soma system.

While all four systems were designed for similar grasping tasks using similar (and sometimes even identical) components, we expect that differences between the three designs arise from task requirements, integration requirements, and ECE implications.

7.5.2 Comparing Component Integration Aspects

Early and continuous integration is crucial to creating a complex robotic system because a system's performance depends on the interaction and cooperation of the used components that can only be evaluated after integration.

A system's design is affected by the task and the integration effort required to combine system components. Integration efforts increase as the complexity of a system increases due to the increased number of components. It further increases when developers are separated geographically due to less frequent and spontaneous interaction between developers. Hence, it is important to analyze the integration effort before comparing general system-building aspects.

Triantafyllou et al. (2021) proposed a methodology to analyze integration effort for robotic system building and distinguished between *high-level* and *low-level* integration. With each level, we present the proposed conditions for different integration strategies and how these strategies were applied in the considered systems described above.

Little information is provided about the DN system components' high- and low-level integration aspects. Our assessment is based entirely on the fact that Dex-Net and the used robot platform are ROS-enabled. However, the developers might have used different software solutions to plan or control the two arms or end-effectors. Therefore, we include the system in our comparison based on the previously drawn assumption, but we are not going to discuss in detail this system's high- and low-level integration strategies.

High-Level Integration Strategies

With high-level integration, we need to analyze software compatibility and assess the effort required to resolve compatibility conflicts. The overall system is developed in a native development environment defining the version and type of the chosen operating system and available drivers and libraries. With compatibility analysis, we evaluate if a component's software requirements can be fulfilled in the native development environment.

A component can be integrated directly, i.e., *native inclusion* when it is executable directly in the native development environment. This type of integration is straightforward and requires little integration effort. A component may be natively included when the effort is low to resolve software compatibility conflicts (e.g., extending communication interfaces). However, when a component is not executable natively (e.g., missing libraries or drivers), conflict resolution increases integration effort substantially.

System component	**Native inclusion**	**Containerisation**
ROS ecosystem	Soma, DN, APC, SH	—
Perception	Soma, DN, APC	SH
Planning	Soma, DN, APC, SH	—
Control	Soma, DN, SH,	Soma, APC, SH

Table 7.1 Comparing common components compatibility from native integration and containerization point of view for the four systems. Note that a system's component may be assigned to both categories due to variations in sub-components or alternative implementations.

To eliminate conflict resolution effort, a component can be *containerized* in a virtual environment so that its requirements are fulfilled inside the container without modifying the native development environment. Even though containers (e.g., Docker[13], LXC[14], or ROS MultipleMachines[15]) eliminate conflict resolution efforts entirely, containerization requires additional effort to setting up a container and to interface it with the native environment.

In Table 7.1, we present the high-level integration strategies used for each system component of the Soma, DN, APC, and SH systems. Since the DN system could use ROS-enabled components, we assumed native inclusion for all of its components.

As Triantafyllou et al. (2021) pointed out, native inclusion is the general and preferred integration strategy that we also observed in all four systems. For example, the ROS ecosystem offers standardized communication interfaces and distributed system architecture. It supports both Python and C++ programming languages for native inclusion. Moreover, countless general software solutions are available in this ecosystem for planning, perception, and control that can be readily used or easily augmented. Thus, ROS fosters native inclusion.

On the other hand, containerization is frequently required for components implementing controllers. Industrial robots have proprietary controller implementations; oftentimes, these implementations are not native to ROS. All three systems used containerization due to robot drivers. However, we must note that the respective systems used customized hardware components, and some used alternative hardware components, which increased the probability of driver compatibility conflicts.

The difference between native and containerized perception highlights a challenge frequently faced by researchers. The SH system's visual perception component was

[13]Docker containerization: https://www.docker.com/
[14]Linux containers: https://linuxcontainers.org/
[15]Distributed ROS environment for driver or library conflicts: http://wiki.ros.org/ROS/Tutorials/MultipleMachines

containerized because the developed software solution was not compatible with their native development environment. The incompatibilities arose due to unmaintained libraries. A similar issue would arise now with one of Soma's ECTO[16]-based vision component because ECTO is not maintained since 2018. The respective component would require containerization or re-implementation. The latter option brings us to low-level integration strategies.

Low-Level Integration Strategies

With low-level integration, we analyze the interoperability of the components within a system and the components' quality and performance levels. A component's current development state must be assessed both in isolation and within the system to estimate the associated integration effort. Triantafyllou et al. (2021) proposed to analyze five conditions of a component:

1. the component has a directly usable interface without any modification,
2. its programming language or framework is supported by the native environment,
3. its execution and resource usage is efficient,
4. it is well documented and tested, and
5. it is actively maintained.

If a component fulfills all five conditions mentioned above, it can be *directly* integrated into a system because the integration effort is the lowest. The effort increases depending on which conditions are unmet and the importance of those conditions. The effort is medium when unmet conditions require component *augmentation* (e.g., interface extension or performance improvements). A component needs *re-implementation* when key functionalities or interoperability are not met.

The proposed method provides a quantitative evaluation to assess a component's integration effort and duration. The quantitative evaluation considers how well the five conditions are met, weighted by priority. Since such evaluation is not available for the APC and DN systems, we only compare the applied component-wise low-level integration choices for all four systems. We encourage the reader to learn more details about this quantitative evaluation from Triantafyllou et al. (2021).

[16]ECTO framework: https://plasmodic.github.io/ecto/

System component	**Direct integration**	**Augmentation**	**Re-implementation**
ROS ecosystem	Soma, DN, APC, SH	—	—
Perception	APC, DN	Soma, SH	Soma
Planning	APC, DN, Soma	Soma	SH
Control	DN	Soma, SH	Soma, APC, SH

Table 7.2 Comparing common components integration via direct inclusion, augmentation, or re-implementation for the three systems. Note that a system's component may be assigned to multiple categories due to variations in sub-components or alternative implementations.

Table 7.2 compares the four systems for low-level integration design choices. As for high-level integration, we assumed that the DN system directly integrated all of its components since there are ROS-enabled solutions. Since this is a strong assumption, we excluded the DN systems from the below discussion.

We can observe two patterns. First, all three systems (Soma, APC, and SH) directly integrated ROS because this meta-operating system provides well-structured communication interfaces and off-the-shelf software solutions that can be directly integrated into a system.

Secondly, the APC system used more direct integration than the Soma or SH systems. We motivate this for two reasons: the developers were geographically localized, and they developed novel components. Close collaboration enables fluid and quick knowledge sharing. Moreover, novel software or hardware solutions have limited or no requirement conflicts. The APC system was created by one research group, and they developed novel vision, planning, and execution components. In contrast, the other two systems (SH and Soma) were developed by multiple groups located apart geographically and built on existing software and hardware infrastructures.

As an example, the visual perception component of the APC system was developed from scratch. In contrast, Soma and SH used existing frameworks, ECTO and TensorFlow, respectively. While SH developers had to adapt the interface of a neural network in TensorFlow for integration, the Soma consortium had to augment ECTO to interface it with ROS. However, the Soma consortium also re-implemented bin and object detection outside ECTO due to curiosity to see if increased visual accuracy improves the overall system performance.

