An Optimization Approach to Certified Manipulation

by

Bernardo Aceituno

Submitted to the School of Engineering
in partial fulfillment of the requirements for the degree of

Ph.D. in Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

Author .....
Bernardo Aceituno
Dec 15, 2022

Certified by .....
Alberto Rodriguez
Associate Professor
Thesis Supervisor

Accepted by .....
Nicolas Hadjiconstantinou
Department Graduate Officer

An Optimization Approach to Certified Manipulation

by

Bernardo Aceituno

Submitted to the School of Engineering
on Dec 15, 2022, in partial fulfillment of the
requirements for the degree of
Ph.D. in Engineering

Abstract

The goal of this thesis is to explore the problem of contact-rich robotic manipulation from an optimization perspective. We plan to study the interplay between contact mechanics, geometry, and machine learning to synthesize manipulation plans with varying theoretical properties. More specifically, we propose a quasi-dynamic mechanics model for contact-trajectory optimization and apply it to solve long-horizon manipulation problems in conjunction with randomized planning. We also discuss a machine learning pipeline to solve this problem from video demonstrations, leveraging novel tools from differentiable optimization and learning. Finally, we aim to explore the issue of certification for planar manipulation tasks in the frictionless plane. We propose a theory of certification that enables us to generate long-horizon manipulation plans that are robust to bounded pose uncertainty. The desired outcome of these techniques is to validate them over a wide range of standard manipulation tasks in 2D environments. Our current results demonstrate the ability of model-based approaches at synthesizing high-quality manipulation plans with varying properties, such as optimality, convergence, robustness, and computation speed.

Thesis Supervisor: Alberto Rodriguez
Title: Associate Professor

Acknowledgments

Through the past five years, I have had the privilege of being supported, advised, and cared for by incredible individuals and institutions. First and foremost, I must thank G-d and my family. My achievements are nothing but the result of the last 27 years of care, love, and advice I have received from them. My family taught me to be kind, learn from others, be open-minded, love books, and have a passion for technology.

I need to start by expressing my gratitude to my advisor Alberto Rodriguez. Through the past five years, he has provided me with guidance and support that have been invaluable and have shaped me into a critical engineer and researcher. Alberto has an innate ability to identify important problems, articulate questions, and empathize with others which made him one of the most meaningful mentors I have encountered in my life. I will always look up to him as a role model and be inspired by his care for his students, his attention to detail, and his kindness. I also want to thank my thesis committee Tomas Lozano-Perez and Sangbae Kim who offered insightful advice and guidance in presenting my work and conducting research.

I thank all the members of the MCube Lab for their incredible support over these past few years. They are great colleagues, advisors, and friends. I will cherish the many hours of discussions on research ideas, directions, and optimization problems. I must also acknowledge the contributions of our external collaborators in our projects: Hongkai Dai (from Toyota research institute), Anastasiia Varava (from KTH), Mustafa Mukadam, Shubham Tulsani, and Abhinav Gupta (from Facebook AI Research). Many of the key insights and discussions that shaped this thesis were a result of their command of robotics, optimization, and deep learning. Their work ethic, kindness, and expertise have been great examples to follow.

I want to thank my co-founders and advisors on Stack. Founding this startup has been a journey that has taught me invaluable lessons. I have been privileged to work with Antoni and Melissa and many other collaborators. Their excellence, attention to detail, discipline, and passion are major sources of inspiration.

Finally, I want to thank all the friends that accompanied me on this journey: my

v(b)ery good friends from the MIT different grad programs, the MIT and Harvard Venezuelan community, the French, Latin-American, and Israeli family, the colleagues of the rowing club, my friends at MIT Hillel and Chabad, and NYC family. All of these people have brought joy into my life and have a very special place in my heart.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Robotics has reached a major milestone in its history, now ready to revolutionize warehouses, manufacturing, recycling, and even home assistance. Achieving success in these areas requires robots to master the art of re-arranging their surroundings. We call this problem **robotic manipulation**. Having the ability to manipulate objects flexibly and skillfully is essential to the complex process of re-arranging an environment to complete a task (e.g. picking up an object and using it). Robots today lack this capability, limiting many of the robots used in factories to pre-programmed and human-crafted sequences of pick-and-place tasks. Dexterous manipulation is one of the keys to unlocking the application of autonomous robots in homes, warehouses, hospitals, and other environments that are not carefully designed to accommodate repetitive and hand-crafted sequences of actions.

When looking at humans, manipulation skills are ubiquitous in our day to day. From taking our phones out of our pockets to assembling furniture, we consistently re-arrange objects to complete tasks. A key component to achieving such a wide range of dexterity is our subconscious command of contact mechanics when moving an object by grasping, pushing, pulling, folding, sliding, and many other skills. Enabling this level of command in a robot implies moving away from pre-designed sequences an into an autonomous understanding of manipulation from a perspective of geometry, physics, and logic. In order to explore this goal of providing robots with the versatile ability to manipulate objects, this thesis will consider three few questions related to

Figure 1-1: Examples of common dexterous manipulation tasks in our day to day: pulling a phone from a pocket, packing groceries, assembling furniture, picking a book from a shelf, opening a dresser, or preparing a package at a warehouse. Images generated with CLIP [103].

the dexterity of robotic tasks. Here, we will consider a manipulation task any problem that requires re-locating at least one object from an initial pose in space to another pose. An example of this is picking a book from a table and storing it on a shelf. With this perspective, we focus this thesis on the following questions:

How to achieve versatile dexterity? To enable dexterity we require our algorithms to efficiently reason about dynamics, geometry, and, perhaps most challenging, contact mechanics. Contact-rich interaction is the key enabler in this context. Unfortunately, the addition of contacts creates discrete events that require combinatorial decision-making (e.g. push an object from the side or pull it from the top). This makes the problem of finding policies very challenging without any kind of guidance. Today, there exist two main streams of work that aim to solve this problem: 1) research that prescribes a set of *primitives* (e.g. grasping, pushing, or pivoting) and concatenates them to manipulate an object [140, 29, 138, 66], which restricts the set

of solvable tasks, or 2) research that formulates a large, non-differentiable, and non-convex optimization problem that can solve the task by implictly reasoning about contact interaction [100, 91], which is computationally intractable without careful seeding. Can we find a rich model of task and mechanics that allows for a efficient discovery of manipulation skills?

Can we scale dexterity from experience? Following on our previous question, even with the right algorithm, model-based methods can be computationally expensive, are unable to exploit past experience, and require a degree of hand engineering, as they assume access to known object shape, pose, and desired trajectory. On the other hand, Many tasks can be easily described through videos and have the potential to scale robot manipulation planning and inference, but decoding relevant information from them remains a challenging problem. While recent approaches employing deep learning from images and videos have seen remarkable success in various prehensile robotic tasks [122, 137, 15, 136], there has been limited progress in the direction of dexterous manipulation tasks. An underlying struggle lies in building representations that can reason over object geometry, rigid-body mechanics, and the combinatorial decisions of choosing contact modes. Can we leverage scalable data sources, such as video, and our understanding of mechanics to find manipulation policies at scale?

Can we certify that a policy will succeed? Certification is central problem in robotics and automation. Challenging robotic tasks benefit from the ability to certify their safety and performance in the face of uncertainty. While feedback control often allows a system to adapt to disturbances and uncertainty, these algorithms rarely provide any explicit guarantees of where an arbitrary system will be stabilized. Over the past decade, there has been significant progress in certification by leveraging tools from Lyapunov theory [121, 87]. such frameworks allow the synthesis of feedback controllers with formal stability guarantees. This has led to impressive results in self-driving cars, aerial robots, and mobile robots. Perhaps the main limiting factor of these methods is their scalability, restricting their usage to simple dynamical systems

(one or two states) [61] with accurate sensing, unsuitable to manipulation tasks with large state spaces, and hybrid constraints. Moreover, these approaches are dependent on state-space representations, ubiquitous in control systems, which struggle to capture the important geometric and topological properties of the manipulation problem, limiting robustness to the mechanic's space of the system. Can there be a principled approach to finding manipulation policies that are certified to succeed under a set of uncertainty?

1.1 Contributions

In order to offer an answer to these three questions, this thesis has three main contributions:

1. **Planning Framework:** We propose a novel hierarchical framework to efficiently plan optimal 2D manipulation tasks, combining tools from dynamic programming, randomized motion planning. and mixed-integer optimization.

2. **Learning Framework:** We design a framework to solve manipulation tasks using a small set of examples given animated video descriptions of 2D tasks. To achieve this, we combine tools from deep learning and differentiable programming.

3. **Certification Model:** Finally, we derive a model to find manipulation policies certified to succeed, despite bounded object-pose uncertainty, for planar manipulation tasks under frictionless contact and point-finger manipulators.

Naturally, we have to restrict our work to a limited scope of problems where we can explore the nature of each of the three problems. To this end, we operated under spaces with polygonal objects in 2D environments. We also operate under a reduced model for our robots, assuming that our robot manipulators are a set of points that can float in space. These assumptions limit the scope of our algorithms to 2D problems. Nonetheless, these assumptions still allow us to capture some of the

main complexities of the manipulation problem, such as object geometry, environment interaction, and contact mechanics.

Our first contribution delves in the question of how to achieve dexterous behavior. We explore this problem under the task of moving a single object from an initial pose to a goal pose in a 2D environment. To this end, we derive a model of the contact interaction of planar manipulation tasks using quasi-dynamics and polytopic objects and environments. This allows us to cast contact mechanics under a mixed-integer convex optimization problem. Our model retains global optimality, global infeasibility and convergence, at the cost of combinatorial complexity. Then, we propose a framework to solve long-horizon manipulation tasks with alternating sticking contact interactions using dynamic programming, sampling-based planning, and mixed-integer optimization. We validate the applicability of this quasi-dynamic model and our proposed framework applied to manipulation tasks in challenging environments without a prescribed object motion, contact schedule or primitives.

Our second contribution explores the question of how to connect well-known mechanics models with deep learning to find policies for manipulation tasks. To this end, we present a differentiable learning framework that can learn to solve manipulation tasks from an animated video description of the problem in 2D. Our framework consists of a deep neural network that decodes a set mechanical parameters from video (such as object shape, object trajectory, contact-facet assignments, and friction cones) and a differentiable convex optimization model that finds a trajectory for the robot fingers that executes the task on the video. We restrict this model to 2D manipulation problems using a point robot finger and polygonal objects. We self-supervise this pipeline using a differentiable simulator that captures intermittent contact. We validate this approach under a variety of toy problems of picking and placing a set of objects from an initial pose to a goal pose in a toy 2D setup.

Our third and final contribution explores the question of certification in 2D manipulation tasks. We propose a method for synthesizing of robust manipulation policies that are certified to succeed in a frinctionless environment despite a bounded pose uncertainty. To this end, we derive a model of certificates that builds sequences of

invariant sets of configuration space, or funnels, to constrain the set of possible configurations of an object and funnel it towards a goal. We validate this theory with a wide variety of planar manipulation tasks, where the robot fingers and the environment are used to achieve certification, and in a real-world setup for the problem of sensorless planar grasping.

1.2 Thesis Outline

The remainder of this thesis is structured as follows: Chapter 2 covers the literature and key concepts referenced through this work. Chapter 3 describes our quasi-dynamic model and hierarchical framework to solve manipulation tasks in long horizons. Chapter 4 describes our differentiable learning framework to solve manipulation tasks given a video description. Chapters 5 and 6 describe our certification model for manipulation tasks, starting with the sub-problem of grasping and, then, extending toward general planar manipulation tasks. Finally, 7 concludes on the contribution of each of the previous chapters and proposes future steps and extensions to this work.

Chapter 2

Background

In this chapter we review some of the previous research most relevant to this work and introduce the concepts that we reference through the section.

2.1 Literature review

We split our review in four sections, each of relevance to its respective chapter on this thesis.

2.1.1 Planning Manipulation

Trajectory Optimization Trajectory Optimization is a popular tool to generate robot motions that need to respect some constraints [63, 75, 113]. The main benefit it provides comes from its versatility, as it generalizes between systems and problems. A significant body of work has been dedicated apply this tool to contact-rich motion generation [50, 82, 91, 100, 116, 124, 127]. Since contact is fundamentally discontinuous and dependent on geometry, modelling it requires the introduction of non-smooth constraints or discrete decision variables to determine its schedule. Hence, solution to this problem is subject to a dichotomy, either rely on fast nonlinear optimization tools [91, 100, 116], which rigorously model the problem at the cost of dependence on initialization and lack of convergence guarantees, or pose the problem as a large

combinatorial optimization, which has useful theoretical properties subject to an exponential growth in complexity [50, 124, 127]. Despite the aforementioned challenges, many have successfully applied trajectory optimization through contact to locomotion problems [38]. Manipulation, however, introduces a higher dimension of complexity, as geometry and hybrid dynamics play a larger role in the generation of object motions.

Quasi-Dynamic Models The idea of applying approximate physics to solve contact-rich planning problems has been pursued for many decades [93], promising a trade-off between performance and fidelity. On one extreme, some attempt to accurately model the dynamics of the entire system [100], while other completely drop the non-smooth/nonlinear aspects of the model [123] for performance. In the middle, and most relevant to us, some works approximate specific constraints of the problem while retaining other relevant aspects of contact interaction [28, 66, 124]. In the case of [28], a quasi-dynamic relaxation allows them to formulate the prehensile pushing problem with gravity in terms of motion cones. On the other hand, [124] shows how a simplified model guided by geometry can be used to generate long-horizon trajectories involving objects and tools. A relevant example is that of [127], where the hybrid aspects of the quadruped locomotion problem are accurately modeled, while the nonlinear relations of angular dynamics are relaxed. Our work borrows from this philosophy, where relaxations are applied to some elements of the problem, such that we can still capture the general behavior of the system and provide better performance and theoretical guarantees.

Motion Planning Motion planning is a well studied problem within robotics. There are two main approaches relevant to this work. First, Sampling-based motion planning algorithms, such as RRT [80], which have been a successful approach to solving problems with plenty of local minima. On the other hand, optimization-based techniques, such as CHOMP or GPMP [104, 48], have also seen success at getting high-quality solutions in high-dimensional setups, although local in nature. In con-

trast to these two approaches, Dynamic Programming (DP) [21] can solve optimal control problems to optimality with the caveat of high-computational complexity, which restricts this approach to short horizon problems.

Manipulation Planning The most common approach to manipulation is planning motions with *primitives*, such as grasping [101], pushing or pivoting. Many models can accurately optimize motions with these primitives [93]. Different primitives can be concatenated to manipulate a simple object [66]. A key limitation, however, is that primitives are specific to an object and an environment, which makes them hard to generalize to different tasks. Another line of research has focused on optimizing the contact interaction as part of the task [100, 91, 50], without assuming a primitive. However, optimizing contact interaction requires scheduling contact modes along the motion, which leads to combinatorial complexity, or modeling contacts implicitly through complementarity conditions, which are not differentiable. This makes it very challenging to optimize a manipulation plan without making assumptions on the contact interaction or providing significant seeding [3].

Manipulation on Long-Horizons Recent work has tried to alleviate these issues through sampling-based motion planning [32], which removes assumptions on the contact schedule and object motion. However, this approach is unable to guarantee the quality of its solution. Morever, as with RRT, this approach will struggle to reason globally on long-horizons, leading to slow computation. Other recent work has shown that the robot *Contact-Trajectory* can be globally optimized efficiently [4, 31], with the caveat that the object trajectory must be prescribed. Our work draws inspiration from these two philosophies and will aim to find object motions using a sampling-based approach, guided by a high-level search, and optimize the robot contact-trajectory using mixed-integer optimization.

2.1.2 Learning Manipulation

Learning manipulation from video Prior work in visual manipulation has involved learning predictive models from videos of a robot interacting with its environment. These models are used to find actions with simulation roll-outs, in a model-predictive fashion [54, 8, 122, 77, 135, 119, 40]. These approaches have been successful at solving prehensile manipulation problems where the dynamics are hard to model. When the dynamics have a known model, while these approaches remove the need to analyze visual data, mechanics or geometry, they struggle in generalizing to different tasks and geometries and can be data inefficient. Our work aims to address this limitation, under the planar non-prehensile manipulation task, by learning mechanical parameters from video and optimizing robot finger actions with a known rigid-body mechanics model, instead of learning a fully predictive model.

Bridging learning and model based approaches Recent research has tried to bridge the gap between deep learning and model-based techniques by introducing differentiable computational techniques as part of the learning pipeline. The first relevant line of work involves the introduction of differentiable optimization solvers [14, 13, 6], which introduce a parametric convex optimization program as a differentiable layer in a neural network. We leverage one of these solvers [6] to solve a quadratic program that takes as input mechanical parameters from a task (derendered from video) and outputs robot finger trajectories within our network. The second relevant line of work comes from the implementation of differentiable contact-mechanics simulators [41, 102], which execute a parametric physical simulation and allow to back-propagate from a measured output. We modify an existing differentiable simulator [41], using finite-differences, to handle intermittent contact dynamics and evaluate the error when executing the learned robot finger trajectories as part of a loss function.

2.1.3 Caging and Grasping

Sensorless Grasping. Stemming from the foundational works on sensorless manipulation by [51], and by [58] on sequences of squeezing grasps, this line of work aims to find grasping strategies that reliably bring an object to a known configuration, despite initial uncertainty in the object pose. In [58], Goldberg proposes an algorithm to find squeezing grasps that can reorient any convex polygon. This can be seen as a particular case of conformant path planning, as in [85, 51, 18], which synthesizes motions that drive a robot from an initially uncertain pose towards a goal, possibly under uncertain dynamics. This section maintains the spirit of these works and studies the case of general point-based manipulators, and general planar polygonal objects.

From caging to grasping. One way to constrain the object configuration to an invariant set is to cage it, a concept introduced to robotics by [106]. While not all cages lead to a grasp, as shown in [111], these always provide a certificate that the object is bounded to some compact set. More importantly, a subset of these cages are guaranteed to have a motion of the fingers that drives the cage into a grasp of the object. In this context, a grasp is a configuration in which a set of fingers immobilize an object. We are interested in synthesizing this type of cages that lead to a grasp in a unique configuration.

Computational models for caging. Many algorithms for cage synthesis have been studied since its introduction in [106]. The most relevant to this work is the optimization model in [1], which poses the caging condition in terms of convex-combinatorial constraints. We exploit the properties of this model to include requirements of convergence of the grasp process and observability of the final grasp. This optimization approach to caging, however, has a few limitations: it has only been applied to 2D cages, it has exponential bounds on complexity, and assumes that our robot has point fingers. Caging has also been studied in the context of randomized planning, such as in [131, 129], making no assumptions on shapes, and graph-search

defined on contact-space, in [10, 25], with polynomial bounds in complexity.

2.1.4 Certifying Manipulation

Certification and Robustness The ability to find policies that can certify the success of a particular task has not been achieved until recent years. One of the first relevant works to explore this problem came from Burridge et. al, [26], which described how controllers could be sequenced to achieve the success of a task via "funneling". Since then, many algorithms have emerged to solve this problem, leveraging tools such as Lyapunov functions [121, 87], Contraction theory [83, 115], and barrier functions [12, 120]. We remark the contribution of [88] to synthesize nonlinear controllers with Lyapunov certificates that can be sequenced to execute long-horizon tasks in continuous systems (vehicles and drones). Perhaps the biggest limitation of these tools is their limitation to continuous dynamics. Despite initial work on applying some of these tools into manipulation problems, it has been restricted to approximate certification[74, 62, 112] or specific tasks [11, 47, 46]. One way to constrain the object configuration to lie within a set is to "cage" it, a concept introduced to robotics by [106]. While not all cages lead to a goal object pose when the robot fingers move, as shown in [111], these always provide a certificate that the object is bounded to some compact set. We borrow heavily from this idea to transcribe it as a set of constraints [1, 2].

Conformant Path Planning Our work partly stems from the works on sensorless manipulation by [51], and by [58] on sequences of squeezing grasps. We aim to find manipulation strategies that reliably move an object towards a goal configuration, despite initial uncertainty in the object pose or along its trajectory. These problems are categorized under the context of conformant path planning, as in [85, 51, 18], which describe motions that drive a robot from an initially uncertain pose towards a goal, possibly under uncertain dynamics. This chapter follows the vision of these works and studies the case of moving a planar polygonal object along a trajectory with a point-based manipulator and a line-segment external environment. Our work

proposes a computational model to optimize grasping trajectories that would be certified to succeed despite initial (bounded) uncertainty on the object pose. We achieve this by concatenating a sequence of cages that gradually close the object pushing it to a goal configuration.

2.2 Mixed-Integer Programming

A Mixed-Integer Convex Program (MIP) [56] is an optimization problem of the form:

$$\min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}, \mathbf{z})$$

subject to:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} \in \mathcal{H},\ \mathbf{x} \in \mathbb{R}^{\dim(\mathbf{x})},\ \mathbf{z} \in \{0,1\}^{\dim(\mathbf{z})},$$

where $f(\mathbf{x}, \mathbf{z})$ is a *convex* cost function, $\mathcal{H}$ is a *convex* set of constraints, $\mathbf{x}$ is a vector of continuous variables and $\mathbf{z}$ a vector of binary variables. Mixed-Integer Programs have many useful properties and applications, as well as off-the-shelf solvers.

In particular, we make use of two useful properties of these models:

- **Search guarantees.** If the problem has a solution, the algorithm will find it. If it does not, it will report so. If a convex cost function is provided, we can always find a global optimal, provided one exists.

- **Integer constraints:** binary variables allow to encode logical implications as linear constraints. A common technique is known as the big-M formulation. For example, given a binary variable $\mathbf{y}$, we can incorporate logical constraints of the form

  $$\mathbf{y} = 1 \Rightarrow Ax \leq b,$$

  as the inequality

  $$Ax \leq b + M(1 - \mathbf{y})$$

where M is a large positive number. The satisfaction of this inequality implies that if $\mathbf{y} = 1$ then $Ax \leq b$ is enforced, otherwise when $\mathbf{y} = 0$ the inequality is largely relaxed, or effectively nullified if M is large.

These properties come at the cost of exponential bounds on computation time that grow with the number of binary decision variables variables. Most algorithms to solve this problem rely on "branch-and-bound" and "cutting-plane" methods, which perform and iterative search over a tree of the binary decision variables and rely on information from the optimization problem to quicken the search. These techniques operate over this discrete search space, discarding large sets of branches by solving a hierarchy of relaxations of the original problem [22]. Because of this, these optimization problems often scale much better, reducing the exponential complexity burden [134].

However, numerical conditioning can hinder the performance of this process, and the conditions under which the search algorithms are efficient are not well characterized. In practice, we solve mixed-integer programs using off-the-shelf optimization software, which have provided satisfactory performance.

Chapter 3

Optimal Planning of Manipulation Tasks

3.1 A Global Quasi-Dynamic Model of Contact-Trajectory Optimization

Since the early years of robotics, resolving contact interaction has hindered the deployment of robots for general manipulation tasks. Contact has been, and still is, difficult to observe, predict and control. The last decade has seen significant advance in planning-through-contact tools for general motion generation. Their wide-spread use in robotics, however, has been limited by coarse approximations of the mechanics of contact, or by numerical difficulties that arise from the choice of problem formulation – e.g., hybrid dynamics, non-unique solutions, non-convex constraints. These consequences are difficult to avoid in a problem often specified as the optimization of the interaction between a robot and an object, both free variables, to achieve a loosely specified behavior.

In this work we focus on a different perspective: We assume as input a desired trajectory of the object in its environment, and ask how, where, and when should the robot make contact to achieve it. We refer to this problem as **Contact-Trajectory Optimization**, in contrast to the more general trajectory optimization problem,

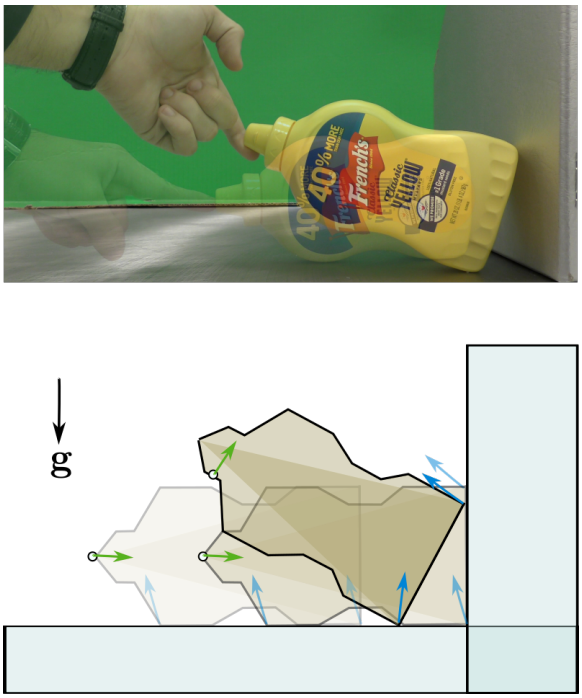


Figure 3-1: By reasoning about an approximate world, with quasi-dynamics and polytopic shapes, we reliably optimize alternated-sticking contact interactions to achieve complex manipulation tasks. Our model receives an object motion and outputs a contact-trajectory (positions and forces).

where the object trajectory is also a free variable [100]. This sub-problem retains some of the key challenging aspects of planning through contact, namely hybridness and non-convexity. We will see, however, that this provides multiple benefits. It leads to a better defined problem, with more numerically stable solutions, and ultimately a global model for planning-through-contact with certificates: global optimality, global infeasibility and convergence.

In order to achieve this goal, we rely on a quasi-dynamic approximation of the motion equations and polygonal (convex or non-convex) descriptions of objects and environments. The term quasi-dynamics refers to an approximation of the equations that describe the time-evolution of the system that ignores high-order inertial terms, assumes uniform pressure contact distributions, and exploits a mixed integer-convex approximation of nonlinear relations. In contrast to previous work that use exact contact-dynamics ([100]), we provide a global model. In contrast to previous works that rely on approximate contact-dynamics ([123, 124]), we implement approxima-

tions that obey non-penetration and contact complementarity, fundamental properties in manipulation. The key tools to achieve this come from mixed-integer convex optimization, including piecewise McCormick envelopes and disjunctive constraints. Hencer, in this section, we present two main contributions:

- **Modeling** of the contact-trajectory optimization problem with quasi-dynamics and polytopic objects and environments as a mixed-integer convex optimization problem. Our model retains global optimality, global infeasibility and convergence.

- **Validation** of the model on multi-contact manipulation behaviors on planar environments, sagittal and transversal, both in simulation and on real robot environment.

Fig. 3-1 illustrates our vision for this model: by optimizing contact-trajectories in simple environments we can generate motion plans to complete tasks that involve complex contact interactions. We show how our model optimizes contact-trajectories for planar tasks consistently in less than 1 s.

The remainder of this section is organized as follows: Sec. 3.1.1 provides an overview of the model, its assumptions and properties. Sec. 3.1.2 describes the proposed model in detail, while Sec. 3.1.3 discusses its implementation as a Mixed-Integer Convex Program. Finally, Sec. 3.1.4 demonstrates the model with simulation and real experiments.

3.1.1 Approach Overview

In this subsection we provide an overview of our model, describe its properties and discuss its assumptions. Given the trajectory of a polytopic object, this model will find a sequence of contacts interactions that achieve this motion and minimize a convex objective.

Inputs and Notation

Our model receives as inputs the trajectory of the object and the geometry of the object and environment. We introduce the following notation and variables:

1. **Object:** A polytopic rigid-body $\mathcal{O}$ with N_v vertices and N_f facets. Each facet $\mathbb{F}_f$ has N_v^f vertices, with nominal positions $\mathbf{v}_v^f$, with a corresponding friction cone $\mathbb{FC}_f$, represented with R_d rays. In the 2D case we have $N_f = N_v$ and $N_v^f = 2$.

2. **Trajectory:** A set of object poses over T discrete time-steps. We describe each pose, at a time-step t, as $\mathbf{q}(t) \in \mathcal{C}$, where $\mathcal{C}$ is the configuration space of the object. In the 2D, $\mathcal{C}$ corresponds to the x, y, θ coordinates of $SE(2)$.

3. **Manipulator:** A set of N_c contacts points. We describe the c_{th} contact-point, at time-step t, as $\mathbf{p}_c(t) \in \mathcal{W}$, where $\mathcal{W}$ is the workspace. In the 2D case, the workspace simply corresponds to each position (x, y) that can be reached by the robot.

4. **Environment:** A polytopic environment with N_e facets, described as planes with friction cones $\mathbb{FC}_e$. Additionally, the free-space between the object and environment is segmented in N_R convex polytopic regions $\mathcal{R}_r = \{\mathbf{x} \in \mathcal{W} \mid A_r\mathbf{x} < b_r\}$.

A diagram describing these elements, at a fixed time-step, can be seen in Fig. 3-2.

Desired Properties

The proposed model describes the time-evolution of the multi-contact system while maintaining the following desirable properties:

1. **Versatility** the model must capture rigid-body (quasi) dynamics and hybrid (sticking) contact interactions while remaining agnostic to the task, object, and environment.

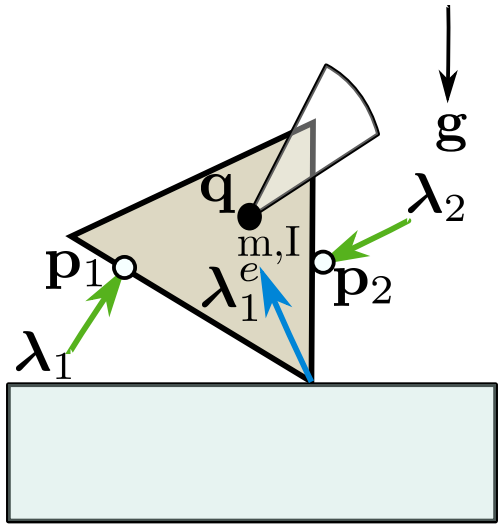


Figure 3-2: For a trajectory of T time-steps, our model solves for the finger positions $\mathbf{p}$ (dots) and contact forces $\boldsymbol{\lambda}$ (green), as well as reaction forces with the environment $\boldsymbol{\lambda}^e$ (blue) represented by separate surrogate variables.

2. **Global Optimality** If there is a feasible set of contact trajectories that achieve this motion, the model will always find it. Moreover, if there is a convex cost function, the model can always be used to find the solution that globally minimizes it.

3. **Infeasibility Detection** If the model is infeasible, then it implies that the object motion cannot be achieved.

Throughout the upcoming sections we will discuss how the constraints of the model preserve these properties.

Modeling Assumptions In order to achieve the desired properties of the model, we make the following assumptions:

1. Objects are rigid, with uniform contact surfaces (such that line contacts can be approximated by contact with two vertices), and approximated as combinations of "simple" polytopes.

2. Object motions occur at low speeds, such that high order intertial effects are negligible.

3. Robot fingers have small masses, such that there is no impacts between the robot and the object.

4. Interactions between the object and robot are a sequence of alternating sticking contacts, such that friction cones can be approximated as R_d rays.

The upcoming sections will describe how these modeling decisions translate into a mixed-integer convex model.

3.1.2 Global Quasi-Dynamic Model

In this subsection we present our global quasi-dynamic model. Each paragraph describes a set of constraints, for which we discuss their derivation and numerical implementation.

Force-Motion Equations

By solving the Lagrange equations for a an object, assumed to be rigid, we obtain the standard force-motion equation that describes the dynamics of the system:

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} = \tau_g(\mathbf{q}) + H^T(\mathbf{q})\mathbf{\Lambda},$$

where the given object configuration is denoted by $\mathbf{q} = \begin{bmatrix} \mathbf{q}_t \\ \mathbf{q}_\theta \end{bmatrix}$, where $\mathbf{q}_t$ corresponds to the position of the object and $\mathbf{q}_\theta$ to its orientation, $H(\mathbf{q})$ maps the effect of the contact forces $\mathbf{\Lambda} = \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\lambda}^e \end{bmatrix}$ as body wrenches, τ_g describes the gravitational wrench, $M(\mathbf{q})$ is a mass matrix, and $C(\mathbf{q}, \dot{\mathbf{q}})$ maps velocities $\dot{\mathbf{q}}$ into Coriolis forces. When velocities are low and inertial effects are small, the equation of motion reduces to:

$$M(\mathbf{q})\ddot{\mathbf{q}} = \tau_g(\mathbf{q}) + H^T(\mathbf{q})\mathbf{\Lambda}.$$

From this form, we can decouple the equations of motion into translational and rotational components:

Translational Motion The top rows are directly Newton's second law applied to the center of mass of the object:

$$m\ddot{\mathbf{q}}_t(t) = m\mathbf{g} + \sum_c \boldsymbol{\lambda}_c(t) + \sum_n \boldsymbol{\lambda}_n^e(t) \tag{CT1}$$

Where m is the object mass, $\boldsymbol{\lambda}_c$ is the force applied by the c_{th} finger $\mathbf{p}_c$, and $\boldsymbol{\lambda}_n^e$ is the environmental force applied over the n_{th} vertex of object. Eq. (CT1) is a linear sum of unknown terms, which is a convex constraint.

Rotational Motion The bottom rows, recasting the Jacobian $H^T(\mathbf{q})$ as a sum of cross-product operations, become:

$$I\ddot{\mathbf{q}}_\theta(t) = \sum_c (\mathbf{p}_c(t) - \mathbf{q}_t(t)) \times \boldsymbol{\lambda}_c(t) + \sum_n R^r(q_\theta(t))\mathbf{v}_n \times \boldsymbol{\lambda}_n^e(t)$$

where I is the object's moment of inertia w.r.t its center of mass, $\mathbf{p}_c$ is the position of the c_{th} finger, and R^r is a rotation matrix from 0 to $\mathbf{q}_\theta$. Unfortunately, the cross-product operation is a bilinear term, making each torque a non-convex constraint. For now, we replace each cross-product with a surrogate term and substitute the rotational dynamics with:

$$I\ddot{q}_\theta(t) = \sum_i \boldsymbol{\tau}_i(t) \; + \; \sum_n R^r(\mathbf{q}_\theta(t))\mathbf{v}_n \times \boldsymbol{\lambda}_n^e(t) \tag{CT2}$$

where $\boldsymbol{\tau}_i \approx (\mathbf{p}_i - \mathbf{q}_t) \times \boldsymbol{\lambda}_i$ is mixed-integer convex approximation, the method by which this approximation is computed is described in detail in Sect. 3.1.2. Note that, fortunately, the terms corresponding to external forces are linear constraints on the external force, since $R^r(\mathbf{q}_\theta(t))$ is known a priori.

Equations (CT1) and (CT2) are often referred to as the "centroidal dynamics" of a system, ubiquitous in the legged locomotion community, see [38, 99].

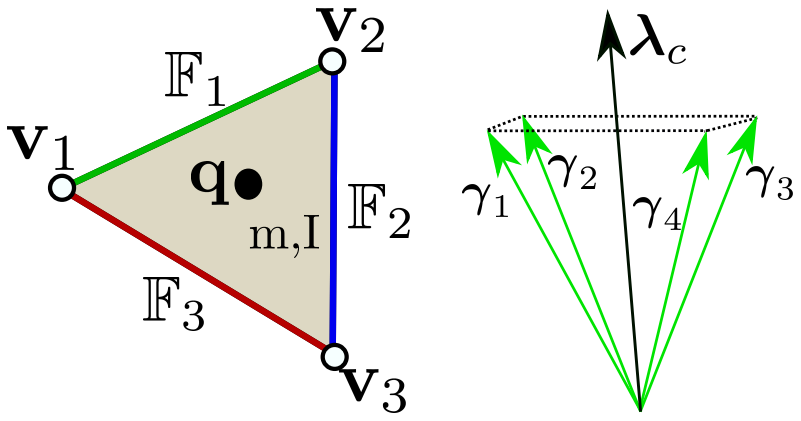


Figure 3-3: (left) Contact Assignment: the object is decomposed in N_f facets and N_v vertices, and contacts forces are constrained to lie within the respective friction cone. (right) Friction cone approximation as $R_d = 4$ rays.

Contact Scheduling

In order to ensure consistency between equations (CT1)-(CT2) and the position of the fingers, forces $\boldsymbol{\lambda}_c$ must be active only if $\mathbf{p}_c$ is in contact with one of the facets of the object. Moreover, this contact force must be constrained to lie within its corresponding friction cone. This leads to a hybrid condition, as the constrains change depending on the position of the fingers and the shape of the object. To achieve this, we leverage the object representation as a polytope with N_f facets $\mathbb{F}_f$ and the fingers described as points, as shown in Fig. 3-3 (left).

To include this constraint, at each time-step t, we introduce a binary matrix as part of the decision variables $\mathbf{T}(t) \in \{0,1\}^{N_f \times N_c}$ that maps the position of each contact c to some facet f of the object via the constraint:

$$\mathbf{T}_{f,c}(t) = 1 \Rightarrow \begin{cases} \mathbf{p}_c(t) \in \mathbb{F}_f(t) \\ \boldsymbol{\lambda}_c(t) \in \mathbb{FC}_f(t) \end{cases} . \tag{CT3}$$

This constraint enforces that forces are only active when the fingers are in contacts with a facet. Since each facet $\mathbb{F}_f$ has N_v^f vertices with position $\mathbf{v}_n^f$, we model the facet

assignment constraint as:

$$\mathbf{p}_c(t) \in \mathbb{F}_f(t) \Leftrightarrow \mathbf{p}_c(t) = \sum_j \rho_j(t)\mathbf{v}_j^f,\ \sum_j \rho_j(t) = 1,$$

which constrains the finger position to be a convex combination of the facet vertices, where ρ are assignment weights. Then, we constrain the force to lie on its friction cone $\mathbb{FC}_f$, described with R_d rays $\boldsymbol{\gamma}_{f,1}, \ldots, \boldsymbol{\gamma}_{f,R_d}$, with $R_d = 2$ for 2D and $R_d \geq 3$ for 3D, depicted in Fig. 3-3 (right), as:

$$\boldsymbol{\lambda}_c(t) \in \mathbb{FC}_f(t) \Leftrightarrow \boldsymbol{\lambda}_c(t) = \sum_k \alpha_k(t)\boldsymbol{\gamma}_{f,k},\ \alpha_k(t) > 0,$$

which constrains each contact force to be a conic combination of the friction cone rays, where α are also assignment weights. Finally, since forces cannot be active if the fingers are not in a facet, we add the following constraint:

$$\sum_f \mathbf{T}_{f,c}(t) = 0 \Rightarrow \boldsymbol{\lambda}_c(t) = 0 \tag{CT4}$$

We transcribe all $\Rightarrow$ operator using big-M formulation. The constraints defined by Eqs. (CT3) and (CT4) are equivalent to a complementarity constraint over the contact force [100], since the contact force will only be non-zero once the finger touches the object.

Alternated-Sticking For robustness of the optimized trajectories, we further require that finger contacts are not sliding between time-steps, as our model assumes that contact-trajectories are a sequence of sticking contacts, we model this constraint as:

$$\mathbf{T}_{f,c} = 1 \Rightarrow \mathbf{p}_c(t+1) = \sum_j \rho_j(t)\mathbf{v}_j^f, \tag{CT5}$$

which enforces that if the finger is in contact at time t then it must remain sticking at time-step $t+1$ before switching to a different contact location. If we were to allow sliding-contact, we would need to represent the border of the friction cone which

is a non-convex constraint [60]. This is possible, but would extend the size of the optimization problem.

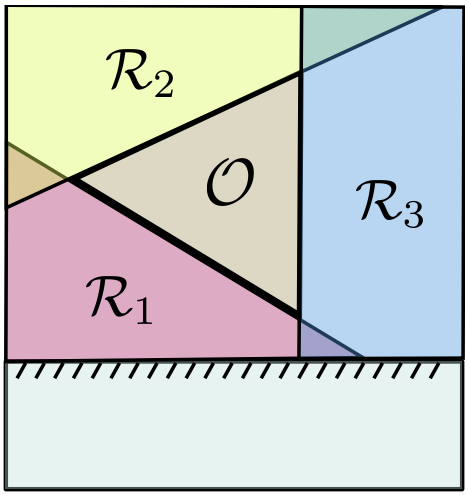


Figure 3-4: Non-Penetration Constraint: The free-space between the object and the environment is decomposed in N_R convex regions $\mathcal{R}$, including the facets of the object.

Environment Contacts Since we provide the object motion as an input and we assume an uniform pressure distribution on the object facets, the contact schedule between the object and the environment is also known. Hence, the model only needs to constrain reaction forces at each object vertex $\mathbf{v}_n$, to lie within their respective friction cone $\mathbb{FC}_n^e(t)$, with R_d rays $\boldsymbol{\gamma}_{n,1}^e(t), \ldots, \boldsymbol{\gamma}_{n,R_d}^e(t)$. This constraint is imposed, for the n_{th} vertex at time-step t, as:

$$\boldsymbol{\lambda}_n^e(t) \in \mathbb{FC}_n^e(t) \tag{CT6}$$

As before, the friction cone constraint is included as $\boldsymbol{\lambda}_n^e(t) = \sum_k \alpha_k^e(t)\boldsymbol{\gamma}_n^e(t),\ \alpha_k^e(t) \geq 0$, where α^e are assignment weights for each ray. Since these vertex contacts can also slide, some of the assignment weights must fixed to zero to have the force in the border of the friction cone. This can be done when setting-up the optimization problem, since the contact modes between each vertex and the environment are known from the given trajectory.

Non-Penetration

Naturally, fingers cannot penetrate the object nor the environment. A strategy to enforce this is to segment the free-space into N_R convex, possibly overlapping, regions $\mathcal{R}_r$ that cover the free space of the workspace [105, 78, 127], as shown in Fig. 3-4. Each finger is then constrained to lie within one of this regions. This constraint is added through a binary decision matrix $\mathbf{R}(t) \in \{0,1\}^{N_R \times N_c}$, such that:

$$\mathbf{R}_{r,c}(t) = 1 \Rightarrow \mathbf{p}_c(t) \in \mathcal{R}_r(t) \tag{CT7}$$

with $\sum_r \mathbf{R}_{r,c}(t) = 1, \ \forall c$. Here, region assignment is done as:

$$\mathbf{p}_c(t) \in \mathcal{R}_r(t) \Leftrightarrow A_r(t)\mathbf{p}_c(t) < b_r(t)$$

which are all linear constraints on the finger positions. Finding these regions is a separate optimization problem, examples include [44].

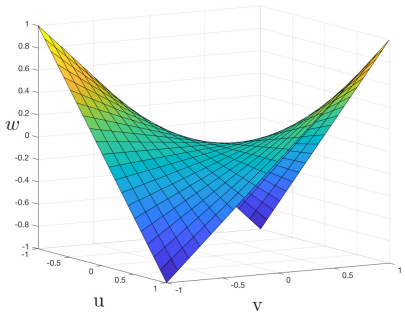
Modeling Approximations via McCormick Envelopes

The constraints defined by Eq. CT2 include a cross-product operation, which is a non-convex function due to the presence of bilinear equalities. For the purpose of this section, we refer to a bilinear equality constraint as a relation of the type:

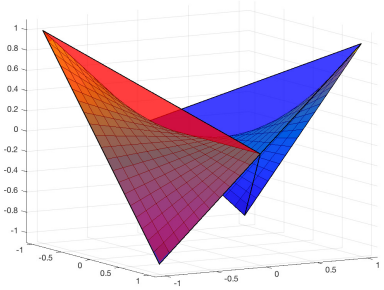
$$w = u \cdot v$$

where u, v and w are decision variables. Concretely, each cross-product $\boldsymbol{\tau}$ adds $4|\boldsymbol{\tau}|-6$ bilinear equalities to the model. To illustrate their non-convexity, we plot the surface $w = uv$ in Fig. 3-5a.