In summary, native inclusion is the preferred high-level integration strategy, but hardware components and deprecated software libraries often require containerization. Moreover, low-level integration strategies span all three strategies within a complex

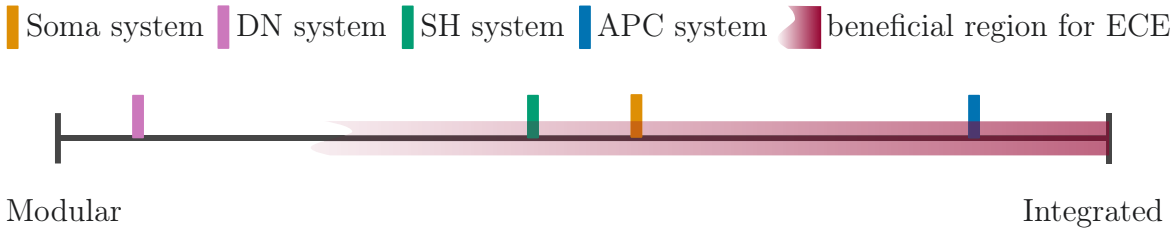


Figure 7.9 Comparing system building aspects of the four grasping systems along the *modular* and *integrated* design spectrum, where the red band indicates the region beneficial system designs for ECE usage.

system, especially for heterogeneous hardware platforms and geographically distant developer teams.

7.5.3 Comparing General System Building Aspects

We want to analyze general system-building aspects beneficial for ECE-based manipulation systems. Eppner et al. (2018) recommended characterizing general system building along four dimensions, and each dimension proposes a spectrum of approaches. They explored these dimensions and identified problem characteristics that can match with regions along some dimensions. We want to extend their analysis by identifying beneficial regions for an ECE-based manipulation system along each dimension.

We compared the four systems for the four general system-building aspects by placing each system on the respective spectrum and highlighting beneficial regions for ECE-based manipulation systems.

Modular — Integrated Spectrum

A robotic system can be decomposed into individual modules by breaking down a task into simpler sub-problems which helps solve complex tasks. However, a wrong factorization of the problem into sub-problems can make solving the task unnecessarily difficult. Moreover, the system's performance can only be determined by how these modules collaborate to solve a given task and not by the individual module's performance.

A modular design is preferred when the effort is increased for high- or low-level components integration. It is especially beneficial when system components are developed across distant developers (geographically or by expertise). Collaborative development of an integrated solution requires a shared understanding of the problem and the solution. However, using sub-components requires well-defined interfaces that can be more easily communicated than knowledge.

Next, we explain the placement of each system on the spectrum by starting from the *Modular* side and progressing toward more *Integrated* systems, as shown in Figure 7.9.

We placed the DN system closer to the *Modular* end of the spectrum than the other systems because its components individually solved sub-problems. At the same time, there was a minimal coupling between components.

The SH system is somewhat further away from the *Modular* end than the DN system. The system subdivided the packing process and used dedicated planning and control components for grasping and placement that increased its modularity moderately. However, it also coupled sensing, control, and hardware using a geometrical ECE-based surface-grasp strategy similar to ours.

Our system is in the middle of the spectrum but slightly shifted to the *Integrated* end than the SH system. The Soma system was designed to be a tightly integrated system using hybrid automaton to couple planning and control and by adapting grasping strategies to our compliant end-effector's embodiment. However, we broke down vision and planning into multiple sub-components, increasing the system's modularity to simplify the integration of components developed by different project partners.

The APC system is toward the *Integrated* end of the spectrum because its development process, similarly to the Soma systems, was tailored to create a tightly integrated system. This system also tightly coupled planning and control using a hybrid automaton control scheme, and it adapted the picking strategies to their robot's embodiment and the requirements for object recognition. Compared to the SH and Soma systems, the APC system did not decompose planning into grasp planning and motion trajectory generation. It only used a simple grasp planner to choose one out of two grasp strategies and omitted motion trajectory generation by leveraging the mobility of its hardware platform.

We propose that an integrated system design is beneficial for ECE-based manipulation. ECE-based manipulation aims to generate robust behavior rather than controlling specific state variables. To generate a manipulation behavior, we need to tightly couple system components which we observed for all three systems that used ECE for grasping. ECE-based manipulation systems tightly couple perception with planning and control for reactive behavior generation and couple control and hardware via co-designing these components.

The beneficial region for ECE-based systems extends toward the *Modular* end of the spectrum because a modular design can reduce the effort to combine different hardware and software technology for complex robotic systems and for distant developer teams. The Soma and SH systems were developed to be modular to ease integration for

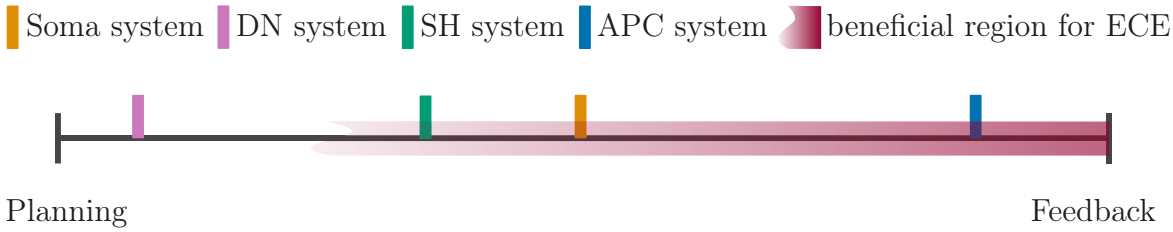


Figure 7.10 Comparing system building aspects of the four grasping systems along the *planning* and *feedback* design spectrum, where the red band indicates the region beneficial system designs for ECE usage.

geographically distant project partners and to incorporate various robot arms and/or end-effectors.

Planning — Feedback Spectrum

With *planning*, we search for global solutions using a world model. For example, a motion planner uses a model of the environment and the robot to compute a valid motion trajectory from a given start to a goal. When global information is difficult to obtain, locally obtained information can be used as *feedback*.

With *feedback*, we search for local solutions and reduce uncertainty using the information provided by physical interaction. As an example, visual servoing uses visually perceived information to minimize an error between the desired and actual position of a feature. With this example, we use continuous feedback. However, discrete changes can be used as feedback as well. Such discrete feedback is used in the hybrid automaton control scheme to switch between active controllers. Moreover, sensory information can also be integrated into planning using the POMDP representation of a planning problem.

Since EC-based planning integrates discrete feedback into planning (for grasping or motion trajectory generation), we expect the respective systems to shift toward the *Feedback* end of the spectrum.

The DN system is at the *Planning* end of the spectrum because it computed a globally optimal grasp pose for one of its end-effectors based on the 3D scan of the whole scene. At the same time, it used no feedback from physical interaction nor searched for a local solution.

The SH system is almost at the center of the spectrum but closer to the *Planning* end because it relied on motion planning to grasp, transport, and place an object into the toolbox while using open-loop joint-trajectory control. However, they used discrete feedback when surface-grasping objects. Unlike the DN system, where objects were

pick-and-placed iteratively, the bi-manual simultaneous packing created the need for motion planning for the SH system because the two arms shared the drop-off location where collision avoidance is required. Even if the arms had been mounted on a mobile base, as in the APC system, the robots would have still shared the goal regions for tool placement. This task requirement also affected the system's design along the next spectrum.

We settled the Soma system at the middle of the spectrum because it relied on planning and feedback. It used a symbolic planner to choose an object, EC, and strategy, and it also could use a motion planner to find feasible joint space trajectories. At the same time, the system integrated continuous and discrete feedback into planning and control. While the motion planning sub-component was optional, the consortium used several alternative solutions (sampling-based planners, geometrical reasoning, or based on prior knowledge) depending on the application domain. As Eppner et al. (2018) observed, a different embodiment could circumvent the need for motion planning. For example, a mobile base would have significantly simplified the execution of sliding motions in our commercial food handling task, and a redundant robot arm would have solved the kinematic limitation of the 6-DOF P-Rob2 arm.

The APC system is placed at the *Feedback* end of the spectrum because it used feedback extensively, similarly to the Soma system but relied on very simple planning. The planner chose between two grasping strategies based on visual information and prior knowledge. The hybrid automaton integrated discrete feedback to switch between motion primitives, and the motion primitives were executed with feedback-guided controllers. Strikingly, the system robustly solved the task without using motion planning for collision avoidance or to avoid the kinematic limitations of the used robot arm by leveraging the mobility of their hardware platform.

With this spectrum, EC-based systems are situated toward the *Feedback* end because ECE offers observable contact events that can be integrated into planning and execution for decision-making and for reactive behavior generation. Moreover, contact-exploiting motions oftentimes require a feedback controller to regulate interaction forces.

Computation — Embodiment Spectrum

A robot's behavior is determined by the *computation* performed by software components and the *embodiment* of its hardware components. Software components are easily adapted to task requirements, and grasping problems can be defined as pure computational problems, such as computing a force closure grasp. Similarly, visual perception

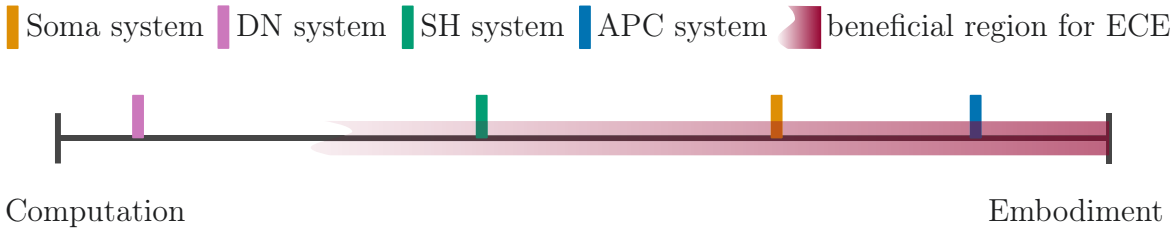


Figure 7.11 Comparing system building aspects of the four grasping systems along the *computation* and *embodiment* design spectrum, where the red band indicates the region beneficial system designs for ECE usage.

can be posed as a computational problem by processing raw sensor measurements to extract relevant task information.

On the other hand, customizing hardware components take longer and can significantly increase a system's cost. However, task-tailored hardware solutions are simple and robust, and appropriate embodiment simplifies software-based computation. In the context of grasping, under-actuated and compliant robotic hands can passively adapt to the shape of an object and consistently reduce the need for accurate object shape and pose detection. Furthermore, a camera's field of view can be greatly increased by mounting it on a robot's arm.

The DN system used the most software-based computation compared to the three other systems. It used large amounts of synthetic data to compute a grasp policy using machine learning. Then, the learned model computed a grasp pose, and a planner computed the arm's motion to reach the pose. However, we must note that the DN system leveraged their robot's embodiment because the system could learn to grasp with two end-effectors. By having two different end-effector embodiment complementing each other, the system's grasp performance increased significantly compared to its previous versions Dex-Net 2.0 and 3.0.

The SH system used significant software-based computation for motion planning and control, but their compliant hands also complemented the grasping behavior. The system had to use software-based computation for collision avoidance due to the simultaneous packing with two arms. However, visual detection and grasp pose calculation was simplified using the passive adaptability of their mechanical compliant hands to an object's shape when grasping.

Our system is closer to the *Embodiment* end of the spectrum than the SH system because the consortium's soft-bodied intelligence research aimed to leverage a robot's embodiment to solve grasping tasks robustly. Thus, embodiment was essential to generate robust grasping behaviors and simplify object and EC detection. Still, the

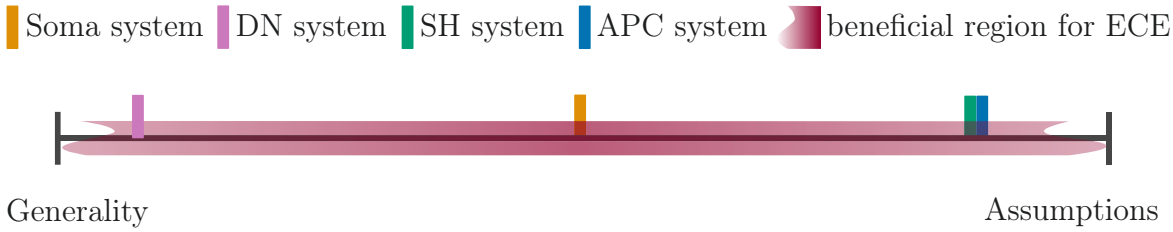


Figure 7.12 Comparing system building aspects of the four grasping systems along the *generality* and *assumptions* design spectrum, where the red band indicates the region beneficial system designs for ECE usage.

Soma system also used software-based computation for reliable execution, keeping the system closer to the middle of the spectrum than to the *Embodiment* end.

We placed the APC system at the *Embodiment* end of the spectrum because the developers chose a well-designed combination of hardware components to simplify software computation. A good example is their use of a holonomic mobile base to significantly relax the kinematic constraints when reaching into a shelf by simply driving the base forward. Another example is the custom-built end-effector with an elongated thin nozzle and tip-mounted suction cup that could easily fit between objects and pick 24 out of the 25 objects. The system performed most computation for object detection, which was also simplified using priors about the end-effector.

We think that the *Embodiment* region of the spectrum benefits systems using ECE for manipulation. ECE-based manipulation behavior depends on a robot's embodiment, and it can replace computational aspects of control, planning, and perception traditionally performed with software components.

Generality — Assumptions Spectrum

General solutions apply to a wide range of problems, while *assumptions* narrow the scope of applicability. Ideally, a system generalizes for variable properties of a given problem and uses strong assumptions about invariant problem properties. We can appropriately categorize these properties when we deeply understand the problem.

Some robotic problems are already well understood for which general approaches were applied successfully. Examples of such approaches are task-generic navigation and motion planning algorithms (A*, PRM, or RRT), various robot arm controllers (joint space, operation space, or impedance control), or Bayesian-based state estimators.

In the scientific community, roboticists frequently assess how well an approach generalizes because it shows a deeper understanding of the solved problem, which was also the goal of Part I to generalize ECE for motion planning. Generalization is less

important for the industry because the goal is to efficiently and reliably solve a task with a fixed scope. Thus, all invariant problem properties are used as assumptions to keep the problem as simple as possible. For example, Part II and III leveraged as many assumptions as the industry-inspired tasks and ECE offered.

The DN system aimed to solve universal picking. Their task was less inspired by an industrial application than the other three tasks for the Soma, SH, and APC systems. Since the latter three systems tackled real-world industry-related grasping problems, we think that all three systems leveraged as many invariant problem properties as possible.

All four systems used readily available or general solutions such as ROS, motion planning, or controllers generating robot arm motions. Moreover, all systems used custom-made end-effectors. Except for the DN system, the other three systems customized their high-level planning component for task-specific decision-making and visual detection by combining object and environment-specific assumptions.