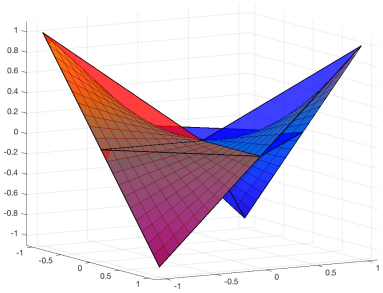
While there are several methods to approximate or relax non-convex constraints of this type, we are interested in an approximation that 1) can be embedded in a global model and 2) preserves the hybrid structure of contact. A technique that achieves this purpose is that of *Piecewise McCormick Envelopes*. Initially proposed



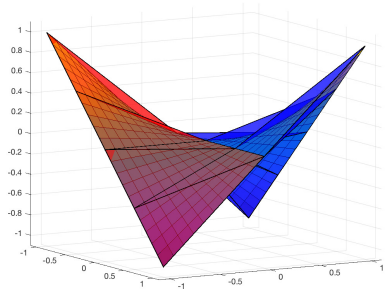
(a) $w = uv$



(b) $M = 1$



(c) $M = 2$



(d) $M = 4$

Figure 3-5: (a) Bilinear equality curve for $w = uv$, (b)-(d) Piecewise McCormick Envelopes with different levels of accuracy.

by McCormick in [95], this approximation covers the bilinear surface $w = uv$ with M convex envelopes, each being the convex-hull of the surface between a segment on the uniform set $-u_{M/2}, \ldots, u_{M/2}$ and $v \geq 0$, $v \leq 0$. Examples of these envelopes are shown in Fig. 3-5b, blue for $v \geq 0$ and red for $v \leq 0$. Then, each approximation of w is constrained to lie within the envelopes with a binary decision matrix $\mathbf{W}(t) \in \{0,1\}^{2\times M}$ as:

$$\mathbf{W}_{1,k}(t) = 1 \Rightarrow \begin{cases} w \geq u_{k-1} v \\ w \geq u_k v + u - u_{k-1} \\ w \leq u_{k-1} v + u - u_k \\ w \leq u_k v \\ u_{k-1} \leq w \leq u_k \\ v \geq 0 \end{cases} \tag{CT8}$$

$$\mathbf{W}_{2,k}(t) = 1 \Rightarrow \begin{cases} w \leq u_{k-1} v \\ w \leq u_k v - u - u_{k-1} \\ w \geq u_{k-1} v - u - u_k \\ w \geq u_k v \\ u_{k-1} \geq w \geq u_k \\ v \leq 0 \end{cases}, \tag{CT9}$$

which constraint w to lie within one of the envelopes, depending on the sign of v. This approximation is not exact; however, it provides several useful properties:

1. The segmentation provides upper and lower bound on the quality of the approximation.

2. A larger M makes the approximation arbitrarily tight, at the cost of a larger binary matrix.

3. The approximation preserves the hybrid and nonlinear structure of the bilinear surface, as:

$$\begin{aligned} u \cdot v \geq 0 \quad &\Rightarrow w \geq 0 \\ u \cdot v \leq 0 \quad &\Rightarrow w \geq 0 \\ u \cdot v = 0 \quad &\Rightarrow w = 0 \end{aligned}$$

4. If the relaxed envelope constraint is infeasible then the original bilinear constraint is also infeasible, as the envelopes cover the original curve in their solution space.

Examples of approximations of different sizes are shown in Fig. 3-5a-3-5d.

3.1.3 Contact-Trajectory Optimization

From the constraints presented above, we formulate a Mixed-Integer Optimization problem. We summarize all the decision variables required for our model in Table

Name	Description	Size	C/B
$\mathbf{p}$	Point Locations	$D \times N_c \times T$	C
$\boldsymbol{\lambda}$	Contact Forces	$D \times N_c \times T$	C
$\boldsymbol{\lambda}^e$	External Forces	$D \times N_v \times T$	C
$\boldsymbol{\tau}$	Torque Approximation	$(4\|\boldsymbol{\tau}\| - 6) \times N_c \times T$	C
ρ	Facet weights	$\sum_f N_v^f \times N_c \times T$	C
α	Friction Cone Weights	$R_d \times N_c \times T$	C
α^e	External Cone Weights	$R_d \times N_v \times T$	C
$\mathbf{T}$	Contact Assignment	$N_f \times N_c \times T$	B
$\mathbf{R}$	Non-Penetration	$N_R \times N_c \times T$	B
$\mathbf{W}$	McCormick Envelope	$2M \times (4\|\boldsymbol{\tau}\| - 6) \times N_c \times T$	B

Table 3.1: Summary of Decision Variables (C: Continuous, B: Binary).

3.1. For notation convenience we define three sets of decision variables: contact-trajectories $\mathcal{X} = \{\mathbf{p}, \boldsymbol{\lambda}, \boldsymbol{\lambda}^e, \boldsymbol{\tau}\}$, assignment weights $\mathcal{Y} = \{\rho, \alpha, \alpha^e\}$, and binary matrices $\mathcal{T} = \{\mathbf{T}, \mathbf{R}, \mathbf{W}\}$.

Optimization problem Adding the constraints and all the decision variables, we transcribe the optimization problem into **MIQP1**:

$$\textbf{MIQP1} : \underset{\mathcal{X},\mathcal{Y},\mathcal{T}}{\text{minimize}}\ J = \begin{bmatrix}\mathcal{X} & \mathcal{Y} & \mathcal{T}\end{bmatrix} Q \begin{bmatrix}\mathcal{X} \\ \mathcal{Y} \\ \mathcal{T}\end{bmatrix} + q^T \begin{bmatrix}\mathcal{X} \\ \mathcal{Y} \\ \mathcal{T}\end{bmatrix}$$

subject to:

1. For time-step $t = 1$ to $t = T$:
   (a) Quasi-Dynamics (CT1)-(CT2).
   (b) For fingers $c = 1$ to $c = N_c$
      - Contact-Trajectory Assignment (CT3)-(CT5).
      - Non-Penetration (CT7).
   (c) Environmental Contact (CT6).
      - Pre-fix weights α^e when sliding.
   (d) For bilinear terms $b = 1$ to $b = N_c(4D - 6)$:

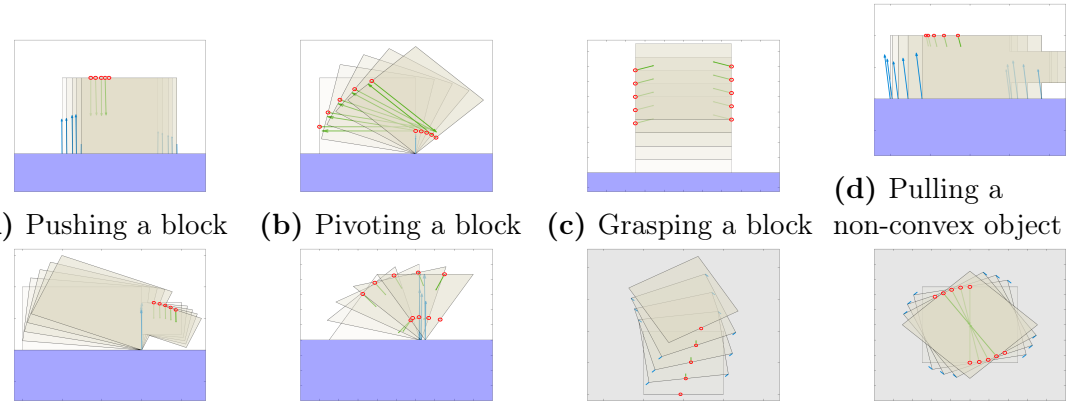
- Piecewise McCormick Envelope (CT8)-(CT9).

Where Q is a positive-semi-definite (PSD) square matrix and q is a column vector of appropriate size, these matrices can be chosen according to the problem.

Properties of the Model The formulation of the problem can be categorized as a Mixed-Integer Quadratic Program (MIQP); this type of problem has several useful properties [56]. Mainly, if given sufficient time (with worst-case exponential complexity), a solver can always find the global solution to the optimization problem. This also implies that it does not require any form of initialization or warm-start. Finally, If **MIQP1** results infeasible we can guarantee that the original problem, with exact bilinear equality constraints, is also infeasible [95].

The complexity of the program grows exponentially with the number of binary variables in the model, defined by the number of fingers, tightness of the McCormick envelopes, shape of the object and environment.

3.1.4 Validation and Applications



(a) Pushing a block **(b)** Pivoting a block **(c)** Grasping a block **(d)** Pulling a non-convex object

(e) Pivoting a non-convex "tool" **(f)** Pivoting a triangle **(g)** Sliding with curvature **(h)** Rotating 90° in place

Figure 3-6: Primitive behaviors optimized with our model, without any initialization or seeding. Red points are the finger locations, green arrows the applied forces, and blue arrows represent the environmental contacts.

To demonstrate some of the capabilities of our model, we implement **MIQP1** and asses its application for set of traditional manipulation problems. First, we aim to

validate the model's ability to optimize simple manipulation behaviors and detect infeasible ones. We then show different applications of our model for manipulation problems that involve interaction with the robot and the environment. Finally, we execute a set of open-loop experiments to illustrate how this model transfers to a real-world set-up.

We generate all the trajectories in MATLAB R2019b, running on an Intel Core i9 laptop with Mac OS X High Sierra. We use Gurobi 8.1.0 [59], an off-the-shelf optimization software, as our MIP solver. All of our tests are done in two-dimensional set-ups, such that $|\boldsymbol{\tau}| = |\mathbf{p}| = N_v^f = R_d = 2$. We fix the accuracy of each piecewise McCormick envelops to $M = 4$. We manually generate each object trajectory and segment the free-space of each task into convex regions $\mathcal{R}$. For all problems, we add a quadratic cost-function that minimizes the applied force and smooths the finger trajectories:

$$J = \sum_{t=1}^{T} \sum_{c=1}^{N_c} ||\boldsymbol{\lambda}_c(t)||^2 + ||\ddot{\mathbf{p}}_c(t)||^2 - \beta_c(t)$$

Second derivatives are computed within the model with backwards-Euler scheme for simplicity and numerical stability. The term β is a lower-bound convex-approximation of the distance between each contact-force and the border of its friction cone, computed as in [3, 66], this term has to be linear in order for the cost-function to be convex.

Model Validation We start by validating the functionality of the model in several simple manipulation problems. In particular, we show its ability to optimize contact-trajectories for "primitive" object motions and to detect when an object motion is infeasible. We generate a set of object trajectories in the sagittal $-XZ-$ plane for which the optimal solution can be intuitively found. For this, we use three objects: 1) a block, 2) a triangle, a 3) a non-convex object. Unless otherwise specified, all surfaces have a friction coefficient of $\mu = 0.1$. Then, we generate the following trajectories of $T = 5$ time-steps:

Block 1-finger sliding (Fig. 3-6a), 2-finger pivoting (Fig. 3-6b), 2-finger grasping

(Fig. 3-6c). We observe that one finger is sufficient to slide the object by pushing it.Grasping directly from the ground requires two or more contacts in order to lift the object. In the case of pivoting, a surface with $\mu = 0.1$ is not sufficient for 1-finger pivoting and the solver only finds a solution when $N_c \geq 2$, which allows to create internal force to increase friction.

Triangle 2-finger pivoting (Fig. 3-6f). Consistent with physical intuition, $\mu = 0.1$ does not provide enough friction for the object to pivot without sliding; hence, the model chooses to place a second finger to push on the ground a generate additional reaction force, providing enough torque. Similar to the block case, fixing $N_c = 1$ leads the solver to report infeasibility.

Non-Convex Object 1-finger sliding (Fig. 3-6d) and 1-finger pivoting (Fig. 3-6e). For both tasks, one finger is sufficient. In contrast to the other objects, the non-convex "end" of the tool can be used to generate sufficient torque and pivot with only one finger.

Transverse manipulation 1-finger curved-pushing (Fig. 3-6g) and 2-finger in-place rotation (Fig. 3-6h). For straight line pushing or pushing with small curvature, one finger is sufficient, as it can generate enough torque to slide and rotate. However, two fingers are required rotate the block around its geometric center.

As a reference, all trajectories are optimized in the range of 0.04 s to 0.44 s of computation. We stress the benefits of global optimality in this problem, since small changes in the friction coefficient or geometry of the object lead to different solutions. Furthermore, the ability to report when a task is infeasible, either because it requires more than one finger or because it is physically impossible, also provides useful insight on the primitive itself.

Applications Once we have proven that our framework can effectively generate simple behaviors, we aim to show how it can also reason about longer horizon tasks that involve complex contact interactions. For this, we show two examples of "extrinsic" dexterity [34], where contacts with the environment are essential part of the manipulation problem, on two different set-ups with $T = 10$ time-steps. In both

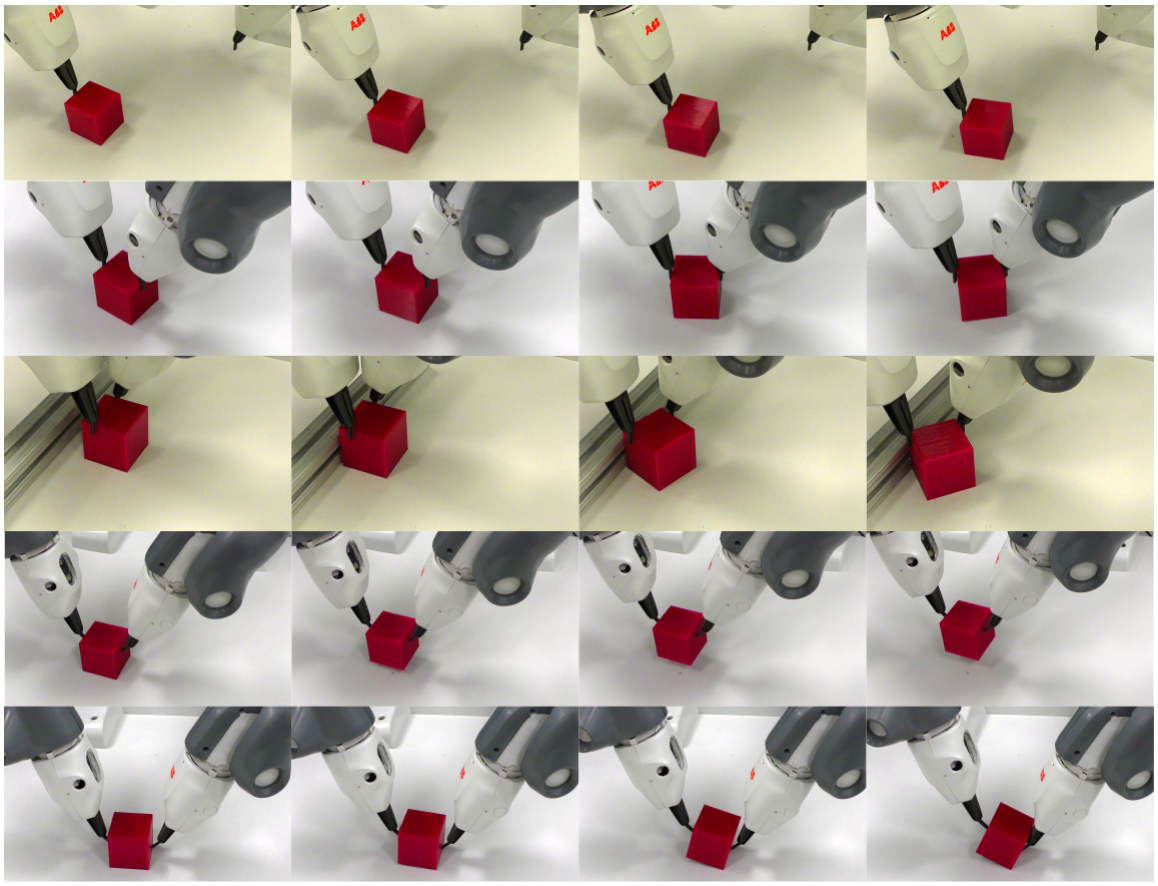
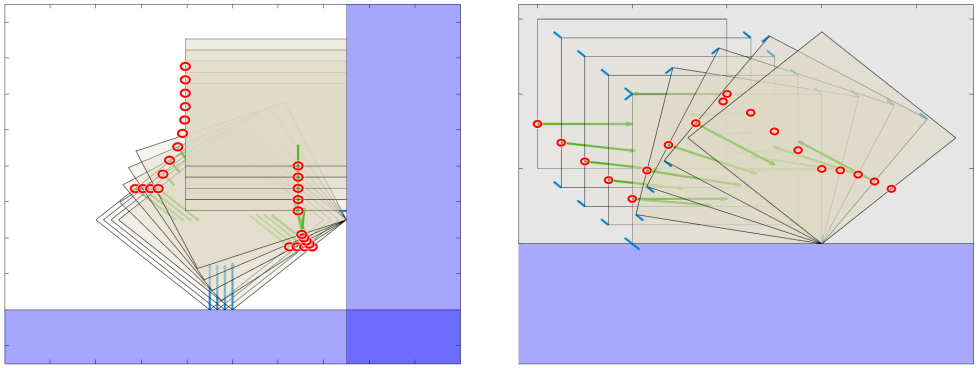


Figure 3-7: Experimental validation of our model Our optimized behaviors can be applied for real world execution, snapshots of each open-loop experiment are shown (each blue box corresponds approximately to the goal pose of the object). Top to bottom: 1) transversal pushing with a desired angle. 2) Rotating 45° in-place. 3) Transversal pivoting against a wall. 4) Grasping vertically. 5) Sagittal pivoting.



(a) Pivoting against a wall with gravity

(b) Planar sliding with two fingers and a wall

Figure 3-8: Two applications of our model for extrinsic dexterity in the sagittal and transverse planes.

cases, the object trajectories were specified manually on a set-up with a block and a wall. We optimize these "exrinsic" re-orienting strategies in two separate planes:

Sagittal Accounting for the effect of gravity, the object is pushed to a wall, pivoted w.r.t. it, and lifted vertically. The execution of this motion is shown in Fig. 3-8a. The optimal solution is computed in under 10 sec.

Transverse In this case, the object slides against a wall and is then pivoted with respect to it. This trajectory is shown in Fig. 3-8b. The optimal solution finds a sticking contact location for each finger. The problem is solved in 2 sec.

Experimental Validation In order to demonstrate that the contact-trajectories generated by this model translate to the real world, we conduct experiments on a two-arm robot manipulating and a cubic object on a table.

Our robotic platform is an ABB YuMi® (IRB-14000) robot, which has two 7 DOF arms with a custom point-finger attached to each end-effector. We work with a Robot Operating System (ROS) setup, interfaced with MATLAB R2019, and run all the demonstrations in an open-loop fashion, guided through position commands.

Transverse Manipulation Our first demonstration focuses on planar sliding on an uniform surface, replicating the behaviors in Figs. 3-6g, 3-6h, and 3-8b. Snapshots of the execution of each task are shown in Fig. 3-7 (top to middle). Despite the open-loop fashion of these demonstrations, we achieve a reliable execution of the optimized

contact-trajectory, as long the trajectory is executed slow enough, otherwise impacts occur and inertial effects are noticeable.

Sagittal Manipulation Our second demonstration shows how our model can accurately generate motions that interact with gravity. For this, we replicate the pivoting and grasping behaviors, depicted in Figs. 3-6b and 3-6c. Executions are shown in Fig. 3-7 (bottom two). In this example, however, trajectories are brittle and highly-dependent on the initial pose of the object and the robot contacts. This points at the importance of tracking position along with forces, as contacts must remain sticking in order to execute these tasks correctly.

In both cases, accuracy is achieved by placing the object in the precise initial condition. The general scenario, with uncertainty, requires a controller and perception to execute each motion.

Model Limitations Perhaps the main limitation of this approach comes from needing to specify as input the object motion. While in many cases object trajectories are intuitive to specify, many simple motions lead to infeasible contact-trajectories, such a lifting a steep triangle. The second main limitation is the restriction to alternating sticking-contacts. This is not necessarily a limitation in the 2D case since the sliding condition can be included through a binary decision matrix that encodes the contact mode of each finger at every time-step [67]. The 3D case is more troubling, since the friction cone border constraints are non-convex and these would have to be approximated in some way [60].

Potential Extensions The first set of extensions to this work arise from its main limitations: incorporating sliding contact, as described above, and mitigating the dependence on an provided trajectory. Further validation in more complex 3D tasks [70] is also important. Since our model outputs trajectories with position and force, a natural extension would be to combine this model with a state-of-the-art feedback controller [17, 50, 61, 67, 112]. This is particularly relevant under recent advances on localized tactile sensing [49, 66, 33], which could allow us to generate more dynamic

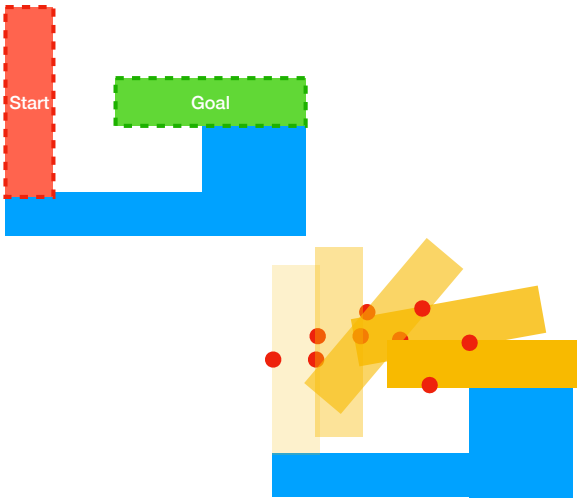


Figure 3-9: Given an object, an environment, a start and a goal, our framework finds a manipulation plan to complete the task.

motions under uncertainty.

3.2 A hierarchical framework to plan manipulation in long-horizons

In this section, we explore the problem of efficiently planning object trajectories with robot contact interaction to solve a manipulation task. We refer to this problem as **Object-Contact Trajectory planning**. In contrast to using pre-designed primitives, we several the model described in the previous section to design a hierarchical algorithm that reasons about the object motion and the robot contact interaction as part of the plan. Our goal is to incorporate multi-contact interaction between the object, the robot, and the environment, without sacrificing efficiency in long-horizon trajectories. Solving this problem has three key challenges: 1) dynamics, which determine the motion of the object, 2) geometry, which constrains the set of actions and configurations, 3) and non-smoothness, which is required to describe rich contact interactions.

Our approach is to decompose the problem into three stages, which can be evaluated hierarchically and solved efficiently: 1) A dynamic-programming search over

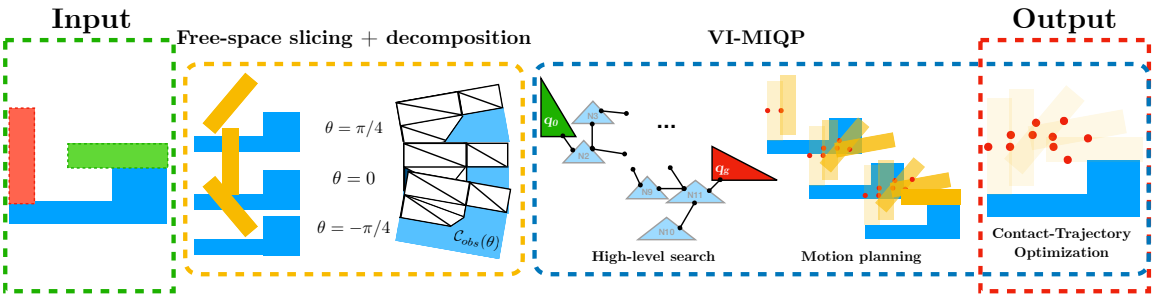


Figure 3-10: Our proposed framework receives an object with a start and goal pose. We discretize the free-space in slices and decompose them into polygonal regions. Our algorithm performs a high-level search over the graph of regions, building a roadmap. At each new edge of the roadmap, we sample an object trajectory that connects the two regions. For each object trajectory, we optimize a manipulator contact-trajectory that executes the motion.

regions of the object configuration space, 2) A sampling-based motion planner for the object trajectory between connected regions, and 3) A contact-trajectory optimization to find a manipulator contact-trajectory (finger trajectory + forces) to execute the sampled object trajectory. These three stages can be solved iteratively in a hierarchy with backtracking, where the result of each stage informs the previous one. We name this approach *VI-MIQP*, acronym for Value Iteration (VI) with Mixed Integer Quadratic Programming (MIQP). We restrict our implementation in this section to 2D scenes, over which it is simpler to reason about geometry. While this is a limitation, 2D manipulation problems encompass a wide range of skills that can be composed to solve practical tasks. The main contributions of this section are:

- **Framework** to solve manipulation tasks with alternating sticking contact interactions using dynamic programming, sampling-based planning, and mixed-integer optimization.
- **Validation** of this approach applied to manipulation tasks in challenging environments without a prescribed object motion, contact schedule or primitives.

The remainder of this section is organized as follows: Sec. II reviews concepts and literature relevant to this work. Sec. III provides an overview of our framework, its assumptions and properties. Sec. IV describes the proposed algorithm in detail, while Sec. V discusses its implementation. Sec. VI demonstrates the algorithm with

simulated experiments, and we conclude in Sec. VII summarizing the contributions and limitations of the work.

3.2.1 Problem Setup

In this subsection we provide an overview of our framework inputs and discuss its assumptions. Given a polygonal object, our algorithm will find a sequence of contact interactions that move it toward a goal pose.

Inputs and Notation Our algorithm receives as inputs the initial and final pose of the object and the geometries of the object and environment. We introduce the following notation and variables:

1. **Object:** A polygonal rigid body $\mathcal{O}$ with N_v vertices and N_f facets in a workspace $\mathcal{W}$. Each facet $\mathbb{F}_f$ has 2 vertices, with nominal positions $\mathbf{v}_v^f$, with a corresponding friction cone $\mathbb{FC}_f$, represented with 2 rays.

2. **Trajectory:** A set of object poses over discrete time steps, the length of the plan T is not known a-priori. We describe each pose, at a time step t, as $\mathbf{q}(t) \in \mathcal{C}$, where $\mathcal{C}$ is the configuration space of the object. $\mathcal{C}$ corresponds to the x, y, θ coordinates of $SE(2)$. The starting configuration is $\mathbf{q}_0$ and the goal configuration is $\mathbf{q}_g$.

3. **Manipulator:** A set of N_c contacts points. We describe the c_{th} contact-point, at time-step t, as $\mathbf{p}_c(t) \in \mathcal{W}$, where $\mathcal{W}$ is the workspace.The workspace corresponds to each position (x, y) that can be reached by the robot. We describe the force applied by the manipulator to the object, at time step t, as $\lambda_c(t) \in \mathbb{R}^2$.

4. **Environment:** A polygonal environment $\mathcal{E}$ with N_e facets, described as planes with friction cones $\mathbb{FC}_e$. Additionally, we segment the free-space between the object and environment into N_R convex polytopic regions $\mathcal{R}_r = \{\mathbf{x} \in \mathcal{W} \mid A_r\mathbf{x} < b_r\}$. We label the force between the environment and vertex v of the object as

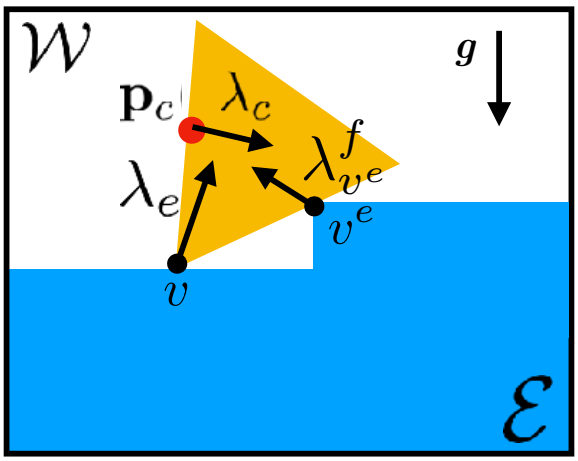


Figure 3-11: Elements of our problem: a polygonal object $\mathcal{O}$, a point-finger manipulator $\mathbf{p}_c$ and an environment $\mathcal{E}$. The contact between object and environment leads to a reaction force.

λ^e_v and the force applied by the vertex v^e of the environment into the object facet f as $\lambda^f_{v^e}$.

Then, we operate our planning on the following spaces:

1. **Configuration Space:** A set $\mathcal{C}$ of x, y, θ coordinates in $SE(2)$ where the object can move. The set of configurations where the object penetrates the environment is called C-obstacles or $\mathcal{C}^{obs}$ and is defined by the Minkowski sum between the object geometry (across all orientation of the configuration space) and the environment edges.

2. **Free-Space:** a set of configurations $\mathcal{C}^{free}$ where $\mathcal{O}$ does not penetrate the environment, defined as $\mathcal{C}^{free} = \mathcal{C} - \mathcal{C}^{obs}$, which is a subset of $SE(2)$. For practical purposes, we will discretize this space over the orientation component with a sample S $\mathcal{C}-$slices.

A diagram describing these elements, at a fixed time-step, can be seen in Fig. 3-11.

Modeling Assumptions Similar to last section. in order to build the manipulation planning problem, we make the following assumptions:

1. Objects are rigid, with uniform contact surfaces (such that line contacts can be approximated by contact with two vertices), and approximated as combinations of "simple" polygons.

2. Object motions occur at low speeds, such that high-order inertial effects are negligible.

3. Robot fingers have small masses, such that there are no impacts between the robot and the object.

4. Interactions between the object and robot are a sequence of alternating sticking contacts, at which friction is captured by a cone represented by two rays.

The upcoming sections will describe how these modeling decisions translate into our planning framework.

3.2.2 Hierarchical Framework

In this section, we present our hierarchical approach to long-horizon manipulation planning. Each paragraph describes a stage of the hierarchy and provides technical and implementation details. Our framework has three stages, which help alleviate some of the key issues of contact-rich manipulation:

1. **High-Level:** The first stage constructs a roadmap over a decomposition of the free-space into convex regions. We represent this roadmap as a graph G with edges E. We search over this roadmap to find a path from the start to goal configuration.

2. **Motion Level:** The second stage finds object trajectories that connect edges of the roadmap. These trajectories are concatenated while searching on the graph. If there is not a feasible trajectory corresponding to an edge, then we remove the edge from the roadmap.

3. **Contact Level:** This stage verifies that the motion-level trajectory can be executed with the manipulator. This problem is known as Contact-Trajectory

Optimization. If the optimization has no solution then we reject the object trajectory as infeasible.

These three stages are evaluated in a hierarchy where stage informs the previous one. Our approach leverages dynamic programming, sampling-based motion planning, and mixed-integer optimization to construct the full pipeline, illustrated in Fig. 3-10.

High-Level Search

An effective approach to guide a long-horizon planning problem is to build a high-level roadmap that outlines a coarse plan for a low-level finer planning stage. This outline ensures a certain level of optimality on the resulting trajectory. Without a roadmap, sampling-based planning tools, such as RRT, take a long time to solve long-horizons problems, and the resulting solution often has poor quality.

Roadmap construction To construct our roadmap we operate under the following decomposition of the free-space:

- Slicing: we discretize the free-space of the object, which lies in $SE(2)$, into slices of constant orientation. We refer to $\mathcal{C}^{free}$-slice of orientation θ as $C^{free}(\theta)$. Each slice is a 2D region of coordinates in $\mathbb{R}^2$. We showcase some examples in Fig. 3-12 (left).
- Segmentation: we further decompose each $\mathcal{C}^{free}$-slice into convex regions, as demonstrated in Fig. 3-12 (right).

We use each of the convex regions as a node in our graph G, where edges E are the overlapping side or area with regions in the same slice or in adjacents slices[1]. We label the node of the i_{th} region as N_i, where $E_{i,j}$ is the edge between N_i and N_j. This graph is the roadmap used to plan our high-level path. In practice, we compute $C^{free}(\theta)$ as the Minkowski sum $C^{free}(\theta) = \mathcal{E} \bigoplus R(\theta)\mathcal{O}$, where $R(\cdot)$ is a rotation matrix, and perform the convex decomposition using Delaunay triangulation.

[1]The separation between slices can cause two regions not to overlap, which can lead to path

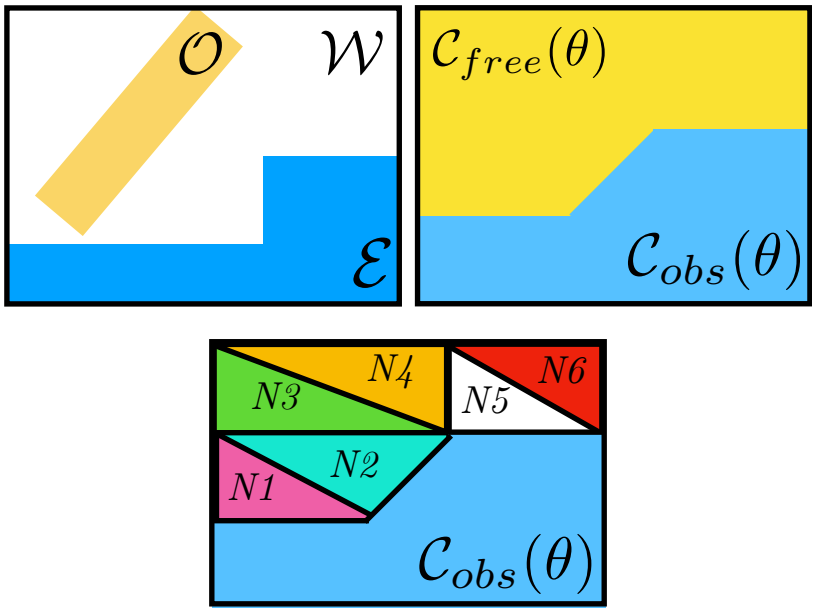


Figure 3-12: Free-space decomposition for high-level planning. Top: Workspace $\mathcal{W}$ and free-space slice at $C^{free}(\theta)$. Bottom: convex decomposition of the slice $C^{free}(\theta)$.

Dynamic programming Once we construct the roadmap G, E we need to search over the nodes to find the best path to the goal and determine the distance from each node to the goal. Since we need to find an object motion as we search through the graph, as part of the motion-level planning, it is more reasonable to pre-compute a value function for each edge $V(E_{i,j})$ using a heuristic. With this value function, we iteratively explore different paths in subsequent stages.

We start by finding start and goal nodes such that:

$$\mathbf{q}_0 \in N_0, \ \mathbf{q}_g \in N_g,$$

and set

$$V(E_{i,j}) = \infty, \ V(E_{i,g}) = d(i,g),$$

where $d(i,j)$ is the normalized $SE(2)$ distance between the center of N_i and N_j[2]. Finally, we determine the value of each node by running value iteration on the graph,

non-existence. Hence, this makes the resulting plan dependent on the resolution of the slicing

[2]We normalize this distance to avoid over-emphasis on the rotation component.

using the Bellman equation [21]:

$$V(E_{i,j}) = d(i,j) + \min_k V(E_{j,k}) \ , \ k \neq j$$

This relation is evaluated at each node iteratively until convergence. If $V(E_{0,k}) = \infty$ then there is no path from the start to the goal under the current slicing.

Forward pass Finally, after computing the value of each node, we proceed to explore the tree in a forward pass. We create a list of nodes $N = [N_0]$, including the start node, and a list of configurations $q = [\mathbf{q}_0]$, including the start configuration. We then proceed to explore the graph. We set $N_t = N_0$ and choose:

$$N_{t+1} = \operatorname*{argmin}_k V(E_{t,k})$$

Then, we verify with the motion-level plan if there is a trajectory from node N_t to N_{t+1}. We then follow the logic:

1. If the motion-level planner finds a feasible motion $\mathbf{q}_{new}$ and contact-trajectory for the entire motion $\mathbf{p}_c, \lambda_c$: we push the solution $q = [q, \mathbf{q}_{new}]$, $N = [N, N_{t+1}]$, and set $N_t = N_{t+1}$.

2. If the motion-level planner cannot find a trajectory, then we remove the edge from the graph by setting $V(E_{t,t+1}) = \infty$.

If all the edges of node N_t have $V(E_{t,t+1}) = \infty$, then we remove N_t from the list of nodes and pop the latest $\mathbf{q}_{new}$ from the list of configurations. We continue this process until $N_t = N_g$, at which point we run the motion-level planner to find a motion to the goal.

3.2.3 Motion-Level Planning

While running the forward pass through the optimal graph, at each edge with nodes N_t and N_{t+1}, we need to verify if there is a motion for the robot to move the object.

This requires finding a feasible contact-trajectory for the given path, which results in a non-smooth and nonlinear optimization problem, such as in [100]. This problem is hard to solve and usually requires a warm-start and depends heavily on numerical conditioning, which often makes it impractical.

In our case, we decouple the problem by randomly sampling object motions between N_t and N_{t+1}, since each node is associated with a convex polygon, using the function $Sample(\cdot)$, which randomly samples an $SE(2)$ configuration within a convex polygon. For each sample we solve an optimization problem $CTO()$ to find a contact-trajectory, discarding the sample if $CTO()$ is infeasible. We sample up to N_{MAX} trajectories before deciding an edge is not feasible. We summarize this procedure in algorithm 1.

Algorithm 1 RRT_{CTO}

Require: $\mathcal{O},\ \mathcal{E},\ N_t,\ N_{t+1},\ N_{MAX}$
  trial = 0
  while $trial < N_{MAX}$ **do**
    $\mathbf{q}_{new} = [\mathbf{q}_0, Sample(N_t), Sample(N_t \cap N_{t+1})$
    $\mathbf{p}_{new}, \lambda_{new} = CTO([\mathbf{q}, \mathbf{q}_{new}])$
    if Success **then**
      return $\mathbf{q}_{new}, \mathbf{p}_{new}, \lambda_{new}$
    else
      trial = trial + 1
    end if
  end while
  return Failure

Note that $CTO(\cdot)$ is called over the entire object trajectory, including previous nodes. We remark that the $Sample(\cdot)$ function needs to account for vertices and edges to have completeness, since each node is tied to a convex polygon. For this reason, we sample: 1) uniformly inside the polygon with probability p_1, 2) uniformly inside of a random polygon facet with probability p_2, and 3) in a random polygon vertex with probability $1 - p_1 - p_2$. This guarantees that the sampler will consider trajectories where the object traverses across facets or vertices of the node. We note that this is particularly important since environmental interaction will occur only at facets of vertices of each node. In practice, we always sample along a straight line during the

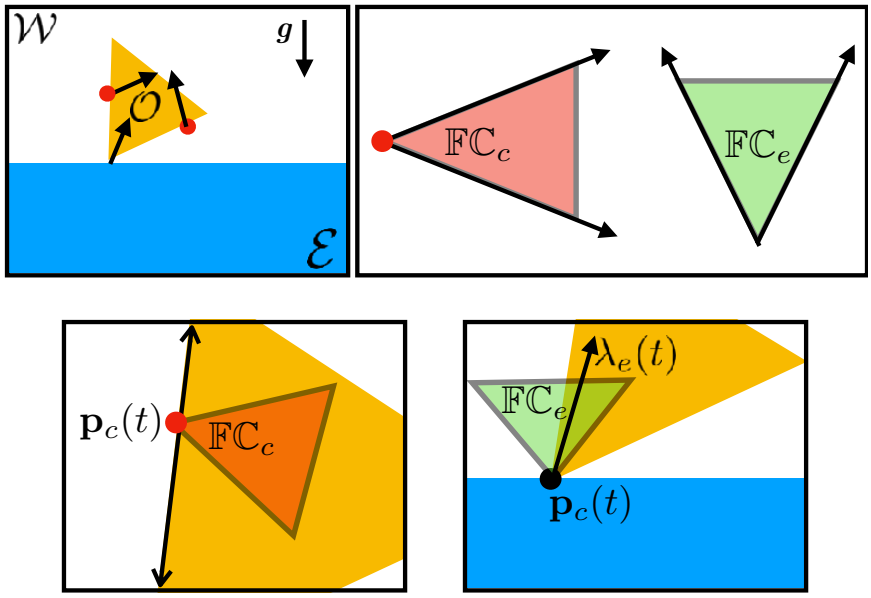


Figure 3-13: Once the object trajectory is sampled, we optimize a set of finger motions and contact forces that execute the trajectory. This problem involves the finger trajectories, applied forces, reaction forces, and the object facets and friction cones.

first trial, to bias towards a smooth motion, and sample freely in subsequent trials.

3.2.4 Contact-Trajectory Optimization

Once we sample an object trajectory, we need to verify that the manipulator can execute it. To do this, we need to solve an optimization problem that verifies that contact dynamics allow for such execution, under the components depicted in Fig. 3-13. One effective method to solve this problem is to apply Mixed-Integer Programming (MIP) once the object trajectory is prescribed, as in our case.

The mixed-integer programming formulation receives the object-trajectory as an input and returns the contact-trajectory of the manipulator. Note that once we sample a trajectory we also obtain the contact schedule of the environmental contacts, encoded by $\mathbb{FC}_e$, and the manipulator free-space regions $\mathcal{R}_r$. To achieve this, we apply the following modeling assumptions, based on 3.1.1, which models constraints using the following convex relations:

Quasi-Dynamics We model the linear object dynamics, at each time-step t, under the quasi-dynamic relation[3]:

$$m[\ddot{\mathbf{q}}_x(t), \ddot{\mathbf{q}}_y(t)]^T = \sum_c \lambda_c(t) + \sum_v \lambda_v^e(t) + \sum_{v^e} \lambda_{v^e}^f(t) - m\mathbf{g}, \tag{CT10}$$

where $\mathbf{g}$ is the gravity vector, note that this equation disregards Coriolis effects. Here, we compute time derivatives using second-order finite differences. For rotational dynamics, since torques require a non-convex bilinear cross product, we use the following approximate model:

$$I\ddot{\mathbf{q}}_\theta(t) = \sum_c \tau_c(t) + \sum_v R(\mathbf{q}_\theta(t))\mathbf{v}_v \times \lambda_v^e(t) + \sum_{v^e} \mathbf{v^e}_v \times \lambda_{v^e}^f(t), \tag{CT11}$$

where $\tau_c \approx (\mathbf{p}_c - [\mathbf{q}_x, \mathbf{q}_y]^T) \times \lambda^c$ and $R(\cdot)$ is a rotation matrix. We approximate the bilinear cross product ($\times$) in τ_c using the McCormick Envelopes technique [95]. McCormick Envelopes are a piecewise outer approximation of the bilinear product $w = x \cdot y$. Finally, we determine the value of environmental forces via the friction cone:

$$\lambda_v^e(t) \in \mathbb{FC}_e^v(t), \lambda_{v^e}^f(t) \in \mathbb{FC}_f(t), \tag{CT12}$$

where the value of $\mathbb{FC}_e(t)$ is determined once the trajectory is sampled.