Along this spectrum, the DN system is placed close to the *Generality* end, while the three other systems were placed radically closer to the *Assumptions* end. This is because the DN system was designed to provide a universal picking solution evaluated with 25 unseen objects. Even though its grasping solution generalizes to unseen objects, it is difficult to attain a deeper understanding of the grasping problem because knowledge was encoded into a Convolutional Neural Network.

While Soma, SH, and APC systems are very close to the *Assumptions* end of the spectrum, we shifted the Soma system toward *Generality* for two reasons. First, our system was designed for two application domains and to work with various hardware components, while the DN, APC, and SH systems had a fixed hardware setup and aimed to solve a single task. The Soma system was made to be re-configurable for each application domain using configuration files. These files can be combined such that all types of ECs (surface, wall, corner, edge, and granular) can be detected, and all grasp strategies can be used in both application domains.

Secondly, our commercial food handling task involved grasping real fruits and vegetables with irregular shapes, mass, and colors, as illustrated in Figure 7.13. Compared to the DN system's grasping problem with 50 known and 25 unseen objects, the Soma system aimed to grasp uncountable unseen objects because there are no two identical real fruit or vegetable. Nevertheless, the Soma system used strong assumptions to simplify grasping, while the DN system used no assumptions about the considered objects or the environment.

Figure 7.13 Illustrative size, color, and shape difference between two instances of a fruit.

The SH and APC systems are side-by-side and closest to the *Assumptions* end of the spectrum because these systems used strong assumptions about invariant problem properties from their tasks. Since their object sets were fixed, their solutions generalized less than the Soma system.

We propose that the beneficial region is centered between *Generality* and *Assumptions* ends, as opposed to the three other spectra where the beneficial regions for ECE-based systems were rather one-sided. This is because ECE-based manipulation offers strong priors to be used as assumptions and also generalizes for various manipulation and motion planning tasks as well.

ECE provides assumptions about manipulation behavior because the produced manipulation behavior relies on interaction regularities. When we understand the physics of such regularities, we can identify problem properties that do not affect the regularity and use them as assumptions. Hence, a system using ECE can also use assumptions about the respective interaction regularities. An excellent example of ECE-based assumption is that granular ECE provides a high-level force pattern implicitly controlling objects' position relative to a hand pushing into a pile. We used this assumption to simplify planning and control, as shown in Chapter 6. As another example, we assumed that ECE provides detectable contact events, and based on this assumption, we efficiently computed contingent motion plans in Chapter 4.

At the same time, ECE offers generalization because a simple grasp strategy can reliably pick a large variety of objects. We have shown that more object properties become invariant when geometrical ECs are used in combination with granular ECE. For example, the cardinality of a pile becomes an invariant property when the pile is next to a wall. As a further example, ECE mechanically reduces the position

uncertainty of an object, an end-effector, or the configuration of a robot arm, that we leveraged for conformant motion planning in Chapter 3, for contingent motion planning in Chapter 4, and for grasping in Chapter 6.

7.6 Conclusion and Further Considerations

The third and final part of this thesis focused on the implications of ECE on robotic system design. To achieve that, we combined ECE-based motion planning and grasping methods from Part I and II, respectively, into the Soft Manipulation (Soma) system.

The Soma system was designed to serve as an evaluation platform and demonstrator of software and hardware technologies. These technologies resulted from researching soft-bodied intelligence that involved environmental constraint exploitation to a great extent. We evaluated the realism and relevance of these technologies with the Soma system and demonstrated the system's capabilities with two real-world industrial applications. Therefore, the system's design was driven to use ECs. After a detailed description of our system's components, we explained the practical implications of ECE-based manipulation for each component by comparing it to a traditional grasping approach and by providing practical implementation details. To extend our insights on general system building, we compared the Soma system with three other complex systems solving real-world pick-and-place tasks. Based on our analysis, we proposed beneficial characteristics of robotic systems that use ECE-based manipulation. We proposed that ECE-based manipulation systems should have an integrated design, use continuous and discrete feedback for planning and control, and leverage a robot's embodiment to simplify software-based computation. Hence, we accomplished our goal of revealing how ECE-based manipulation affects the design of robotic systems.

7.6.1 Limitations of the Soma System

The Soma system achieved Technology Readiness Level 6 by demonstrating its capabilities in relevant environments. Nonetheless, we only reached the first milestone ahead of many more before our technologies could be integrated into a production line.

With the commercial food handling application, we tackled increasingly complex grasping scenarios to challenge our approaches, improve our understanding of the grasping problem, and continuously improve the system. For example, we started with a single object in the bin, then increased the number of objects, and finally grasped from piles of objects. One could further increase the problem's complexity by having

one or multiple layers of objects covering the bottom of the bin. In such a situation, our system would have difficulties grasping. It would require new grasping strategies, such as a top-down scooping motion, tactile sensing on the fingers for controlled sliding on a layer of objects, or a new end-effector design. The complexity of the scenarios can be further increased by considering objects loosely packed or with less round packaging. While we worked on such a problem, more research is required to provide reliable solutions.

With human-robot interactions, the consortium analyzed perceived realism, comfort, and safety using a relatively simple handover scenario. Despite its simplicity, the consortium gained essential insights into what properties of a robot's behavior influence human's willingness for interaction. To give a concrete example, the smoothness of a motion and delayed reactions can increase perceived realism and comfort. Our robot's simple handover behavior could be further improved by perceiving human intentions and reacting appropriately. If a robot could detect human intention from visual cues, such as body posture, then it could adjust its handover motion to increase a human's willingness for interaction.

With the implication of ECE on robotic system design, we analyzed and compared four systems solving similar pick and place tasks. However, ECE is useful for other prehensile and non-prehensile manipulation tasks, such as learning from demonstrations or collaborative work between humans and robots or multiple robots. Robotic systems solving such manipulation tasks can also benefit from our observations and propositions for system building.

8

Final Discussion

The thesis tackled motion planning under uncertainty by focusing on one core idea: a well-structured environment provides contact-based information via contact-exploiting motions and contact sensing to simplify motion planning under uncertainty. Such use of the environment is called environmental constraint exploitation (Deimel et al., 2013). First, we revisit the main challenges of motion panning from the introduction and summarize our main insights. Secondly, we finish this chapter and the thesis by discussing the implications of environmental constraint exploitation from a problem factorization perspective.

In Chapter 1, we explained that motion planning is a complex decision-making problem. Its computational complexity arises from the fact that we need many variables to uniquely describe the state of an object or a robot, making the state space high-dimensional. Moreover, we pointed out that uncertainty makes decision-making challenging. In robotics, uncertainty originates from inaccuracies in a robot's motion, sensing, belief about its environment, or understanding of physics.

The thesis handled motion generation challenges associated with state uncertainty and high dimensionality by integrating contact-exploiting motion, contact sensing, and contact-based structural context into planning.

Part I focused on motion planning under uncertainty and applied geometrical environmental constraint exploitation to find robust solutions under motion and perception inaccuracies efficiently. We derived two methods that used contact with the environment to simplify planning in high-dimensional configuration spaces. The main difference between the two methods is in uncertainty handling.

The first method handled uncertainty by searching for robust action in free space unaffected by uncertainty or contact-exploiting motions reducing uncertainty. We proposed a balanced exploration of free-space and contact-exploiting motions because configuration space connectivity depends on environment geometry and accumulated

uncertainty through past motions of a robot. We efficiently approximated task-relevant environmental constraint regions using workspace decomposition and mapped these workspace regions onto a combined configuration and action space. We showed that our search space reduction makes motion planning under uncertainty scalable to high-dimensional problems and large workspaces.