Geometry We need to constrain the manipulator fingers to only lie in the free-space between the object and the environment. To achieve this, we introduce a binary decision matrix $\mathcal{H} \in \{0,1\}^{N_c, N_R, T}$ that maps each contact $\mathbf{p}_c(t)$ to a convex region $\mathcal{R}_r(t)$, at each time-step t. This is encoded by the mixed-integer linear constraint

$$\mathcal{H}(c, r, t) = 1 \Rightarrow \mathbf{p}_c(t) \in \mathcal{R}_r(t), \tag{CT13}$$

where the $\Rightarrow$ operator is encoded using the big-M formulation [92].

[3] we distinguish between $\lambda_v^e(t)$ and $\lambda_v^e(t)$ because environmental forces have pre-fixed friction cones while robot friction-cones need to be found by the model.

Contact mechanics We encode the hybrid mechanics of applied contact by introducing a binary decision matrix $\mathcal{T} \in \{0,1\}^{N_c,N_f,T}$. This matrix map each manipulator contact to a facet of the object and a friction cone as:

$$T(c,f,t) = 1 \Rightarrow \begin{cases} \mathbf{p}_c(t) \in \mathbb{F}_f(t), \\ \lambda_c(t) \in \mathbb{FC}_f(t), \end{cases} \tag{CT14}$$

and

$$\sum_f T(c,f,t) = 0 \Rightarrow \lambda_c(t) = 0, \tag{CT15}$$

which are all linear constraints with big-M formulation.

We aggregate these sets of constraints into a single optimization problem. This results in the following mixed-integer optimization problem:

$$\textbf{CTO} : \underset{\mathcal{T},\mathcal{H},\mathbf{p},\lambda}{\text{minimize}}\ J = \begin{bmatrix} \mathcal{T} & \mathcal{H} & \mathbf{p} & \lambda \end{bmatrix} Q \begin{bmatrix} \mathcal{T} \\ \mathcal{H} \\ \mathbf{p} \\ \lambda \end{bmatrix} + q^T \begin{bmatrix} \mathcal{T} \\ \mathcal{H} \\ \mathbf{p} \\ \lambda \end{bmatrix}$$

subject to:

1. For time-step $t = 1$ to $t = T$:
   (a) Quasi-Dynamics (CT1)-(CT2).
   (b) Environmental Contact (CT3).
   (c) For fingers $c = 1$ to $c = N_c$
      - Non-Penetration (CT4).
      - Contact-Trajectory Assignment (CT5)-(CT6).

This optimization problem has the following properties: 1) Given a convex cost function, we will always find the optimal solution, and 2) The model only reports infeasibility if the original non-convex problem is also infeasible, thanks to the McCormick envelope approximation used in (CT2). This is comes with the drawback

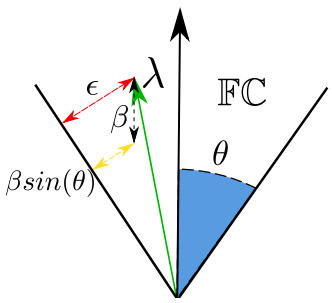


Figure 3-14: Contact robustness margin: we compute a convex inner approximation of the stability margin of our motion.

of combinatorial complexity, which grows exponentially with the number of binary variables. However, we find that this optimization problem can be solved very quickly in practice, on the order of tens of milliseconds.

Moreover, this formulation also allows for the inclusion of additional constraints, such as kinematics or robustness margins [35], provided that these can be formulated in a mixed-integer linear fashion. In the final $CTO(\cdot)$ call of the algorithm, when reaching the goal pose, we add a quadratic cost function that minimizes the applied force, smooths the finger trajectories, and maximizes contact robustness:

$$J = \sum_{t=1}^{T} \sum_{c=1}^{N_c} ||\boldsymbol{\lambda}_c(t)||^2 + ||\ddot{\mathbf{p}}_c(t)||^2 - \beta_c(t)$$

The term β is a lower-bound convex-approximation of the distance between each contact-force and the border of its friction cone, computed as in [3, 66], this term has to be linear in order for the cost-function to be convex. We depict this β term in Fig. 3-14.

3.2.5 VI-MIQP

Putting all the previous stages together, we formulate our planning framework under the stages outlined in the previous section. This algorithm receives an object shape, initial pose, goal pose, and free-space $\mathcal{C}^{free}$ sliced over orientations. We summarize our proposed framework in algorithm 2.

This formulation has a few benefits and theoretical properties:

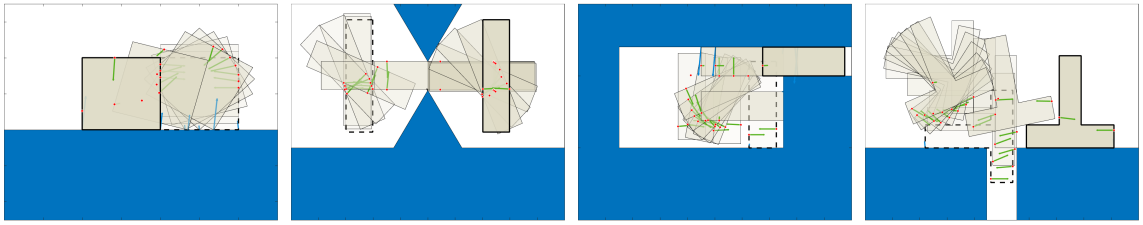
Algorithm 2 VI-MIQP

Require: $\mathcal{O},\ \mathcal{E},\ \mathbf{q}_0,\ \mathbf{q}_g$
  $G, E\ =\ \text{Delaunay}(\mathcal{C}^{free})$ ▷ Roadmap construction
  $V(E_{i,j}) = \infty,\ \forall\ i, j$
  $V(E_{i,g}) = 0,\ \forall\ i$
  while V not converged **do**
    $V(E_{i,j}) = d(i,j) + \min_k V(E_{j,k})$ ▷ Value iteration
  end while
  $N_t = N_0$ ▷ Starting node
  $N = [N_t]$ ▷ Starting list
  while $N_t \neq N_g$ **do** ▷ Forward pass
    for $N_{t+1} = \text{argmin}\ V(E_{t,t+1})$ **do**
      $\mathbf{q}_{new}, \mathbf{p}_{new}, \lambda_{new} = RRT_{CTO}(q, N_t, N_{t+1})$
      if Success **then**
        $\mathbf{q} = \text{push}(\mathbf{q}, \mathbf{q}_{new}),\ \mathbf{p} = \mathbf{p}_{new},\ \lambda = \lambda_{new}$
        $N = [N, N_{t+1}]$ ▷ Adds edge to plan
        $N_t = N_{t+1}$
        break **for**
      else
        $V(E_{t,t+1}) = \infty$ ▷ Remove edge
      end if
      if $V(E_{t,t+1}) = \infty\ \forall N_{t+1} \in Neigh(N_t)$ **then**
        $N_t = N_{t-1}$ ▷ Removes node
        $\text{pop}(q),\ \text{pop}(N)$
      end if
    end for
  end while

- First, the dynamic programming over a roadmap yields an optimal solution to the high-level plan, conditioned on the resolution of the slicing of the free-space and in the limit of sampling of the motion planner.

- Thanks to the mixed-integer optimization formulation, we can guarantee that the contact-trajectory found by $CTO(\cdot)$ is **globally optimal**, conditioned to the object trajectory found in the motion-level.

These properties are very useful to understand the feasibility of a task and the quality of the solution.



(a) Pivoting a block and sliding **(b)** Moving in a narrow corridor **(c)** Peg-in-wall for a rectangle **(d)** Unpegging a T from a hole

Figure 3-15: Long-horizon manipulation tasks solved with our algorithm. Green dots correspond to the point-fingers of our manipulator. Doted object contour corresponds to the initial pose $\mathbf{q}_0$ and bold object contour corresponds to goal pose $\mathbf{q}_g$.

3.2.6 Validation and Results

To demonstrate the capabilities of our framework, we implement algorithm 2 and asses its application to a set of long-horizon manipulation problems. First, we aim to validate the model's ability to find solutions to complex manipulation tasks and detect infeasible ones. We then show how the algorithm performance scales over problems with varying complexity.

We generate all the trajectories in MATLAB R2021b, running on an Intel Core i9 laptop with Mac OS X Big Sur. We use Gurobi 9.1.0 [59], an off-the-shelf optimization software, as our MIP solver. All of our tests are done with a $\mathcal{C}^{free}$ sampled in 7 slices between $-90°$ and $90°$[4]. We use piecewise McCormick envelopes of 12 segments. We segment the free space of each task into convex regions $\mathcal{R}$ using Delaunay triangulation. We compute second derivatives within the mixed-integer model with the backwards-Euler scheme for simplicity and numerical stability.

Simulated Validation We start by validating the functionality of the algorithm in several manipulation problems. In particular, we show its ability to solve problems that demand long horizon and multi-contact reasoning. These tasks require alternating contact switching with the robot and the environment. All the tasks are solved with a 2-finger manipulator with kinematic constraints based on an ABB YuMi robot,

[4]This choice is made to leave a gap of $30°$ between slices, which is enough to find paths in the shapes used. Environments with narrower paths or more non-convex shapes may require further slicing.

encoded in $CTO(\cdot)$. For this, we use three objects: 1) a block, 2) a rectangle, a 3) a non-convex T object. The contacts between all surfaces have a friction coefficient of $\mu = 0.1$. We solve the following manipulation problems:

Block Pivoting: we ask the planner to rotate a block $-90°$ in the sagittal plane and slide it $200mm$ in the $-X$ direction (Fig. 3-15a). Due to the low friction with the surface, our algorithm finds a trajectory that: 1) grasps the block with two fingers, slowly rotating in the process, 2) pivots the object with one finger, while pushing it to the goal, and 3) uses the two fingers to immobilize in the final goal pose. This trajectory demonstrates the contact-rich nature of our approach, since the solution to a problem can choose to grasp an object and then proceed to use a single finger to complete the task. This trajectory is found in 0.2s.

Rectangle across Narrow Corridor: we ask the planner to grasp a rectangle and move to the other side of a region with a very narrow corridor in the moddle (Fig. 3-15b). The algorithm performs a strategy of: 1) grasping the object, 2) rotating the object $-90°$, 3) sliding the object through the corridor with one finger, 4) grasp the object the two fingers, and 5) pivot it back for $0°$. This trajectory demonstrates the long-horizon reasoning that our planner can achieve, considering contact switches and global path. This type of trajectory would likely require a very large amount of exploration from randomized sampling-based planner. This trajectory is found in 7.2s.

Rectangle Peg-in-wall: we ask the planner to pick a rectangle from the ground and insert it into a tight hole in a wall, rotated by $-90°$ (Fig. 3-15c). The algorithm performs a strategy of: 1) grasping the object, rotating it while grasped, 2) pushing it to the wall with one finger, 3) sliding it up to the "ceilng", and 4) pushing it with the two fingers, against the ceiling contact, to insert the object in the wall. This trajectory demonstrates multi-contact interaction with the robot and the environment. Here, the wall and ceiling contacts are used to enable more stability and to make the trajectory geometrically feasible. This trajectory is found in 2.5s.

T Unpeg: we ask the planner to pick a T object from a hole and rotate it by $90°$ into the floor (Fig. 3-15d). The algorithm finds a strategy of: 1) grasping the object

from the hole, 2) pivoting it while grasped, 3) switching the grasp configuration, and 4) placing the T in the goal location. This tasks demonstrates the ability of our algorithm to reason over non-convex geometry in the objects and environment. This trajectory is found in 6.3s.

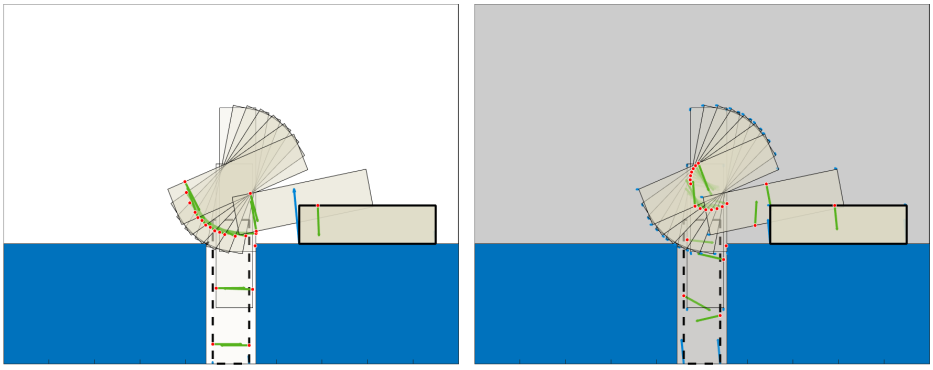
As a reference, all trajectories are optimized in the range of 0.2 s to 7.2 s. We stress the ability of $CTO(\cdot)$ to report when a task is infeasible, either because it requires an additional contact force or because it is geometrically infeasible, which can help the planner quickly discard edges and find a path. We note that all these tasks are performed in the sagittal plane.

Application to Sagittal and Traversal tasks A key question is how this algorithm performs across the sagittal –XZ– plane and one problem in the traversal –XY– plane. The traversal plane adds the presence of Coulomb friction forces in the object surface, following the maximum discipation principle. We can easily model this patch contact in our $CTO(\cdot)$ problem by transforming the friction cone at each object vertex to the vector:

$$\mathbb{FC}_e^v(t) = \frac{-\mu_e}{\sqrt{\dot{v}_x^2(t) + \dot{v}_x^2(t)}}[\dot{v}_x(t), \dot{v}_y(t)],$$

where $v(t) = [v_x(t), v_y(t)]$ is the position of the object vertex v at time-step t. We then contrast how our model performs in both traversal and sagittal scenarios with the following tasks:

Saggital Unpeg: we ask the planner to pick a rectangle from a hole and rotate it by 90° into the floor (Fig. 3-16a). The algorithm finds a strategy of: 1) pushing the object to one side of the hole to create some clearance, grasping it up from the hole, 2) pivoting it with respect to one of the vertices of the hole, 3) grasping it outside of the hole, and 4) dropping it into the goal location in the floor. Similar to before, this trajectory demonstrates multi-contact interaction and dynamics with the robot and the environment. Here, the planner effectively finds a space clearance to pivot the object against the environment and then use gravity to drop it to the goal. This trajectory is found in 3.2s.



(a) Sagittal unpeg-from-hole **(b)** Traversal unpeg-from-hole

Figure 3-16: Comparison between the same task solved in a sagittal and traversal environment

Traversal Unpeg: we ask the planne to pick a rectangle from a hole and rotate it by 90° into the floor (Fig. 3-16b). The algorithm finds a similar strategy of: 1) pushing the object to one side of the hole to create some clearance, grasping it up from the hole, 2) pivoting it with respect to one of the vertices of the hole, 3) grasping it outside of the hole, and 4) pushing it into the goal location in the floor. This trajectory shows how the presence of traversal contact, dictated by the maximum dissipation principle, forces the planner to find a different contact schedule for the trajectory. This trajectory is found in 9.6s.

These trajectories are optimized in the range of 3.2 s to 9.6 s. This demonstrates the versatility of our model when accounting for new constraints and dynamic effects.

Complexity Analysis One natural question is understanding the computational complexity of this algorithm and how efficient it can be at solving problems of different scale. Assessing the speed of a mixed-integer optimization problem can be very challenging, since off-the-shelf tools will use different techniques to solve the problem. Nevertheless, we can analyze other metrics that relate to the complexity of the problem, such as number of nodes of explored N_N, $CTO(\cdot)$ calls N_{MIP}, and value iteration steps N_{VI}.

We come back to the task of "rectangle across a corridor" and sample several intermediary goals. We measure the $SE(2)$ distance between $\mathbf{q}_s$ and each of these

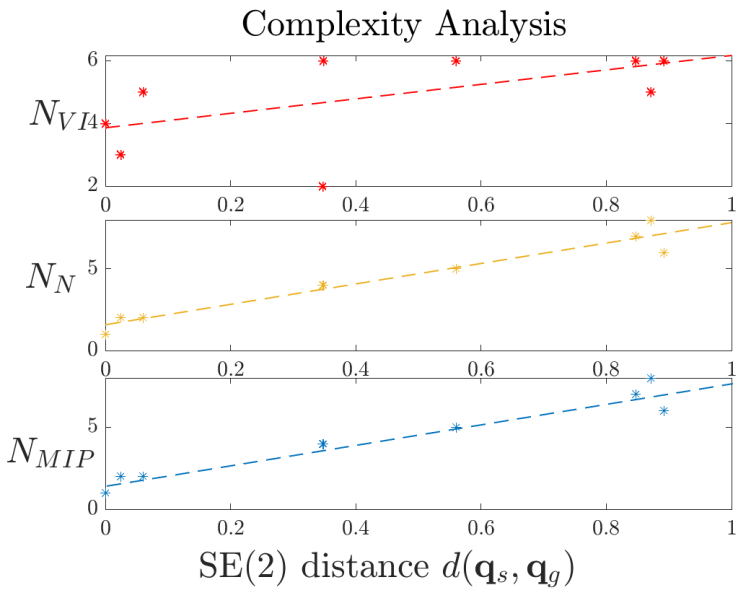


Figure 3-17: Complexity analysis for our model

goals and then run our algorithm, recording: number of value iteration steps, number of nodes explored, and number of MIP calls. We report the results in Fig. 3-17. We find the complexity our framework to grow approximately linearly with the distance to the goal. This comes with the caveat that MIP calls can often take up to a second, becoming the main source of delays. However, this also suggests that our approach can scale well to very long-horizon tasks.

Source Code The entire source code used as part of this work is publicly available on GitHub: https://github.com/baceituno

Chapter 4

Differentiable Learning of Manipulation Tasks from Video

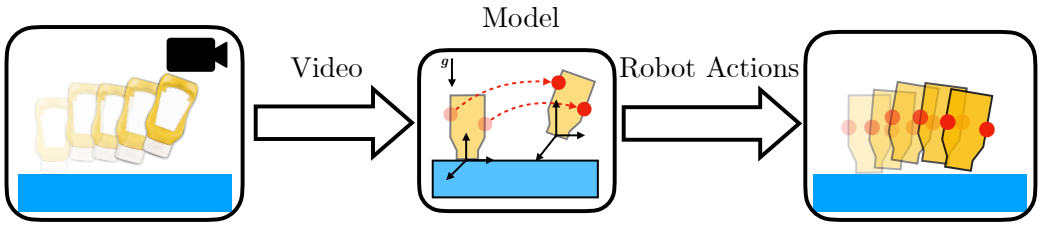


Figure 4-1: We tackle the problem of visual non-prehensile planar manipulation where given a pre-segmented video of an object in planar motion (left) the goal is to find the robot actions (middle) to reproduce the same object motion (right). In learning for manipulation, we study the role of structure, intermediate representations, and end-to-end differentiation.

Can bringing mechanics representations and priors into deep learning models aid in decoding videos and learning dexterous manipulation? This chapter studies this question as part of the problem of video-based non-prehensile planar manipulation. Here, given a video of an object in planar motion in some environment (i.e. input video), the goal is to find contact and force trajectories for the robot fingers (i.e. output robot actions) that when executed will reproduce the same object motion in the same environment. This problem is illustrated in Fig. 4-1. Our approach is to build intermediate representations based on mechanics that serve as a glue in combining upstream neural models that decode the video with downstream differentiable priors

that solve for robot actions. We limit the problem scope to planar settings, with the input video being pre-segmented, and the robot having point-fingers (standard in robot manipulation literature [140, 112, 52, 4, 32]). These simplifications allow us to assess the impact of structured learning without the confounding effects of sensor or actuator noise, and enable us to make a reasonable first step towards general approaches that would be able to learn dexterous manipulation by watching large scale human videos [39].

We present an architecture, Differentiable Learning for manipulation (DLM) (illustrated in Fig. 4-2), that works in three stages and can be progressively trained on each stage: (1) a neural model that derenders the input video to mechanical parameters i.e. object shape, object trajectory, discrete contact-mode decisions, and allowable forces between robot fingers and object, (2) a differentiable convex optimization module based on cvxpylayers [6, 45] that solves inverse dynamics from the mechanical parameters to find contact and force trajectories for the robot fingers, and finally (3) a differentiable contact-mechanics simulation module based on lcp-physics [41] and finite-differencing that executes the robot actions to simulate the object motion. We train the first two stages with supervision from an expert contact trajectory optimizer [4] while the third stage is trained end-to-end over the desired object trajectory.

To investigate the role structure plays in learning, we compare our approach with various architectures including neural models without any mechanics priors on visual non-prehensile planar manipulation tasks. In our experiments, we train the third stage of DLM on a dataset that is smaller relative to the training dataset used for the first two stages to resemble a more realistic setting where pre-training is done on a large simulation dataset with easy to obtain supervision and fine-tuning is done on a smaller real-world dataset that is expensive to collect and harder to label. We find that our structured model is able to outperform other learning-based architectures on unseen objects and motions, while being more computationally efficient compared to the non-learning expert.

4.1 Visual Non-Prehensile Manipulation Inference

In this section, we define our problem of inferring manipulation actions from a video which serves as a task specification. We limit our focus to the context of 2D non-prehensile manipulation given 2D pre-segmented videos of desired object motion. In the discussion section, we elaborate on possible avenues to extending our formulation and approach to 3D, real videos, and real robots. We start by defining the main components of our problem, its assumptions, and our notation:

1. **Task:** a video $\mathcal{V}$ with T frames showing a sequence of object poses. The tasks involves a set of object poses $\mathbf{r}(t) \in SE(2)$, sampled in T time-steps t.

2. **Object:** a polygonal rigid-body $\mathcal{O}$ with mass matrix $\mathbf{M}$ and N_F facets. Each facet $\mathbb{F}_f$ has a corresponding friction cone $\mathcal{FC}_f$.

3. **Action Space:** a set of N point-fingers[1] moving freely around space. We describe the position of finger c, at time-step t, as $\mathbf{p}_c(t)$. The contact interaction between the object and the finger at $\mathbf{p}_c(t)$ applies a force $\lambda_c(t)$.

4. **Environment:** a polygonal environment described as planes with friction cones $\mathcal{FC}_e$. The contact interaction between the object and the environment occurs at the contact point $\mathbf{p}_e(t)$ and results in a reaction force $\lambda_e(t)$.

We show a high-level representation of our problem in Fig. 4-1. One of the challenges in solving this problem come from interpreting video data, in order to extract an implicit representation of the object shape, the object motion, and finding the appropriate contact interactions between the robot fingers and the object. Specifically, finding a contact interaction is divided in two steps: (i) select a sequence of *contact modes*, indicating where each contact is applied at each time-step, and (ii) *invert the dynamics* to resolve the exact contact locations and forces to apply. In this context, inverse dynamics (ID) is an optimization problem that receives an object motion and

[1]The use of point-fingers does not account for robot kinematics. While this is a limitation, it is a standard approach to evaluate planning and learning pipelines for manipulation problems (see [140, 71, 112, 2, 4, 32]).

outputs forces and locations, formulated in the form:

$$\textbf{ID:} \qquad \min_{\mathbf{p},\mathbf{\Lambda}} \quad \sum_{t=0}^{T} J^{ID}_{p,\lambda}(t) \tag{CT1}$$

$$\text{subject to:} \quad \mathbf{M}\ddot{\mathbf{r}}(t)+\mathbf{G}(\mathbf{r}) = J(\mathbf{r})^T\Lambda(t), \quad \mathbf{p}_c(t) \in \mathbb{F}_c(t), \quad \lambda_c(t) \in \mathcal{FC}_c(t), \quad \lambda_e(t) \in \mathcal{FC}_e(t) \tag{CT2}$$

where $J^{ID}_{p,\lambda}(t)$ is a cost function that typically penalizes actuation efforts, $\mathbf{G}(r)$ represents gravity and inertial effects, and $J(r)$ is a Jacobian mapping contact forces $\Lambda = [\lambda_1, \ldots, \lambda_N, \lambda_e]$ into wrenches. Solving **ID**, however, assumes knowledge of the exact shape of the object and the contact modes to be applied [53, 30, 69], encoded in the facets $\mathbb{F}_c(t)$ and friction cones $\mathcal{FC}_c(t)$. Solving this problem without assuming known contact modes is often intractable and a significant body of research has focused on studying it [100, 42, 4]. Our approach addresses this by extracting object shape and contact modes (as facets and friction cones) as parameters through a deep neural model, which can be trained with ground-truth labeled data and augmented with self-supervision through differentiable optimization and differentiable simulation.

4.2 Differentiable Visual Non-Prehensile Manipulation

In this section, we present our approach Differentiable Learning for Manipulation (DLM). Each subsection describes a stage of the model and provide technical and implementation details. Our approach leverages deep neural networks, differentiable optimization, and simulation, to construct the full pipeline, illustrated in Fig. 4-2.

Mechanical Derendering One challenge of visual manipulation is to decode geometry and motion from video. While pre-segmented videos can be obtained from realistic videos with state-of-the-art computer vision tools [65], it is challenging to translate explicit task information into an implicit latent space. This is due to discrete variables such as facets, vertices, and intersections. Hence, the first stage of

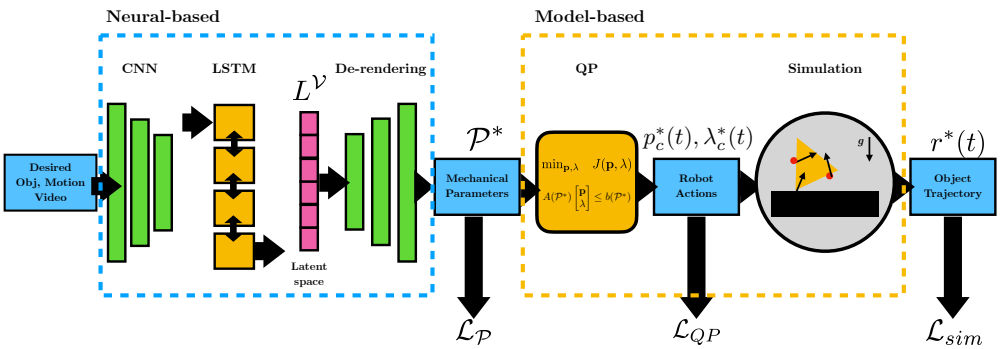


Figure 4-2: Our proposed fully differentiable pipeline, DLM (Differentiable Learning for Manipulation). In contrast to approaches that represent policies as feed-forward neural networks, our approach leverages model-based priors from mechanics to process mechanical parameters via a differentiable quadratic program and evaluate the resulting policy through a differentiable simulator.

our model encodes the video frames $\mathcal{V}$ into an implicit latent space $\mathcal{L}^{\mathcal{V}}$. To achieve this, we first encode each frame I_k through a set of convolution layers $CNN(\cdot)$ and combine these embeddings with a Long Short-Term Memory (LSTM) layer, to retain their temporal relation, as:

$$L^{\mathcal{V}} = LSTM(CNN(I_1), CNN(I_2), \ldots, CNN(I_T)) \tag{CT3}$$

Given this encoding, we find all the parameters required for **ID** with differentiable optimization, in similar fashion to [73]. This set of mechanical parameters [4] includes: (i) the Jacobian matrix $J(\mathbf{r})(t)$, (ii) the location of external contacts $\mathbf{p}_e(t)$, and (iii) parameters that implicitly encode contact modes $\mathbb{F}_c$, $\mathcal{FC}_c$, and $\mathcal{FC}_e$. We derender this set of mechanical parameters $\mathcal{P} = \{\mathbf{r}(t), J(\mathbf{r})(t), \mathbb{F}_c(t), \mathcal{FC}_c(t), \mathbf{p}_e(t), \mathcal{FC}_e(t)\}$ through an MLP[2] and the loss function:

$$\mathcal{P}^*(t) \ = MLP^{\mathbf{P}}(L^{\mathcal{V}})(t), \ \forall t, c, \tag{CT4}$$

$$\mathcal{L}_{\mathcal{P}} = ||\mathcal{P}^*(t) - \mathcal{P}(t)||_2^2 \tag{CT5}$$

As illustrated in Fig. 4-3 these parameters are an implicit representation of the finger

[2]Facets $\mathbb{F}$ are represented by two vertices in the world frame, while friction cones $\mathcal{FC}$ are represented with two rays originating at the contact point.

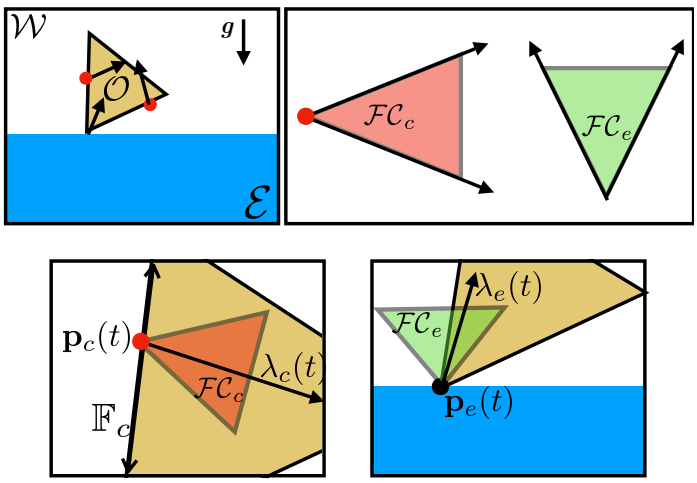


Figure 4-3: Interactions in our setup (top-left) and the mechanical parameters from the scene (bottom). Different contact modes are encoded implicitly via the friction cones and the object facet at each contact (top-right). Note that the input video only demonstrates the object moving without any robot actions.

contact modes and the local object geometry for a given task, since the contact mode only encodes which facet is in contact and what is the friction cone.

Differentiable Inverse Dynamics Given the derendered parameters, we recast **ID** as the following quadratic program which solves for robot actions $p_c(t), \lambda_c^*(t)$:

$$\textbf{QP:} \quad \min_{\mathbf{p}_c, \lambda, \epsilon} \quad J_{QP} = \sum_{t=0}^{T} ||\Lambda_c(t)||_2^{Q_\lambda} + ||\ddot{\mathbf{p}}_c(t)||_2^{Q_p} + q|\epsilon(t)|$$

subject to:

$$\mathbf{M}\ddot{\mathbf{r}}^*(t) = J(\mathbf{p}, \mathbf{r})^{*^T} \Lambda(t) - \hat{\mathbf{G}}(\mathbf{r}^*) + \epsilon(t), \;\; \lambda_c(t) \in \mathcal{FC}_c^*(t), \;\; \lambda_e(t) \in \mathcal{FC}_e^*(t), \;\; \mathbf{p}_c(t) \in \mathbb{F}_c^*(t) \tag{CT6}$$

Under this linearization, **QP** has the property of being a convex optimization problem [24]. This has a few benefits: (i) its solution is always the global optima, (ii) if $\mathcal{P}^*(t) = \mathcal{P}(t)$ then its solution is guaranteed to match the ground-truth optima, and (iii) **QP** can be added as a differentiable layer of our model [14, 6], which inputs $\mathcal{P}^*$ and outputs the optimal $\mathbf{p}_c^*, \lambda_c^*, \epsilon$ for such parameters. There are a few considerations to make **QP** a differentiable layer. In particular, **QP** must have a solution–or be feasible– for any choice of parameters $\mathcal{P}^*(t)$ [6]. To make the problem always feasible,

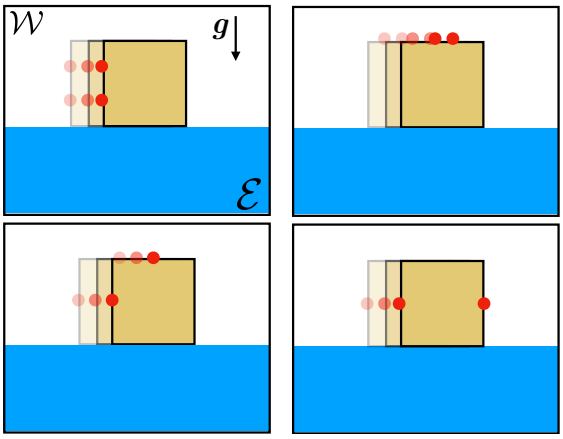


Figure 4-4: Example of four different valid solutions to a simple manipulation task of pushing a block from left to the right. Since the video does not show what actions to follow, our model may output any of these actions depending on its supervision.

we relax the dynamics constraints by adding a slackness term $\epsilon(t)$ and penalize it in the loss function J_{QP}, with weight q. After solving **QP**, we supervise its solution with ground-truth labels, under known parameters, with the loss

$$\mathcal{L}_{QP} = q|\epsilon| + ||\mathbf{p}_c^* - \mathbf{p}_c||_2^2 + ||\lambda_c^* - \lambda_c||_2^2, \tag{CT7}$$

which drives the model towards a set of actions that are both consistent with the physical structure, such that $\epsilon(t) \to 0$, the ground-truth physical parameters $\mathbf{p}_c^*, \lambda_c^* \to \mathbf{p}_c, \lambda_c$. After training, we set $\epsilon = 0$ to ensure consistency.

Evaluation A drawback of learning directly from the action space is the dependence on *ground-truth* finger trajectories to assess the generality of the learned model. The existence of multiple solutions, e.g. as depicted in Fig. 4-4, makes it ambiguous to assess a model with unseen labeled data. To resolve this ambiguity in the evaluation, we simulate the task using inferred actions $p_c^*(t), \lambda_c^*(t)$. We then compare the simulated object trajectory with the desired object trajectory, and use this error as

a metric [19]. Algebraically, a simulator is represented by

$$\mathbf{r}^*(t) = Sim(\mathbf{p}_c^*(t)) \begin{cases} \mathbf{r}(t+1) = \mathbf{r}(t) + \Delta T \dot{\mathbf{r}}(t), \\ \dot{\mathbf{r}}(t+1) = \dot{\mathbf{r}}(t) + \Delta T \ddot{\mathbf{r}}(t), \\ \mathbf{M}\ddot{\mathbf{r}}(t) + \mathbf{G}(\mathbf{r}) = J(\mathbf{r})^T \mathbf{\Lambda}^f(t) \end{cases} \tag{CT8}$$

and:

$$\mathbf{\Lambda}^f = [\lambda_1^f(\mathbf{p}_1^*(t)), \ldots, \lambda_N^f(\mathbf{p}_N^*(t)), \lambda_e^f(\mathbf{p}_e^*(t))]$$

is a functional representation of the contact forces. These forces need to be determined based on the inertial properties of the system. To resolve these forces we solve a Linear Complementary Problem (LCP), based on [41], which outputs the corresponding contact forces between the object, fingers, and the environment. These contact forces are activated discretely as:

$$\lambda_k^f((\mathbf{p}_k^*(t))) = \delta(D(\mathbf{p}_k^*(t), \mathbf{r}(t))), \quad D(\mathbf{p}_k^*(t), \mathbf{r}(t)) \geq 0, \delta(x) = \begin{cases} 1, x = 0 \\ 0, x > 0 \end{cases} \tag{CT9}$$

where $D(\mathbf{p}_k^*(t), \mathbf{r}(t))$ is the distance between point $\mathbf{p}_k^*(t)$ and the object at pose $\mathbf{r}(t)$. Similar relations are used to represent the friction cones and contact modes, which we omit for simplicity. Then, we use the simulation for the evaluation metric using the loss function

$$\mathcal{L}_{sim} = ||Sim(\mathbf{p}_c^*(t)) - \mathbf{r}(t)||_2 \tag{CT10}$$

where $\mathbf{p}_c^*(t)$ are finger trajectories obtained by solving **QP**. It is important to note that $\mathcal{L}_{sim}$ is not an injective mapping from the finger trajectory error $||\mathbf{p}_c^*(t) - \mathbf{p}_c(t)||_2$. Therefore, having actions near the ground-truth might not necessarily lead to a lower $\mathcal{L}_{sim}$.

Supervision via Simulation Having access to the loss function $\mathcal{L}_{sim}$ can also tune our model subject to ground-truth physics. This reduces (or potentially removes) the burden of having labeled data over the parameters $\mathcal{P}(t)$ and the finger-trajectories

$\mathbf{p}_c(t)$, leading to a fully self-supervised approach. Training our model with $\mathcal{L}_{sim}$ requires the function $Sim(\cdot)$ (i.e. the simulator) to be differentiable in all its domain. However, the function $\delta(\cdot)$ only returns gradients at the origin. While this is a pervasive problem, many researchers have successfully included contact mechanics as differentiable functions by approximating $\delta(\cdot)$ with a smooth function [100]. In our case, we approximate $\delta(\cdot)$ as the smooth function:

$$\delta(D(\mathbf{p}_k(t)) \approx \sigma(-D(\mathbf{p}_k(t), \mathbf{r}(t)), \kappa)$$

where $\sigma(\cdot, \kappa)$ is a sigmoidal function with factor κ. This approximation can be made arbitrarily tight as $\kappa \to \infty$. With this smoothing, the simulator gradients are computed via finite differences as:

$$\nabla \mathbf{r}^*(t) \approx (Sim(\mathbf{p}_k^*(t) + h_p) - Sim(\mathbf{p}_k^*(t) - h_p))/2h$$

where $h_p = h\mathbf{I}_{3\times 3}$. While automatic differentiation is a more efficient approach to obtain gradients, we found it incompatible with the smoothed contact model of our current implementation. We leave this enhancement for future work. Minimizing $\mathcal{L}_{sim}$ drives the learned finger trajectories to push the object through the desired object trajectory. A consequence of this approximation is that $\delta(\cdot) \approx \sigma(\cdot)$ leads to effects such as contact at distance and small penetration, although these are attenuated for a large $\kappa >> 1$ in practice.

4.3 Experiments

We implement all models in PyTorch [96] with Python 3. To generate our datasets, we use MATLAB R2020b and solve optimization problems with the Gurobi [59] solver. For all experiments, we use Adam as our network optimizer and CVXPY layers [45, 7] as differentiable solver for **QP**. In all experiments we train over each loss function for 100 iterations with a learning rate of 10^{-4}. We implement our simulator on top of the LCP-based scheme in [41] by modifying it to accommodate our problem setting.

We run all training, testing, and timing experiments on a Macbook Pro with 2.9 GHz 6-Core Intel CPU.

Datasets: We built four datasets of non-prehensile manipulation tasks in the *sagittal*[3] plane with randomized trajectories in $SE(2)$, all ground truth robot actions are found by solving Contact-Trajectory Optimization (CTO) [4]:

- *Training set* of 20 tasks with randomly generated polygonal objects of 4 to 6 facets, including all mechanical parameters and robot actions for each task.

- *Fine-Tuning set* of 20 tasks with new random objects of 4 to 6 facets, with robot actions for each task.

- *Test set* of 40 tasks with new random objects of 12 facets, with robot actions for each task.

- *Family Test set* of 40 tasks with new random objects of 4-6 facets (same shape and task distribution as training and fine-tuning sets), with robot actions for each task.

Each task has a video of $T = 5$ RGB frames of size 50×50. We find optimal finger trajectories for each task using a Mixed-Integer Quadratic Program (MIQP) [4], for $N = 2$ point fingers. We set $\kappa = 0.5$, $q_\epsilon = 0.1$ and $dim(\mathcal{L}^{\mathcal{V}}) = 75$. We illustrate example objects we generate for each dataset in Fig. 4-5 (left). Also see Appendix for more details on data generation.

Architectures: We asses the performance of our approach, Differentiable Learning for Manipulation (DLM) against four different architectures and a model-based oracle (see Fig. 4-5).

- NN: The Neural Network (NN) architecture replaces **QP** in DLM, with a 3-layer MLP. This is akin to a pixel-to-actions style policy network [55, 15] and is trained to generate MIQP solutions with a loss function $\mathcal{L}_{NN} = ||\mathbf{p}_c^*(t) - \mathbf{p}_c(t)||_2^2$.

- NNM: The Neural Network Manipulation (NNM) architecture extends NN by also fine-tuning by minimizing $\mathcal{L}_{sim}$, akin to the examples used in [41]. This network is pre-trained with 100 iterations of minimizing $\mathcal{L}_{\mathcal{NN}}$.

[3] side-view of the scene with gravity pointing down in the image plane.

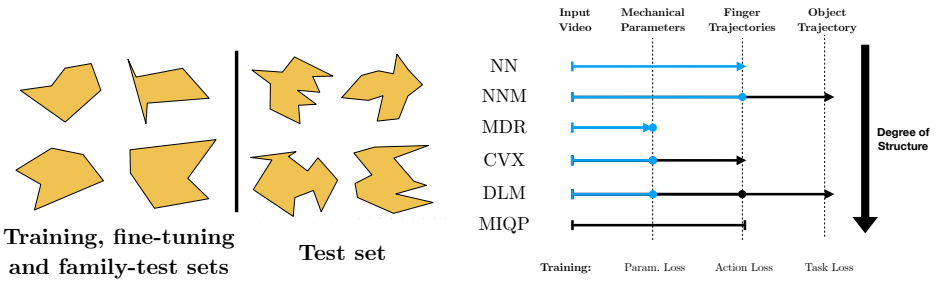


Figure 4-5: Left: Examples shapes of objects in our different datasets. **Right:** Architectures used to evaluate our framework. Blue segments are stages in the architecture with a neural model, black segments are differentiable model-based stages. Dots represent points with labeled data used for pre-training and arrow ends are the final loss function for the full architecture. The MIQP approach has privileged access to all the parameters of the problem.

- MDR: In the Mechanical DeRendering (MDR) architecture, DLM is cut off when it is trained to extract parameters from video by minimizing the loss function $\mathcal{L}_{\mathcal{P}}$. At test time robot actions are obtained by solving **QP**. This is akin to the approach used to learn integer solutions to mixed-integer programs [68, 27].

- CVX: In the ConVeX optimization (CVX) architecture, DLM is cut off after being trained by minimizing $\mathcal{L}_{QP}$. This network is pre-trained with 100 iterations of minimizing $\mathcal{L}_{\mathcal{P}}$.

- DLM: Our proposed approach is trained by minimizing $\mathcal{L}_{Sim}$ after it is pre-trained with 100 iterations of minimizing $\mathcal{L}_{\mathcal{QP}}$ which in turn is pre-trained with 100 iterations of minimizing $\mathcal{L}_{\mathcal{P}}$.

- MIQP (Oracle): A model-based solution to each manipulation task, solved via CTO [4] and provided with all ground-truth parameters (such as object shape and desired object trajectory in $SE(2)$).

Network details: The $CNN(\cdot)$ and $DeConv(\cdot)$ operators represent 3 (de)convolution operations with 6, 12, and 24 channels. The $LSTM(\cdot)$ function is a 5-layer bi-directional recurrent LSTM. Each $MLP(\cdot)$ has 3 fully-connected hidden layers of equal size to their input. All the layers use ReLU activation functions. We add a 0.2 dropout and batch-normalization to each fully-connected layer of the network.

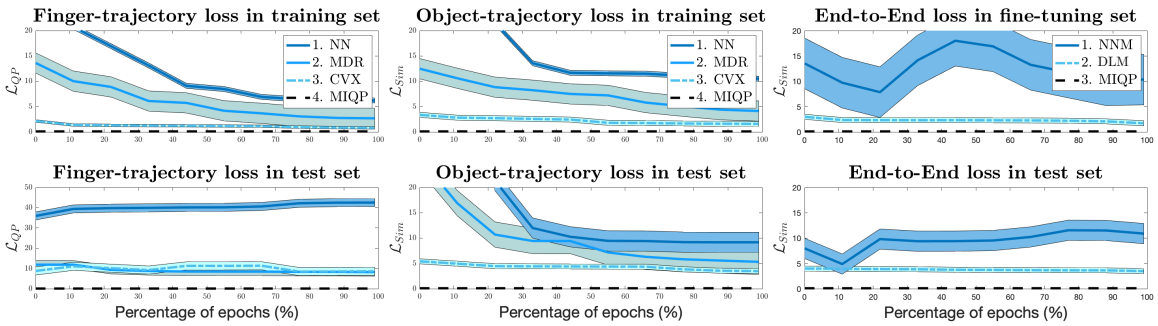


Figure 4-6: Left: In contrast to NN, MDR and CVX closely match the finger-trajectory solution from MIQP (Oracle). **Center:** While each network is trained to match finger trajectories from MIQP, the more structured networks (MDR and CVX) generalize better to unseen object trajectories and shapes. **Right:** Each network is trained by propagating gradients through the simulator. The DLM architecture performs better than the unstructured NNM without over-fitting.

Learning Policies from Data We first analyze the ability to infer finger trajectories from the ground truth data obtained via CTO.

Finger trajectory loss: We measure the ability of NN, MDR, and CVX methods to learn specific finger actions from the training set and evaluate their generalization to unseen data from the test set. We present the results of running each network for 100 epochs in Fig. 4-6 (left). The NN network consistently falls in a local minima. In contrast, the MDR and CVX result in lower training loss closer to the MIQP (Oracle) reference. CVX further refines the learned weights for more accurate predictions, since it is pre-trained with learned weights from MDR.

Simulated object trajectory loss: As we mention in Section IV.D, when we presented new data, the networks struggle to generalize finger trajectories to new object shapes. This is a consequence of each problem having multiple solutions. To resolve this ambiguity, we measure the *object-trajectory* loss by simulating the learned actions at each learning iteration. We show the object trajectory loss curves for each of our datasets in Fig. 4-6 (center). Fig. 4-6 (center) demonstrates how the inferred actions lead to better execution on unseen objects, despite little improvement on the finger-trajectory loss. We note how training with CVX not only refines the loss of MDR, but also prevents over-fitting. Examples of the qualitative performance of each model are shown in Fig. 4-7.

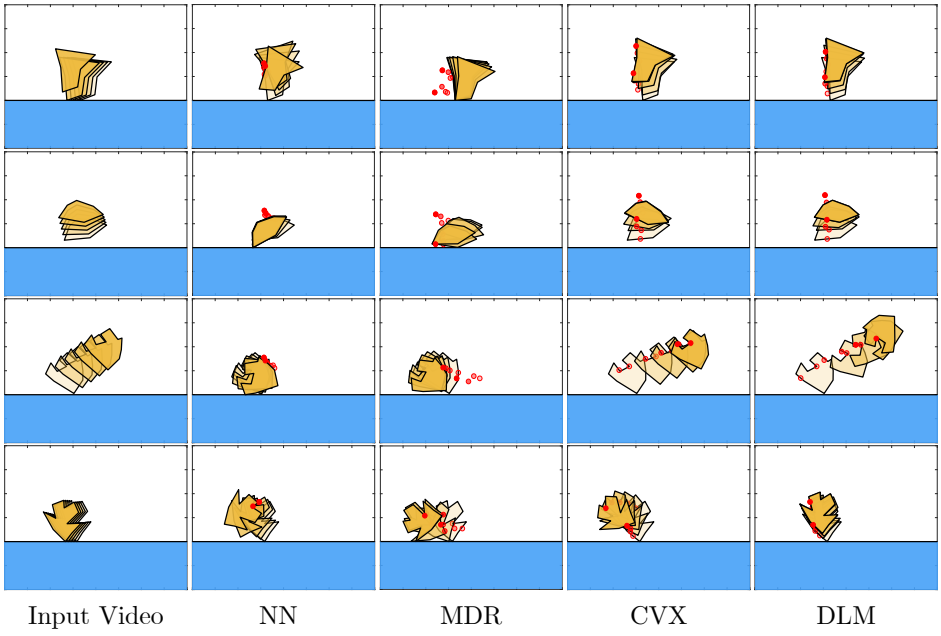


Figure 4-7: Qualitative examples of each network applied to four trajectories from the *family* (top two) and *test* (bottom two) sets. These shapes and motions are not seen during the training phase of each model. Networks with more embedded mechanical structure (MDR, CVX, DLM) tend to perform better than less structured ones (NN). The DLM pipeline provides a layer of self-supervision to the model which helps it to generalize better to unseen scenarios.

Learning Policies from Simulation Next we study the impact of learning actions in a self-supervised fashion. We train DLM and NNM networks on the Fine-tuning set, leveraging the differentiability of our simulator. Both networks are pre-trained with the training set. We plot the resulting object trajectory loss in Fig. 4-6 (right). The DLM architecture decreases the test loss without overfitting on the Fine-tuning set, while the NNM network oscillates around a local minima. This test presents the most similarity to a real-world experiment where, without ground-truth labels on the parameters, the network learns to execute a video motion by iterating over experiments. Examples of qualitative performance of DLM model is shown in Fig. 4-7. We show an example of how this approach can be applied to real object, given a segmented video, in Fig. 4-1. Also see Appendix for results on shapes from the OmniPush dataset [20].

Table 4.1: Object-Trajectory Loss and computation times (in seconds) for the different architectures on each dataset.

Model	Training	Fine-Tuning	Test	Family test	Backward Pass (s)	Forward Pass (s)
NN	8.5 ± 0.5	N/A	9.5 ± 2	10.0 ± 1.5	0.05 ± 0.01	0.06 ± 0.02
NNM	N/A	15.0 ± 5.00	11.0 ± 2.00	15.0 ± 5.0	40.00 ± 5.00	0.06 ± 0.02
MDR	4.5 ± 2.0	N/A	7.5 ± 2.5	4.5 ± 2.0	0.10 ± 0.01	0.11 ± 0.02
CVX	2.0 ± 0.5	N/A	4.0 ± 0.5	3.5 ± 0.5	0.23 ± 0.01	0.11 ± 0.02
DLM	-	1.5 ± 0.5	3.5 ± 0.5	3.5 ± 1.0	40.00 ± 5.00	0.11 ± 0.02
MIQP	-	-	-	-	-	0.57 ± 0.25

Comparative Analysis Finally, we compare the final performance of the different networks in terms of object-trajectory loss and computation times (per data-point), shown in Table 4.1. Inference time in a neural model is smaller than MIQP (Oracle) [4], as our model involves simple operations and solving **QP**. We note that training the DLM model is slow, due to the use of finite difference in the backward pass.

Chapter 5

Certified Grasping

5.1 Caging as Optimization

A cage is an arrangement of obstacles that bounds the mobility of an object such that it cannot escape arbitrarily far from its initial configuration. A cage can be used to manipulate without rigid immobilization, alleviating common issues from jamming, wedging, and general over-constrained interactions (e.g. turning the handle of a door). A cage can also be used as a way-point to a grasp [133, 111], providing a guarantee that the object will not escape in the process [2]. This connection of cages to invariant regions and to robustness has attracted significant attention from the robotic manipulation community.

In practice, however, the application of caging algorithms has been limited. In this work, we propose to rethink the conventional topological/geometric approach to characterize caging—focused on describing the set of finger configurations that cage an object—and instead, we develop an approach that directly synthesizes a caging configuration for some particular objective.

Figure 5-1 describes the motivation and long-term goal of this project: How do we synthesize a manipulation strategy to cage an object while respecting the robot and gripper kinematic limitations, and possibly exploiting the environment? The work in this section is a first step in that direction. We present a reformulation of the caging condition as a set of mixed-integer convex constraints with the *versatility*

to incorporate the caging condition in the context of a larger manipulation planning framework. Moreover, this proposed formulation also provides useful *guarantees* in terms of optimality and convergence of the optimization problem. These properties, however, come at the expense of exponential bounds on computation time, and resolution completeness of the algorithm.

In particular, this section focuses on caging planar polygonal objects—described as the union of convex polygons—with a manipulator described by an arbitrary number of point fingers in the plane. To do this, the proposed approach provides a set of sufficient conditions expressed as convex mixed-integer constraints to cage such object. The main contributions of this section are:

- **Cage optimization** algorithm based on the proposed convex-combinatorial caging model. Cages can be computed efficiently with off-the-shelf mixed-integer solvers yielding global convergence guarantees.

- **Proof of correctness**. If a cage exists within the resolution of the representation, the optimization algorithm will report it. If the algorithm reports a solution, then that solution is a cage, and if a cage does not exist, the algorithm will report so.

- **Validation** of the proposed cage synthesis algorithm on random planar polygons.

- **Application** of the caging algorithm to find cages that exploit the environment (walls) or that take into account kinematic limits in finger motions.

The remainder of this section is organized as follows: Sect 5.1.2 presents relevant concepts and notation, and Sect 5.1.3 provides an overview of the framework to synthesize cages. Sects 5.1.4 and 5.1.5 describe the constraints that represent the proposed our convex-combinatorial model for planar caging. Sect 5.1.6 details the theoretical guarantees of the approach. Finally, Sect 5.1.7 shows the results obtained from implementing the cage synthesis algorithm on random polygons and shows examples of synthesis of complex cages exploiting the environment and considering

kinematic constraints to finger motions.

5.1.1 Related Work

The problem of caging a rigid polygon was originally proposed by Kuperberg [79] as:

> "Let P be a polygon in the plane, and let C be a set of n points in the complement of the interior of P. The points capture P if P cannot be moved arbitrarily far from its original position without at least one point of C penetrating the interior of P. Design an algorithm for finding a set of capturing points for P."

Examples of cages can be seen in Fig. 5-1. This notion was later imported to the robotics manipulation community by Rimon and Blake [107] as the problem of surrounding an object by a robotic hand such that the object is not completely immobilized, but cannot escape the "cage" formed by the fingers. The first set of works that addressed the caging problem focused on algorithmic search—often based on computation geometry techniques—of caging configurations of planar polygons for two- and three-fingered manipulators [107, 118, 126]. In these and most caging works, the manipulator "fingers" are usually modelled as points. More recently, we have also seen techniques that search caging configurations with two and three fingers directly in contact space [9, 25]. In contrast to these works, the approach we present in this section is not limited a specific number of fingers; instead, we present a framework for finding cages with arbitrary number of fingers.

The complexity of the caging problem significantly increases for three-dimensional objects. Pipattanasomporn and Sudsang [98] developed an algorithm for computing all two-finger cages of non-convex three-dimensional objects.

Most works on caging, as the work presented in this section, considers fingers a points. Taking into account the shape of the manipulator can be beneficial, especially when dealing with classes of objects with shape features such as "waists" and "rings"[89, 90], "holes" [117], and "narrow parts" [132]. These approaches avoid reasoning about the entire configuration space, and instead identify specific shape

features that facilitate specific types of cages. Searching for local shape features (in workspace) rather than cages (in configuration space), reduces the complexity of the caging search problem.

In the present section, we do not assume that objects have any specific shape features, but we do build on the concept of searching for a particular type of cage. As described in Sect 5.1.3, we search within those cages that are defined by a sequence of loops in adjacent slices of configuration space which jointly define an enclosing of the object. This yields a rich and computationally favorable family of cages. One of the key contributions is a representation of the loop in each slice, and how it changes between adjacent slices. This representation makes it possible to represent the problem as a convex mixed-integer optimization problem with search guarantees.

An alternative approach for cage verification is to build an explicit approximation of the configuration space of the object, either in two-dimensional [86] or three-dimensional [128, 130] space. While this approach offers more generality, as it is applicable to manipulators and objects of arbitrary shapes, it is computationally expensive due to the high dimension of the configuration space. To mitigate this problem, Varava et al. [128, 130] represent configuration spaces as a collection of lower-dimensional slices with fixed orientation values. This is similar to our choosing of decomposing the configuration space into discrete orientation slices. In contrast to our proposed approach, however, Varava et al. [128, 130] assume a predefined rigid configuration of the manipulator and do not optimize relative finger placements. This is done such that the configuration space of the object is defined. In our work, we directly optimize the distribution of finger placements around the object.

The majority of works in caging consider cage verification and synthesis as separate problems. This has made integrating caging into motion planning and trajectory optimization frameworks relatively unaddressed in the literature. Some scenarios where caging has been integrated in manipulation planning are multi-agent transportation of rigid objects [97] and multi-agent herding of a group of mobile agents [131]. In these scenarios, maintaining caging provides robustness—the object or the agent cannot escape. In general MIP has been applied to a diverse set of engineering problems

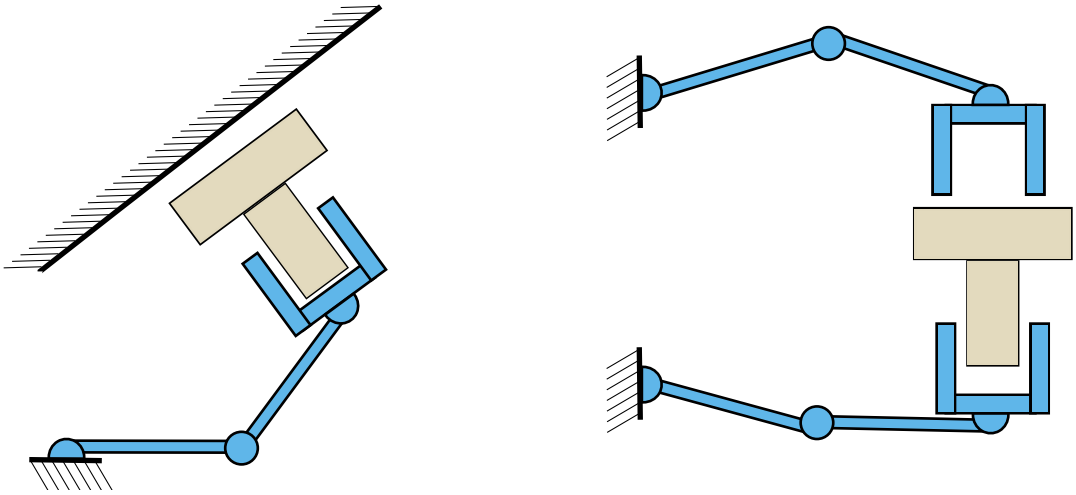


Figure 5-1: Illustrative examples of caging in the context of robot manipulation: trapping against a wall and caging under kinematic constraints, e.g., limited gripper opening.

in robotics, including design, scheduling, planning, and control [76, 64, 43, 36, 3, 125].

5.1.2 Preliminaries

Here we introduce the notation and definitions that we will use through this section.

Notation

Given a rigid object $\mathcal{O}$ in a planar workspace $\mathcal{W} \subset \mathbb{R}^2$, we denote its configuration as $q = [q_x, q_y, q_\theta]^T$ in its *Configuration Space* $\mathcal{C} = SE(2)$ [84]. The object $\mathcal{O}$ can be represented as the union of M convex polygons $\mathbf{P}_1, \ldots, \mathbf{P}_M$ covered by L facets $\mathbf{F}_1, \ldots, \mathbf{F}_L$. We refer to a hyperplane of $\mathcal{C}$ with fixed orientation component θ as a $\mathcal{C}-$slice, denoted $\mathcal{C}(\theta)$. The manipulator $\mathcal{M}$ is an arrangement of N point fingers in the workspace $\mathcal{W}$, with positions $\mathcal{M} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_N\} \in \mathcal{W}^N$. We summarize this notation in Table 5.1 and depict it in Fig. 5-2.

In configuration space $\mathcal{C}$, and at each $\mathcal{C}-$slice, we refer to the set of configurations where the object intersects a finger as $\mathcal{C}$-obstacles (also known as Collision Space). Then, the free space of the object $\mathcal{C}_{\text{free}}$ is the subset of $\mathcal{C}$ not intersecting any of the $\mathcal{C}$-obstacles. Note that we allow the object to be in contact with the obstacles, and

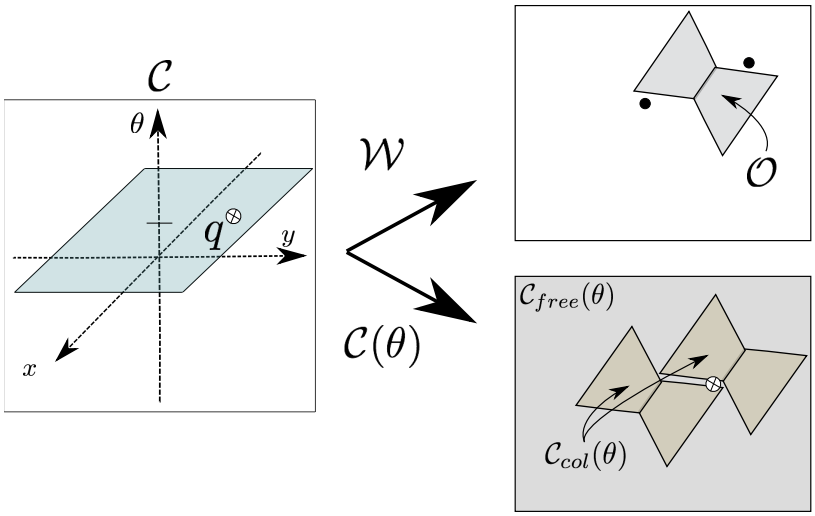


Figure 5-2: Depiction of some of the central concepts relevant to this chapter: 1) The configuration space of a planar object $\mathcal{C}$, under a configuration $q = [x, y, \theta]$, 2) the workspace $\mathcal{W}$ with the object $\mathcal{O}$ and two point obstacles, and 3) The visual representation of the slice of C-space $\mathcal{C}(\theta)$.

so the free space is a closed set. For a more detailed definition of free configuration space and its properties, see Rodriguez and Mason [109].

Caging

Using the language of configuration space, we re-state the caging problem as

> Given a planar object $\mathcal{O}$ at configuration $q = [q_x, q_y, q_\theta]^T$ and an N-finger manipulator $\mathcal{M}$, find a configuration of $\mathcal{M}$ such that q lies in a compact connected component of $\mathcal{C}_{\text{free}}$. We denote that compact connected component as $\mathcal{C}_{\text{free}}^{\text{compact}}$.

This formulation of the problem, based on a topological characterization, is equivalent to the more traditional geometric condition that there must exist no continuous path for the object to go arbitrarily far from the manipulator.

5.1.3 Approach Overview

In this section, we overview our approach to represent the problem of caging a planar object.

Table 5.1: Summary of notation used in this chapter

$\mathcal{O}$	object
$\mathcal{W} \subseteq \mathbb{R}^2$	workspace
$\mathcal{C} \subseteq SE(2)$	C-space
$\mathcal{C}_{\text{free}} \subset \mathcal{C}$	free space
$\mathcal{C}_{\text{col}} \subset \mathcal{C}$	C-obstacles
$\mathcal{C}(\theta)$	slice of the C-space
$\mathcal{C}_{\text{free}}(\theta)$	free space of a slice
N	number of robot fingers
M	number of convex polygons composing $\mathcal{O}$
S	number of $\mathcal{C}-$slices
R	number of convex regions in $\mathcal{W}\backslash\mathcal{O}$

Conditions for Caging

Based on the definition in Section 5.1.2, synthesizing a cage is equivalent to creating a compact connected components in the free space $\mathcal{C}_{\text{free}}^{\text{compact}}$. In general, the set of possible compact connected components is very large. In [111], Rodriguez and Mason showed that it might be indeed too large for it to be a useful characterization of cages and grasps. Here, we introduce a set of conditions that characterize a rich set of cages that are both useful and will make the search more efficient. We restrict to those cages where the compact connected component $\mathcal{C}_{\text{free}}^{\text{compact}}$ has a single local maxima and minima along the orientation θ axis, as illustrated in Figure 5-3. To formalize this idea, we first introduce the concept of limit orientations

Definition 1 (Limit Orientation)**.** *Given a compact connected component $\mathcal{A}$ of the configuration space of the object $\mathcal{C}$, we define its upper limit orientation as*

$$\theta_U = \sup_{\Theta} \theta \text{ where } \Theta = \{\theta \text{ s.t. } [q_x, q_y, \theta] \in \mathcal{A}\}$$

and its lower limit orientation as

$$\theta_L = \inf_{\Theta} \theta \text{ where } \Theta = \{\theta \text{ s.t. } [q_x, q_y, \theta] \in \mathcal{A}\}$$

For some $[q_x, q_y] \in \mathbb{R}^2$. The key idea we propose in this chapter is that the com-

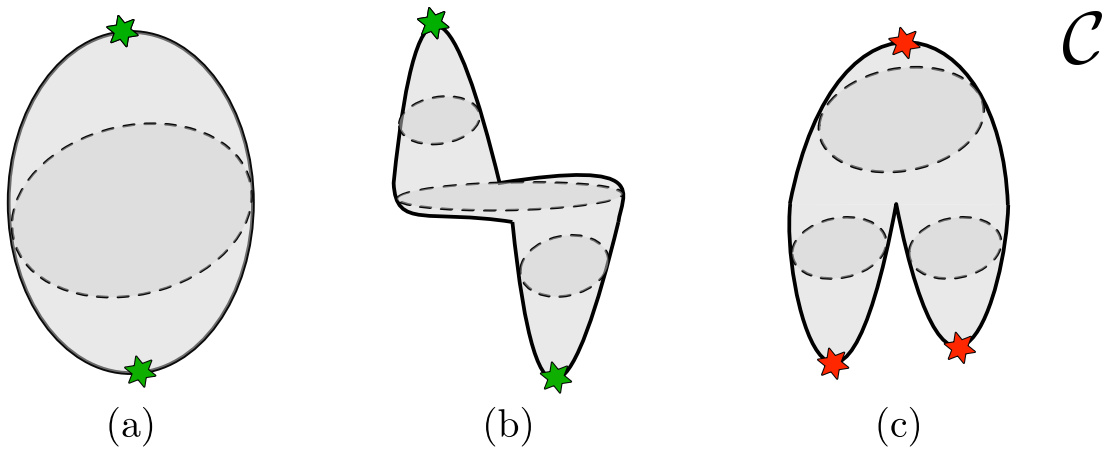


Figure 5-3: Different types of compact connected components. The conditions described in this section describe cages (a) and (b), as both of these components have a pair of limit orientations where the component opens and closes (green stars). However, these conditions are not sufficient to create a cage with component (c), as there are two local minima in the orientation of the component (red stars).

bination of the following three constraints generate a compact connected component $\mathcal{C}_{\text{free}}^{\text{compact}}$, and that these constraints can be expressed with the language of mixed-integer programming:

1. The mobility of the object is bounded in the θ axis by two limit orientations θ_U and θ_L, or it is unbounded and it cycles, i.e., object can rotate indefinitely.

2. In all $\mathcal{C}-$slices between the two limit orientations there is a loop of $\mathcal{C}-$obstacles enclosing the free configurations q of the object. These loops must be connected between adjacent slices in a way that they form an enclosing of the configuration of the object.

3. At the $\mathcal{C}-$slices of the limit orientations, when they exist, the free space component enclosed by the loop must degenerate to zero volume. In our case, this will mean being reduced to a line segment or a single configuration.

Figure 5-4 illustrates the process in the workspace, slices, and configuration space where these conditions hold. While these conditions do not represent the complete set of cages of an object, they include a large subset. Figure 5-3 gives some intuition of the type of cages we do and do not characterize.

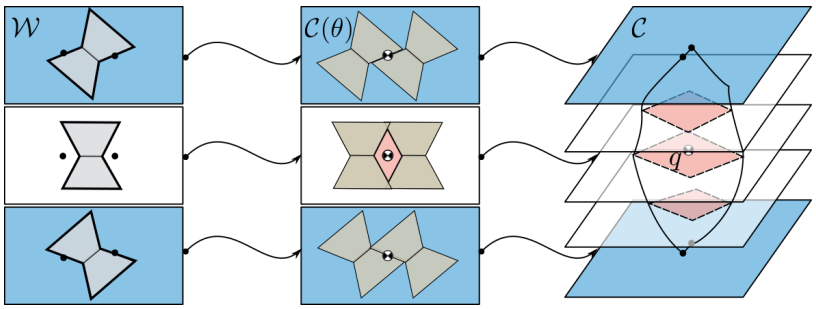


Figure 5-4: Overview of the conditions for caging. Each plane shows a $\mathcal{C}-$slice during a cage between two limit orientations (blue) with a compact connected component of free space (red). Note that in the *limit orientations* the object is constrained to a line segment of translational motion by the collision space. Also, the object is only caged if the component remains compact and connected between slices.

Model Overview

To make it possible to formulate the problem as a convex mixed-integer program, we rely on the following assumptions:

- The object $\mathcal{O}$ is represented as the union of M convex polygons, with a boundary composed of L line segments.
- The manipulator $\mathcal{M}$ is represented as a set of N point-fingers.
- We discretize $\mathcal{C}$ in S $\mathcal{C}$-slices, similar to [128]. We will impose that the manipulator bounds the object in each slice and will also impose regularity between consecutive slices to provide caging guarantees.

These assumptions allow us to: 1) describe the compact-connected component at each $\mathcal{C}-$slice as a chain of interconnecting convex polygons, and 2) describe limit orientations and point-line intersection constraints. Formulating this model requires continuous variables to represent the positions of the manipulator fingers, and binary variables to represent the discrete-combinatorial connectivity relation between $\mathcal{C}$-obstacles. Therefore, we will introduce three types of constraints to our model:

- **Limit orientations:** we constrain that either the object is caged for all 360° or that $\mathcal{C}_{\text{free}}^{\text{compact}}$ is bounded by two *limit orientations*.

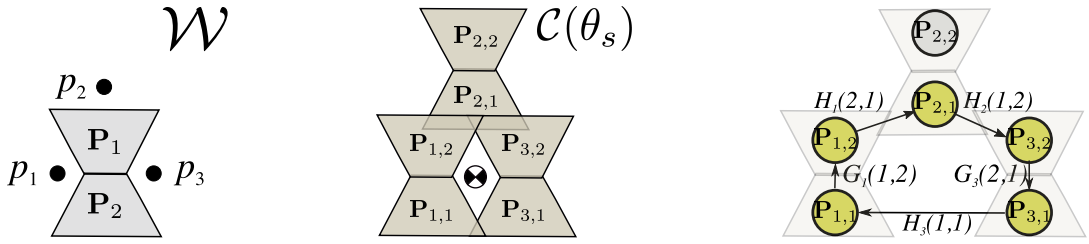


Figure 5-5: The enclosing loop for a slice with three fingers in $\mathcal{W}$ (left), $\mathcal{C}(\theta)$ (center) and its corresponding connection graph (right)

.

- **At each slice:** We require that a subset of the N $\mathcal{C}-$obstacles, each induced by a finger, forms a loop around the configuration of the object $[q_x, q_y]^T$. This ensures that there is a compact connected component at each slice

- **Between slices:** We require that the loop at each $\mathcal{C}-$slice maps continuously, without breaking, into the loop of its adjacent $\mathcal{C}-$slices. We impose this by requiring that the connectivity between $\mathcal{C}-$obstacles at each slice remains consistent across $\mathcal{C}-$slices, forming the component $\mathcal{C}_{\text{free}}^{\text{compact}}$ that encloses q.

In Sections 5.1.4 and 5.1.5, we show how to formulate these conditions and that they are sufficient to guarantee that the object $\mathcal{O}$ is caged by the manipulator $\mathcal{M}$.

5.1.4 Constructing enclosing loops at configuration space slices

In this section, we derive sufficient conditions for an object to be caged in a configuration space slice $\mathcal{C}(\theta)$, that is given an orientation of the object θ, we derive conditions that guarantee that the object cannot escape under planar translations.

The approach to derive these conditions is to construct a continuous loop in the $\mathcal{C}$-slice that encloses the configuration of the object q. To that end, we impose that the configuration of the N manipulator fingers is such that the union of the $\mathcal{C}$-obstacles induced by a subset of these fingers contains such an enclosing loop. We do this in three steps: In Section 5.1.4 we describe how to enforce that a loop exists, in Section 5.1.4 we impose that the loop encloses the object, and in Section 5.1.4 we require that the manipulator fingers that define the loop live in the free space of the object $\mathcal{C}_{\text{free}}$.

We start by defining the notion of enclosing loop. Consider an object $\mathcal{O}$, its configuration space $\mathcal{C}$, and a slice of constant orientation $\mathcal{C}(\theta)$, a manipulator $\mathcal{M}$, and the $\mathcal{C}(\theta)$-obstacle it defines in the the $\mathcal{C}(\theta)$. We define:

Definition 2 (Enclosing loop)**.** *Continuous closed curve contained in* $\mathcal{C}(\theta)$-obstacle *that cannot be contracted to a point inside* $\mathcal{C}(\theta)$-obstacle*.*

And we make the following remark:

Remark 1. *An object in configuration* $q = [q_x, q_y, q_\theta]$ *is caged under planar translations in* $\mathcal{C}(\theta)$ *if* q *lies in the interior of an enclosing loop in* $\mathcal{C}(\theta)$-obstacle*.*

Since the configuration space of a planar object with a fixed orientation is a subset of $\mathbb{R}^2$, the statement immediately follows from Jordan's polygon theorem [114]. The connected component of the free space containing q must be bounded by definition, and since the free space is closed, it must also be compact.

For notation convenience, we will refer to the free space of $\mathcal{C}(\theta)$ as $\mathcal{C}_{\text{free}}(\theta)$, and its enclosed component $\mathcal{C}_{\text{free}}^{\text{compact}}(\theta_s)$ as $\mathcal{C}_c(\theta_s)$. To create $\mathcal{C}_c(\theta_s)$, we require that the $\mathcal{C}-$obstacles in $\mathcal{C}(\theta)$ form a loop.

Existence of a loop

We assumed in Section 5.1.3 that the object can decomposed into the union of M convex polygons $\mathbf{P}_1 \dots \mathbf{P}_M$, such that:

$$\mathcal{O} = \bigcup_{i=1}^{M} \mathbf{P}_i$$

This property becomes key to formulate the existence of a loop between a subset of the N $\mathcal{C}(\theta)-$obstacles induced by the N fingers. Because the obstacle associated to manipulator finger i in the slice $\mathcal{C}(\theta)$ has the shape of a rotated and translated version of the object $\mathcal{O}$, it can also be decomposed into the union of M convex polygons $\mathbf{P}$. We will note the convex polygons that decompose the $\mathcal{C}(\theta)$-obstacle associated to finger j as $\mathbf{P}_{1,j} = \mathbf{p}_j \bigoplus \mathbf{P}_1, \dots, \mathbf{P}_{M,j} = \mathbf{p}_j \bigoplus \mathbf{P}_M$, where $\bigoplus$ is the Minkowski sum

operator, such that:

$$\mathcal{C}_{col}(\theta) = \bigcup_{j=1}^{N} \bigcup_{i=1}^{M} \mathbf{P}_{i,j}$$

Then loop existence is equivalent to constructing a loop in a graph where nodes are each of the convex polygons $\mathbf{P}_{i,j}$ $\forall i, j$, and edges represent the intersection or adjacency among them. This is illustrated in Figure 5-5.

For the algorithm to be general to any number of fingers, it must determine which of the fingers or polygons on each $\mathcal{C}-$obstacles are part of the enclosing loop, to account for the possibility that not all fingers are necessary. For representational purposes it must also determine the direction of each of these edges in a directed graph, which prevents degenerate loops. We address the construction of enclosing loops without using all fingers at the end of this section, but for now, for simplicity of exposition, we assume that all fingers must be part of this loop.

Loops that use all fingers Here we formulate the constraint that the finger obstacles form a loop of constraints around the configuration of the object. We assume, without loss of generality, that the indices of the fingers $1 \dots N$ are in the same order as they eventually appear in the loop of constraints, i.e. finger i is nearby finger $i+1$ and this nearby finger $i+2$. When fingers can move freely, this assumption comes without loss of generality since an MIP will ultimately optimize over finger locations, and the caging property is invariant to their identity. However, if the fingers have any kinematic constraints, the numbering of fingers can have an impact on the set of possible cages. Moreover, this assumption implies that all fingers are part of the cage, forbidding the model from that ability to discriminate when a finger is not necessary in the cage.

In practice, this means that when encoding intersections between finger obstacles we can only consider adjacent fingers. To do so, for each finger n, we introduce a binary matrix $H_n \in \{0,1\}^{M \times M}$ that encodes the intersections or connectivity between its convex polygons and those of its adjacent finger. That is, $H_n(i,j) = 1$ if polygon $\mathbf{P}_{i,n}$ intersects polygon $\mathbf{P}_{j,n+1}$. Imposing an intersection between those two polygons

can be written as

$$H_n(i,j) = 1 \Rightarrow \exists\ \mathbf{r}_n \in \mathbb{R}^2 \text{ s.t. } \mathbf{r}_n \in \mathbf{P}_{i,n} \cap \mathbf{P}_{j,n+1} \tag{CT1}$$

Note that the values of H are variables that depend on the configuration of the fingers $\mathbf{p}_1, \ldots, \mathbf{p}_N$. The coordinates of the point $\mathbf{r}_n$ also become continuous variables in the optimization problem, and imposing that $\mathbf{r}_n$ belongs to $\mathbf{P}_{i,n}$ and $\mathbf{P}_{j,n+1}$ is a series of linear constraints, given that both are convex polygons. we can also transcribe the operator $\Rightarrow$ (implies) as linear constraint through the big-M formulation [22].

To make all fingers a part of the loop, we enforce the following constraint for each finger n:

$$\sum_{i,j} H_n(i,j) = 1 \tag{CT2}$$

This imposes that there is one and only one directed edge from $\mathcal{C}(\theta)$-obstacle n to $\mathcal{C}(\theta)$-obstacle $n+1$. Since these connections are enforced for $n = 1, \ldots, N$, they lead to a directed loop of obstacles in $\mathcal{C}(\theta)$.

In addition to using H_n to represent the connectivity between different $\mathcal{C}(\theta)$-obstacles, we also introduce a binary matrix $G_n \in \{0,1\}^{M \times M}$, for each finger n, to represent the internal connectivity between the convex polygons that form each $\mathcal{C}(\theta)-$obstacle. That is, $G_n(i,j) = 1$ if polygon $\mathbf{P}_{i,n}$ and $\mathbf{P}_{j,n}$ are connected. Note that the values of G are constants pre-determined by the shape of the object $\mathcal{O}$ and its decomposition into convex polygons.

We can then formulate the loop closure condition between all N fingers as:

$$H_{n-1}(i,j) \Rightarrow \exists\ k, \exists\ l \text{ s.t. } G_n(j,k) + H_n(j,l) = 1 \tag{CT3}$$

$$G_n(i,j) \Rightarrow \exists\ k \neq i, \exists\ l \text{ s.t. } G_n(j,k) + H_n(j,l) = 1 \tag{CT4}$$

(CT3) and (CT4) combined guarantee that for each node with an inbound edge, there is one and only one outbound edge. When combined, this defines a loop in $\mathcal{C}(\theta)$. Note that in the special case of a two-finger manipulator, since $H_{n,n+1}$ and $H_{n-1,n}$ have the

same value, we need to further constrain that $l \neq i$ in (CT3).

Extension to arbitrary loops The previous constraints allow us to find cages that employ all fingers. However, the flexibility of the model to use integer variables also facilitates finding loops containing only a sub-set of the fingers. For this, we can extend each H matrix into a higher dimensional tensor that considers all possible finger combinations. Essentially, $H_n(i,j,m) = 1$ would encode an intersection between $\mathbf{P}_{i,n}$ and $\mathbf{P}_{j,m}$. This would extend the dimension of each H matrix to $M \times M \times (N-1)$ (note the $N-1$ term represents connection with the other fingers). With this new formulation, we have recast constraint (CT2) and replace it by

$$\sum_{i,j} \sum_{n,m} H_n(i,j,m) + G_n(i,j) \geq 1$$

adding the requirement to have at least one edge in the closed directed graph. Similarly, equations (CT3) and (CT4) are modified as:

$$H_n(i,j,m) \Rightarrow \exists\ k, \exists\ l, \exists\ o \text{ s.t. } G_m(j,k) + H_m(j,l,o) = 1$$

$$G_n(i,j) \Rightarrow \exists\ k \neq i, \exists\ l\ \exists\ o \text{ s.t. } G_n(j,k) + H_n(j,l,o) = 1$$

With this extended formulation, the model can consider interconnections between arbitrary pairs of fingers. This removes any dependence on finger numbering, as any pair of fingers can be a part of the loop, and allows the model to distinguish "minimal" cages without redundant or unnecessary fingers, as in [97]. Naturally, this comes at the cost of a much larger H matrix, which has a significant impact in the computational complexity of the Mixed-Integer model.

Object enclosing

The previous constraints ensure the existence of a compact connected component $\mathcal{C}_{\text{free}}^{\text{compact}}(\theta)$ in each slice. It is important to note that the enclosing loop implies the existence of a piecewise-polygonal curve in each slice $\mathcal{C}(\theta)$ that traverses through the

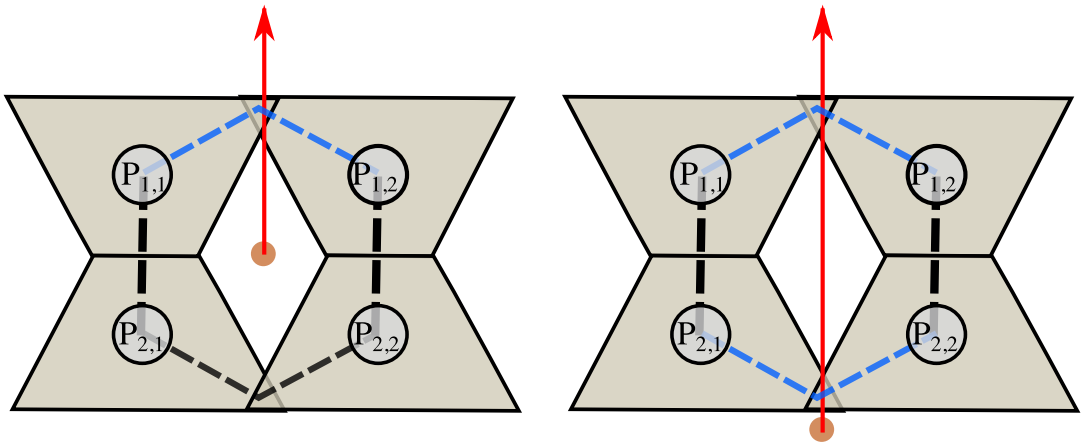


Figure 5-6: A point lies within a closed curve (dashed line) if a ray originating from that point has an odd number of intersections (blue) with the edges of the loop. Left: the point lies inside the polygon, and the ray has an odd number (one) of intersections. Right: the point lies outside the polygon, and the ray has an even number (two) of intersections.

inside of some of the $\mathcal{C}(\theta)$-obstacles. We construct this curve by connecting points that live in the intersection of polygons in the loop.

However, this does not guarantee that $q = [q_x, q_y, q_\theta]$ is contained in $\mathcal{C}_{\text{free}}^{\text{compact}}$, since the configuration q could lie on the outside of the component. Note that enclosing q in one of the slices $\mathcal{C}_{\text{free}}^{\text{compact}}(\theta)$ implies requiring that $[q_x, q_y]^T$ is enclosed in $\mathcal{C}_{\text{free}}^{\text{compact}}(q_\theta)$. To formalize this constraint, we use the following remark:

Remark 2 (Point inside a curve [114])**.** *Given a closed curve and a point r in the plane, and given a ray starting at r. If the ray intersects the curve an odd number of times, then r is in the interior of the curve.*

This remark is valid except for degenerate cases, for example when the ray is parallel to a segment of the curve, in which case the intersection can be a segment rather than a point, or when the curve encloses zero area. These scenarios can be avoided in our analysis if all the polygons have non-zero area (none of them is a point or a line). In this case, we can define the enclosing curve such that its segments do not coincide with the facets of the $\mathcal{C}-$obstacles. This prevents the enclosing curve from being completely parallel to the ray or having zero area.

Making use of Remark 2 we impose the configuration q to be enclosed by the

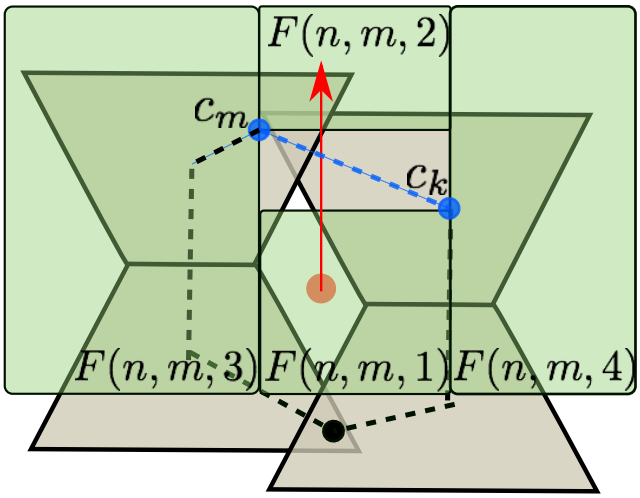


Figure 5-7: Counting ray (red arrow) intersection with enclosing polygonal loop (dashed line). For any given segment (dashed blue line) of the enclosing loop, the configuration q (red dot) must lie in one of the four regions $F(n,m,1)$, $F(n,m,2)$, $F(n,m,3)$, $F(n,m,4)$. Note that since the regions are aligned with the direction of the ray, the ray intersects the mentioned segment when the configuration when q lies in $F(n,m,1)$.

curve constructed above by constraining the number of intersections between a ray and the line segments to be odd. We will see that we can impose this as a convex-combinatorial constraint.

To count the number of times the ray crosses the enclosing polygonal loop, we count the number of edges it crosses. We do that by creating binary variables that encode the relative location of the configuration q and the ray with respect to each polygon in the loop. Verifying if a ray intersects a line segment requires a bilinear non-convex constraint, which is incompatible with our framework. However, we can recast this constraint by checking whether the origin of the ray lies on, over, under, or to the side of the segment. To model this constraint, we start by assuming a direction of the ray (in our case we use the positive $y-$axis) and by decomposing the surrounding area around each segment in the polygonal loop into four boxes orthogonally aligned with the direction of the ray. Figure 5-7 illustrates an example where the ray is in the direction of positive Y, and the four regions for the blue dashed edge. For each segment, and their associated four regions, we introduce binary decision variables $F \in \{0,1\}^{N\times M\times 5}$, such that:

1. $F(n, m, 1)$ through $F(n, m, 4)$, when equal to one, assign $[q_x, q_y]$ to that particular region of the line segment starting in the m_{th} polygon of the n_{th} finger.

2. $F(n, m, 5)$ is set to 1 if the m_{th} polygon of the n_{th} finger is not part of the loop.

We strategically construct the regions such that region number 1 is parallel to the ray and below the line segment. By construction then, the ray will intersect the segment if $F(n, m, 1) = 1$, as illustrated in Figure 5-7.

With these variables, we can constraint the ray to intersect an odd number of times with the polygonal loop by imposing

$$\begin{cases} \sum_{n,m} F(n, m, 1) \text{ is an odd number} \\ \sum_{i=1}^{5} F(n, m, i) = 1 \ \ \forall n, m \end{cases} \tag{CT5}$$

note that the second constraint is necessary to only assign configuration q to one of the four regions. This also effectively reduces the area where the configuration may lie to the intersection of the actual caging region and the orthogonal region.