The second method also used free-space and contact-exploiting motions but integrated contact-based sensing into planning to distinguish between outcomes of noisy motions. We used the fact that environmental constraint exploitation produces detectable contact events to rule out parts of the state space when these events are distinguishable from sensor measurements. For each perceivably different contact event, we planned appropriate contingencies and reused partial solutions to reduce the effort for planning yet unsolved contingencies. We showed that contingency planning also benefits from limiting the configuration space search to task-relevant regions. In future work, the connectivity of task-relevant regions should also be used to limit the action space when planning contingencies.

In this part, we simplified planning by searching for task-relevant motion behaviors. To search for those motion behaviors, we modeled the effect of geometrical environmental constraints on a robot's state. Therefore, this approach is limited to environmental constraints for which we can accurately model the interaction physics.

Part II focused on robotic grasping from piles of nearly identical objects using environmental constraint exploitation. We discovered a complex environmental constraint emerging from dynamic and seemingly chaotic interaction between movable objects when a hand is shoved into a pile. The novel granular environmental constraint effectively centered and rolled an object onto the hand without controlling individual objects' motion or applied forces on them. This regularity can be leveraged to generate robust grasping behavior and simplify perception, control, and planning.

Our empirical study of the granular environmental constraint provided alternative grasp strategies using granular and geometrical environmental constraints and realistic assumptions to simplify grasp planning. We reduced uncertainty purely with a task-tailored sequence of environmental constraint exploitations without explicitly reasoning about it. Thus, the proposed grasp planner ignored uncertainty. To further simplify the planning problem, we discretized the state space into environmental constraint regions and devised task-tailored actions for each region prior to planning. Since the granular and geometrical environmental constraints provided a high-level motion constraint, we abstracted away details about the multi-body interaction physics. As a result, grasp

planning became a symbolic search problem of finding a sequence of environmental constraint exploitations leading to a successful grasp.

Our approach greatly simplified perception, control, and planning by devising robust manipulation behaviors. To devise such robust manipulation behaviors, we needed a deep understanding of the involved environmental constraints. While we provided an empirical process to gain understanding, a theoretical approach should also be developed to find and characterize other complex environmental constraints.

Part III finally combined our motion planning and grasping approaches in a complex robotic system aimed to solve real-world industrial applications. Environmental constraint exploitation affects all system components. In Part I, geometrical ECE affected robust motion trajectory generation under uncertainty. In Part II, geometrical and granular ECE affected perception, control, hardware, and high-level reasoning about grasping. Therefore, our robotic system combined software and hardware components depending on the requirements of the used ECs. We achieved synergistic component combinations by integrating contact sensing into planning and control for reactive behavior generation and by co-designing control, hardware, and the environment based on the required motion behaviors. We then investigated the general implications of environmental constraint exploitation on robotic system designs by comparing the state-of-the-art grasping system with three other systems that use environmental constraint exploitation. For environmental constraint exploitation, we propose an integrated system design with extensive use of feedback and leveraging a robot's embodiment.

8.1 Decomposition Paradigms of Robotics Problems

We close the thesis with a discussion about decomposing robotics problems. There are two decomposition paradigms: the *functional decomposition* and the *behavioral decomposition*. Below, we compare the two paradigms and argue that environmental constraint exploitation fosters behavioral decomposition.

The traditional and still predominant *functional decomposition* subdivides complex problems into subproblems that are easier to solve. The general subproblems are sensing, planning, and acting. Perception interprets sensor measurements to compute a model of the environment for planning. Planning computes the entire collision-free path from start to goal. The planned path is followed with controllers computing control signals to generate the desired motions. These subproblems are solved individually with dedicated modules having well-defined functionality.

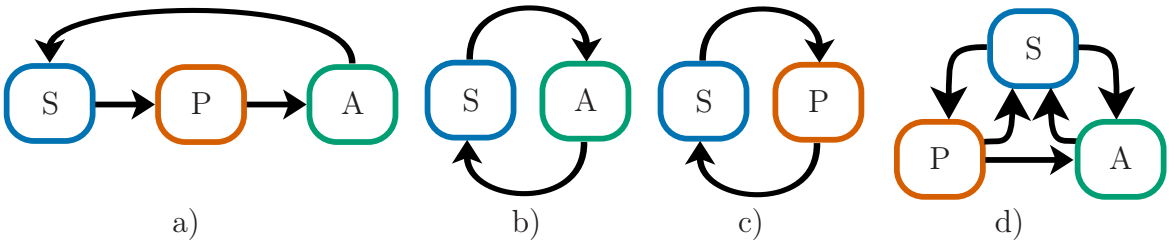


Figure 8.1 Different combinations of sensing (S), planning (P), and acting (A) for reactive behavior generation, where *a)* is sequential sense-plan-act architecture, *b)* is reactive control, *c)* is reactive planning, and *d)* is hybrid reactive planning and control.

One combination of these modules is *sense-plan-act* as described above. This combination is suited for well-structured, -defined, and -controlled environments. However, it is not applicable in the presence of uncertainty because it gathers information only once at the very beginning, and the planner plans too far into an uncertain future. Researchers proposed alternative solutions to generate motions that react to changes (caused by uncertainty) by integrating sensing in various ways into control and planning (Kappler et al., 2018).

Reactiveness requires integrating sensing into planning and acting, and it can be defined as a functional requirement. One example is *sequential sense-plan-act*, where a robot request information at intentionally chosen moments during an execution (Figure 8.1.a). While it improves robustness against some sources of inaccuracies, it is unsuitable for handling fast environmental changes. As another example, *reactive control* (e.g., feedback control or visual servoing) uses continuous sensor information (Figure 8.1.b). Thus, it can cope with fast local changes and handle sensing and motion uncertainty. However, such an approach can be subject to local minima because it lacks planning, i.e., decision-making with global information. A further example is *reactive planning* which can overcome local minima by making decisions with global information updated with locally gained new information (Figure 8.1.c). *Hybrid reactive planning and control* can handle local and globally affecting changes, where both planning and acting (or only one) are equipped with reactive functionalities (Figure 8.1.d). However, reactiveness can be interpreted as a behavior.

Behavioral decomposition divides a complex behavior into simpler behaviors, where the complex behavior is a solution to a given problem. We propose to differentiate between two types of motion behaviors: *reactive* and *adaptive* behavior. Reactive motion behavior handles uncertainty. The thesis provides two examples of reactive motion behavior generation. First, we used manipulation funnels (geometrical and

granular ECE) to reduce uncertainty mechanically. Secondly, we used contact events to reduce uncertainty by differentiating between reached funnel entrances. Adaptive behavior adjusts motions to a given environment. The thesis provides two examples of adaptive motion behavior generation as well. First, we used an ECE-graph to adapt grasping from a pile concerning its cardinality and geometrical ECs near the pile. Secondly, we adapted motion generation to a task in a given environment with guided configuration space exploration (in free space and on geometrical ECs). Thus, environmental constraint exploitation fosters the behavioral decomposition of motion generation problems into reactive and adaptive behaviors.

With behavioral decomposition, first, we need to find task-relevant motion behaviors. Secondly, we need to select appropriate components and define their functionalities for each desired behavior. For the former, we have shown that ECE can be used to find adaptive behaviors in Part I and II. For the latter, we need to consider other constraints, such as robot kinematics or parts of the environment not used as ECs, to select components and define required functionalities, which we have shown with the Soma system in Part III.