Finally to transform (CT5) into a set of (mixed-integer) linear constraints, we use the fact that $(1\ XOR\ 1) = (0\ XOR\ 0) = 1$ and $(1\ XOR\ 0) = (0\ XOR\ 1) = 1$, hence we obtain the following lemma

Lemma 1. *The summation of binary variables $\sum_{i=1}^{n} b_i$ is an odd number if and only if $b_1\ XOR\ b_2\ \ldots\ XOR\ b_n = 1$.*

Where the XOR operator can be transcribed as linear constraints on the binary variables as described in [72].

Non-Penetration Constraints

Finally, for the previous constraints to be valid, we must prevent the fingers from penetrating the object at each slice where there must exist a loop. Unlike the enclosing loop constraint which we represented in the configuration space of the object, non-penetration is easier to represent in the workspace of the object. For this, we partition

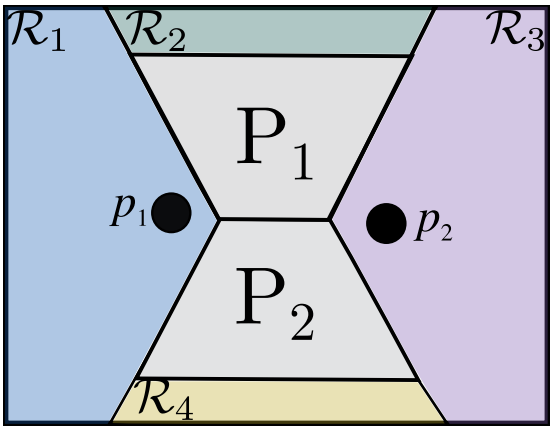


Figure 5-8: Decomposition of the free workspace $\mathcal{W} \setminus \mathcal{O}$, for some orientation θ_0, into the union of convex regions and assignment of each finger to a region. In this example with two point fingers, the free workspace is segmented into $R = 4$ convex polygonal regions $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4$, finger $\mathbf{p}_1$ is in region $\mathcal{R}_1$ (that is $\boldsymbol{R}_{1,1}(\theta) = 1$) and finger $\mathbf{p}_2$ is in region $\mathcal{R}_3$ (that is $\boldsymbol{R}_{3,2}(\theta_0) = 1$).

the 2D free workspace around the object $\mathcal{W} \setminus \mathcal{O}$ into a set of R convex regions, as illustrated in Figure 5-8. We represent each convex region as the union of half-spaces:

$$\mathcal{R}_i = \{x \in \mathbb{R}^2 | A_i x \leq b_i\}$$

and then constrain that each finger $\mathbf{p}_n$ to lie in one of these regions. For this, we introduce the binary decision matrix $\boldsymbol{R}(\theta) \in \{0,1\}^{R \times N}$ that assign fingers to a region at each slice with:

$$\boldsymbol{R}_{r,n}(\theta) \Rightarrow A_i \mathbf{p}_n \leq b_i \quad \text{and} \quad \sum_{r=1}^{R} \boldsymbol{R}_{r,n}(\theta) = 1, \forall n \tag{CT6}$$

this ensures that each finger lies in one and only one of the regions. We again transcribe the $\Rightarrow$ operator via big-M formulation. Note that this constraint, although formulated in the workspace of the object, it also ensures that q lies in each $\mathcal{C}$-slice without penetrating any $\mathcal{C}-$obstacle.

The following section we discuss how the loops in each slice interconnect to create a compact connected component $\mathcal{C}_{\text{free}}^{\text{compact}}$ or cage.

5.1.5 Constructing a cage from loops

To guarantee a cage, we will require that $\mathcal{C}_{\text{free}}^{\text{compact}}$ either repeats periodically in the orientation dimension or closes at two *limit orientations*, in order to remain compact. In Section 5.1.5 we present a set of sufficient conditions for that to be the case.

Further, for the object to be fully caged, the enclosing loops at each of the $\mathcal{C}-$slices constructed in Section 5.1.4 must be interconnected such that there are no escaping paths in the space between slices. In Section 5.1.5 we show that we can impose sufficient regularity conditions to guarantee it.

Limit Orientations

We distinguish between two different ways to cage an object:

1. A cage where the object rotation is not limited, also known as a *fence*,

2. A cage where the reorientation of the object is limited.

The former case is a practical, but often inefficient way to cage an object that employs many fingers. The later case requires the existence of *limit orientations* in the configuration space between which the mobility of the object is bounded.

We are specially interested in the construction of cages of the latter type. For that, the model must impose the existence of and determine a pair of limit orientations θ_U, θ_L in $\mathcal{C}_{\text{free}}^{\text{compact}}$. To capture this in the formulation, we introduce a vector of binary variables $\Theta \in \{0,1\}^S$, one variable for each slice, such that $\Theta_s = 0$ implies that slice s with orientation θ_s lies between two limit orientations. When a slice lies between two limit orientations, we must impose the necessary constraints for that slice to contain an enclosing loop (existence of a loop and non-penetration). We impose this relation through the following constraints

$$\Theta_s = 0 \Rightarrow \begin{cases} \sum_{r=1}^{R} \boldsymbol{R}_{r,n}(\theta_s) = 1 \\ \sum_{i,j} H_n(i,j)(\theta_s) = 1 \end{cases} \tag{CT7}$$

Where $\boldsymbol{R}_{r,n}(\theta_s)$ are the binary variables defined for non-penetration, and $H_n(i,j)(\theta_s)$ are the binary variables defined to construct the enclosing loop for slice θ_s. With this constraint CT7, if slice $\mathcal{C}(\theta_s)$ lies between two limit orientations then the loop existence and non-penetration constraints are active. If the slice, however, lies outside the limit orientations then the loop existence and non-penetration constraints are not enforced.

We constrain that $\Theta_s = 1$ when the component of $\mathcal{C}_{\text{free}}(\mathcal{O})$ in the corresponding slice degenerates to zero area. Moreover, assuming slices are ordered by increasing orientation, we add the condition that:

$$\Theta_s = 1 \implies \begin{cases} \Theta_{s+1} = 1, & \theta_s \geq \theta_U \\ \Theta_{s-1} = 1, & \theta_s < \theta_L \end{cases}$$

Which removes all constraints for the slices after the limit orientations. The condition that imposes that the cage in a slice degenerates to zero area depends on the number of fingers and on the boundary of the object. Figure 5-9 illustrates some scenarios that satisfy this zero-area condition:

- **Two fingers**: the fingers make contact with two parallel but opposite facets of the object, as in Figure 5-9 (top).
- **Three fingers**: three fingers are in simultaneous contact with non-parallel facets, as in Figure 5-9 (center).
- **More fingers**: four or more fingers are in simultaneous contact with non-co-directional facets (without the same normal), as in Figure 5-9 (bottom).

where opposite refers to facets with normals that are parallel but with opposite direction. Since the fingers create a loop that encloses q, any *concave vertex* in the object can be considered opposite and non-parallel to the facets with a 180° difference to some angle in arc of the vertex.

Finally, for the optimization problem to be able to determine automatically the limit orientations θ_U and θ_L among the vector of orientations Θ, we characterize all

geometric object-finger configurations that define a limit orientation and force that one of these configurations is satisfied at the limit orientations. To achieve this, introduce at each slice Θ_s a binary matrix $T_s \in \{0,1\}^{N \times L}$. Each term in the matrix is constrained so that $T_s(n, f) = 1$ implies that finger n is in contact with facet $\mathbf{F}_f$, for some configuration of the object within the slice s.

Given that the shape of the object is known, we can determine which combination of contacts with the facets of the object induce a limit orientation. Note that each component of T represents one possible combination of contacts. We pre-compute which elements of T induce a limit orientation to obtain the set $\mathcal{L}_{\mathcal{O}}$, comparing each pair of facets in the objects, and impose constraint:

$$T_s \in \mathcal{L}_{\mathcal{O}} \Rightarrow \Theta_s = 1 \tag{CT8}$$

which guarantees that the model will indeed assign $\Theta_s = 1$ only if it is a real limit orientation.

As an aside note, and in relation with previous works in caging theory, the idea of caging between two limit configurations was key for Rimon and Blake's original caging formulation [106], and is equivalent to finding a critical point of the inter-finger distance function in the contact space of the object, as proposed by Allen et al. [9] and Bunis et al. [25].

Continuous Boundary Variation

The constraints described in Section 5.1.4 and in Section 5.1.5 ensure the existence of a compact connected component of free space in each sampled $\mathcal{C}$-slice, ensure that q lies in one of those slices, and ensure that the free space closes in upper and lower limit orientations. Due to the discretization of slices, the object might still escape through the space in between the $\mathcal{C}$-slices, through complex motions that combine translations and rotations. See the example in Figure 6 in Rodriguez et al. [111]. To avoid this, we impose conditions to guarantee that the polygonal intersections that define the enclosing loops in Section 5.1.4 are maintained between slices. This

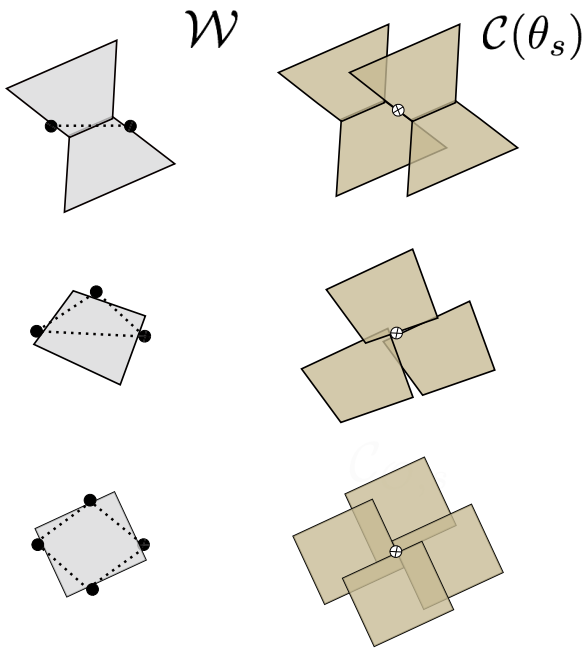


Figure 5-9: Examples of limit orientations: (top) Two fingers in contact on facets with opposite normals yield a line in free space. (center) Three fingers in contact on non-parallel facets yield a single point in free space. (bottom) Four points in contact with with four non-co-directional facets, each facet has a different normal vector.

will ensure sufficient regularity to guarantee that the configuration q is in a compact connected component of $\mathcal{C}_{\text{free}}$. In this section we describe the constraints, and in Section 5.1.6 we give a proof of correctness.

In essence, to impose regularity between slices, we impose that the active set of discrete connections between polygons in a slice remains active in adjacent slices, as illustrated in Figure 5-10. In practice, we model this constraint by enforcing that there is a point x that remains inside the intersections across adjacent slices. Given two fingers $\mathbf{p}_n$ and $\mathbf{p}_{n+1}$ whose $\mathcal{C}$-obstacles contribute to define an enclosing loop, we have noted their respective intersecting convex polygons as $\mathbf{P}_{i,n}$ and $\mathbf{P}_{j,n+1}$. We define now a new decision variable $x_{j,n+1}$, at each slice, denoting a point that belongs to the intersection of $\mathbf{P}_{i,n}$ and $\mathbf{P}_{j,n+1}$. Note that by the subscript $_{j,n+1}$ we mean to indicate that x is expressed in local relative coordinates to $\mathbf{P}_{j,n+1}$.

We start by imposing the constraint:

$$x_{j,n+1} \in \mathbf{P}_{j,n+1} \tag{CT9}$$

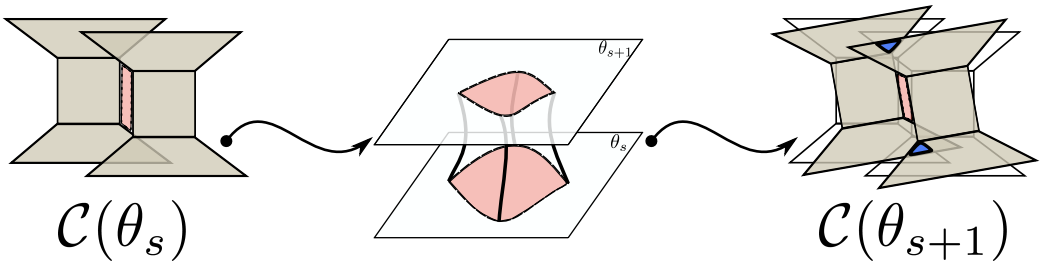


Figure 5-10: Continuous boundary variation. If the intersecting convex polygons that define the enclosing loop have a continuous intersection between adjacent slices (blue triangular regions in figure on the right), then the caged free space has a continuous boundary between slices θ_s and θ_{s+1} (red area), which will induce a compact connected component between them.

To constrain that x is also in $\mathbf{P}_{i,n}$ when transitioning from slice θ_{s+1} to slice θ_s, we explicitly construct the trajectory of x in $\mathbf{P}_{i,n}$'s coordinates given by a relation of $\Delta\theta = \theta_{s+1} - \theta_s$:

$$x_{i,n}(\Delta\theta) = x_{j,n+1} + \boldsymbol{R}(\Delta\theta)^T(c_{j,n+1} - c_{i,n})$$

where $\boldsymbol{R}(\Delta\theta)$ is a planar positive rotation matrix of angle $\Delta\theta$, and $c_{i,n}, c_{j,n+1}$ are the origins or centers of rotation of $\mathbf{P}_{i,n}$ and $\mathbf{P}_{j,n+1}$ respectively.

The trajectory $x_{i,n}(\theta)$ describes a circular arc with center $c_{j,n+1}$ and radius $|x_{j,n+1} - c_{i,n+1}|$, as illustrated in Figure 5-11. Constraining the trajectory of x to stay inside the polygon $\mathbf{P}_{i,n}$ is a non-convex constraint because of the curvature of the arc. Instead we impose the stricter constraint that a triangle enclosing the arc must lie inside the polygon $\mathbf{P}_{i,n}$. We construct the arc-enclosing triangle with the two ends of the arc $(x_{i,n}(\theta_s), x_{i,n}(\theta_{s+1}))$ and the point x_v at the intersection of the two arc tangents at those two ends. A sufficient condition for the triangle and the arc to be inside the polygon $\mathbf{P}_{i,n}$ is that all three vertices $\{x_{i,n}(\theta_s), x_{i,n}(\theta_{s+1}), x_v\}$ are inside the polygon. See this construction in Figure 5-11 (bottom).

Note that we can construct x_v as a linear function of decision variables:

$$x_v = x_{i,n}(\theta_s) + \left(\mathbf{I}_{2\times 2} + \boldsymbol{R}_{-90^\circ} \tan\frac{\Delta\theta}{2}\right)(c_{j,n+1} - c_{i,n}) \qquad \text{(CT10)}$$

with $\Delta\theta = \theta_{s+1} - \theta_s$.

We activate these regularity constraints only for those polygons that define the enclosing loops with the following constraints:

$$H_n(i,j)(\theta_s) \quad \Rightarrow \quad x_{i,n}(\theta_s), x_{i,n}(\theta_{s+1}), x_v \in \mathbf{P}_{i,n} \tag{CT11}$$

Through constraints (CT9), (CT10) and (CT11) we ensure that the intersecting polygons of $\mathcal{C}(\theta)$ remain connected for all orientations between $\mathcal{C}(\theta_s)$ and $\mathcal{C}(\theta_{s+1})$. Hence, the free space component boundary only expands or contracts continuously when rotating between slices.

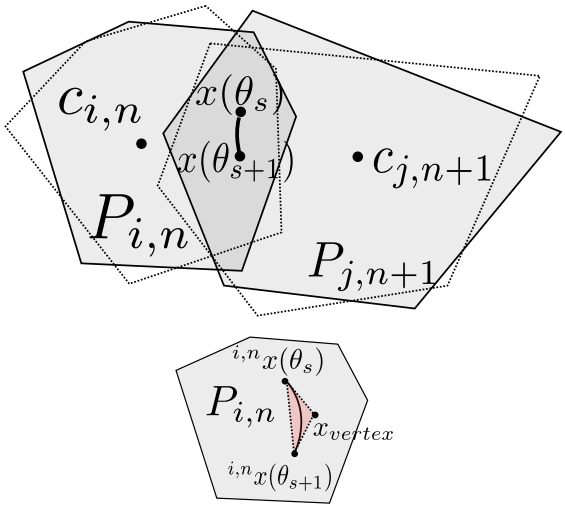


Figure 5-11: Loop closure continuity between $\mathcal{C}$-slices θ_s and θ_{s+1}. We require that the convex polygons defining the C-obstacles intersect at a point x during the rotation between slices. (top) Example polygons defining the C-obstacles rotate from θ_s (darker shaded grey) to θ_{s+1} (lighter shaded grey). Point x lies at the intersection of both polygons at both slices, and its trajectory induced by the rotation between θ_s and θ_{s+1} is a circular arc. (bottom) The arc, observed from $P_{i,n}$'s coordinates. It is contained within the triangle formed by vertices $\{x_{i,n}(\theta_s), x_{i,n}(\theta_{s+1}), x_v\}$.

5.1.6 Theoretical properties of our approach

This section derives a proof of correctness for the solutions obtained from this model.

We want to demonstrate that all manipulator configurations found by our algorithm are guaranteed to cage the object in its configuration q. The proposed formulation relies on a discretization of the configuration space into slices. We now demonstrate that the enclosing loop does not break between two consecutive slices. Formally, the following theorem describes the sufficient conditions that are the basis for constructing cages in Section 5.1.4 and Section 5.1.5.

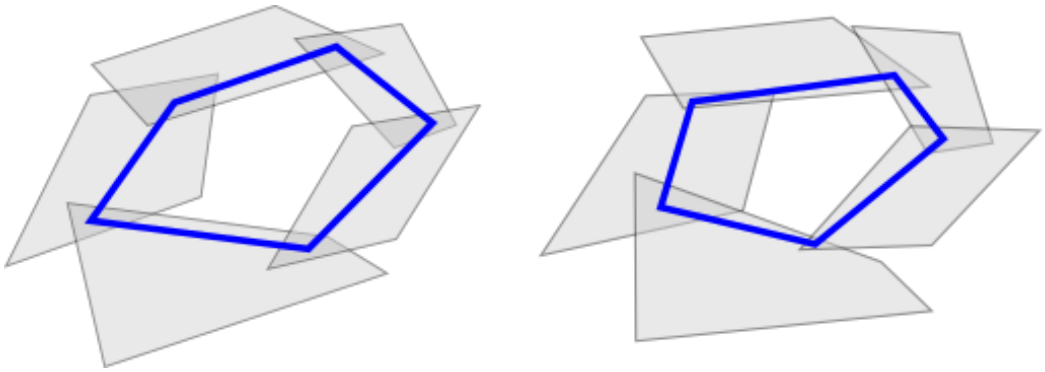


Figure 5-12: The set of intersections between the obstacle polygons remains the same for consecutive slices; there is an enclosing loop that is preserved between them.

Intuitively, we would like to demonstrate that, given a caging configuration q that is caged under translations in a orientations θ_s and θ_{s+1}, it remains caged for all orientations from the respective interval. Formally, we want to prove the following statement:

Theorem 1. *Consider two consecutive slices corresponding to orientations θ_s and θ_{s+1}, and assume that the following conditions are satisfied (see Fig. 5-12):*

1. *There are enclosing loops $\gamma(\theta_s)$ and $\gamma(\theta_{s+1})$ that enclose the configuration of the object q within the C-obstacles defined in C-slices θ_s and θ_{s+1} respectively, and these are formed by the same polygonal intersections;*

2. *The set of intersections between the obstacle polygons remain the same for all orientations $\theta \in [\theta_s, \theta_{s+1}]$;*

If the object's movements are restricted to arbitrary rigid translations in $\mathbb{R}^2$ and rotations between orientations θ_s and θ_{s+1}. Then, any point x such that $x \in \operatorname{int}\gamma(\theta_s)$

and $x \in \operatorname{int}\gamma(\theta_{s+1})$*, is caged in* $\mathbb{R}^2 \times [\theta_s, \theta_{s+1}]$.

To derive the final statement, we will need to make several preliminary observations. First, notice that the enclosing loop is always located inside the obstacles and there is a lower bound on its distance from their boundary. Let $\alpha(\theta)$ denote the distance from $\gamma(\theta)$ to the boundary of the obstacles. Formally, we prove the following statement:

Lemma 2. *There exists* $\alpha_{min} > 0$ *such that* $\alpha(\theta) \geq \alpha_{min}$ *for any* $\theta \in [\theta_s, \theta_{s+1}]$.

Proof. Let $\gamma_i(\theta)$ be a segment of the polygonal enclosing loop $\gamma(\theta)$, corresponding to an obstacle polygon $P_i(\theta)$. Then, $\alpha(\theta)$ can be expressed as the minimum distance from a segment to the respective polygon:

$$\alpha(\theta) = \min_i d(\gamma_i(\theta), \partial P_i(\theta))$$

Let us consider a polygon $P_i(\theta)$, the respective segment of the enclosing loop $\gamma_i(\theta)$, and its endpoints $x_{\gamma_i,1}(\theta)$ and $x_{\gamma_i,2}(\theta)$. On the one hand, according to the condition (CT9), one of the endpoints $x_{\gamma_i,1}$ is fixed in the internal coordinates of P_i, therefore its distance to the $\partial P_i(\theta)$ does not depend on θ and is equal to some constant $a_i^{(1)} = d(x_{\gamma_i,1}(\theta_s), \partial P_i(\theta_s)) > 0$. On the other hand, constraint (CT11) ensures that another endpoint $x_{\gamma_i,2}$ is restricted to a triangle T_i, lying inside P_i (in its internal coordinates). Therefore, $d(x_{\gamma_i,2}(\theta), \partial P_i(\theta)) \geq d(T_i, \partial P_i) > 0$. As T_i is fixed in the internal coordinates of P_i, $a_i^{(2)} = d(T_i, \partial P_i)$ is independent on θ, and is determined by the choice of $x_{\gamma_i,2}(\theta_s)$ and $x_{\gamma_i,2}(\theta_{s+1})$. Since P_i is convex, and $\gamma_i(\theta)$ is a segment inside it, $d(\gamma_i(\theta), \partial P_i(\theta)) = min(d(x_{\gamma_i,1}(\theta), \partial P_i(\theta)), d(x_{\gamma_i,2}(\theta), \partial P_i(\theta))) \geq min(a_i^{(1)}, a_i^{(2)})$. Let $\alpha_i = min(a_i^{(1)}, a_i^{(2)}) > 0$. Then $\alpha(\theta) = \min_i d(\gamma_i(\theta), \partial P_i(\theta)) = \min_i \min(a_i^{(1)}, a_i^{(2)}) = \min_i \alpha_i > 0$. By denoting $\alpha_{min} = \min_i \alpha_i$ and using independence of α_i on θ, we conclude the proof. For a visual depiction, see Fig. 5-13. □

Now, this observation allows us to show that in any orientation θ, any point from the free space, and in particular from the compact-connected component that defines the cage, is separated from the corresponding enclosing loop $\gamma(\theta)$ at least by α_{min}.

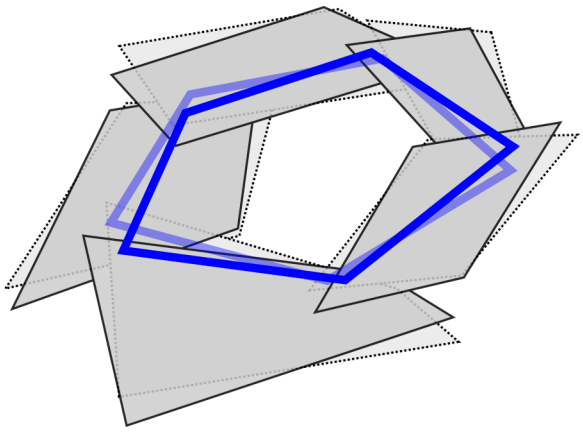


Figure 5-13: For slices that are close to each other, the respective enclosing loops are close to each other and homotopic in the intersection of the $\mathcal{C}-$obstacles of the slices.

Lemma 3. *For any point x, lying in the $\mathcal{C}_{\text{free}}^{\text{compact}}(\theta)$, the distance $d(x, \gamma(\theta))$ is greater than α_{min}.*

Proof. Consider a segment $\gamma_i(\theta)$, such that $d(x, \gamma(\theta)) = d(x, \gamma_i(\theta))$, and a corresponding obstacle polygon $P_i(\theta)$. Let the $x_{\gamma_i(\theta)}$ be a point on this segment, where $d(x, \gamma_i(\theta)) = d(x, x_{\gamma_i(\theta)})$, and s be a segment, connecting x and $x_{\gamma_i(\theta)}$. Let $x_\partial = s \cap \partial P_i(\theta)$ (such x_∂ exists, as x lies outside $P_i(\theta)$, and $x_{\gamma_i(\theta)}$ lies inside $P_i(\theta)$ due to the convexity of $P_i(\theta)$). Then $d(x, \gamma(\theta)) = d(x, x_{\gamma_i(\theta)}) > d(x_\partial, x_{\gamma_i(\theta)}) \geq d(\partial P_i(\theta), \gamma_i(\theta))$. According to the Lemma 2, $d(\partial P_i(\theta), \gamma_i(\theta)) \geq \alpha_{min}$. □

Now, let us define the *radius* of a polygon as the distance from its rotation center c to the furthest point in the polygon:

$$\boldsymbol{R}(P) = \max_{x \in P} d(c, x)$$

For the set of polygons $\bigcup_i P_i$ forming the enclosing loop, let R denote the maximum radius: $R = \max_i \boldsymbol{R}(P_i)$, where c_i is a rotation center of P_i.

Furthermore, let us define the *displacement* of a point $x \in P$ induced by a rotation of the polygon P through an angle $\Delta\theta$ around the rotation center c by $\mathrm{D}_{\Delta\theta}(x) = 2\sin(\frac{\Delta\theta}{2}) \cdot d(c, x)$. Then, to make sure that the displacement of any point $x \in \bigcup_i P_i$ does not exceed ε, we need to limit the rotation angle to $\Delta\theta(\varepsilon) = 2\arcsin\frac{\varepsilon}{2R}$.

We now want to show that the movement of the enclosing loop is bounded, and the allowed amount of displacement ε cannot be exceeded if the orientation interval is sufficiently small:

Lemma 4. *Consider $\theta \in [\theta_s, \theta_{s+1}]$, the corresponding enclosing loop $\gamma(\theta)$, and an arbitrary point x. Let us fix $\varepsilon \in (0, 2R)$. For any $\theta' \in \mathcal{D}_{\Delta\theta(\varepsilon)}(\theta) \cap [\theta_s, \theta_{s+1}]$ and the corresponding enclosing loop $\gamma(\theta')$, we have $d(x, \gamma(\theta')) \geq d(x, \gamma(\theta)) - \varepsilon$.*

Proof. First, note that the displacement of any point x in $\bigcup P_i$ does not exceed ε when rotated through an angle $\Delta\theta$:

$$\mathrm{D}_{\Delta\theta}(x) = 2d(x, c)\sin(\Delta\theta/2) = \varepsilon\frac{d(x, c)}{R} \leq \varepsilon.$$

Let us show that $d(x, \gamma(\theta')) \geq d(x, \gamma(\theta)) - \varepsilon$. Suppose the opposite is true, i.e. $d(x, \gamma(\theta')) < d(x, \gamma(\theta)) - \varepsilon$. Consider the point $x_{\gamma(\theta')} \in \gamma(\theta')$, where $d(x, \gamma(\theta')) = d(x, x_{\gamma(\theta')})$, and its image $x'_{\gamma(\theta')} \in \gamma(\theta)$. Then $d(x, x_{\gamma(\theta')}) < d(x, \gamma(\theta)) - \varepsilon \leq d(x, x'_{\gamma(\theta')}) - \varepsilon \leq 0$, leading to contradiction. □

Finally, we can demonstrate the correctness of our approach by proving Theorem 1:

Proof. Suppose the opposite is true, and there is a continuous escaping path $\rho : [0, 1] \to \mathbb{R}^2 \times [\theta_s, \theta_{s+1}]$, such that:

$$\rho(0) = \{x_0, \theta_s\}, x_0 \in \operatorname{int}\gamma(\theta_s)$$

$$\rho(1) = \{x', \theta'\}, x' \notin \operatorname{int}\gamma(\theta'),$$

where $\theta' \in (\theta_s, \theta_{s+1}]$.

Since x_0 lies in the interior of $\gamma(\theta_s)$, for any $\delta > 0$ there exists some $\lambda \in [0, 1 - \delta]$ such that $\rho_\lambda = \{x_\lambda, \theta_\lambda\}$, and $x_\lambda \in \mathcal{C}_{\text{free}}(\theta_\lambda) \cap \operatorname{int}\gamma(\theta_\lambda)$, while $x_{\lambda+\delta} \notin \mathcal{C}_{\text{free}}(\theta_{\lambda+\delta}) \cap \operatorname{int}\gamma(\theta_{\lambda+\delta})$. As ρ is continuous in $[0, 1]$, its corresponding coordinates x and θ are also continuous in $[0, 1]$. Let us fix α_{min}, according to the definition of Lemma 2, and R as described in the proof of Lemma 4. Using the continuity, let us choose δ such that for

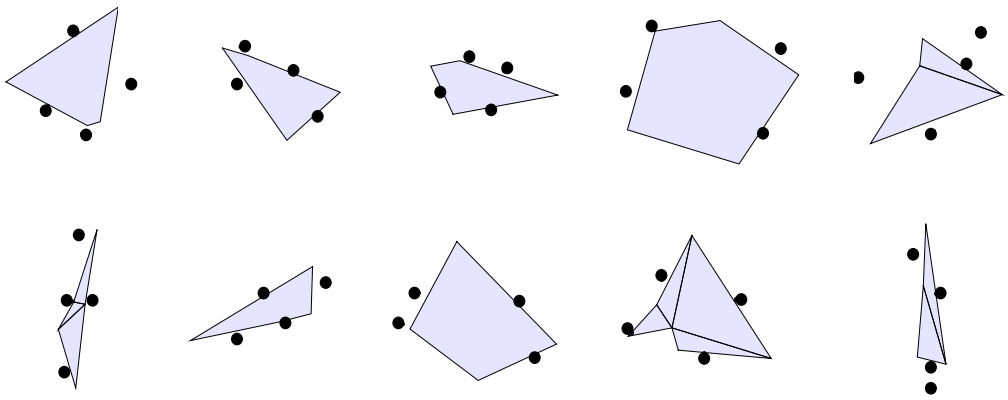


Figure 5-14: Cage synthesis for random polygonal objects, decomposed into the union of convex polygonal objects. In this experiment, the proposed optimization framework searches for possible cages of random polygons with up to four fingers.

corresponding λ $\Delta x = d(x_\lambda, x_{\lambda+\delta}) < \alpha_{min}/2$ and $\Delta\theta = |\theta_{\lambda+\delta} - \theta_\lambda| < 2\arcsin\frac{\alpha_{min}/2}{2R}$. According to the definition of α_{min} in Lemma 2, for all θ and i, $\alpha_{min} \leq d(\gamma_i(\theta), \partial P_i(\theta))$. In particular, $\alpha_{min} \leq d(\gamma_i(\theta_s), \partial P_i(\theta_s)) \leq d(\gamma_i(\theta_s), c_i) + d(\partial P_i(\theta), c_i) < 2R$ due to the triangle inequality.

Let Ω_1 and Ω_2 be two disks with the center in x_λ and radii α_{min} and $\alpha_{min}/2$, respectively. According to Lemma 3, $\Omega_1 \in \operatorname{int}\gamma(\theta_\lambda)$. Applying Lemma 4 to $\varepsilon = \alpha_{min}/2 < 2R$, we obtain $\Omega_2 \in \operatorname{int}\gamma(\theta_{\lambda+\delta})$. On the other hand, $d(x_\lambda, x_{\lambda+\delta}) < \alpha_{min}/2$, therefore, $x_{\lambda+\delta} \in \Omega_2 \in \operatorname{int}\gamma_\theta(\lambda+\delta)$, leading to contradiction. □ □

5.1.7 Implementation and Results

In this section we implement an optimization-based cage-finding algorithm based on the proposed model. We also evaluate the performance and versatility of this formulation by synthesizing cages for different planar geometries, and under different sets of extra constraints.

Formulation of Caging as an Optimization Problem

Given an object segmented in M polygons, a manipulator with N fingers and S sample slices of the configuration space $\mathcal{C}$, we formulate the cage-finding algorithm as

Variable	Dimension	Role	C/B
Θ	S	Limit orientations	B
H	$N \times M^2 \times S$	Loop creation	B
G	$N \times M^2 \times S$	Loop creation	B
R	$N \times R$	Non-Penetration	B
F	$N \times M \times 5$	Object enclosing	B
T	$N \times L \times S$	Limit orientations	B
$\mathbf{p}$	$2 \times N$	Finger positions	C
x	$2 \times N \times S$	Boundary continuity	C

Table 5.2: Summary of Decision Variables in **MIP1** (B: Binary, C: Continuous)

the feasibility problem:

$$\mathbf{MIP1}: \min_{\Theta,H,G,R,T,p} \quad J(\Theta, H, G, R, T, p)$$

subject to

- For all S slices:
  - Existence of a loop (CT1)—(CT4).
  - Non-penetration (CT6).
  - Limit orientation constraints (CT7)—(CT8).
  - Continuous Boundary Variation (CT9)—(CT11).
- For slice containing $\theta_s = q_\theta$:
  - Configuration enclosing (CT5).

Where $J(\cdot)$ is a convex cost function. All constraints in problem **MIP1** are convex, and the decision variables are either real or binary, hence it is a Mixed-Integer Convex Program. A summary of the variables involved in **MIP1** is presented in Table 5.2.

This provides useful properties and guarantees:

- First, given sufficient time, a solver will always find the global optimal solution, providing a convergence guarantee with worst-case exponential complexity. We evaluate in Section 5.1.7 the performance of the algorithm caging random polygons.

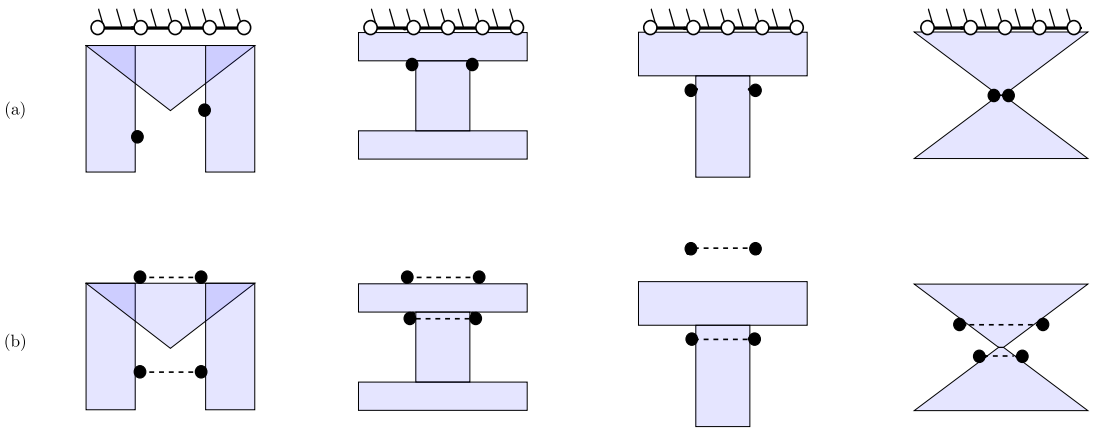


Figure 5-15: Cage synthesis with additional kinematic constraints: (a) Cage using two fingers and a fixed environment, in this case a wall segment modelled as a set of non-movable fingers (b) Caging with four fingers that are subject to kinematic constraints, in this case the constraints induced by these points being the fingers of two parallel jaw grippers, i.e, limited opening and moving parallel to each other.

- Second, the formulation is versatile since we can add additional mixed-integer convex constraints without sacrificing its properties. In Section 5.1.7 we apply the formulation to find cages exploiting an environmental wall as extra constraint, and in Section 5.1.7 we explore different convex cost functions to minimize or maximize the separation between fingers.

Model Validation

To evaluate the cage synthesis algorithm, we transcribe and solve the optimization problem using off-the-self optimization software. We use Gurobi 8.0.0 [59] as our MIP solver. All the tests are performed in MATLAB R2018b, on a Intel Core i9 laptop running Mac OS X High Sierra. For all tests, we set the parameter S to 9 slices, evenly distributed in the range between $-90°$ and $90°$. As a proof of concept, Figure 5-16 shows an example of a cage found on a classically complex object extracted from Figure 6 in [111].

Caging random polygons We generate a set of 20 random polygonal shapes by sampling vertices within a circular area and connecting them to form the object facets,

which then we segment into convex polygons with Delaunay triangulation [57]. We then call on the optimizer to find a cage for a manipulator with up to four fingers, under the cost function:

$$J = 0$$

which leads the solver to return the first feasible cage found. Figure 5-14 shows the results on 10 of these polygons. We see that the model efficiently finds a cage, handling non-convex and non-symmetric shapes.

We note however that when we ask the optimizer to find two- or three-finger cages, the optimizer sometimes reports infeasibility because the uniform slicing of the configuration space does not contain any limit orientation. This can be alleviated by using heuristics to more adequately slice the configuration space by possibly taking into account symmetries in the shape of the object and the angular separation between facets. However, it remains an open question to find optimal slicing strategies. When caging with four or more fingers, a cage is almost always found with uniform slicing because limit orientations are more abundant.

Caging with kinematic constraints The optimization nature of the formulation allows us to include additional constraints in the caging problem. For this, we study two cases of interest: (1) Caging with two fingers against a fixed-environment (wall); and (2) caging with the kinematic constraints of two grippers each with two fingers of limited opening. To incorporate the wall in case (1), we distribute 5 point-fingers with fixed positions along the wall and allow the two robot fingers to move freely. Note that other methods could also be used to model the wall, such as computing the $\mathcal{C}-$obstacle associated to a line segment and directly including it as part of the loop. We choose the model as a set of points for simplicity and consistency with the current implementation. To model the kinematics of (2) we use the dimensions of an ABB YuMi robot and constraint the distance between pairs of fingers to the maximum and minimum opening of the grippers. For simplicity, and to avoid having to deal with collision checking between arms and between the gripper, which leads to a non-convex constraint, we also enforce that the two grippers can only translate

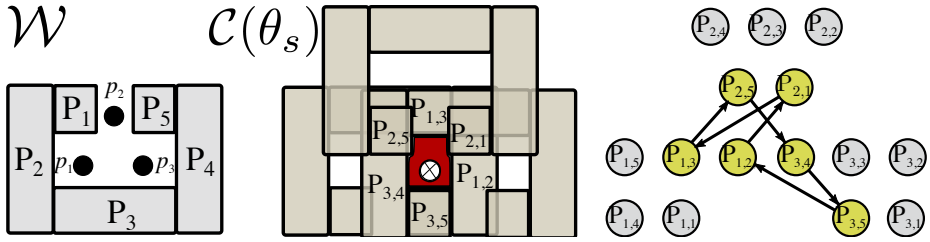


Figure 5-16: Example of an object difficult to cage extracted from Figure 6 in [111]. The figure shows (left) an example cage found with the algorithm **MIP1**, (center) as well as its representation in a $\mathcal{C}$-slice of the configuration space with orientation $\theta = 0$, and (right) the enclosing loop in the connection graph encoded by the binary matrices H_n and G_n.

parallel to each other but not rotate.

Figure 5-15 shows cages for four different polygonal objects under the above constraints. In contrast to traditional caging approaches, where these conditions would be difficult to include into the algorithm, we need to only include additional constraints that describe the kinematics of the manipulator $\mathcal{M}$ into the model. It is interesting to note that by adding extra constraints the optimization time is often reduced. The reason for this, maybe contradictory, speed-up is that, in a convex mixed-integer program, additional constraints reduce the search space by allowing the solver to discard infeasible branches. Computational complexity is largely affected by the number of variables rather than by the number of constraints.

Caging with different cost functions The convexity of the formulation makes it possible for the algorithm to find global optimal solutions when given a convex cost-function. In this section, we explore the effect of adding two different cost functions:

- **Minimize separation** (for a set of unconstrained fingers): We define the cost function

$$J_1(\mathcal{M}) = \sum_{i \neq j} (\mathbf{p}_i - \mathbf{p}_j)^2,$$

  which is a convex function. Such cost-function penalizes the separation between each pair of fingers.

- **Maximize separation** (for a one-parameter coupled 3-finger gripper): In general, maximum distance is in a non-convex cost-function; However, it can be convex for some particular kinematic relations. One useful case is when the finger configurations are coupled by one-parameter, similar to [94]. Hence, In this experiment we define parametrizations for *two-fingers*:

$$\mathbf{p}_1 = \begin{pmatrix} \mathbf{p}_{1,x} \\ \mathbf{p}_{1,y} \end{pmatrix}, \ \mathbf{p}_2 \begin{pmatrix} \mathbf{p}_{1,x} + \alpha \\ \mathbf{p}_{2,y} \end{pmatrix},$$

and *three-fingers*:

$$\mathbf{p}_1 = \begin{pmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ \mathbf{p}_2 = \begin{pmatrix} \mathbf{p}_x \\ \mathbf{p}_y \end{pmatrix} + \alpha \begin{pmatrix} -1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix},$$

$$\mathbf{p}_3 = \alpha \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix},$$

with $\alpha \in [0, \infty)$. In such case, we can maximize the gripper opening by minimizing the cost-function

$$J_2(\alpha) = -\alpha$$

which is convex.

We apply the cage-synthesis model to find cages to the same objects in Figure 5-15 under a variety of cost functions.

Figure 5-17 shows different solutions to our model under the loss functions J_1 and J_2. In both cases, we exploit convenient parametrizations of the robot kinematics, which result in linear constraints. However, finding these parametrizations might not always be possible. For more complex kinematic chains, these could be integrated in the model using using piece-wise affine constraints [36].

A key remark from these results is the impact that our approximations play in the model, in particular those from (CT5). Since (CT5) is used to ensure that the object lies inside of the compact-connected component, having a tighter constraint

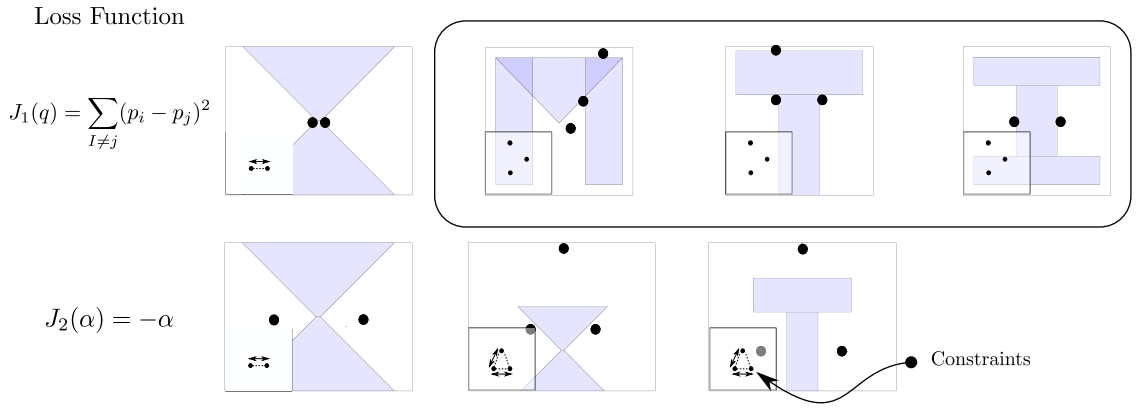


Figure 5-17: Synthesis of cages with different convex cost functions. **Top**: finding minimum and maximum distance cages for a parallel jaw gripper (left) and an unconstrained 3-finger gripper (right). **Bottom**: finding maximum cages for a parallel jaw gripper and a one-parameter 3-finger gripper. Since all the cost-functions are convex, the solutions we obtain are globally optimal (to the tolerance of our approximations and slicing).

also constrains the set of cages that satisfy it. This causes that the solution found by our model to be different from the theoretical minimum. An example can be seen in Fig. 5-17 (M-object), here the third finger is placed separate from the other two fingers, while the theoretical minimum would have two of the fingers in the same position.