We want to illustrate behavioral decomposition and how to define components and their functionalities for a specific motion behavior. Since environmental constraint exploitation is about generating motion behaviors, a sequence of ECE is a behavioral decomposition. Thus, let us re-examine our bin-picking task from a motion behavior generation perspective.

In Figure 8.2, we decomposed the bin-picking task into simple motion behaviors and proposed specific combinations of sensing, planning, and acting for each simple behavior. Approaching a pile inside a bin can be achieved with reactive control, for example, visual servoing. Visual servoing is a non-contact-based manipulation funnel since it can bring a robot's hand from a large set of possible states to a narrower set of states. For lowering and sliding the hand, we proposed using hybrid reactive planning and control because the hand must reach and maintain contact with the bottom of the bin. At the same time, the robot must avoid collision with other parts of the environment. Collision avoidance was an additional constraint on the two motion behaviors besides the horizontal surface EC. We generated these behaviors with a motion planner and hybrid automaton control scheme (Section 3.2.4). However, we showed that simple controllers could successfully lower and slide the hand without using motion planning (Section 6.1) when the environment was composed only of the involved environmental constraints, so collision avoidance was not required. Finally, the grasping behavior is achievable with two controllers running in parallel. We used

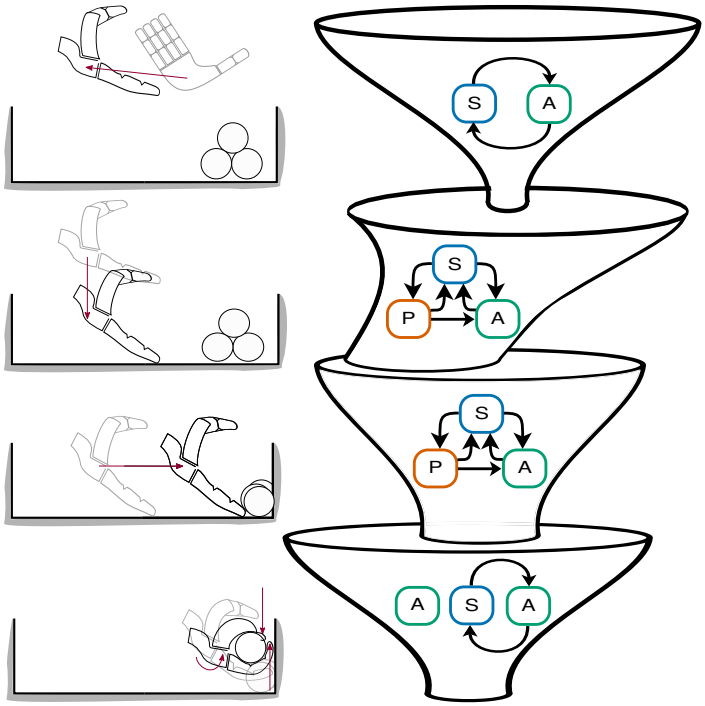


Figure 8.2 Behavioral decomposition of grasping from a pile inside a bin: 1) move the hand above the bin, next to the pile, and with its palm facing up, achievable with reactive control; 2) lower the hand until it touches the bottom of the bin without other collision, achievable with hybrid reactive planning and control; 3) slide the hand on the horizontal surface into the pile until the selected wall is reached without any other collision between the robot and the bin, achievable with hybrid reactive planning and control; 4) close the hand while rolling the object into the palm, achievable with an open-loop controller for finger closure and a reactive controller for the wrist motion.

an open-loop controller to close our compliant hand and an impedance controller to maintain contact between the hand and the wall while rotating the wrist so that the object rolls toward the palm.

As another example, Burridge et al. (1999) proposed a formal approach to sequence robot behaviors. The behaviors were represented as manipulation funnels and implemented with reactive controllers. Each manipulation funnel defines the required functionalities and interactions between control and sensing to produce desired behaviors.

Hence, behavioral decomposition does not commit to one specific functional decomposition but allows tailoring components and functionalities for the desired behavior.

ECE defines a behavioral decomposition and simplifies components' functionalities compared to components derived from a traditional decomposition. Perception can detect ECE affordances without computing an accurate model of the whole environment. Planning can compute only segments of a complex motion behavior where other constraints are imposed on the motion without computing the entire path. Control

generates compliant motions (unless a robot's hardware provides structural compliance) to follow a manipulation funnel without accurately following a path.

The thesis explored the use of geometrical and granular environmental constraints inspired by how humans use their environment. We expect that humans use other contact and non-contact-based environmental constraints unconsciously. To discover and understand new constraints, robotics should expand its multi-disciplinary field of computer science and engineering with other analytical science fields, such as behavioral biology, neuroscience, and psychology. Fortunately, such initiatives have been started, such as the *Science of Intelligence*[1] research cluster.

I believe that behavioral decomposition is the path toward generating artificially intelligent behavior, and environmental constraint exploitation is one fundamental tool for motion behavior generation. The thesis revolved around contact-based environmental constraint exploitation. I hope the explained insights will inspire and help researchers unlock the full potential of robots to help humanity and protect the environment.

[1]SCIoI project page: https://www.scienceofintelligence.de/

Bibliography

Jessica Abele and Oliver Brock. First Analysis of Environment Design for Motion Planning with Contact. RSS workshop: Revisiting Contact - Turning a problem into a solution, 2017.

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*, 2007.

A. Amagai and K. Takase. Implementation of dynamic manipulation with visual feedback and its application to pick and place task. In *IEEE International Symposium on Assembly and Task Planning (ISATP2001)*, pages 344–350, 2001.

Sankalp Arora, Sanjiban Choudhury, and Sebastian Scherer. Hindsight is Only 50/50: Unsuitability of MDP based Approximate POMDP Solvers for Multi-resolution Information Gathering, 2018.

T. Asfour, D.N. Ly, K. Regenstein, and R. Dillmann. Coordinated Task Execution for Humanoid Robots. In *Experimental Robotics IX*, volume 21, pages 259–267. Springer Berlin Heidelberg, 2006. Series Title: Springer Tracts in Advanced Robotics.

Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A Ngo. Monte Carlo Value Iteration for Continuous-State POMDPs. In *Algorithmic foundations of robotics IX*, pages 175–191. Springer, 2010.

Amit Bhatia, Lydia E. Kavraki, and Moshe Y. Vardi. Sampling-based motion planning with temporal goals. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2689–2696, 2010.

Aditya Bhatt, Adrian Sieler, Steffen Puhlmann, and Oliver Brock. Surprisingly Robust In-Hand Manipulation: An Empirical Study. In *Robotics: Science and Systems*, 2021.

Timothy Bretl. Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *The International Journal of Robotics Research*, 25(4):317–342, 2006.

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)*, volume 2, pages 1469–1474 vol.2, 2001.

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. Single-Query Motion Planning with Utility-Guided Random Trees. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3307–3312, April 2007. ISSN: 1050-4729.

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.

Lucian Buşoniu, Előd Páll, and Rémi Munos. Continuous-action planning for discounted infinite-horizon nonlinear optimal control with Lipschitz values. *Automatica*, 92: 100–108, 2018.

John Canny. On computability of fine motion plans. 1:177–182, 1989.

Manuel G Catalano, Giorgio Grioli, Edoardo Farnioli, Alessandro Serio, Cristina Piazza, and Antonio Bicchi. Adaptive synergies for the design and control of the Pisa/IIT SoftHand. *The International Journal of Robotics Research*, 33(5):768–782, 2014.