Limitations of the model

The properties of this model come at the expense of some model limitations. First, the discretization of the configuration space into slices makes the model sensitive to the selection of orientations used for finding a cage. Second, the enclosing constraints (Section 5.1.4) are sufficient but not necessary, since the configuration $[q_x, q_y]$ is forced to lie in one of the 4 aligned regions covering each line segment, that constraint is sometimes unnecessarily infeasible (this can be seen often for example when trying to cage a triangle with 3 fingers). Finally, we build our implementation of the model assuming that all fingers are part of the enclosing loop and, hence, cannot search for cages with a minimal number of fingers.

Potential extensions

We are interested in making the model more flexible and efficient. First, by studying how object shape information can be leveraged to derive heuristics to automatically slice the configuration space. In general, it is important to further explore opportunities to reduce the complexity of the model. One avenue is to exploit the property that adding constraints often helps the solver quickly discard large branches of the search space. From a theoretical perspective, while the conditions in this chapter are sufficient to guarantee a cage, in particular those derived from Theorem 1, we suspect that these could also become necessary and sufficient through a dense enough slicing, for any cage of tye type described in Fig. 3 (a) an (b). Hence, the potential *"resolution completeness"* of the model deserves further study.

Source Code

The MATLAB source code implementing the caging problem in used as part of this work is available on GitHub: https://github.com/baceituno/Grasping

5.2 Certified Grasping

The key question we study in this section is that of robustness in the process of grasping an object. Can we ever certify that a planned grasp will work?

The common approach to grasping is to plan an arrangement of contacts on the surface of an object. Experimental evidence shows an intuitive but also paradoxical observation: On one hand, most grasps do not work as expected since fingers do not deliver exactly the planned arrangement of contacts; on the other hand, many planned grasps still end up working and produce a stable hold of the object.

These natural dynamics work within all grasping algorithms, often to their benefit, sometimes adversarially. [94] put it as: if we cannot put the fingers in the right place, can we trust the fingers to *fall where they may*? In this section we study the possibility to synthesize grasps for which the fingers have no other option than to do so.

The notions of robustness and certification are central to the robotics community. However, formal approaches to synthesize robustness in grasping have been mostly limited to study the set of forces that a grasp can resist, see [23], neglecting the key importance of the reaching motion towards that grasp.

Both the reaching motion and the end-grasp can encode robustness. In this section we study the problem of synthesizing trajectories of a set of point fingers that converge onto an intended grasp of a polygonal planar object without friction, naturally encoding robustness to uncertainty as part of the grasping process.

We start by proposing three different types of certificates that one can formulate at different stages of the grasping process, depicted in Figs. 5-1 and 5-19:

- **Invariance Certificate:** At the beginning of the grasping process, the object lies in an invariant set of its configuration space. In this section we study the case when the object is geometrically trapped by fingers around it, i.e., the object is caged by the fingers.

- **Convergence Certificate:** All configurations in the invariant set are driven towards a given end-grasp configuration. Intuitively, this is analogous to driving down the value of a scalar/energy function with only one minimum.

- **Observability Certificate:** The configuration of the object in the end-grasp is identifiable with the robot's contact or proprioceptive sensors after completing the grasp. In this work we characterize when the location of fingers is enough to recover the pose of the object, for which the condition is analogous to first-order form closure.

Sections 5.2.2, 5.2.3, and 5.2.4 derive a model for a particular formulation of each of these certificates. We achieve this under the assumption that objects are polygons, robot fingers are points, and that there is no friction between objects. These models build on tools from convex-combinatorial optimization that decompose

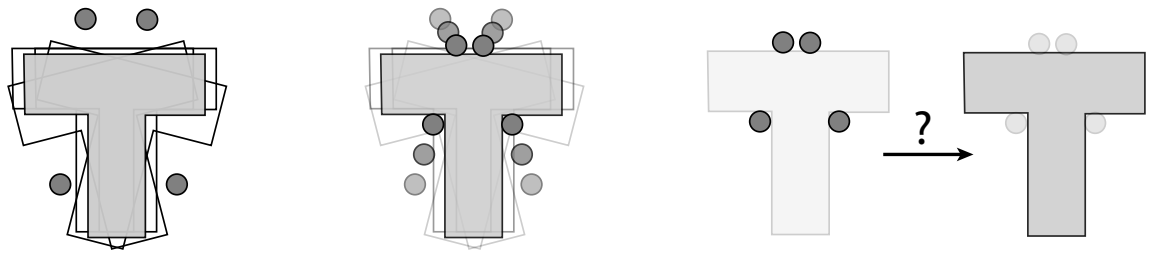


Figure 5-18: Overview of frictionless grasping with certificates. From a configuration space perspective, we say a grasp is certified to succeed when: 1) The robot bounds the object pose within an invariant set, and 2) the free-space converges to a single configuration. From this initial bound, we obtain an invariant set of configurations for which the grasp will always succeed. A third certificate, also valid for non-converging grasping processes, comes from requiring that the end-grasp configuration is observable.

the configuration space of an object surrounded by fingers into free regions, and is based on recent work to formulate the caging synthesis problem as an optimization problem, see [1]. Section 5.2.2 summarizes the approach.

The combination of the models for each of the three certificates yields a global geometric model to synthesize grasping motions that reach certifiable grasps. Section 5.2.5 describes the application of this model to robust grasping of planar polygons, and provides experimental evidence of the value of the approach by a direct comparison between certified grasping and force-closure grasping.

The formulation we provide in this section for each of the proposed certificates presents limitations–and opportunities for future work–which we detail in Chapter 7. Most notably, the presented formulation is purely geometrical, and does not take into account friction uncertainty, which can yield undesired behaviors between fingers and object such as jamming and wedging.

5.2.1 Problem description

The problem of interest in this section is that of finding a grasping motion that is certified to succeed. Formally, we define this problem as:

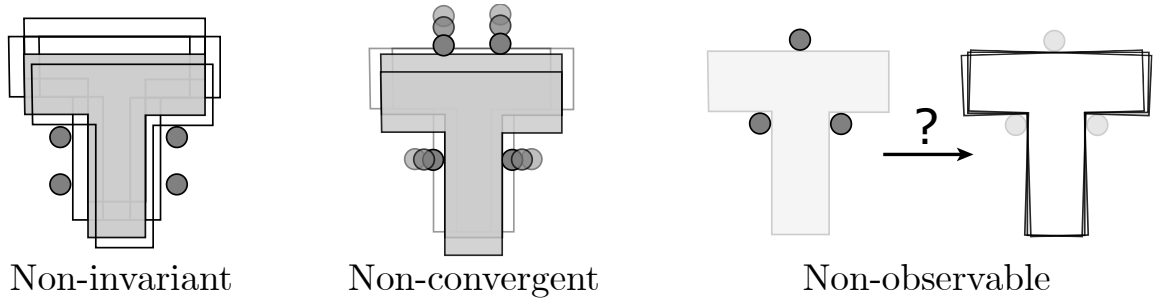


Figure 5-19: Examples of grasps that break each of the certificates. We show three grasps that break each of the individual certificates: 1) The object configurations does not lie on an invariant set, the object can escape by moving up, 2) The object configurations do not converge to a unique configuration, 3) It is not possible to identify a single pose of the object by using proprioceptive sensing.

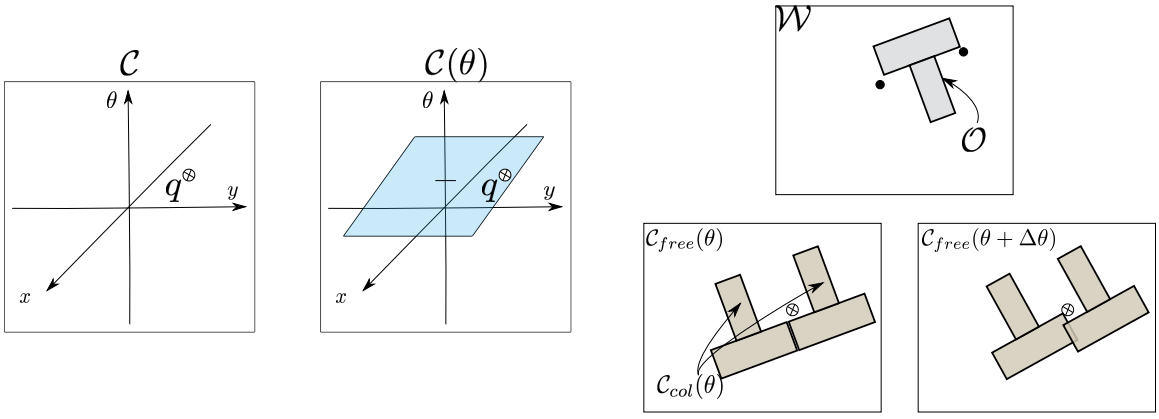


Figure 5-20: Key concepts used to develop our framework. A visual representation of the following concepts: 1) a configuration space $\mathcal{C}$, 2) a $\mathcal{C}-$slice , and 3) the mapping between workspace $\mathcal{W}$, $\mathcal{C}(\theta)$ and $\mathcal{C}(\theta + \Delta\theta)$.

Problem 1 (Certified Grasping): Given an object $\mathcal{O}$, a manipulator $\mathcal{M}$, S samples of $\mathcal{C}-$slices, and a goal object configuration $\boldsymbol{q}$, find a manipulator trajectory $\rho_{\mathcal{M}} = \{\mathcal{M}(t) \mid t \in \{1, \ldots, N_T\}\}$ and an invariant set $Q_0 \subset \mathcal{C}(\mathcal{O})$, such that $\rho_{\mathcal{M}}$ will drive any configuration of the object $\hat{\boldsymbol{q}} \in Q_0$ towards an observable grasp on $\boldsymbol{q}$.

This problem can be seen as a particular case of the general problem known as LMT [85], and as a generalization of Goldberg's squeezing plans [58] for non-convex objects and point-finger contacts. For an object on a plane without friction, a solution to this problem results from implementing the certificates described in the previous section as a three-step process (discretized as a manipulator trajectory of N_T time-

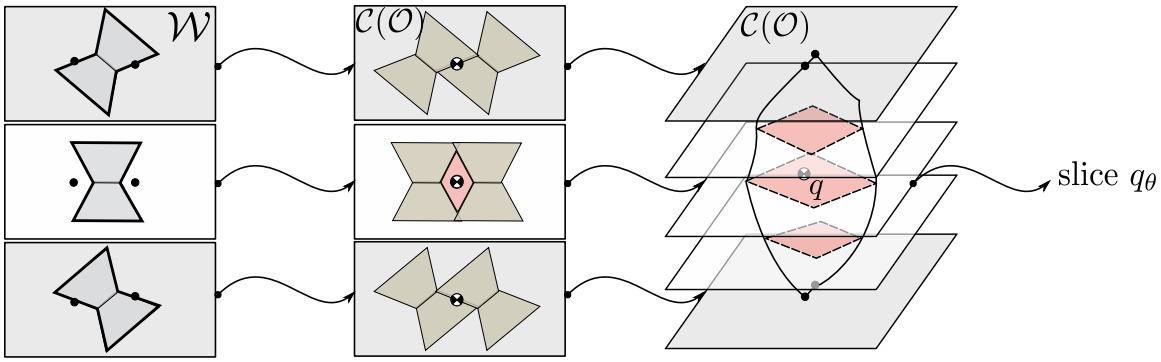


Figure 5-21: Invariance Certificate. Example of a cage in $\mathcal{W}$ (left), the $\mathcal{C}-$slices (center) and the configuration space $\mathcal{C}(\mathcal{O})$ (right). Note how the configuration $\boldsymbol{q}$ lies in a compact connected-component of the free-space (pink), bounded by two limit orientations (gray). Image adapted from [1].

steps):

- Invariance: The configuration of the object $\boldsymbol{q}$ lies in a compact-connected component of its free-space. We will impose this condition at $t = 1$ with the convex-combinatorial model of caging from [1].

- Convergence: The manipulator path drives all configurations in the initial invariant set (cage) towards the goal $\boldsymbol{q}$. To meet this condition, once the object is caged ($t \in \{2, \ldots, N_T\}$), the manipulator follows a penetration-free path over which the compact-connected component contracts. Then, at the final time-step of the path, the $\mathcal{C}$-obstacles reduce $\mathcal{C}_{free}^{compact}(\mathcal{O}, N_T)$ to a singleton $\{\boldsymbol{q}\}$.

- Observability: As a consequence of the fingers motion, the final contact configuration can recover the object pose at $\boldsymbol{q}$ through proprioceptive sensing. We call such a configuration an *observable grasp* and is a condition solely required at the end of the path ($t = N_T$).

The satisfaction of these constraints would give a geometric certificate that any configuration of the object in the set $Q_0 = \mathcal{C}_{free}^{compact}(\mathcal{O}, t = 1)$ will be driven towards and immobilized in the goal grasp. The following three sections provide a model for each of these three steps, which then will be combined into an optimization problem (**MIQP1**) for certified grasping of polygonal objects.

5.2.2 Invariance Certificate

As explained above, one way to constrain an object to an invariant set is to cage it geometrically. Under the model presented in [1], the following are a set of sufficient conditions for invariance:

1. The component $\mathcal{C}_{free}^{compact}(\mathcal{O})$ is bounded in the orientation coordinate by two limit orientations.

2. At all $\mathcal{C}-$slices between the two limit orientations there is a loop of $\mathcal{C}-$obstacles enclosing a segment of free-space. Each $\mathcal{C}-$obstacle has the shape of the object rotated according to the limit orientation. All these loops must be connected, enclosing a component of free-space in between adjacent slices. At the slice with q_θ, the loop must enclose $\boldsymbol{q}$ (as illustrated in the middle column of Fig. 5-21).

3. At the $\mathcal{C}-$slice of a limit orientation (if these exist) the free-space component enclosed by the loop has zero area. Thus, becoming a line segment or a point.

The union of these conditions define a set of constraints to enclose the configuration q, as illustrated in Fig. 5-21. We can transcribe these conditions as a convex-combinatorial model composed of two sets of constraints, described below, and explained in more detail in the previous section.

Creating loops at each $\mathcal{C}$-slice

To construct a loop of $\mathcal{C}-$obstacles at each slice, we formulate the problem as that of finding a closed directed graph within the intersections between polygonal obstacles. In such graph, each node represents a convex polygon of the decomposition of a $\mathcal{C}-$obstacle, while each edge imposes an intersection between polygons. We denote the polygon i of $\mathcal{C}-$obstacle n as $\boldsymbol{P}_{n,i}$. Including this condition in the model, at each time t, is done through the following constraints:

Existence of a Loop. This is encoded through two binary matrices: $\boldsymbol{H}_n \in \{0,1\}^{M\times M}$ and $\boldsymbol{G}_n \in \{0,1\}^{M\times M}$. $\boldsymbol{H}_n$ encodes edges between $\mathcal{C}-$obstacle n and $\mathcal{C}-$obstacle $n+1$, such that $\boldsymbol{H}_n(i,j) = 1 \Rightarrow \boldsymbol{P}_{n,i} \cap \boldsymbol{P}_{n+1,j} \neq \emptyset$. $\boldsymbol{G}_n$ encodes edges within $\mathcal{C}-$obstacle n, such that $\boldsymbol{G}_n(i,j) = 1 \Rightarrow \boldsymbol{P}_{n,i} \cap \boldsymbol{P}_{n,j} \neq \emptyset$. These matrices are constrained so that the resulting graph is closed and directed. We show an example of this loop and its graph in Fig. 5-22 (b) and (c).

Configuration Enclosing. We include this condition by introducing a binary tensor $\mathbf{F} \in \{0,1\}^{N\times M\times 3}$, where $\mathbf{F}^{\boldsymbol{q}}(n,j) = 1$ imposes that a ray with origin $\boldsymbol{q}$ has an intersection with the line segment connecting polygon[1] j at $\mathcal{C}-$obstacle n and the next polygon in the loop (encoded by $\boldsymbol{H}_n$ and $\boldsymbol{G}_n$), while other values of k assign the ray to the complement of the segment. Following Jordan's Polygon Theorem, the constraints needed to enclose $\boldsymbol{q}$ is to impose $\sum_{(i,j)} \mathbf{F}(i,j)$ to be odd. An illustration of this condition is shown in Fig. 5-22 (d).

Non-Penetration Constraints. We impose this constraint by introducing a binary matrix $\boldsymbol{R} \in \{0,1\}^{N\times R}$ that encodes the location of finger points $\boldsymbol{p}$ in the convex decomposition of the free workspace $\mathcal{W}\setminus\mathcal{O} = \bigcup_{k=1}^{R} \mathcal{R}_k$. $\boldsymbol{R}(i,r) = 1$ assigns finger i to region r in $\mathcal{W}\setminus\mathcal{O}$, or $\boldsymbol{R}(i,r) = 1 \Rightarrow \boldsymbol{p}_i \in \mathcal{R}_r$, with $\sum_r \boldsymbol{R}(i,r) = 1, \forall i$. A visualization of this is shown in Fig. 5-22 (e).

Combining all of these constraints ensures the existence of a loop at each $\mathcal{C}-$slice and that $\boldsymbol{q}$ is enclosed by one of these loops.

Constructing a cage from loops

The next step is to impose that these constraints are only active for slices between two limit orientations (when these exist) while also enclosing a component of free-space between slices.

[1] In practice we use the line segment defined by the geometric centers of the polygons.

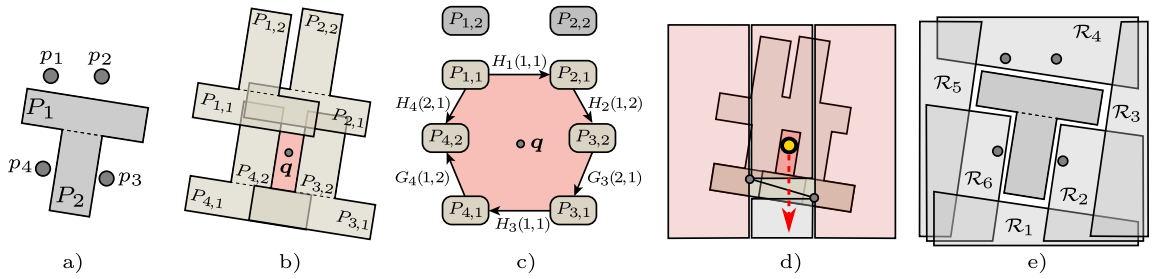


Figure 5-22: Caging Model. (a) Illustration of the cage of an object composed of two polygons ($M = 2$), caged with four fingers ($N = 4$) in a configuration space slice of constant orientation defined by six polygonal regions ($R = 6$), and with a boundary with eight edges ($L = 8$). (b) The model forms a polygonal loop at each slice of $\mathcal{C}(\mathcal{O}, t)$, (c) defining a graph of polygonal intersections that enclose $\boldsymbol{q}$. (d) We test that the configuration $\boldsymbol{q}$ is enclosed by the loop by checking a ray (red arrow) has an odd number of intersections with the loop. In this case the ray intersects the loop once, in the black line between two polygons. (e) Slightly exploded view of the (intersecting) polygonal regions that define the non-penetration space where the fingers can move.

Constraint Activation. To determine which slices must contain a closed loop of $\mathcal{C}-$obstacles, we first determine if a slice is limit orientations. To include this constraint, we introduce a binary vector $\Theta \in \{0,1\}^S$, where $\Theta_s = 1$ imposes that a limit orientation must be reached before slice s, deactivating all loop constraints in such slice. In this context, *before* means a greater or equal angle if the slice lies in the negative orientation half-space or a smaller or equal angle if it lies in the positive one. We then introduce the constraint $\sum_s^S \Theta_s = 2$.

Limit Orientations. A limit orientation occurs when the loop encloses a zero-area component, a condition defined by the contacts between the fingers and some translation of the object. At any particular $\mathcal{C}-$slice, we define this condition when the contact normals between the fingers in contact with the object span $\mathbb{R}^2$ in positive linear dependent set. To verify this limit orientation condition, we define a binary matrix $\boldsymbol{T}_s \in \{0,1\}^{N \times L}$, such that $\boldsymbol{T}_s(i,l) = 1 \Rightarrow \boldsymbol{p}_i \in \boldsymbol{L}_l$ imposes that finger i must be in contact with facet l at slice s with contact normal λ_i^s. Using this variable we pre-compute $\mathcal{L}_\mathcal{O}$ as the set of contact assignments that lead to a limit orientation, such that $\sum_c^N \alpha_c^s \lambda_c^s = 0,\ \alpha_c^s > 0\ rank(\begin{bmatrix}\lambda_1^s \cdots \lambda_N^s\end{bmatrix}) = 2$. Then, we impose $\boldsymbol{T}_s \in \mathcal{L}_\mathcal{O} \Rightarrow$

$\Theta_s = 1$, which constrains that two limit orientations must exist in each cage.

Continuous Boundary Variation. In order for $\mathcal{C}_{free}^{compact}(\mathcal{O})$ to be compact and connected, the swept value of the loop created at each $\mathcal{C}-$slices must also enclose a segment of free-space in between adjacent slices. [1] shows that a sufficient condition for this is to have the boundary of such loops to vary continuously unto the boundary of the loop in the adjacent $\mathcal{C}-$slices. Mathematically, this constraint requires 1) that the topology of the loop at each $\mathcal{C}-$slice is preserved in adjacent slices and 2) that the swept volume of each polygon in the loop remains in contact with the swept volume of its connecting polygons in the loop between slices. We integrate a set of sufficient convex constraints for this condition as part of the model .

By satisfying these conditions, we ensure that the configuration $\boldsymbol{q}$ is enclosed by a compact-connected component of free-space. For more details on implementation and proofs on the correctness of these conditions, the reader is referred to [1].

5.2.3 Convergence Certificate

Given an initial cage, the convergence certificate is satisfied if the process drives a set of bounded configurations towards the goal $\boldsymbol{q}$. The main insight that allows us to integrate this stage in the framework comes from the following remark:

Remark 1: Given an object $\mathcal{O}$, at some time-step t, with a configuration $\boldsymbol{q}$ enclosed in a compact connected component of free-space $\boldsymbol{q} \in \mathcal{C}_{free}^{compact}(\mathcal{O}, t)$ and bounded between limit orientations $\theta_l(t)$ and $\theta_u(t)$, a collision-free manipulator path $\rho_{\mathcal{M}}$ where the upper and lower limit orientations get closer monotonically, satisfying $\frac{d}{dt}\left(\theta_u(t) - \theta_l(t)\right) < 0$, until $\mathcal{M}(N_T)$ immobilizes $\mathcal{O}$ uniquely at configuration $\boldsymbol{q}$ will drive all configurations $\hat{\boldsymbol{q}} \in \mathcal{C}_{free}^{compact}(\mathcal{O}, t)$ towards $\boldsymbol{q}$.

The conditions specified in Remark 1, shown in Fig. 5-23, are sufficient but might not be necessary. However, these allow us to optimize a manipulator path that satisfies the convergence certificate. This also allows us to characterize the set of

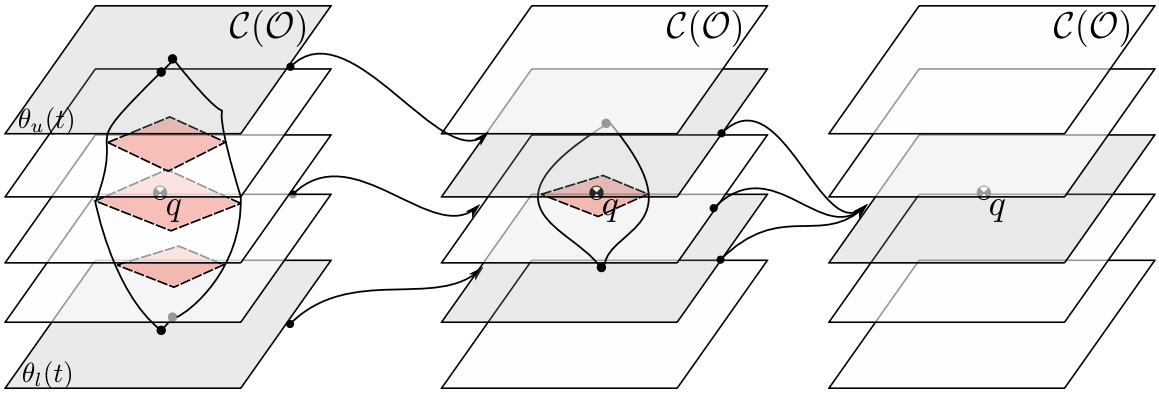


Figure 5-23: Convergence Certificate. This condition is trivially satisfied when the range of limit orientations (gray) decreases, converging at $t = N_T$. The initial cage at $t = 1$ (left) is equivalent to the set of configurations.

initial configuration that will certifiably converge to $\boldsymbol{q}$. Hence, by relying on the model described in the previous section, we derive a linear model to certify convergence as detailed below.

Certificate Model

In order for the conditions detailed in remark 1 to hold, we require that:

1. The object configuration must lie in a cage at all times.
2. The separation between *limit orientations* must decrease monotonically between time-steps, until they converge at $t = N_T$.
3. The cage at $t = N_T$ must uniquely enclose the goal configuration $\boldsymbol{q}$.

Algebraically, the conditions to impose a cage at each time-step are posed as:

$$\begin{cases} \sum_s \Theta_s(t) = 2 \\ \theta_s(t) \in [\theta_l(t), \theta_u(t)] \Rightarrow (\text{loop existence})|_{(s,t)} \end{cases} \tag{CT12}$$

for all $t \in \{1, \ldots, N_T\}$. Then, in order to ensure that the cage topology does not break between time-steps, we introduce the following constraint at each slice:

$$\boldsymbol{H}_n(i,j)|_{t=k} \Rightarrow \exists\ r_t \in \mathbb{R}^2\ s.t.\ r_t \in \boldsymbol{P}_{n,i,k+1} \cap \boldsymbol{P}_{n+1,j,k+1} \tag{CT13}$$

Note that this condition is sufficient, as the intersection occurs between convex polygons and the path is linearly interpolated. Because of this, we introduce the following remark:

Remark 2: Since all initial configurations of the object are caged at $t = 1$ and the topology of the enclosing loop does not change between adjacent time-steps, the conditions for $\boldsymbol{q} \in \mathcal{C}_{free}^{compact}(\mathcal{O}, t)$ are trivially satisfied for all $t > 1$.

Furthermore, for the final cage to uniquely immobilize the object a one configuration, we require that the *limit orientations* at $t = N_T$ are infinitesimally close, reducing $\mathcal{C}_{free}^{compact}(\mathcal{O}, t = N_T)$ to a singleton. Algebraically, we add this constraint as:

$$\begin{cases} \boldsymbol{T}_u(N_T) = \boldsymbol{T}_l(N_T) \in \mathcal{L}_{\mathcal{O}} \\ |\theta_u(N_T) - \theta_l(N_T)| \approx 0 \end{cases} \tag{CT14}$$

This condition effectively ensures that the object is trapped at an unique configuration at time N_T, since it cannot translate due to the loop and it cannot rotate beyond θ_u or under θ_l (which are infinitesimally close). Finally, the limit orientations converge monotonically under the constraint:

$$\begin{cases} \theta_u(t+1) < \theta_u(t) \\ \theta_l(t) < \theta_l(t+1) \end{cases} \tag{CT15}$$

This, along with the caging model, certifies that the grasp will always succeed within a set of certified initial configurations Q_0.

Optimizing Region of Attraction.

A desirable property of a grasp is for it to maximize the set of configurations caged a $t = 1$, since all these configurations would converge to the goal. This condition is

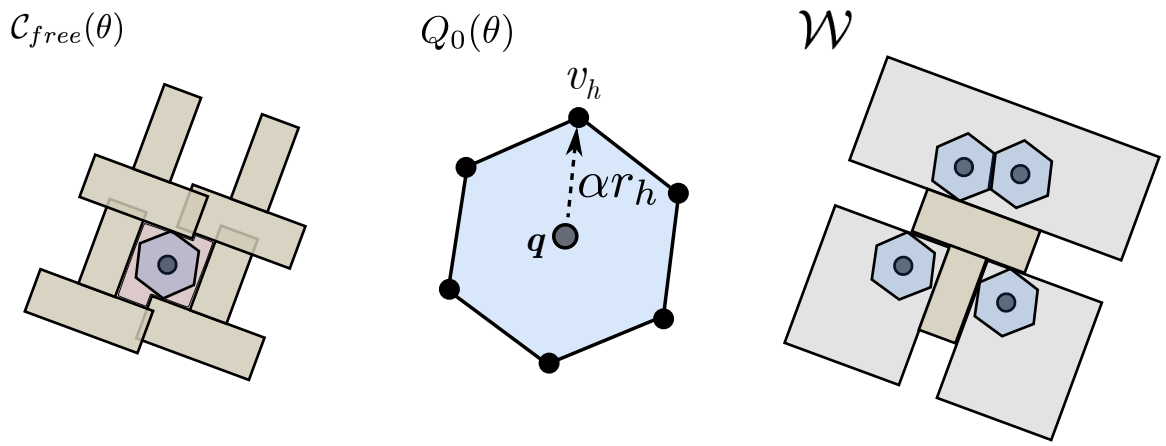


Figure 5-24: Region of Attraction Model. We can constrain a set of configuration Q_0 (blue) to lie in the free-space of the initial cage (red). Such constraint is implemented in $\mathcal{C}_{free}(\theta)$ and $\mathcal{W}$ by enforcing that all finger positions lie in a free-space region $\mathcal{R}$ when translated by the vertices of v_h of Q_0.

equivalent to constraining that $\mathcal{C}_{free}^{compact}(\mathcal{O})$ encloses an arbitrary set of initial conditions Q_0. Unfortunately, constraining volume of Q_0 cannot be integrated in general within this convex-combinatorial model. However, we can use an inner convex approximation of Q_0 in the form: $Q_0 = \{q \in \mathcal{C}(\mathcal{O}) \mid q \in ConvexHull(v_1, v_2, \ldots, v_n) \times [\theta_1, \theta_2]\}$ by including two additional constraints:

1. All configurations in the convex hull defined by $v_1, v_2, \ldots, v_n$ must be enclosed by the loop, satisfying $\sum_{(i,j)} \mathbf{F}^{v_k}(i,j),\ k \in \{1, \ldots, n\}$ to be odd for all points in the convex hull.

2. All configurations in $ConvexHull(v_1, v_2, \ldots, v_n)$ need to lie outside of the $\mathcal{C}-$obstacles. In the workspace, this condition is equivalent to imposing that the fingers lie in one of the R regions of $\mathcal{W} \backslash \mathcal{O}$ when the object is translated around the configuration in the convex hull.

Since this polygon is convex, our model only needs to enforce that the polygon vertices are enclosed by the loop and that the fingers lie in a single convex region $\mathcal{R}$ when they are translated by $-v_1, v_2, \ldots, -v_n$. These conditions can be expressed algebraically as:

$$\theta_s \in [\theta_1, \theta_2] \Rightarrow \quad \sum_{(i,j)} \mathbf{F}^{v_h}(i,j) \text{ is odd, } \forall h$$
$$\boldsymbol{R}(i,r) = 1 \Rightarrow \boldsymbol{p}_i - v_h \in \mathcal{R}_r, \forall h \tag{CT16}$$

These constraints ensure that that all points in $ConvexHull(v_1, v_2, \ldots, v_n)$ are enclosed by the cage. An useful description of this convex hull is under the following decomposition:

$$v_h = \boldsymbol{q} + \alpha r_h, \ \alpha > 0$$

where r_h is a vector in the direction of $v_h - \boldsymbol{q}$. Under this definition, we can incorporate α as a decision variable of our model which can be maximized to find the cage the encloses the biggest convex hull. This process is summarized in Fig. 5-24.

5.2.4 Observability Certificate

Once a planned grasp process has been executed, we can also certify the immobilization at the goal configuration if the grasp is *observable*, i.e. such that we can retrieve the object pose from sensor readings. In this section, we present a definition of grasp observability and derive sufficient constraints for a grasp to be locally observable under proprioceptive sensing (e.g. joint encoders). In practice, this adds an extra constraint to the type of end grasp that we are interested in.

Definitions

Given a vector of n_r sensor readings $\boldsymbol{s} \in \mathbb{R}^{n_r}$, we define:

Definition 1 (Sensor Model): Given a final grasp G achieved by a manipulator configuration $\mathcal{M}(N_T)$, we define a sensor model F_G as a mapping from object configurations to sensor readings:

$$\begin{aligned} F_G: \quad \mathcal{C}(\mathcal{O}) &\longrightarrow \mathbb{R}^{n_r} \\ \hat{\boldsymbol{q}} &\longmapsto \boldsymbol{s} = (s_1, ..., s_{n_r}) = F_G(\hat{\boldsymbol{q}}). \end{aligned}$$

Definition 2 (Grasp Observability): Given a grasp G, a sensor model F_G and

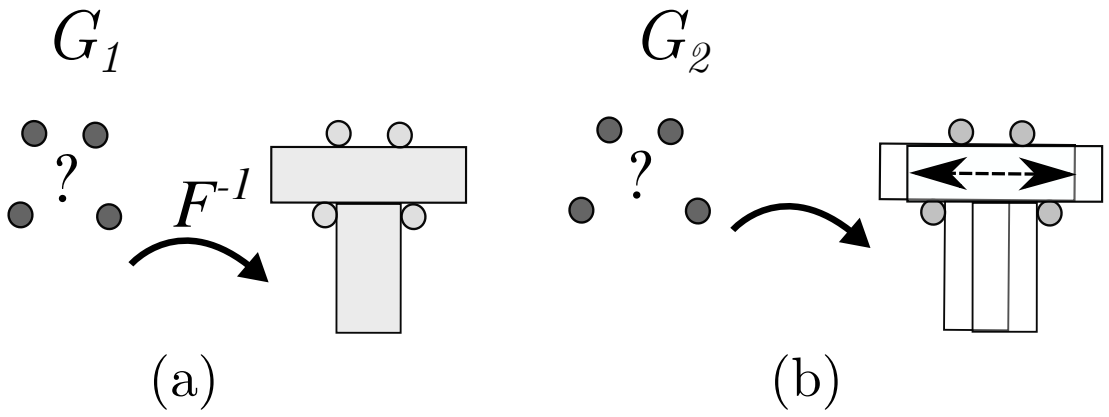


Figure 5-25: Observability Certificate. (a) An observable grasp G_1. (b) A non-observable grasp G_2, the object can slide between the fingers.

a final object configuration $\boldsymbol{q}$, we will say that G is observable if and only if F_G is locally invertible around $\boldsymbol{q}$.

Remark 3: If $n_r \geq 3$ and the sensor model F_G satisfies that its Jacobian $JF_G(\boldsymbol{q}) \in \mathbb{R}^{n_r \times 3}$ is full rank, then the grasp G is observable and only 3 sensor readings are necessary for observability.

Fig. 5-25 shows an example of grasp observability. In general, F_G can be hard to define in closed form, as it depends on the object and manipulator geometries. Hence, we restrict our analysis to first order effects, see [108].

Proprioceptive Sensor Model: In order to give an intuitive notion of a sensor reading, we characterize a sensor model for point-contact sensing to first order effects. Intuitively, for an object in contact, this model reports local changes based on a gap function at each contact point, $\psi_i(\bar{\boldsymbol{q}}, \boldsymbol{p}_i)$, as it is commonly used to formalize the study of grasp stability, as in [101]. More concretely, sensor readings should only report changes in the object pose that imply decrements of the gap (causing penetration), ignoring changes that preserve or break contact (no applied force). Therefore, we characterize a sensor reading with the result of applying the sensor model Jacobian

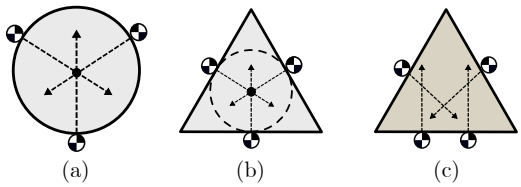


Figure 5-26: Examples of grasps with different proprioceptive observability conditions: (a) Not observable, (b) First-Order Not-Observable, and (c) First-Order Observable.

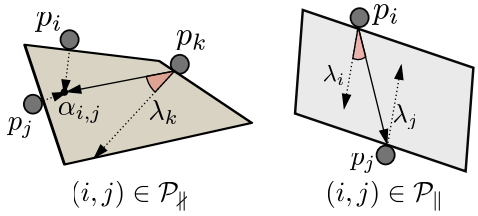


Figure 5-27: Examples of finger arrangements for non-coincidence of normal vectors in the case of non-parallel (left) and parallel (right) facet assignments. Note that these arrangements are illustrative examples and not grasps.

to an infinitesimal object configuration variation $d\boldsymbol{q}$ from $\boldsymbol{q}$:

$$JF_G(\boldsymbol{q}) = \begin{pmatrix} \frac{ds_1}{d\boldsymbol{q}}(\boldsymbol{q}) \\ \vdots \\ \frac{ds_{nr}}{d\boldsymbol{q}}(\boldsymbol{q}) \end{pmatrix},$$

$$\frac{ds_i}{d\boldsymbol{q}}(\boldsymbol{q})\ d\boldsymbol{q} = \begin{cases} k_i\ \dfrac{d\psi_i}{d\boldsymbol{q}}(\boldsymbol{q}, \boldsymbol{p}_i)\ d\boldsymbol{q}, & \dfrac{d\psi_i}{d\boldsymbol{q}}(\boldsymbol{q}, \boldsymbol{p}_i)\ d\boldsymbol{q} < 0, \\ 0, & \dfrac{d\psi_i}{d\boldsymbol{q}}(\boldsymbol{q}, \boldsymbol{p}_i)\ d\boldsymbol{q} \geq 0, \end{cases}$$

where k_i is a real non-zero constant.

The first-order behavior of the proprioceptive sensor model above highlights a relation between observability and first-order form closure. As a result of Remark 3, we will consider only three sensor readings, $n_r = 3$.

Remark 4: Given a grasp G of an object in its final configuration $\boldsymbol{q}$, first-order form closure is equivalent to have the matrix $JF_G(\boldsymbol{q})$ be of full rank, where F_G is the proprioceptive sensor model defined above.

Proof. Note that having full rankness of $JF_G(\boldsymbol{q}) \in \mathbb{R}^{3\times 3}$ is equivalent to:

$$[JF_G(\boldsymbol{q})\ d\boldsymbol{q} = \mathbf{0} \Rightarrow d\boldsymbol{q} = \mathbf{0}] \ \Leftrightarrow$$
$$\left[\forall\ i,\ \ \frac{ds_i}{d\boldsymbol{q}}(\boldsymbol{q})\ d\boldsymbol{q} = 0 \Rightarrow d\boldsymbol{q} = \mathbf{0}\right].$$

As a result of the first-order behavior of our virtual sensor model, we have

$$\frac{ds_i}{d\boldsymbol{q}}(\boldsymbol{q})\ d\boldsymbol{q} = 0 \ \Leftrightarrow \frac{d\psi_i}{d\boldsymbol{q}}(\boldsymbol{q}, \boldsymbol{p}_i)\ d\boldsymbol{q} \geq 0,$$

where the implication from right to left is by definition and from left to right is a consequence of $k_i \neq 0$. Therefore,

$$\left[\forall\ i,\ \ \frac{ds_i}{d\boldsymbol{q}}(\boldsymbol{q})\ d\boldsymbol{q} = 0 \Rightarrow d\boldsymbol{q} = \mathbf{0}\right] \Leftrightarrow$$

$$\left[\forall\ i,\ \ \frac{d\psi_i}{d\boldsymbol{q}}(\boldsymbol{q}, \boldsymbol{p}_i)\ d\boldsymbol{q} \geq 0 \Rightarrow d\boldsymbol{q} = 0\right],$$

that is precisely a characterization of first-order form closure (see [101]). Consequently, first-order form closure is equivalent to full rankness of $JF_G(\boldsymbol{q})$, when considering F_G as the proprioceptive sensor model. □

Corollary 1: Given a grasp G and the proprioceptive sensor model F_G, first-order form closure implies grasp observability.

Given the relation between form-closure and observability that we derived above, a planar grasp is first-order observable if there are 4 unilateral contact constraints on the object, see [108]. This is satisfied if the following conditions hold:

1. The object configuration must lie in a singleton of $\mathcal{C}_{free}(\mathcal{O}, t)$. This condition is already implied by (CT14).

2. There must exist no point of coincidence between all the contact normals. This is required because otherwise, to first-order, the object would be free to rotate

infinitesimally around the point of concurrency of the contact normals (see [108]).

Fig. 5-26 shows examples. These are convex-combinatorial constraints on the facet-assignment matrix $\boldsymbol{T}_s$ and manipulator configuration $\mathcal{M}(N_T)$. Algebraically, the non-coincidence condition can be expressed as:

$$\bigcap_i \boldsymbol{p}_i(N_T) + \langle \lambda_i \rangle = \emptyset$$

We note that there are two scenarios for every pair of fingers: 1) Intersecting normals correspond to non-parallel facets and have a single intersection point, and 2) Normal vectors are parallel and thus have infinite intersection points or none. Therefore, if we define the following sets:

- $\mathcal{P} = \{(i,j) \in N^2 \mid i > j\}$ is the set of all different pairs of facet-assignments.
- $\mathcal{P}_{\parallel} = \{(i,j) \in \mathcal{P} \mid \lambda_i \times \lambda_j = 0\}$ is the set of pairs of facet-assignments with parallel normals.
- $\mathcal{P}_{\nparallel} = \{(i,j) \in \mathcal{P} \mid \lambda_i \times \lambda_j \neq 0\}$ is the set of pairs of facet-assignments with nonparallel normals.

where $\times$ is the ordinary cross-product. Then, we can introduce the binary matrix $\boldsymbol{M} = (\boldsymbol{M}_{i,j})_{(i,j)\in\mathcal{P}} \in \{0,1\}^{|\mathcal{P}|_c}$, where $|\mathcal{P}|$ is the cardinality of $\mathcal{P}$, reducing the problem to the following set of convex-combinatorial conditions:

$$\boldsymbol{M}_{(i,j)\in\mathcal{P}_{\nparallel}} \Rightarrow \sum_{k=1}^{N} |(\alpha_{i,j} - \boldsymbol{p}_k(N_T)) \times \lambda_k| > 0 \tag{CT17}$$

$$\boldsymbol{M}_{(i,j)\in\mathcal{P}_{\parallel}} \Rightarrow |(\boldsymbol{p}_i(N_T) - \boldsymbol{p}_j(N_T)) \times \lambda_i| > 0 \tag{CT18}$$

$$\sum_{(i,j)\in\mathcal{P}} \boldsymbol{M}_{i,j} \geq 1, \tag{CT19}$$

where $\alpha_{i,j}$ is the intersection point between the lines defined by the normal vectors starting at $\boldsymbol{p}_i(N_T)$ and $\boldsymbol{p}_j(N_T)$. These conditions guarantee that at least one pair of normals is non-coincident to the rest, providing observability as shown in Fig.