Constantinos Chamzas, Zachary Kingston, Carlos Quintero-Peña, Anshumali Shrivastava, and Lydia E. Kavraki. Learning Sampling Distributions Using Local 3D Workspace Decompositions for Motion Planning in High Dimensions. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1283–1289, 2021.

Lillian Y. Chang, Garth J. Zeglin, and Nancy S. Pollard. Preparatory object rotation as a human-inspired grasping strategy. In *IEEE-RAS International Conference on Humanoid Robots*, pages 527–534, 2008.

Wen-Chung Chang, Chia-Hung Wu, Ya-Hui Tsai, and Wei-Yao Chiu. Object volume estimation based on 3D point cloud. In *International Automatic Control Conference (CACS)*, pages 1–5, 2017.

Nikhil Chavan-Dafle and Alberto Rodriguez. Prehensile pushing: In-hand manipulation with push-primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6215–6222, 2015.

Nikhil(Nikhil Narsingh) Chavan-Dafle. *Dexterous manipulation with simple grippers*. Thesis, Massachusetts Institute of Technology, 2020.

Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T. Mason. Contact Mode Guided Sampling-Based Planning for Quasistatic Dexterous Manipulation in 2D. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6520–6526, 2021.

Matei T. Ciocarlie and Peter K. Allen. Hand Posture Subspaces for Dexterous Robotic Grasping. *The International Journal of Robotics Research*, 28(7):851–867, 2009.

Rod Cross. Measurements of the horizontal coefficient of restitution for a superball and a tennis ball. *American Journal of Physics*, 70(5):482–489, 2002.

Nikhil Chavan Dafle, Alberto Rodriguez, Robert Paolini, Bowei Tang, Siddhartha S. Srinivasa, Michael Erdmann, Matthew T. Mason, Ivan Lundberg, Harald Staab, and Thomas Fuhlbrigge. Extrinsic Dexterity: In-Hand Manipulation with External Forces. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1578–1585, 2014.

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.

Raphael Deimel, Clemens Eppner, José Alvarez-Ruiz, Marianne Maertens, and Oliver Brock. Exploitation of Environmental Constraints in Human and Robotic Grasping. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2013.

Raphael Deimel, Clemens Eppner, José Álvarez-Ruiz, Marianne Maertens, and Oliver Brock. Exploitation of Environmental Constraints in Human and Robotic Grasping. In Masayuki Inaba and Peter Corke, editors, *Robotics Research: The 16th International Symposium ISRR*, Springer Tracts in Advanced Robotics, pages 393–409. Springer International Publishing, Cham, 2016.

Jacques Duran. *Sands, Powders, and Grains: An Introduction to the Physics of Granular Materials*. Springer Science & Business Media, 2012.

Mohamed Elbanhawi and Milan Simic. Sampling-Based Robot Motion Planning: A Review. *IEEE Access*, 2:56–77, 2014.

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 and Oliver Brock. Visual Detection of Opportunities to Exploit Contact in Grasping Using Contextual Multi-Armed Bandits. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 273–278, 2017.

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*, 42(7):1459–1475, 2018.

Michael Erdmann. Using Backprojections for Fine Motion Planning with Uncertainty. *The International Journal of Robotics Research*, 5(1):19 –45, 1986.

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

Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3):189–208, 1971.

Werner Friedl, Hannes Höppner, Florian Schmidt, Máximo A. Roa, and Markus Grebenstein. CLASH: Compliant Low Cost Antagonistic Servo Hands. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6469–6476, 2018.

Siva Ganesh and Vanessa Cave. P-values, p-values everywhere! *New Zealand Veterinary Journal*, 66(2):55–56, 2018.

Keyan Ghazi-Zahedi, Raphael Deimel, Guido Montúfar, Vincent Wall, and Oliver Brock. Morphological Computation: The Good, the Bad, and the Ugly. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 464–469, 2017.

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 (2-4):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.

François Guay, Philippe Cardou, Ana Lucia Cruz-Ruiz, and Stéphane Caro. Measuring How Well a Structure Supports Varying External Wrenches. In Victor Petuya, Charles Pinto, and Erwin-Christian Lovasz, editors, *New Advances in Mechanisms, Transmissions and Applications*, volume 17, pages 385–392. Springer Netherlands, Dordrecht, 2014.

Kaiyu Hang, Andrew S. Morgan, and Aaron M. Dollar. Pre-Grasp Sliding Manipulation of Thin Objects Using Soft, Compliant, or Underactuated Hands. *IEEE Robotics and Automation Letters*, 4(2):662–669, 2019.

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 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, 2000.

W Stamps Howard and Vijay Kumar. On the stability of grasped objects. *IEEE Transactions on Robotics and Automation*, 12(6):904–917, 1996.

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

Feng-hsiung Hsu, Murray S Campbell, and A Joseph Hoane Jr. Deep Blue system overview. In *International Conference on Supercomputing*, pages 240–244, 1995.

Ajinkya Jain and Scott Niekum. Efficient Hierarchical Robot Motion Planning Under Uncertainty and Hybrid Dynamics. In *Conference on Robot Learning*, pages 757–766, 2018.

Peng Jia, Wei li Li, Gang Wang, and Song Yu Li. Optimal grasp planning for a dexterous robotic hand using the volume of a generalized force ellipsoid during accepted flattening. *International Journal of Advanced Robotic Systems*, 14(1): 1729881416687134, 2017.

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

Leslie Pack Kaelbling, Michael Littman, and Anthony Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

Peter Kaiser, David Gonzalez-Aguirre, Fabian Schultje, Julia Borras, Nikolaus Vahrenkamp, and Tamim Asfour. Extracting whole-body affordances from multimodal exploration. In *2014 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pages 1036–1043, Madrid, Spain, 2014.

Daniel Kappler, Lillian Chang, Markus Przybylski, Nancy Pollard, Tamim Asfour, and Rüdiger Dillmann. Representation of pre-grasp strategies for object manipulation. In *IEEE-RAS International Conference on Humanoid Robots*, pages 617–624, 2010.

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, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

Moslem Kazemi, Jean-Sebastien Valois, J. Andrew Bagnell, and Nancy Pollard. Human-inspired force compliant grasping primitives. *Autonomous Robots*, 37(2):209–225, 2014.

Zachary Kingston, Andrew M. Wells, Mark Moll, and Lydia E. Kavraki. Informing Multi-Modal Planning with Synergistic Discrete Leads. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3199–3205, 2020.

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, 2016.

Michael C. Koval, David Hsu, Nancy S. Pollard, and Siddhartha S. Srinivasa. Configuration Lattices for Planar Contact Manipulation Under Uncertainty. In *Algorithmic Foundations of Robotics XII*, pages 768–783. Springer, 2020.

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)*, volume 2, pages 995–1001, San Francisco, CA, USA, 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.

Jean-Claude Latombe. *Robot Motion Planning*. Springer Science & Business Media, 2012.

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

Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

François Lévesque, Bruno Sauvet, Philippe Cardou, and Clément Gosselin. A model-based scooping grasp for the autonomous picking of unknown objects with a two-fingered gripper. *Robotics and Autonomous Systems*, 106:14–25, 2018.

Qiao Lin, Joel Burdick, and Elon Rimon. Computation and analysis of compliance in grasping and fixturing. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 93–99 vol.1, 1997.

Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In Armand Prieditis and Stuart Russell, editors, *Machine Learning Proceedings 1995*, pages 362–370. Morgan Kaufmann, San Francisco (CA), 1995.

Bangshang Liu, Christian Scheurer, and Klaus Janschek. Motion Planning with Cartesian Workspace Information. *IFAC-PapersOnLine*, 53(2):9826–9833, 2020.

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, 2017.

Jeffrey Mahler, Matthew Matl, Vishal Satish, Michael Danielczuk, Bill DeRose, Stephen McKinley, and Ken Goldberg. Learning ambidextrous robot grasping policies. *Science Robotics*, 4(26):eaau4984, 2019.

Christian Mandery, Júlia Borràs, Mirjam Jöchner, and Tamim Asfour. Analyzing Whole-Body Pose Transitions in Multi-Contact Motions. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 1020–1027, 2015.

Matthew T. Mason. The Mechanics of Manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 544–548, 1985.

Matthew T Mason. Kicking the Sensing Habit. *AI Magazine*, 14(1):58–59, 1993.

Matthew T. Mason. Progress in Nonprehensile Manipulation. *The International Journal of Robotics Research*, 18(11):1129–1141, 1999.

Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. PDDL-the planning domain definition language. Technical report, Technical Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

Nik A. Melchior and Reid Simmons. Particle RRT for Path Planning with Uncertainty. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1617–1624, 2007.

Hussein Mnyusiwalla, Pavlos Triantafyllou, Panagiotis Sotiropoulos, Maximo A. Roa, Werner Friedl, Ashok M. Sundaram, Duncan Russell, and Graham Deacon. A Bin-Picking Benchmark for Systematic Evaluation of Robotic Pick-and-Place Systems. *IEEE Robotics and Automation Letters*, 5(2):1389–1396, 2020.

Andrew S. Morgan, Kaiyu Hang, Bowen Wen, Kostas Bekris, and Aaron M. Dollar. Complex In-Hand Manipulation Via Compliance-Enabled Finger Gaiting and Multi-Modal Planning. *IEEE Robotics and Automation Letters*, 7(2):4821–4828, 2022.

Lael U Odhner, Raymond R Ma, and Aaron M Dollar. Open-Loop Precision Grasping With Underactuated Hands Inspired by a Human Manipulation Strategy. *IEEE Transactions on Automation Science and Engineering*, 10(3):625–633, 2013.

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.

OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 2019.

Andreas Orthey, Sohaib Akbar, and Marc Toussaint. Multilevel motion planning: A fiber bundle formulation. *arXiv preprint arXiv:2007.09435*, 2020.

Christos H. Papadimitriou and John N. Tsitsiklis. The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Calder Phillips-Grafflin and Dmitry Berenson. Planning and Resilient Execution of Policies For Manipulation in Contact with Actuation Uncertainty. *Springer*, pages 752–767, 2020.

Robert Platt Jr, Russ Tedrake, Leslie Kaelbling, and Tomas Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proceedings of the Robotics: Science and Systems Conference, 6th (RSS)*, 2010.

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.

Michael Posa, Scott Kuindersma, and Russ Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373, 2016.

Louise Pryor and Gregg Collins. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, 1996.

Előd Páll and Oliver Brock. Analysis of Open-Loop Grasping From Piles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2591–2597, 2021. © 2021 IEEE.

Előd Páll, Arne Sieverling, and Oliver Brock. Contingent Contact-Based Motion Planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6615–6621, Madrid, Spain, 2018. © 2018 IEEE.

Pradeep Rajendran, Shantanu Thakar, Ariyan M. Kabir, Brual C. Shah, and Satyandra K. Gupta. Context-Dependent Search for Generating Paths for Redundant Manipulators in Cluttered Environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5573–5579, 2019.

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, Vancouver, BC, 2017.

Markus Rickert, Oliver Brock, and Alois Knoll. Balancing exploration and exploitation in motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2812–2817, 2008.

Markus Rickert, Arne Sieverling, and Oliver Brock. Balancing Exploration and Exploitation in Sampling-Based Motion Planning. *IEEE Transactions on Robotics*, 30 (6):1552–3098, 2014.

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.

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.

Arne Sieverling, Clemens Eppner, and Oliver Brock. Exploiting Contact for Efficient Motion Planning Under Uncertainty. In *Robotics: Science and Systems Workshop*, Cambridge, 2017a.

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, 2017b.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

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.

Mike Stilman. Global Manipulation Planning in Robot Joint Space With Task Constraints. *IEEE Transactions on Robotics*, 26(3):576–584, 2010.

Matthew B. Stone, David P. Bernstein, Rachel Barry, Matthew D. Pelc, Yee-Kin Tsui, and Peter Schiffer. Getting to the bottom of a granular medium. *Nature*, 427(6974): 503–504, 2004.

Sebastian Thrun. Monte Carlo POMDPs. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

A. Tordesillas, C. A. H. Steer, and D. M. Walker. Force chain and contact cycle evolution in a dense granular material under shallow penetration. *Nonlinear Processes in Geophysics*, 21(2):505–519, 2014.

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.

Marc Toussaint, Jason Harris, Jung-Su Ha, Danny Driess, and Wolfgang Hönig. Sequence-of-Constraints MPC: Reactive Timing-Optimal Control of Sequential Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 13753–13760, 2022.

Pavlos Triantafyllou, Hussein Mnyusiwalla, Panagiotis Sotiropoulos, Máximo A. Roa, Duncan Russell, and Graham Deacon. A Benchmarking Framework for Systematic Evaluation of Robotic Pick-and-Place Systems in an Industrial Grocery Setting. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6692–6698, 2019.

Pavlos Triantafyllou, Rafael Afonso Rodrigues, Sirapoab Chaikunsaeng, Diogo Almeida, Graham Deacon, Jelizaveta Konstantinova, and Giuseppe Cotugno. A Methodology for Approaching the Integration of Complex Robotics Systems: Illustration Through a Bimanual Manipulation Case Study. *IEEE Robotics and Automation Magazine*, 28(2):88–100, 2021.

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.

Ngo Anh Vien and Marc Toussaint. POMDP manipulation via trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 242–249, 2015.

Vincent Wall and Oliver Brock. Multi-Task Sensorization of Soft Actuators Using Prior Knowledge. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9416–9421, Montreal, Canada, 2019.

Vincent Wall and Oliver Brock. Passive and Active Acoustic Sensing for Soft Pneumatic Actuators. *The International Journal of Robotics Research*, 2022.

Peter M Will and David D Grossman. An Experimental System for Computer Controlled Mechanical Assembly. *IEEE Transactions on Computers*, C-24(9):879–888, 1975.

Florian Wirnshofer, Philipp S. Schmitt, Wendelin Feiten, Georg v. Wichert, and Wolfram Burgard. Robust, Compliant Assembly via Optimal Belief Space Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5436–5443, 2018.

J Zachary Woodruff and Kevin M Lynch. Planning and control for dynamic, nonprehensile, and hybrid manipulation tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4066–4073, 2017.

Kwangjin Yang. An efficient Spline-based RRT path planner for non-holonomic robots in cluttered environments. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 288–297, 2013.

Sungwook Yoon, Alan Fern, Robert Givan, and Subbarao Kambhampati. Probabilistic Planning via Determinization in Hindsight. pages 1010–1016, 2008.

Yu Zheng. An Efficient Algorithm for a Grasp Quality Measure. *IEEE Transactions on Robotics*, 29(2):579–585, 2013.

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.