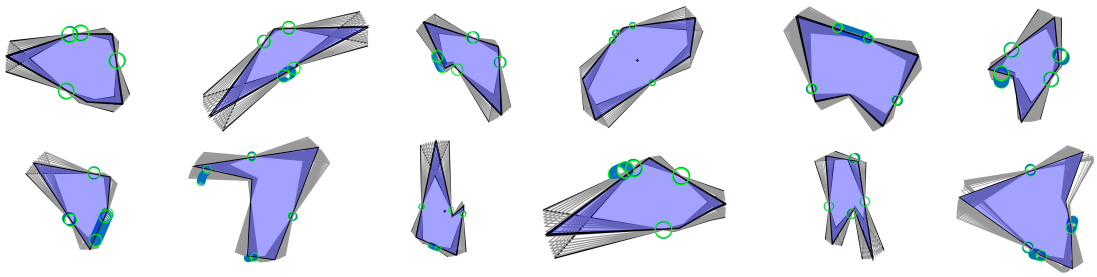


Figure 5-28: Simulation results. 12 random polygons are grasped with trajectories generated with our model. In each case, a set of random initial configurations certified by our model (shown in gray) are driven towards a goal grasp (purple) by using the same trajectory (blue).

5-27. Here, we include absolute value function through slack variables and big-M formulation ([56]).

5.2.5 Application to Sensorless Grasping

This section describes an optimization problem for grasping of planar objects with bounded uncertainty. For this, we formulate a Mixed-Integer program (MIP) using the constraints described in sections 4, 5 and 6. We validate this approach on different polygonal objects, both with experiments and simulations. All the computations are done in MATLAB R2018b on a MacBook Pro computer with Intel Core i9 2.9 GHz processor. All optimization problems are solved with Gurobi 8.0 (see [59]).

Mixed-Integer Programming Formulation

We propose a formulation which receives as inputs the description of the polygonal object $\mathcal{O}$ and the manipulator $\mathcal{M}$. We incorporate the conditions described through the section as constraints and add a quadratic cost term on acceleration to smooth the trajectory, resulting in problem **MIQP1**.

$$\textbf{MIQP1}: \min_{\mathcal{M}(t)} \quad \int \sum_{i=1}^{N} \left|\left| \frac{d^2 \boldsymbol{p}_i(t)}{dt^2} \right|\right|^2 dt$$

subject to:

1. For $t = 1$ to $t = N_T$:

   - Caging (CT12)-(CT13).
   - Convergence Certificate (CT14)-(CT15).
   - Additional possible constraints (kinematics, region of attraction).

2. ($t = 1$) Invariance certificate (CT12)-(CT13).

3. ($t = N_T$) First-Order Grasp Observability (CT17)-(CT19).

We stress the versatility of this formulation, since **MIQP1** is a Mixed-Integer Convex Program, see [56], this problem preserves the properties of the underlying caging model proposed in [1]. Because of this, any solver will always converge to its global solution and will report infeasibility only if no solution exists[2]. We reduce the complexity of the problem by fixing the limit orientations along the path, prescribing the variables in (CT15).

Imposing additional constraints

One of the benefits of this formulation is that it allows for the inclusion of additional conditions over the finger trajectories, as long as these are transcribed in a mixed-integer constraints. Examples of these constraints include: kinematic relations between the fingers as formulated in [35], force-closure in the final grasp as transcribed in [101, 37], and interactions with the environment as in [1]. Furthermore, this framework is also compatible with contact dynamics when there are formulated in the context of trajectory optimization, with the example of [4].

Simulated Experiments

We generate a set of 12 random polygons and optimize a trajectory for each using **MIQP1**. Then, we perform simulations for a set of over 100 different initial conditions, using the open planar manipulation simulator in [139]. We initialize the plan with 3 limit orientations between $-15°$ and $15°$, centered around $\theta = 0$. For simplicity,

[2]We however remark that this solution will only be optimal under the specific slicing of $\mathcal{C}$

do not include constraint (CT16) when solving these simulate problems. Each MIQP has in the ranges of 500-1000 continuous variables and 200-500 integer variables.

In order to generate random polygons with interesting properties, we rely on the heuristics presented in [16], which specify parameters such as irregularity and referential radius. We implement this code in MATLAB and generate the 12 polygons of Fig. 5-28, with 4 to 6 facets. We segment each object with Delaunay triangulation, refer to [57], and construct $\mathcal{W} \setminus \mathcal{O}$ using a triangular decomposition of the free workspace. Is worth noting that algorithms other than Delaunay triangulation might be able to find a decomposition with a small number of convex polygons, as in [81].

For each initial condition, we execute the trajectory with 4 free point fingers. Results for 12 of the random object are reported in Fig. 5-28. Using the same manipulator trajectory, a set of different initial poses (marked in gray) are driven towards $\boldsymbol{q}$ (blue). For all the objects, a trajectory was successfully found in 25 to 45 seconds. However, the time required to find the optimal trajectory ranged from several seconds to around two minutes, depending on the number of integer variables of the problem. We note that fixing the decision variables in (CT15) tends to allow little translational uncertainty, suggesting the need for (CT16) in the general case.

Real Robot Experiments

We demonstrate trajectories generated on four different planar objects in a real experimental set-up with a two-armed robot. We optimize trajectories for each of the objects in Fig. 5-28 and use simulations to determine Q_0. Each trajectory is designed with $N_T = 5$ time-steps and initial limit orientations between $-22.5°$ and $22.5°$. We perform 10 experiments on each object, initializing them at random initial configurations within the invariant set Q_0.

Our robotic platform is an ABB YuMi (IRB-14000) robot, which has two 7 DOF arms with parallel jaw grippers. We work with a Robot Operating System (ROS)

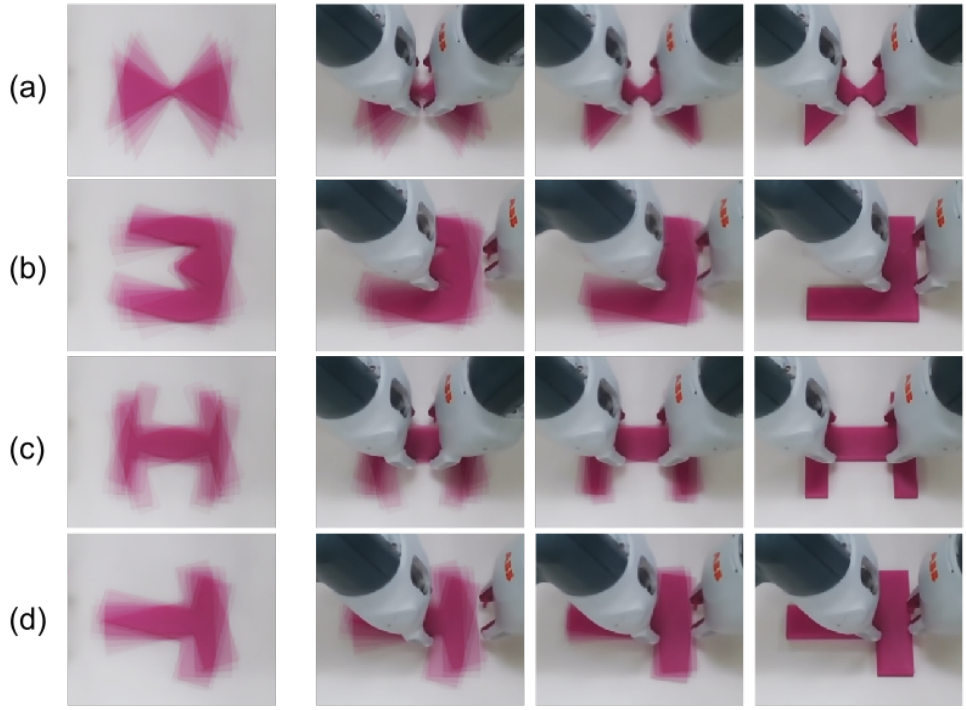


Figure 5-29: Experimental results. Each row shows snapshots from execution of the resulting grasping trajectories for 4 objects, overlaying 10 experiments with initial pose uncertainty (first frame) moving towards a single goal configuration (last frame). Our certification allows for significant rotational uncertainty in the initial object configuration, always converging to the same goal.

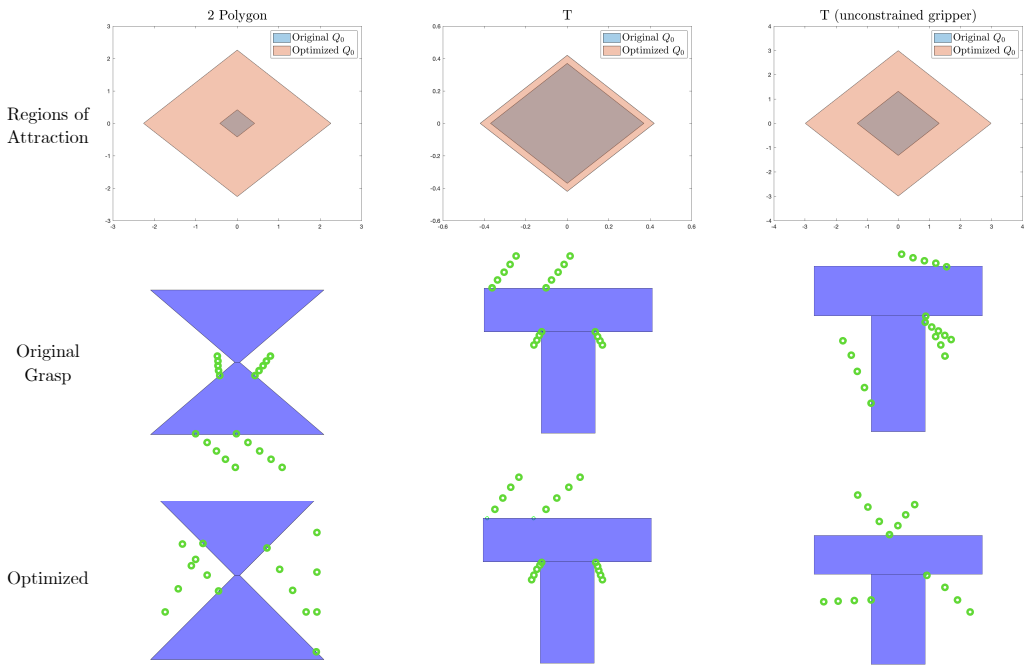


Figure 5-30: Grasps with optimized region of attraction. By incorporating $-\alpha$ as a component of our cost function, we can find the initial cage that encloses the largest area. We show how the size of Q_0, at the $\theta = 0$ slice, increases significantly in all cases, as seen in the *left* examples. However, we also note that kinematic constraints can limit the search for the largest area, as seen for the T object.

setup and an Intel RealSense D415 RGB-D camera calibrated with AprilTag 2 scanning, which we use to place the object within the reach of the robot, and within the invariant set Q_0. Additional constraints are added to **MIQP1** to account for the kinematics of the manipulator. The end-effectors of YuMi are modified to have thin cylindrical fingers. To showcase the robustness of this approach, all experiments are run open-loop.

Fig. 5-29 shows resulting trajectories for 10 different initial conditions of the four objects. Depending on the shape, the resulting trajectories vary from stretching – (a) and (b) – to squeezing (d), and a combination of both (c). In all cases, we are able to handle significant uncertainty in the orientation axis, and varying translational uncertainty (from millimeters to a few centimeters). Videos on the experiments for each of the objects are shown in the supplementary material.

Comparison with pure force-closure grasping

A natural question is how accounting for certification compares to a naive reaching strategy. In order to provide a quantitative answer to this question, we compare our approach to a naive grasping plan which optimizes some criteria of grasp quality, as commonly done in grasp planning algorithms. We design this naive motion by searching for a force close grasp, as defined in [101], and approaching each contact with a trajectory perpendicularly to the goal facet, starting all fingers with the same separation to their contact at the goal pose.

We simulate both strategies to grasp a T-shaped object from 100 different initial conditions. Certified grasping always drives the object to the goal configuration with proprioceptive observability. We measure the L_1 distance to desired object pose, which we call error, after each grasping strategy is executed and report our results in Fig. 5-31. As can be seen in many of these simulations, the naive force-closure grasp does not drive the object towards the goal nor does it provide observability.

Optimizing region of attraction

As we discuss in Section 5.2. we can maximize the set of configurations enclosed by the initial cage Q_0. This in turn, maximizes the set of configurations that will converge to the desired grasp. We incorporate this property into our model by maximizing the set of configurations enclosed by a quadrilateral region on the $\theta = 0$ slice. To this end, we define our convex region with the vertices:

$$v_1 = \boldsymbol{q} + \alpha \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \; v_2 = \boldsymbol{q} + \alpha \begin{pmatrix} 0 \\ -1 \end{pmatrix},$$

$$v_3 = \boldsymbol{q} + \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \; v_4 = \boldsymbol{q} + \alpha \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \; \alpha \geq 0$$

Then we include the constraints defined in (CT16) and expand our cost function to include:

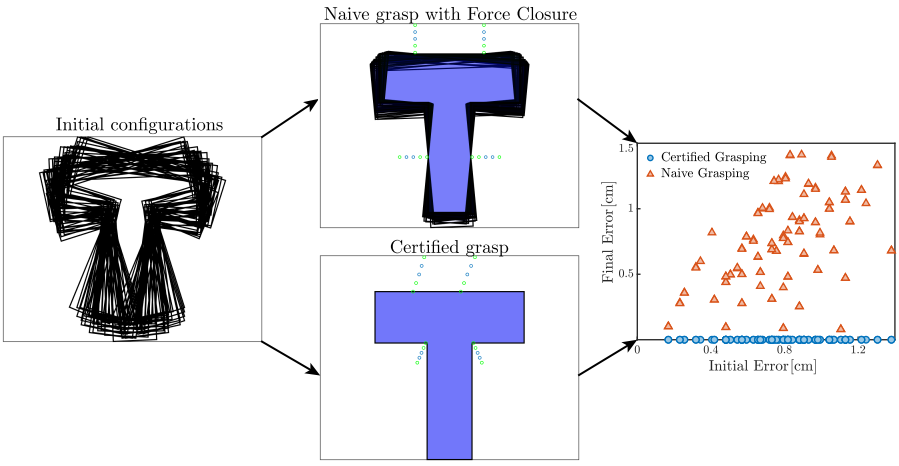


Figure 5-31: Certified Grasping vs. Force-Closure Grasping. We simulate grasps over an object with noisy initial configuration (left). A traditional grasping strategy that maximizes force closure (top-center) fails to handle uncertainty, resulting in significant error in the final pose of several simulations. In contrast, certified grasping (bottom-center) drives the object to its goal pose, always converging to the same configuration. By comparing the L_1 error on the final pose (right), we obtain that certified grasping is orders of magnitude more accurate than a naive policy.

$$\textbf{MIQP2}: \min_{\mathcal{M}(t)} \quad -\alpha + \int \sum_{i=1}^{N} \left|\left| \frac{d^2 \boldsymbol{p}_i(t)}{dt^2} \right|\right|^2 dt$$

We then test this optimization problem over two different geometries (a "T" and a "double-triangle" object), contrasting the solutions of **MIQP1** and **MIQP2**. We report our results in Fig. 5-30. We measure the original Q_0 by solving the **MIQP1**, storing its solution and decision variables, fixing the decision variables in **MIQP2** and then solving to find the maximum α of the original grasp.

Our results show how the size of Q_0 grows noticeably once we include $-\alpha$ in our cost. We also notice how kinematic constraints on the gripper can hinder the search for the largest region, as seen for the "T" object. However, once we remove these constraints, the difference between both regions is much larger. We note that, since we restrict the analysis quadrilateral for areas and only optimize for a common α, all these resulting areas are inner estimates and the true region of attraction might be, in fact, larger.

Chapter 6

Certified Manipulation in the Frictionless Plane

The question of planning robotic manipulation behavior that is robust to uncertainty is almost as old as robotic manipulation itself [85, 51]. Over decades, manipulation research has looked for mechanisms to force the natural dynamics of the manipulation process to contribute to robustness. In many cases, this has been done through educated discovery of actions that yield that natural robustness, for example when pushing an object before grasping it [47], or when squeezing a few times before grasping it [58].

In this chapter we propose a method for synthesis of robust manipulation behavior which is closer to the notion of robustness in control theory where sequences of invariance sets, or funnels, constrain and manipulate the set of possible configurations of a system. Lyapunov analysis, contraction theory, or barrier certificates are common tools for robust behavior in flying, locomoting, or driving robots. However, the interplay between contact mechanics and geometry in manipulation complicates the use of the aforementioned tools in general tasks.

The common approach for planning manipulation behavior is to search for trajectories of manipulator actions that apply the right forces to an object or its environment to move it toward a goal configuration without much consideration to uncertainty. These methods can capture the complexity of the mechanics and the

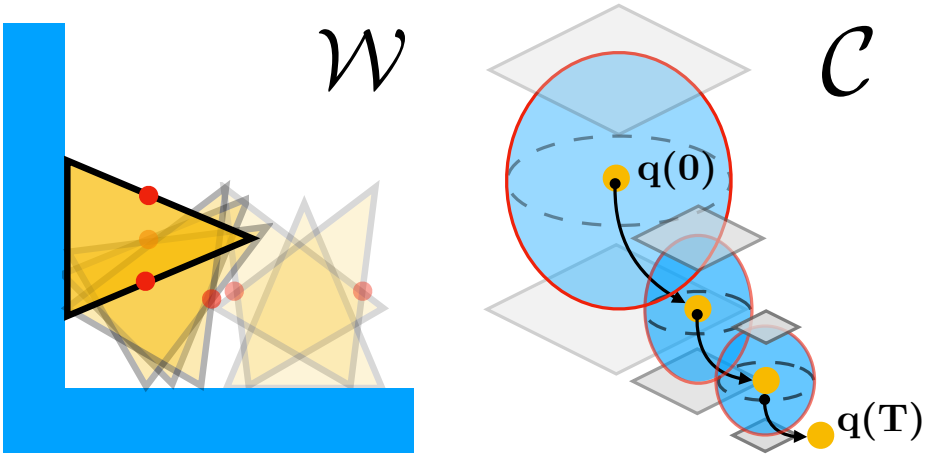


Figure 6-1: We certify manipulation tasks by building a sequence of invariance sets (cages), defined by robot fingers and the environment, which enclose an object along a desired trajectory. This approach generates manipulation strategies that funnel bounded object pose uncertainty to a goal. In this example, we show a triangle being pushed a pivoted against a corner for a bounded set of initial configurations.

geometry of rigid-body interaction[100, 32, 5], however the plans they come up with lead to unreliable executions due to the many sources of uncertainty.

Instead of planning directly for contact trajectories and contact forces when deriving manipulation plans, we propose an alternative method, inspired by the notion of caging [106, 111], where the goal is to derive invariance sets that constraint and manipulate the set of possible configurations of an object. The method that we propose here builds from the contribution from the chapter 5, where an optimization program searches for sequences of manipulator actions that constraint and gradually shrink the free space of an object directly in configuration space leading it to a unique final configuration. We generalize this method to certified manipulation plans by including:

- Motion of the object toward a goal configuration.
- Potential interactions and constraints between object and the environment.

To formulate and solve this problem we make a series of important assumptions: 1)

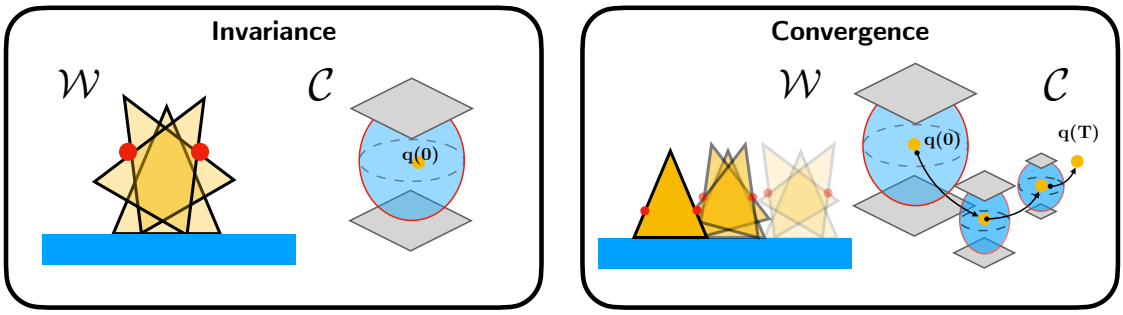


Figure 6-2: Visual depiction of our proposed certificates. **Invariance**: The object initial configuration $\boldsymbol{q}(0)$ is constrained to a compact connected component of free-space $\mathcal{C}^0_{compact}$ (a cage), defined by the manipulator and the environment. **Convergence**: The manipulator drives the object through a sequence of interconnected cages $\mathcal{C}^t_{compact}$ that push the object from its initial configuration through an specified path $\boldsymbol{q}(t)$, converging in a singleton $\mathcal{C}^T_{compact} = \{\boldsymbol{q}(T)\}$.

point manipulators, 2) planar world, 3) frictionless interactions, and 3) polygonal object and environments. Given these assumptions, we plan manipulator motions that manipulate an object through a trajectory toward a goal by combining these two certificates the ensure the success of the manipulation plan:

- **Invariance:** At the beginning of the plan, the object lies in an invariant set of its configuration space. In this chapter we study the case when the object is geometrically trapped by either the manipulator fingers, the environment around it, or a combination of both.

- **Convergence:** All configurations in the initial invariant set are funneled through a series of moving and eventually shrinking invariant sets. At the end of the plan, the invariant set converges to a singleton. This is analogous, in Lyapunov theory, to driving down the value of a scalar/energy function to a unique minimum.

We model these certificates under the language of mixed-integer optimization. This has the caveat of exponential bounds on the computational complexity of solving each optimization problem. Nonetheless, we show how this technique can synthesize certified manipulation strategies that involve open multi-contact interactions between a manipulator, an object and its environment.

Table 6.1: Notation used in the chapter

$\mathcal{O}$	object
$\mathcal{W} \subseteq \mathbb{R}^2$	workspace
$\mathcal{C} \subseteq SE(2)$	C-space
$\mathcal{C}_{\text{free}} \subset \mathcal{C}$	free space
$\mathcal{C}_{\text{col}} \subset \mathcal{C}$	collision space
$\mathcal{C}(\theta)$	slice of the C-space
$\mathcal{C}_{\text{free}}(\theta)$	free space of a slice
$\mathcal{C}_{\text{col}}(\theta)$	collision space of a slice
N_f	number of robot fingers
N_e	number of environment facets
N	number of $\mathcal{C}$-obstacles considered in a loop
M	number of convex polygons composing $\mathcal{O}$
S	number of $\mathcal{C}-$slices
N_r	number of convex regions in $\mathcal{W} - \mathcal{O}$

This chapter is organized as follows: In section 6.1 we summarize relevant work to this problem and introduce the notation we use along the chapter. In section 6.2 we formally describe the problem of this chapter and formulate the approach to solve along with our assumptions. In section 6.3 we describe our models for each certificate. Finally, In section 6.4 we demonstrate how our models can be applied to synthesize certified manipulation plans that are robust to object pose uncertainty

6.1 Background

Given a rigid object $\mathcal{O}$ in a planar workspace $\mathcal{W} \subset \mathbb{R}^2$, we denote its configuration as $q = [q_x, q_y, q_\theta]^T$ in its *Configuration Space* $\mathcal{C} = SE(2)$ [84]. The object $\mathcal{O}$ can be represented as the union of M convex polygons $\mathbf{P}_1, \ldots, \mathbf{P}_M$ covered by L facets $\mathbf{F}_1, \ldots, \mathbf{F}_L$ and vertices $\mathbf{v}_1, \ldots, \mathbf{v}_l$. We refer to a hyperplane of $\mathcal{C}$ with fixed orientation component θ as a $\mathcal{C}-$slice, denoted $\mathcal{C}(\theta)$. The manipulator $\mathcal{M}$ is an arrangement of N_f point fingers in the workspace $\mathcal{W}$, with positions $\mathcal{M} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{N_f}\} \in \mathcal{W}^{N_f}$. The environment $\mathcal{E}$ is an set of N_e facets in the workspace $\mathcal{W}$, defined by line segments $\mathcal{E} = \{\boldsymbol{L}_1, \ldots, \boldsymbol{L}_{N_e}\} \in \mathcal{W}^{N_e}$, where each line segment is defined a $\boldsymbol{L}_k = \overline{v_1^k \; v_2^k}$. For each environment segment, we define two "virtual fingers" which are defined as

$\boldsymbol{p}^k_{(v,1)}, \boldsymbol{p}^k_{(v,2),} \in \boldsymbol{L}_k$[1]. We summarize this notation in Table 6.1.

We refer to the set of configurations where the object intersects a finger or the environment as $\mathcal{C}$-obstacles (also known as Collision Space). Then, the free space of the object $\mathcal{C}_{\text{free}}$ is the subset of $\mathcal{C}$ excluding the $\mathcal{C}$-obstacles. Note that we allow the object to be in contact with the obstacles, and so the free space is a closed set [109]. Each $\mathcal{C}$-obstacle is defined by the Minskowski sum of the robot fingers and the environment. To define a loop one must know set of $\mathcal{C}$-obstacle are enclosing the object configuration at each slice. Hence, define $N = (2 \times N_e + N_f) \times M$, the sum of robot fingers and environment "virtual fingers", as the number of $\mathcal{C}$-obstacles we consider when analyzing the configuration space.

6.2 Certificates for Manipulation

The problem of interest for this chapter is that of finding a manipulation plan that is certified to succeed. Formally, we define this problem as:

Problem 1 (Certified Manipulation): *Given an object $\mathcal{O}$, a manipulator $\mathcal{M}$, S samples of $\mathcal{C}-$slices, and environment $\mathcal{E}$, and a discrete object trajectory $\rho_{\mathcal{O}} = \{\boldsymbol{q}(1), \ldots, \boldsymbol{q}(T)\}$, find a manipulator trajectory $\rho_{\mathcal{M}} = \{\mathcal{M}(t) \mid t \in \{1, \ldots, T\}\}$ and a set $Q_0 \subset \mathcal{C}(\mathcal{O})$, such that $\rho_{\mathcal{M}}$ will drive any configuration of the object $\hat{\boldsymbol{q}} \in Q_0$ including $\boldsymbol{q}(t)$ towards the goal in $\boldsymbol{q}(T)$.*

This problem can be seen as a particular case of the general problem of fine-motion planning, also known as LMT [85]. To solve this problem we make three key assumptions:

1. Friction in between $\mathcal{O}, \mathcal{M},$ and $\mathcal{E}$ is neglegible[2].

2. The object $\mathcal{O}$ is represented as a union of convex polygons.

3. The fingers $\mathcal{M}$ are points in the workspace.

[1] These "virtual fingers" are decision variables used the encode the role of the environment when building a cage around the object.

[2] an intuitive reason for this is to prevent the fingers from jamming or wedging the object

For an object on a plane without friction, a solution to this problem results from enforcing the following certificates as a multi-step process (discretized as a manipulator trajectory of T time-steps):

1. **Invariance**: The initial configuration of the object $\boldsymbol{q}(0)$ lies in a compact-connected component of its free-space $\mathcal{C}^0_{compact} \subset \mathcal{C}_{free}$, This condition is also known as caging [111]. We will impose this condition at $t = 1$ with the convex-combinatorial model of caging proposed in Chapter 5.

2. **Convergence**: The manipulator path drives all configurations towards the goal $\boldsymbol{q}(T)$ through an an overlapping sequence of compact-connected components with decreasing size $\{\mathcal{C}^0_{compact}, \ldots, \mathcal{C}^T_{compact}\}$. These compact connected components are defined to contain each triplet of configurations of the path $\boldsymbol{q}(t-1), \boldsymbol{q}(t), \boldsymbol{q}(t+1) \in \mathcal{C}^t_{compact} \forall t \in 1, \ldots, T-1$. At the final time-step of the path, the $\mathcal{C}$-obstacles reduce $\mathcal{C}^T_{compact}$ to a singleton $\{\boldsymbol{q}(T)\}$.

The satisfaction of these constraints gives a geometric certificate that any configuration of the object in the set $Q_0 = \mathcal{C}^0_{compact}$ will be driven towards and immobilized in the goal configuration $\boldsymbol{q}(T)$. The following sections provide a model for these steps, which then will be combined into an optimization problem (**OPT1**) for certified manipulation of polygonal objects.

6.3 Modeling of certificates

This section describes the modeling of each of our certificates under the language of mixed-integer convex optimization.

6.3.1 Invariance Certificate

As explained above, one way to constrain an object to an invariant set is to cage it geometrically. Extending the model caging proposed in Chapter 5, in order to represent line contacts (such as walls or fingers), the following are a set of sufficient conditions for invariance:

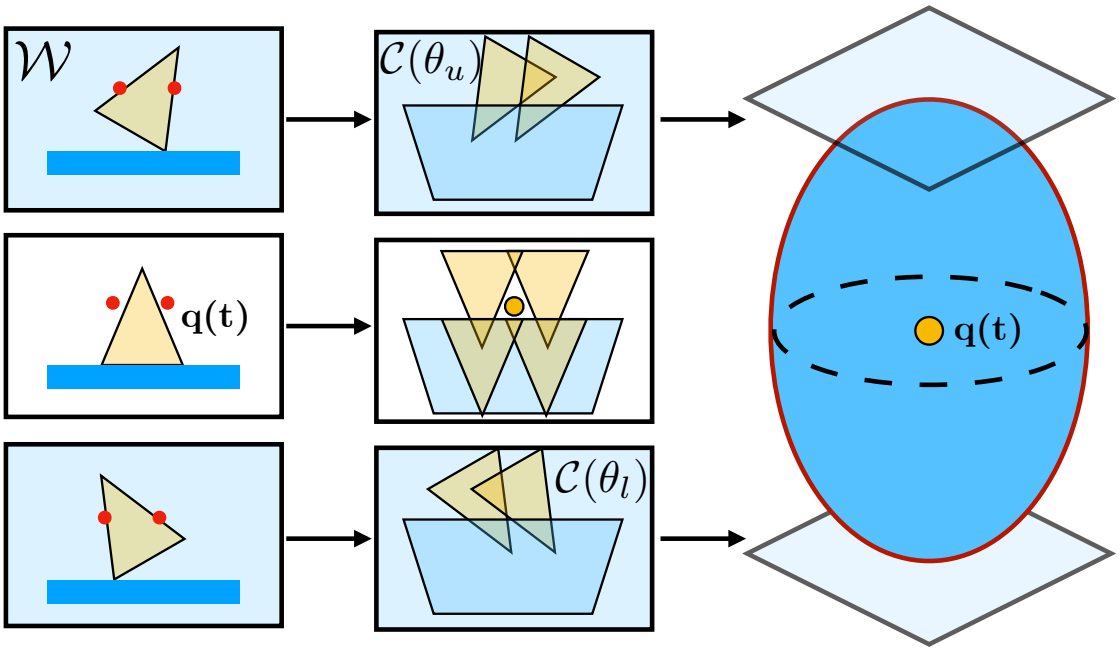


Figure 6-3: Invariance. Example of a cage in $\mathcal{W}$ (left), the $\mathcal{C}-$slices (center) and the configuration space $\mathcal{C}(\mathcal{O})$ (right). Note how the configuration $\boldsymbol{q}$ lies in a compact connected-component of the free-space (pink), bounded by two limit orientations (sky blue).

1. The component $\mathcal{C}^t_{compact}$ is bounded in the orientation coordinate by two limit orientations.

2. At all $\mathcal{C}-$slices between the two limit orientations there is a loop of $\mathcal{C}-$obstacles enclosing a segment of free-space. All these loops must be connected, enclosing a component of free-space in between adjacent slices. At the slice with $q_\theta(t)$, the loop must enclose $\boldsymbol{q}(t)$ (as illustrated in the middle column of Fig. 6-3).

3. At the $\mathcal{C}-$slice of a limit orientation (if these exist) the free-space component enclosed by the loop has zero area. Thus, getting reduced to a line segment or a point.

The union of these conditions define a net of constraints that enclose the configuration $q(t)$, as illustrated in Fig. 6-3.

To construct a loop of $\mathcal{C}-$obstacles at each slice, we transcribe the problem as that of finding a closed directed graph within the intersections between polygonal

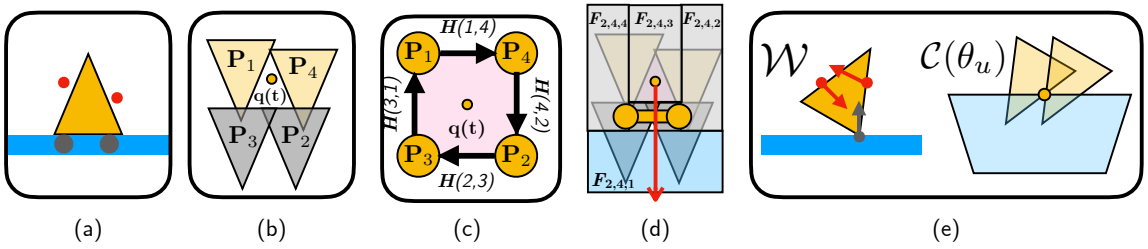


Figure 6-4: Caging Model. (a) Illustration of the cage of an object composed of one polygon ($M = 1$), caged with two fingers and a wall ($N = 4$) in a configuration space slice with a free-space of three polygonal regions ($R = 3$), and with a boundary with three edges ($L = 3$). (b) The model forms a polygonal loop at each slice of $\mathcal{C}(\mathcal{O}, t)$ by using the robot fingers $\boldsymbol{p}$ and introducing "virtual fingers" $\boldsymbol{p}^k_{(v,\{1,2\})}$ inside each environment faces, (c) defining a graph of polygonal intersections that enclose $\boldsymbol{q}(t)$. (d) We require that $\boldsymbol{q}(t)$ is enclosed by the loop by constraining the red ray has an odd number of intersections with the loop. (e) We ensure the cage is bounded by two limit orientations where the contact normals between the object, fingers, and the environment define a positive linear dependent set.

obstacles. In such graph, each node represents a $\mathcal{C}-$obstacle, while each edge imposes an intersection between polygons. We denote the polygon of $\mathcal{C}-$obstacle n as $\boldsymbol{P}_n$. Including this condition in the model, at each time t, is done through the following constraints:

Existence of a Loop. We introduce a binary matrix $\boldsymbol{H}(t) \in \{0,1\}^{N\times N}$ which describes the topology of the loop. $\boldsymbol{H}(t)$ indicates the connection of edges between $\mathcal{C}-$obstacle n and $\mathcal{C}-$obstacle m, at time t, such that $\boldsymbol{H}_{(n,m)}(t) = 1 \Rightarrow \boldsymbol{P}_n \cap \boldsymbol{P}_m \neq \emptyset$. We compute each $\mathcal{C}-$obstacle polygon $\boldsymbol{P}$ as the Minkowski sum between a robot finger $\boldsymbol{p}_k(t)$ or virtual finger from the environment $\boldsymbol{p}^k_{e,\{1,2\}}(t)$ with a convex polygon from the $\mathcal{O}$. This matrix is constrained so that the resulting graph is closed and directed as

$$\boldsymbol{H}_{(n,m)}(t) = 1 \Rightarrow \begin{cases} \sum_{i=1}^{M} \boldsymbol{H}_{(m,i)}(t) = 1 \\ \sum_{j=i}^{M} \boldsymbol{H}_{(j,n)}(t) = 1 \end{cases}$$

. We show an example of this loop and its graph in Fig. 6-4 (b) and (c).

Configuration Enclosing. We include this condition by introducing a binary tensor $\mathbf{F}^{\boldsymbol{q}(t)} \in \{0,1\}^{N\times N\times 4}$, where $\mathbf{F}^{\boldsymbol{q}(t)}(i,j,k=1)=1$ imposes that a ray with origin $\boldsymbol{q}(t)$ has an intersection with the line segment connecting $\mathcal{C}-$obstacle i and $\mathcal{C}-$obstacle j when $\mathbf{H}(i,j)=1$, while other values of $k \in \{2,3,4\}$ denote that the ray lies in the complement of the segment[3]. Following Jordan's Polygon Theorem, the constraint needed to enclose $\boldsymbol{q}(t)$ is to impose $\sum_{(i,j)} \mathbf{F}^{\boldsymbol{q}(t)}(i,j,1)$ to be odd. An illustration of this condition is shown in Fig. 6-4 (d).

Remark 1: *The constraints needed the enclose $\boldsymbol{q}(t)$ can also be used to enclose the set of configurations $\boldsymbol{q}(t) \pm \Delta$ (where $\Delta \in \mathcal{W}$ is a decision variable). Hence $\mathbf{F}^{\boldsymbol{q}(t)}$ can also encode the minimum translational uncertainty enclosed in $\mathcal{C}^t_{compact}$*

Non-Penetration Constraints. We impose this constraint by introducing a binary matrix $\boldsymbol{R}(t) \in \{0,1\}^{N_f\times R}$. $\boldsymbol{R}(t)_{i,r}=1$ assigns finger i to region r in $\mathcal{W}\setminus\mathcal{O}$, or $\boldsymbol{R}(t)_{i,r}=1 \Rightarrow \boldsymbol{p}_i(t) \in \mathcal{R}_r(t)$, with $\sum_r \boldsymbol{R}(t)_{i,r}=1, \forall i,t$.

Constraint Activation. To determine which slices must contain a closed loop of $\mathcal{C}-$obstacles, we must first determine if the cage has limit orientations. To include this constraint, we introduce a binary vector $\boldsymbol{\Theta}(t) \in \{0,1\}^S$, where $\boldsymbol{\Theta}(t)_s=1$ imposes that a limit orientation must be reached before slice s, deactivating all loop constraints in such slice at time-step t. In this context, *before* means a greater or equal angle if the slice lies in the negative orientation half-space or a smaller or equal angle if it lies in the positive one.

Limit Orientations. A limit orientation occurs at the slices where the loop encloses a zero-area component, a condition satisfied when the contact normals of the objects, fingers, and environments define a *positive linear dependent* set. To enforce the existence of limit orientations, we define a set of binary matrices:

1. $\boldsymbol{T}^p_s(t) \in \{0,1\}^{N_f\times L}$, such that $\boldsymbol{T}_s(t,i,l)=1 \Rightarrow \boldsymbol{p}_i(t) \in \boldsymbol{F}_l(t)$ imposes that robot finger i must be in contact with an unique object facet l at slice s.

[3]In practice we use the line segment defined by the geometric centers of the polygons.

2. $\boldsymbol{T}_s^v(t) \in \{0,1\}^{L \times N_e}$, such that $\boldsymbol{T}_s(v,l) = 1 \Rightarrow \boldsymbol{v}_v(t) \in \boldsymbol{L}_l$ imposes that the object vertex v must be uniquely in contact with environment facet l at slice s.

using theses variables and labeling $\mathcal{L}_{\mathcal{O}}$ as the set of contact assignments that is positively linear dependent, we impose $\boldsymbol{T}_s^v \bigcup \boldsymbol{T}_s^p \in \mathcal{L}_{\mathcal{O}} \Rightarrow \boldsymbol{\Theta}_s(t) = 1$. A visualization of this is shown in Fig. 6-4 (e)

Swept volume across slices. In order for the $\mathcal{C}_{compact}^t$ to be compact and connected, the loops created at the $\mathcal{C}-$slices must also enclose a segment of free-space between the slices. Chapter 5 describes a sufficient condition for this, having the boundary of such loops varying continuously towards the boundary of the loop in the adjacent $\mathcal{C}-$slices. This condition is satisfied when the swept volume of the $\mathcal{C}-$obstacles between slices retains the loop topology of the slices. We integrate these constraints as part of the model as well.

Satisfying these conditions ensures that the configuration $\boldsymbol{q}$ is enclosed by a compact-connected component of free-space.

6.3.2 Convergence Certificate

Given an initial cage, the convergence certificate is satisfied if the process drives a set of bounded configurations around a trajectory $\boldsymbol{q}(0), \ldots, \boldsymbol{q}(t)$ towards a goal $\boldsymbol{q}(T)$. The main insight that allows us to integrate this stage in the framework comes from the following remark:

Remark 2: *Given an object $\mathcal{O}$, at some time-step t, in a configuration $\boldsymbol{q}(t)$ enclosed in a compact connected component of free-space $\boldsymbol{q}(t) \in \mathcal{C}_{compact}^t$ and bounded between limit orientations $\theta_l(t)$ and $\theta_u(t)$, a collision-free and frictionless manipulator path $\rho_{\mathcal{M}}$ where the path $\mathcal{M}(t) \rightarrow \mathcal{M}(t+1)$ pushes all points in $\mathcal{C}_{compact}^t$ to $\mathcal{C}_{compact}^{t+1} \supset \boldsymbol{q}(t+1)$, respecting the $\mathcal{C}$-obstacle topology of $\mathcal{C}_{compact}^t$, and satisfying $\frac{d}{dt}(\theta_u(t) - \theta_l(t)) < 0$ will drive any configuration $\hat{\boldsymbol{q}} \in \mathcal{C}_{compact}^t$ towards a configuration on $\mathcal{C}_{compact}^{t+1}$.*

The conditions specified in Remark 2, shown in Fig. 6-5, are sufficient but might not be necessary. However, these allow us to optimize a manipulator path that

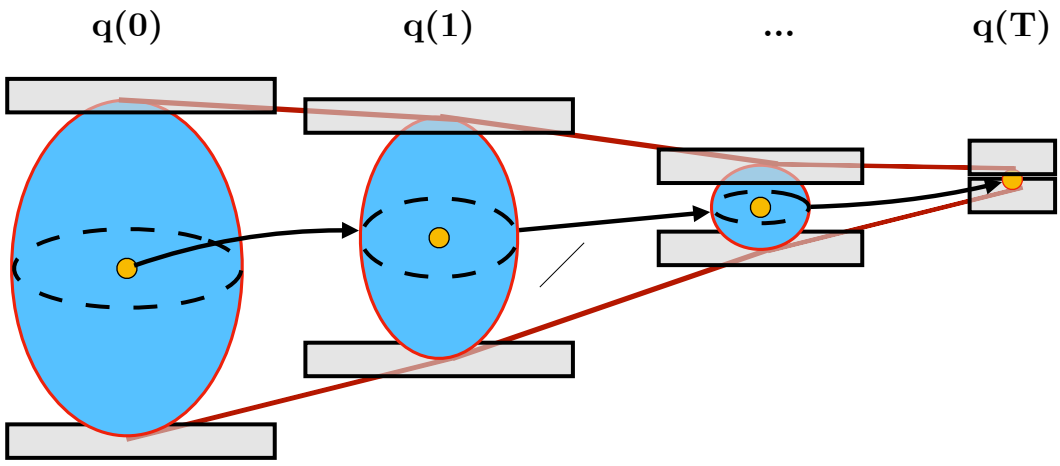


Figure 6-5: Convergence Certificate. We apply this by connecting a sequence of cages with a decreasing range of limit orientations (gray) decreases, converging at $t = T$ to a singleton of $q(T)$. The initial cage at $t = 0$ (left) is equivalent to the set of configurations certified to converge.

satisfies the contraction certificate. This also allows us to characterize the set of initial configuration that will certifiably converge to $\boldsymbol{q}(T)$. Hence, by relying on the model described in the previous section, we derive a model to certify contraction as detailed below.

In order for the conditions detailed in remark 2 to hold, we require that:

1. The object configuration must lie in a cage at all times.

2. Each cage must respect the loop topology of the previous cage, encoded by

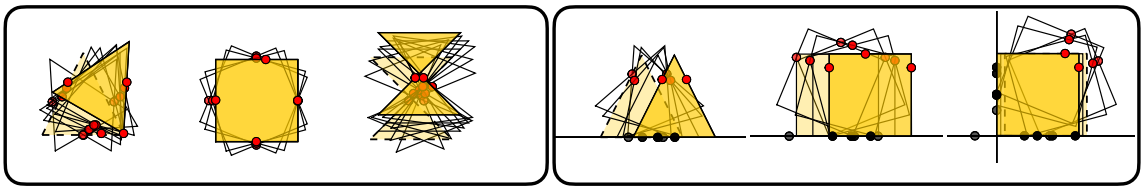


Figure 6-6: Synthesis of short-horizon plans with Invariance and Convergence certificates using **OPT1**. **Left:** Manipulation primitives with point fingers: (1) planar sliding of triangle in an arch, (2) grasping a square, and (3) moving an "hourglass" in a "zig-zag" motion. **Right:** Sliding and trapping tasks with point fingers and external environments: (1)-(2) Sliding a triangle and a square along a wall, (3) trapping a square against a corner.

$\mathbf{H}(t)$, even if choosing a new topology with matrix $\mathbf{H}(t)$.

3. Each cage must contain the object configuration at the specific time-step and its subsequent one.

Algebraically, the conditions to impose a cage at each time-step are posed as:

$$\begin{cases} \sum_s \boldsymbol{\Theta}_s(t) = 2 \\ \theta_s(t) \in [\theta_l(t), \theta_u(t)] \Rightarrow (\text{loop existence})|_{(s,t)} \end{cases} \tag{CT1}$$

for all $t \in \{1, \ldots, T\}$. Then, in order to ensure that the cage topology is respected between time-steps, we introduce the following constraint at each slice:

$$\boldsymbol{H}(t)|_{i,k} \Rightarrow \exists\ r_t \in \mathbb{R}^2\ s.t.\ r_t \in \boldsymbol{P}_{i,t+1} \cap \boldsymbol{P}_{j,t+1} \tag{CT2}$$

Note that this condition is sufficient, as the intersection occurs between convex polygons and the path is linearly interpolated. Finally, we constraint the limit orientations to contract their volume gradually under the constraint:

$$\begin{cases} \theta_u(t+1) < \theta_u(t) \\ \theta_l(t) < \theta_l(t+1) \end{cases} \tag{CT3}$$

This, along with the caging model, certifies that the object configuration will be "funneled" along the trajectory $\boldsymbol{q}(t)$.

For the final time-step, we want to $\mathcal{C}$-obstacles to uniquely enclose the object in $\boldsymbol{q}(T)$. A simple technique to enforce this condition is to require that the two *limit orientations* at $t = T$ are infinitesimally close. Note that this reduces $\mathcal{C}^T_{compact}$ to a singleton. Algebraically, this constraint is added as:

$$|\theta_u(T-1)| \approx |\theta_l(T-1)| \approx \boldsymbol{q}_\theta(t) \tag{CT4}$$

This leads the object to lie in a single configuration where it is unable to rotate,

since the limit orientations enclose the orientation space, or translate, since the limit orientations are defining a positive linear dependent set[4]. With this constraint, we can certify that the entire manipulation plan funnels a set of configurations centered $\boldsymbol{q}(0)$, and along a trajectory $\boldsymbol{q}(t)$, towards a goal $\boldsymbol{q}(T)$.

6.4 Synthesis of certified manipulation plans

Using our model for each certificate, we can pose the following optimization problem that results in a solution to **Problem 1**. We propose a formulation which receives as inputs the description of the polygonal object $\mathcal{O}$, the reference object trajectory $\boldsymbol{q}(t)$, the environment $\mathcal{E}$, and the manipulator $\mathcal{M}$. We incorporate the conditions described through the chapter as constraints and add convex cost term $J(\boldsymbol{p})$, resulting in problem **OPT1**.

$$\textbf{OPT1}: \underset{\mathcal{M}(t),\boldsymbol{H},\boldsymbol{F},\boldsymbol{T}}{\text{minimize}} \int \sum_{i=1}^{N} J(\boldsymbol{p}_i(t))dt$$

subject to:

1. At $t = 0$, Invariance certificate (CT1).
2. For $t = 1$ to $t = T$:
   (a) Caging $\boldsymbol{q}(t-1), \boldsymbol{q}(t)$, and $\boldsymbol{q}(t+1)$ (CT1)-(CT2).
   (b) Convergence of $\mathcal{C}^t_{compact} \rightarrow \mathcal{C}^{t+1}_{compact}$ (CT3).
   (c) Additional task constraints (e.g. kinematic limits [35], robustness.
3. At $t = T$, Caging a singleton (CT4).

We summarize the decision variables of this optimization problem in Table 6.2. We remark the versatility of this formulation, since **OPT1** is a Mixed-Integer Convex Program (MICP), see [56]. Any solver will always converge to its global solution and will report infeasibility only if no solution exists.

[4]This condition is equivalent to a second-order form closure grasp [101]

6.5 Implementation and results

To demonstrate some of the capabilities of our model, we implement **OPT1** and asses its application for set of manipulation tasks. First, we aim to validate the model's ability to optimize simple manipulation behaviors. We then show different applications of our model for manipulation problems that involve interaction with the robot and the environment. [5]

We generate all the trajectories in MATLAB R2020b, running on an Apple M2 processor with Mac OS X Monterrey. We use Gurobi 8.1.0 [59], an off-the-shelf optimization software, as our MICP solver. We implement **OPT1** with the following setup:

1. We set $S = 3$ and pre-compute the set $\mathcal{L}_{\mathcal{O}}$.
2. To simplify formulation we pre-fix the values of $\theta_l(t), \theta_u(t)$, and $\mathbf{\Theta}_s(t)$ in (CT1) and (CT3).
3. We use a schedule such that $\theta_u(t) - \boldsymbol{q}_\theta(t) = \boldsymbol{q}_\theta(t) - \theta_l(t) = \Delta\theta_{UL}(t)$.
4. We manually generate each object reference trajectory $\boldsymbol{q}(t)$.

For all problems, we add a quadratic cost-function that smooths the finger trajectories:

$$J = \sum_{t=0}^{T} \sum_{f=1}^{N_f} ||\ddot{\mathbf{p}}_f(t)||^2$$

Where we compute second derivatives using a second-order finite difference scheme.

6.5.1 Synthesis of manipulation primitives

We first test our model in canonical problems in planar manipulation, without including environment constraints. We use trajectories with $T = 5$ define $\Delta\theta_{UL} =$

[5] See a video with this results at: https://youtu.be/ZJK5PCg7Flw

Table 6.2: Decision variables of **OPT1**

Variable	Dimension	Description
$\mathbf{H}$	$\{0,1\}^{T\times S\times N\times N}$	Topology $\mathcal{C}-$obstacle connections
$\mathbf{F}$	$\{0,1\}^{T\times N\times N\times 4}$	Enclosing of $\boldsymbol{q}(t)$ at each time-step
$\mathbf{T}^p$	$\{0,1\}^{T\times S\times N_f\times L}$	Contact between the fingers and an object facets
$\mathbf{T}^v$	$\{0,1\}^{T\times S\times L\times N_e}$	Contact between object vertices and an environment facets
$\mathbf{R}$	$\{0,1\}^{T\times N_f\times R}$	Non-penetration between fingers a object
$\mathbf{\Theta}$	$\{0,1\}^{S\times T}$	Limit-orientation assignment
$\boldsymbol{p}$	$\mathbb{R}^{N_f\times T}$	Robot fingers
$\boldsymbol{p}^k_{(v,\cdot)}$	$\mathbb{R}^{2\times N_e\times T}$	"virtual" environment fingers

$\{\frac{\pi}{6}, \frac{\pi}{12}, \frac{\pi}{24}, \frac{\pi}{48}, \pi\times 10^{-4}\}$[6]. We use a point finger manipulator with four fingers for the following tasks:

1. **Planar Grasping:** immobilizing a square and an hourglass-shaped object to a single static configuration.

2. **Planar Pushing:** sliding a triangle and an hourglass-shaped object along a trajectory. For the triangle, we perform a translate+rotate motion. For the hourglass, we execute a zig-zag trajectory.

we show the resulting sensorless manipulation plans in Figure 6-6 (Left). Our results show how our model can reliably generate grasps and planar pushing trajectories. Despite the combinatorial complexity of the mixed-integer program, our solver can find optimal solutions to these problems in the range of 0.5 s to 5 s. For each problem the MICP has in the range of $700-900$ integer variables and $400-600$ continuous variables.

6.5.2 Using environment as a part of the plan

We then test problems where environmental constraints play a more important role in certifications. We test two different environments with one and two walls. Again, we use $T=5$ and $\Delta\theta_{UL}=\{\frac{\pi}{6}, \frac{\pi}{12}, \frac{\pi}{24}, \frac{\pi}{48}, \pi\times 10^{-4}\}$. We use a manipulator with two

[6]This choice of schedule is somewhat arbitrary and other schedules will likely work as well

and three fingers as indicated:

1. **Sliding along a wall:** sliding a square (three fingers) and a triangle (two fingers) against a flat wall towards a goal.

2. **Trapping against a corner:** sliding a square, using two fingers, against the 90° corner of two walls.

We depict both trajectories at the right of Figure 6-6. Our results show how our model can represent trajectories that leverage environment interaction to fully object pose uncertainty. In these tasks, our solver finds optimal solutions in the range of 5.5 s to 30 s. For each problem the MICP has in the range of $900 - 3000$ integer variables and $600 - 900$ continuous variables. We also remark the impact of our frictionless contact assumption at solving this task. In the case of high frictional contact, the object can get jammed or wedged against the wall corner or against a finger, preventing the manipulator from executing the remaining of the plan.

6.5.3 Synthesis of mid-horizon plans

Finally, we test our model in the context of longer horizon task that requires switching cage topology and finger configurations along its execution. We select one illustrative problem that is also challenging: certifiably pivot a triangle 120° or $\frac{2\pi}{3}$ using two robot fingers and two walls. This task is inspired by the allen wrench example described in [51]. We set a horizon of $T = 10$ and a schedule that linearly decreases $\theta_{UL} = \{\frac{\pi}{8}, \ldots, \pi \times 10^{-4}\}$.

Our model successfully finds a trajectory to this problem in 2 minutes. The results show how, intuitively, the fingers follow the following strategy: 1) first pivot the triangle 90° against a corner, funneling the largest set of uncertainty, then 2) shifting the fingers to trap one object facet, and finally 3) funneling the remaining uncertainty while pivoting to 120° while trapping the object between the two fingers (in one facet) and the corner between the two walls. We depict this trajectory on Figure 6-7.

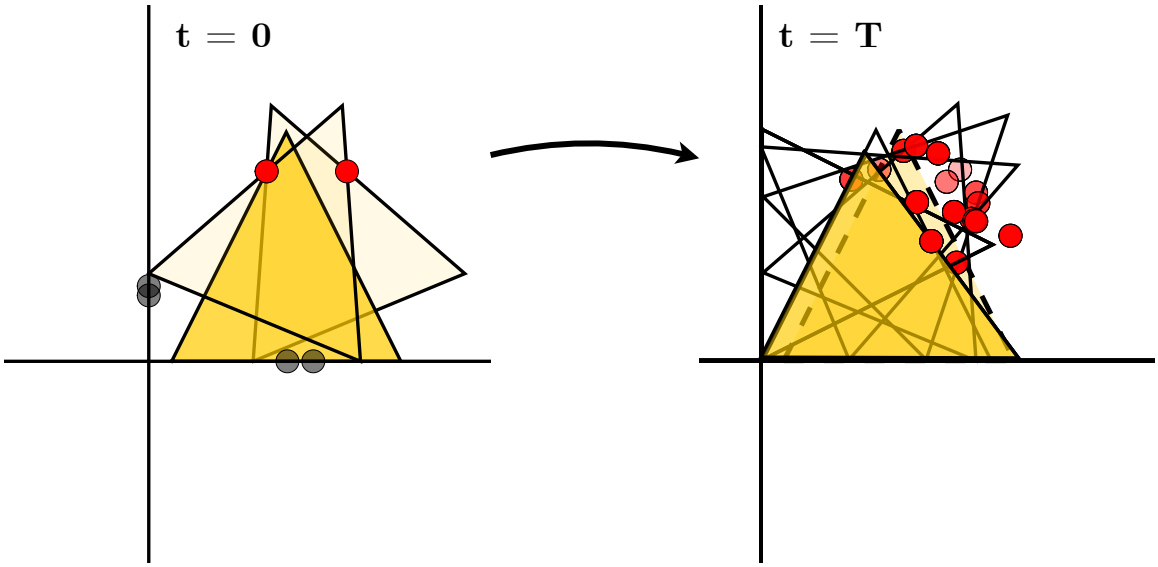


Figure 6-7: Synthesis of mid-horizon sensorless manipulation policy. We solve the task of pivoting a triangle pose 120° using two fingers and two walls.

Chapter 7

Discussion

In this thesis, we have explored the problems of planning, learning, and certifying manipulation policies. In each of the chapters of this thesis we have derived algorithms and computational models that can allow for the discovery of manipulation in skills in planar setups that can leverage our understanding of contact mechanics and the manipulation problem. In this chapter, we will discuss the contributions, limitations and opportunities to explore in future work for each of the chapters of this thesis.

7.1 Planning Manipulation

In Chapter 3, we have presented a hierarchical framework to solve long-horizon object-contact trajectories for manipulation tasks in 2D polygonal environments. We decouple the problem into three stages by searching over a high-level roadmap of a discretized configuration space, planning motion over local free-space regions, and optimizing contact interactions over sampled trajectories. This provides us with a resolution complete algorithm with optimality bounds on high-level search and contact-trajectory. Our framework accounts for object shape, environment contacts, and allows for the inclusion of additional constraints within the contact-trajectory optimization.

We implement this framework using off-the-shelf optimization software and MATLAB. We validate its application on a variety of manipulation tasks with varying

time horizons, physics constraints, and geometries. Our planner returns a solution between 0.2 s to 9.6 s, varying with the horizon of the task and complexity of the problem. Our long-term vision is that the execution of these plans will be robust to uncertainty using geometry and certification, as proposed in [2], and supported by a low-level tactile controller, as demonstrated in [66].

Limitations. This algorithm has a few limitations that constrain its scope. First, the use of mixed-integer programming leads to combinatorial complexity in the contact-trajectory optimization step. Second, we constrain our manipulator to perform alternated sticking, which eliminates the possibility of in-hand sliding as part of our plans. Finally, the sampling nature of this approach will usually lead to non-smooth object trajectories, with jerky motion. Moreover, extending this approach to 3D tasks is not straightforward. Since we rely on free-space slicing over the orientation component to construct a roadmap. A generalization to 3D would require a less naive alternative decomposition of the free-space into convex regions.

Future work. The first set of extensions to this work come from its limitations. First, we can extend the contact-trajectory optimization model to account for sliding motion, with the caveat that it would require more binary variables. Secondly, the trajectories from this model could be post-processed through a nonlinear optimizer, fixing the contact schedule, in order to get smoother object motions. While this approach is not directly applicable to 3D tasks, we can sequentially compose planar tasks between sagittal and traversal environments in order to generate a "2.5D" behavior. In such case, we could also slice over the depth dimension of the space and generate a graph of convex regions over two rotation dimensions, leading to manipulation skills like those of [66]. Another natural extension would be to combine the solutions of this algorithm with a state-of-the-art feedback controller [61, 50, 69]. The use of contact forces could rely on feedback from localized tactile sensing [33, 66].

7.2 Learning Manipulation

In chapter 4 we explore the problem of visual non-prehensile planar manipulation, reconciling tools from model-based mechanics with deep learning. Our proposed Differentiable Learning for Manipulation (DLM) approach: (i) encodes the input video $\mathcal{V}$ in a latent vector $L^{\mathcal{V}}$, (ii) derenders mechanical parameters $\mathcal{P}^*(t)$ for the task, (iii) solves **QP** [14] to obtain robot finger actions $p_c^*(t), \lambda_c^*(t)$, and (iv) evaluates the performance of the model by simulating these actions. We train this model by minimizing $\mathcal{L}_{\mathcal{P}}$, in order to match the ground-truth parameters, and by minimizing $\mathcal{L}_{QP}$, in order to match the ground-truth actions. Moreover, we can self-supervise this approach with new data by back-propagating through the simulator [41] by minimizing $\mathcal{L}_{sim}$, without the need for ground-truth labels on parameters or actions. We assess this method by learning how to solve planar manipulation tasks given a pre-segmented video showing a desired object motion. Our experiments suggest that, when compared to fully neural architectures, our approach can generalize better to unseen tasks and shapes with the same amount of training data.

Limitations. The first limitation of this approach comes from the differentiable simulation scheme. Since our implementation uses finite differences, the slow backward pass hampers the scalability of our model. Similarly, the approximations required to make contact-mechanics differentiable also lead to undesired effects, such as rigid-body penetrations and contact at distance, which can compromise the quality of the results. A second limitation arises from our mechanical assumptions. Assuming that contacts occur at a set of points disregards tasks involving multiple or continuous contacts (e.g. pivoting against a wall). Moreover, the representation of the friction cones brings challenges for extending to 3D scenarios, as it cannot appropriately represent sliding contact. Finally, since our setup uses unconstrained point fingers, we are unable to represent more realistic robot kinematics.

Future work. The next step towards our vision is applying this framework in a real-world system, requiring the capability to handle real-world video data and robot

kinematics. Our vision is to perform training on completely synthetic data generated with MIQP [4], fine-tuned for robustness with the simulator, and execute a task specified with a real-world video. Exploring faster simulation schemes will also be essential to scale this model to larger datasets. Techniques that explicitly resolve the contact mechanics, through differentiable solvers, can alleviate this issue [41, 102]. Extending this framework to more complex manipulation tasks might require an implicit representation for multiple environmental contacts. Finally, moving to 3D tasks in $SE(3)$ is also possible by solving contact-trajectory optimization [4] in a 3D environment [4, 32] also leveraging recent advances in large-scale 3D differentiable simulation [102].

7.3 Certifying Grasping

In chapter 5 we we studied certified grasping of planar objects under bounded pose uncertaint and in frictionless objects. To do this, we extend grasp analysis to include the reaching motion towards the final arrangement of contacts. Under this perspective, we propose three certificates of grasp success: 1) Invariance within an initial set of configurations of the object, 2) convergence from the initial set to a goal grasp, and 3) observability of the final grasp. For each of the these certificates, we derive a mathematical model, which can be expressed with convex-combinatorial constraints, and demonstrate their application to synthesize robust sensorless grasps of polygonal objects. We validate these models in simulation and with real robot experiments, showcasing the value of the approach by a direct comparison with force closure grasping.

Limitations. The first limitation of this approach comes from restricting our the analysis to the configuration space of the object, which neglects frictional interaction between the fingers and the object. Because of this, our model can only find certified grasps for trajectories that do not create stable configurations such as jamming or wedging. Accounting for the role of friction, characterizing undesired scenarios such

as in [60], would allow this framework to provide certification over a larger range of dynamic settings and objects. The second limitation comes from the first-order proprioceptive analysis of observability. Including second-order effects such as curvature of the object, see [108], as well as accounting for more discriminative sensor models that provide shape, texture, or force information, as in [49], could certify success of a grasp without requiring form-closure constraints.

Future work. Given the versatility of convex-combinatorial optimization models, we believe that this approach can be extended to the design of finger phalanges with complex shapes beyond finger points, see [110]. This would allow to certifiably grasp specific objects within a larger set of initial conditions and with a lower number of fingers. Additionally, we are interested in extending this model to invariance sets that are not purely geometrical, for example by considering energy bounds, as in [86], or other type of dynamic constraints on object mobility. This could potentially allow for applications such as sensorless in-hand manipulation, as in [51]. Another promising line of research lies in the study of certification under parametric uncertainty in object shape. Finally, it is important to explore the case for three dimensional grasps and objects, since the definition of our certificates (and specifically **remark 2**) are not limited two-dimensional setups. However, our model for caging is not directly extendable to 3D objects and the notion of limit orientations is not well defined in $SE(3)$ either.

7.4 Certifying Manipulation

In chapter 6 we presented a global model for the *Certified Manipulation* problem on two dimensional polygonal environments. We achieve this by assuming the robot fingers are represented as points, the environment is represented as a set of line segments, and that friction coefficients between all bodies are negligible. Under these assumptions, we propose two certificates, *invariance* and *convergence*, that ensure a manipulation plan will succeed despite initial bounded object pose uncertainty.

We model both certificates under the language of mixed-integer programming. We implement these certificates as part of an optimization model to synthesize certified manipulation plans. We show how our model reliable generates plans for a variety of tasks including simple manipulation primitives with robot fingers, environment interaction, and mid-horizon tasks. Our model provides a principled technique to find manipulation strategies that can be robust to object pose uncertainty.

Limitations. There are several limitations to this approach. First, the computational complexity of our model has exponential bounds on the number of binary variables in the problems. Secondly, our frictionless contact assumption has severe implications on the types of tasks we can execute, since many tasks heavily leverage friction to succeed (e.g. pulling an object). Third, the selection of specific limit orientations can define whether **OPT1** is feasible or not for many shapes.

Future work. There are also some natural extensions to this work. First, this framework can also be applied to solving longer horizon tasks, this can be achieved by solving a sequence of short horizon problems with **OPT1** iteratively to move an object through longer trajectory, similar to [5]. Secondly, there remains the need to rigorously analyze the validity of these certificates across different geometries, shapes, and manipulator schemas. Third, it is key to characterize the role of friction on the convergence certificate in order to certify manipulation plans that can be robust under frictional contact. Finally, it is very relevant to explore the role of energy-bounded cages as part of this framework [86], allowing for less geometrically constrained manipulation plans.

7.5 Final Remarks

After exploring these problems and discussing our contributions, there remain some key directions of work that can get us closer enabling versatile dexterity in our robots.

Spatial planning Our world exists in three dimensions where object geometries are rarely constrained to the plane. Through this thesis we have emphasized the application of our algorithms in the planar setup, since it results in a lower dimensional state space and often captures the main complexities of the manipulation problem. However, for robots to truly achieve dexterous behavior these models must be adapted to the spatial world in $SE(3)$. Significant work has been done in this direction [66, 32], showing that robots can discover dexterous behavior under the appropriate system engineering and algorithm conditioning. Extending our planning, learning, and certification algorithms to the 3D world is paramount for their usage outside of laboratories.

Geometry and robustness Geometry is an often overlooked aspect of manipulation and the central them of Chapters 5 and 6. However, due to the relatively minor role of dynamics, geometry often dictates the robustness and versatility of manipulation systems. In the future, our robots along with their policies must be designed with this in mind. Since robot hands have a monopoly on the manipulation world, their selection of their shapes could greatly benefit from the ideas of quasi-dynamics and certification. Similarly, algorithms and controllers should be aware of the role of geometry in the system. Designing controllers and policies that can implicitly adapt to new changes in geometry, potentially leveraging advances in tactile sensing, can allow our robots to act when they see and feel the environment.

Leveraging video data The internet contains innumerable examples of human dexterity through video, demonstrating the capabilities that enable us to serve at homes, hospitals, warehouses, and stores. This represents a major source of data for robots to extract knowledge of how to achieve manipulation skills. Continuing work that combines video de-rendering along with simple physics representations to learn how to solve manipulation tasks for experience is a promising direction towards scaling robot dexterity in the real world. Naturally, this comes with challenges. Such learning models must be able to decode what are the different elements of an environment,

which ones are relevant to the task, and how should objects behave, whether rigid or deformable. However, the advances of large models show that learning systems can scale to vast amounts of unstructured date.

These ideas present a step towards the dream of versatile dexterity. In the quest to enable robots to co-exist and assist in human work, one of our key goals to bridge the human and robot dexterity significantly beyond current capabilities. Mechanics, geometry, and experience are a stepping stone towards this goal.

Bibliography

[1] Bernardo Aceituno, H. Dai, and A. Rodriguez. A convex-combinatorial model for planar caging. In *IROS*. IEEE, 2019.

[2] Bernardo Aceituno-Cabezas, José Ballester, and Alberto Rodriguez. Certified grasping. *ISRR*, 2019.

[3] Bernardo Aceituno-Cabezas, Carlos Mastalli, Hongkai Dai, Michele Focchi, Andreea Radulescu, Darwin G Caldwell, José Cappelletto, Juan C Grieco, Gerardo Fernández-López, and Claudio Semini. Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *RA-L and ICRA*, 2018.

[4] Bernardo Aceituno-Cabezas and Alberto Rodriguez. A global quasi-dynamic model for contact-trajectory optimization. In *RSS*, 2020.

[5] Bernardo Aceituno-Cabezas and Alberto Rodriguez. A hierarchical framework for long horizon planning of object-contact trajectories. In *IROS*, 2022.

[6] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. *NeurIPS*, 2019.

[7] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *NeurIPS*, 2019.

[8] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *NeurIPS*, 2016.

[9] Thomas F Allen, Joel W Burdick, and Elon Rimon. Two-finger caging of polygonal objects using contact space search. *T-RO*, 2015.

[10] Thomas F Allen, Elon Rimon, and Joel W Burdick. Robust three-finger three-parameter caging of convex polygons. In *ICRA*. IEEE, 2015.

[11] Aaron D Ames, Kevin Galloway, Koushil Sreenath, and Jessy W Grizzle. Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics. *T-AC*, 2014.

[12] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *T-AC*, 2016.

[13] Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J Zico Kolter. Differentiable mpc for end-to-end planning and control. In *NeurIPS*, 2018.

[14] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. *NeurIPS*, 2017.

[15] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *IJRR*, 2020.

[16] Thomas Auer. Rpg-heuristics for the generation of random polygons. Citeseer, 1996.

[17] alp Aydinoglu, Victor Preciado, and Michael Posa. Contact-aware controller design for complementarity systems. In *ICRA*, 2020.

[18] Devin Balkcom and Jeffrey C. Trinkle. Computing wrench cones for planar rigid body contact tasks. *IJRR*, 2002.

[19] Peter W. Battaglia, Jessica B. Hamrick, and Joshua B. Tenenbaum. Simulation as an engine of physical scene understanding. *PNAS*, 2013.

[20] Maria Bauza, Ferran Alet, Yen-Chen Lin, Tomás Lozano-Pérez, Leslie P Kaelbling, Phillip Isola, and Alberto Rodriguez. Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgb-d video. In *IROS*. IEEE, 2019.

[21] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

[22] Dimitris Bertsimas and Robert Weismantel. *Optimization over integers*, volume 13. 2005.

[23] Antonio Bicchi and Vijay Kumar. Robotic grasping and contact: A review. In *ICRA*. IEEE, 2000.

[24] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[25] Hallel A Bunis, Elon D Rimon, Thomas F Allen, and Joel W Burdick. Equilateral three-finger caging of polygonal objects using contact space search. *T-ASE*, 2018.

[26] Robert R Burridge, Alfred A Rizzi, and Daniel E Koditschek. Sequential composition of dynamically dexterous robot behaviors. *IROS*, 1999.

[27] Abhishek Cauligi, Preston Culbertson, Bartolomeo Stellato, Dimitris Bertsimas, Mac Schwager, and Marco Pavone. Learning mixed-integer convex optimization strategies for robot planning and control. *arXiv preprint arXiv:2004.03736*, 2020.

[28] M Chavan-Dafle, R Holladay, and A Rodriguez. Planar in-hand manipulation via motion cones. In *IJRR*, 2018.

[29] Nikhil Chavan-Dafle, Rachel Holladay, and Alberto Rodriguez. In-hand manipulation via motion cones. In *RSS*, 2018.

[30] Nikhil Chavan-Dafle and Alberto Rodriguez. Sampling-based planning of in-hand manipulation with external pushes. In *ISRR*. 2017.

[31] Claire Chen, Preston Culbertson, Marion Lepert, Mac Schwager, and Jeannette Bohg. Trajectotree: Trajectory optimization meets tree search for planning multi-contact dexterous manipulation. In *IROS*, 2021.

[32] Xianyi Cheng, Eric Huang, Yifan Hou, and Matthew T Mason. Contact mode guided sampling-based planning for quasistatic dexterous manipulation in 2d. In *ICRA*, 2021.

[33] E. Chuah, L. Epstein, L. Donghyun, J. Romero, and S. Kim. Bi-modal hemispherical sensor: A unifying solution for three axis force and contact angle measurement. In *IROS*, 2019.

[34] 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 *ICRA*. IEEE, 2014.

[35] Hongkai Dai, Gregory Izatt, and Russ Tedrake. Global inverse kinematics via mixed-integer convex optimization. In *ISRR*, 2017.

[36] Hongkai Dai, Gregory Izatt, and Russ Tedrake. Global inverse kinematics via mixed-integer convex optimization. *IJRR*, 2019.

[37] Hongkai Dai, Anirudha Majumdar, and Russ Tedrake. Synthesis and optimization of force closure grasps via sequential semidefinite programming. In *ISRR*. Springer, 2017.

[38] Hongkai Dai, Andrés Valenzuela, and Russ Tedrake. Whole-body motion planning with centroidal dynamics and full kinematics. In *Humanoids*. IEEE, 2014.

[39] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *ECCV*, 2018.

[40] Neha Das, Sarah Bechtle, Todor Davchev, Dinesh Jayaraman, Akshara Rai, and Franziska Meier. Model-based inverse reinforcement learning from visual demonstrations. *arXiv preprint arXiv:2010.09034*, 2020.

[41] Filipe de Avila Belbute-Peres, Kevin Smith, Kelsey Allen, Josh Tenenbaum, and J Zico Kolter. End-to-end differentiable physics for learning and control. In *NeurIPS*, 2018.

[42] Robin Deits, Twan Koolen, and Russ Tedrake. Lvis: Learning from value function intervals for contact-aware robot controllers. In *ICRA*. IEEE, 2019.

[43] Robin Deits and Russ Tedrake. Footstep planning on uneven terrain with mixed-integer convex optimization. In *Humanoids*. IEEE, 2014.

[44] Robin Deits and Russ Tedrake. Computing large convex regions of obstacle-free space through semidefinite programming. In *WAFR*. Springer, 2015.

[45] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *J-MLR*, 2016.

[46] Mehmet Dogar and Siddhartha Srinivasa. A framework for push-grasping in clutter. *RSS*, 2011.

[47] Mehmet R Dogar and Siddhartha S Srinivasa. Push-grasping with dexterous hands: Mechanics and a method. In *IROS*. IEEE, 2010.

[48] Jing Dong, Mustafa Mukadam, Frank Dellaert, and Byron Boots. Motion planning as probabilistic inference using gaussian processes and factor graphs. In *RSS*, 2016.

[49] Elliott Donlon, Siyuan Dong, Melody Liu, Jianhua Li, Edward Adelson, and Alberto Rodriguez. Gelslim: A high-resolution, compact, robust, and calibrated tactile-sensing finger. In *IROS*. IEEE, 2018.

[50] N Doshi, F Hogan, and A Rodriguez. Hybrid differential dynamic programming for planar manipulation primitives. In *ICRA*, 2020.

[51] Michael A Erdmann and Matthew T Mason. An exploration of sensorless manipulation. *J-RA*, 1988.

[52] Nima Fazeli, Miquel Oller, Jiajun Wu, Zheng Wu, Joshua B Tenenbaum, and Alberto Rodriguez. See, feel, act: Hierarchical learning for complex manipulation skills with multisensory fusion. *Science Robotics*, 2019.

[53] Siyuan Feng, Eric Whitman, X Xinjilefu, and Christopher G Atkeson. Optimization-based full body control for the darpa robotics challenge. *Journal of Field Robotics*, 2015.

[54] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NeurIPS*, 2016.

[55] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *CoRL*, 2018.

[56] Christodoulos A Floudas. *Nonlinear and mixed-integer optimization: fundamentals and applications*. Oxford University Press, 1995.

[57] Steven Fortune. Voronoi diagrams and delaunay triangulations. In *Computing in Euclidean geometry*. World Scientific, 1995.

[58] Kenneth Y Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 1993.

[59] Inc. Gurobi Optimization. Gurobi optimizer reference manual, 2022.

[60] Maximilian Haas-Heger, Garud Iyengar, and Matei Ciocarlie. Passive reaction analysis for grasp stability. *T-ASE*, 2018.

[61] Weiqiao Han and Russ Tedrake. Local trajectory stabilization for dexterous manipulation via piecewise affine approximations. In *ICRA*, 2020.

[62] Kaiyu Hang, Johannes A Stork, Nancy S Pollard, and Danica Kragic. A framework for optimal grasp contact planning. *RA-L*, 2017.

[63] Charles R Hargraves and Stephen W Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of guidance, control, and dynamics*, 1987.

[64] Randy L Haupt. Antenna design with a mixed integer genetic algorithm. *Transactions on Antennas and Propagation*, 2007.

[65] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[66] F Hogan, J Ballester, S Dong, and A Rodriguez. Tactile dexterity: Manipulation primitives with tactile feedback. In *ICRA*, 2020.

[67] F. Hogan and A. Rodriguez. Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics. In *WAFR*, 2016.

[68] F. Hogan, E. Romo, and A. Rodriguez. Reactive planar manipulation with convex hybrid mpc. In *ICRA*, 2018.

[69] Francois R Hogan and Alberto Rodriguez. Reactive planar non-prehensile manipulation with hybrid model predictive control. *IJRR*, 2020.

[70] Yifan Hou, Zhenzhong Jia, and Matthew T Mason. Fast planning for 3d any-pose-reorienting using pivoting. In *ICRA*. IEEE, 2018.

[71] Yifan Hou and Matthew T Mason. Robust execution of contact-rich motion plans by hybrid force-velocity control. In *ICRA*. IEEE, 2019.

[72] D.W. (https://cs.stackexchange.com/users/755/d w). Express boolean logic operations in zero-one integer linear programming (ilp). Computer Science Stack Exchange.

[73] Miguel Jaques, Michael Burke, and Timothy Hospedales. Physics-as-inverse-graphics: Joint unsupervised learning of objects and physics from video. In *ICLR*, 2019.

[74] Aaron M Johnson, Jennifer E King, and Siddhartha Srinivasa. Convergent planning. *RA-L*, 2016.

[75] Mrinal Kalakrishnan, Sachin Chitta, Evangelos Theodorou, Peter Pastor, and Stefan Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *ICRA*. IEEE, 2011.

[76] BK Kannan and Steven N Kramer. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. 1994.

[77] Alina Kloss, Maria Bauza, Jiajun Wu, Joshua B Tenenbaum, Alberto Rodriguez, and Jeannette Bohg. Accurate vision-based manipulation through contact reasoning. In *ICRA*. IEEE, 2020.

[78] Scott Kuindersma, Robin Deits, Maurice Fallon, Andrés Valenzuela, Hongkai Dai, Frank Permenter, Twan Koolen, Pat Marion, and Russ Tedrake. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. *Autonomous robots*, 2016.

[79] Wlodzimierz Kuperberg. Problems on polytopes and convex sets. In *DIMACS Workshop on polytopes*, 1990.

[80] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

[81] Jyh-Ming Lien and Nancy M Amato. Approximate convex decomposition of polygons. *Computational Geometry*, 2006.

[82] C. Karen Liu. Dextrous manipulation from a grasping pose. In *SIGGRAPH*, 2009.

[83] Winfried Lohmiller and Jean-Jacques E Slotine. On contraction analysis for non-linear systems. *Automatica*, 1998.

[84] T Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 1983.

[85] Tomas Lozano-Perez, Matthew T Mason, and Russell H Taylor. Automatic synthesis of fine-motion strategies for robots. *IJRR*, 1984.

[86] Jeffrey Mahler, Florian T Pokorny, Zoe McCarthy, A Frank van der Stappen, and Ken Goldberg. Energy-bounded caging: Formal definition and 2-d energy lower bound algorithm based on weighted alpha shapes. *RA-L*, 2016.

[87] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *ICRA*. IEEE, 2013.

[88] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *IJRR*, 2017.

[89] Satoshi Makita and Yusuke Maeda. 3d multifingered caging: Basic formulation and planning. In *IROS*. IEEE, 2008.

[90] Satoshi Makita, Kensuke Okita, and Yusuke Maeda. 3d two-fingered caging for two types of objects: Sufficient conditions and planning. *International Journal of Mechatronics and Automation*, 2013.

[91] Zachary Manchester and Scott Kuindersma. Variational contact-implicit trajectory optimization. In *ISRR*. Springer, 2017.

[92] Tobia Marcucci and Russ Tedrake. Mixed-integer formulations for optimal control of piecewise-affine systems. In *HSCC*, 2019.

[93] Matthew T Mason. *Mechanics of robotic manipulation*. 2001.

[94] Matthew T Mason, Alberto Rodriguez, Siddhartha S Srinivasa, and Andrés S Vazquez. Autonomous manipulation with a general-purpose simple hand. *IJRR*, 2012.

[95] Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part i—convex underestimating problems. *Mathematical programming*, 1976.

[96] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. Curran Associates, Inc., 2019.

[97] Guilherme AS Pereira, Mario FM Campos, and Vijay Kumar. Decentralized algorithms for multi-robot manipulation via caging. *IJRR*, 2004.

[98] Peam Pipattanasomporn and Attawith Sudsang. Two-finger caging of concave polygon. In *ICRA*. IEEE, 2006.

[99] Brahayam Ponton, Alexander Herzog, Andrea Del Prete, Stefan Schaal, and Ludovic Righetti. On time optimization of centroidal momentum dynamics. In *ICRA*. IEEE, 2018.

[100] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *IJRR*, 2014.

[101] Domenico Prattichizzo and Jeffrey C Trinkle. Grasping. In *Springer handbook of robotics*. Springer, 2016.

[102] Yi-Ling Qiao, Junbang Liang, Vladlen Koltun, and Ming C Lin. Scalable differentiable physics for learning and control. *ICML*, 2020.

[103] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[104] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *ICRA*. IEEE, 2009.

[105] Arthur Richards and Jonathan How. Mixed-integer programming for control. In *ACC*. IEEE, 2005.

[106] Elon Rimon and Andrew Blake. Caging 2d bodies by 1-parameter two-fingered gripping systems. In *ICRA*. IEEE, 1996.

[107] Elon Rimon and Andrew Blake. Caging planar bodies by one-parameter two-fingered gripping systems. *IJRR*, 1999.

[108] Elon Rimon and Joel Burdick. On force and form closure for multiple finger grasps. In *ICRA*. IEEE, 1996.

[109] Alberto Rodriguez and Matthew T. Mason. Path-connectivity of the free space. *T-RO*, 2012.

[110] Alberto Rodriguez and Matthew T Mason. Effector form design for 1dof planar actuation. In *ICRA*. IEEE, 2013.

[111] Alberto Rodriguez, Matthew T Mason, and Steve Ferry. From caging to grasping. *IJRR*, 2012.

[112] Sadra Sadraddini and Russ Tedrake. Sampling-based polytopic trees for approximate optimal control of piecewise affine systems. In *ICRA*. IEEE, 2019.

[113] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *RSS*, 2013.

[114] Moshe Shimrat. Algorithm 112: position of point relative to polygon. *Communications of the ACM*, 1962.

[115] Sumeet Singh, Anirudha Majumdar, Jean-Jacques Slotine, and Marco Pavone. Robust online motion planning via contraction theory and convex optimization. In *ICRA*, 2017.

[116] Jean-Pierre Sleiman, Jan Carius, Ruben Grandia, Martin Wermelinger, and Marco Hutter. Contact-implicit trajectory optimization for dynamic object manipulation. In *IROS*. 2019.

[117] Johannes A Stork, Florian T Pokorny, and Danica Kragic. A topology-based object representation for clasping, latching and hooking. In *Humanoids*. IEEE, 2013.

[118] Attawith Sudsang and Jean Ponce. A new approach to motion planning for disc-shaped robots manipulating a polygonal object in the plane. In *ICRA*. IEEE, 2000.

[119] HJ Suh and Russ Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. *WAFR*, 2020.

[120] Andrew Taylor, Andrew Singletary, Yisong Yue, and Aaron Ames. Learning for safety-critical control with control barrier functions. In *Learning for Dynamics and Control*. PMLR, 2020.

[121] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *IJRR*, 2010.

[122] Stephen Tian, Frederik Ebert, Dinesh Jayaraman, Mayur Mudigonda, Chelsea Finn, Roberto Calandra, and Sergey Levine. Manipulation by feel: Touch-based control with deep predictive models. In *ICRA*. IEEE, 2019.

[123] Emanuel Todorov. A convex, smooth and invertible contact model for trajectory optimization. In *ICRA*. IEEE, 2011.

[124] Marc Toussaint, Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *RSS*, 2018.

[125] James Usevitch and Dimitra Panagou. Determining r-and (r, s)-robustness of digraphs using mixed integer linear programming. *Automatica*, 2020.

[126] Mostafa Vahedi and A Frank van der Stappen. Caging polygons with two and three fingers. *IJRR*, 2008.

[127] Andrés Klee Valenzuela. *Mixed-integer convex optimization for planning aggressive motions of legged robots over rough terrain*. PhD thesis, Massachusetts Institute of Technology, 2016.

[128] Anastasiia Varava, J Frederico Carvalho, Florian T Pokorny, and Danica Kragic. Caging and path non-existence: a deterministic sampling-based verification algorithm. In *ISRR*, 2017.

[129] Anastasiia Varava, J Frederico Carvalho, Florian T Pokorny, and Danica Kragic. Free space of rigid objects: Caging, path non-existence, and narrow passage detection. *WAFR*, 2018.

[130] Anastasiia Varava, J Frederico Carvalho, Florian T Pokorny, and Danica Kragic. Free space of rigid objects: Caging, path non-existence, and narrow passage detection. In *WAFR*, 2018.

[131] Anastasiia Varava, Kaiyu Hang, Danica Kragic, and Florian T Pokorny. Herding by caging: a topological approach towards guiding moving agents via mobile robots. In *RSS*, 2017.

[132] Anastasiia Varava, Danica Kragic, and Florian T Pokorny. Caging grasps of rigid and partially deformable 3-d objects with double fork and neck features. *T-RO*, 2016.

[133] Weiwei Wan, Rui Fukui, Masamichi Shimosaka, Tomomasa Sato, and Yasuo Kuniyoshi. Grasping by caging: A promising tool to deal with uncertainty. In *ICRA*. IEEE, 2012.

[134] H Paul Williams. *Model building in mathematical programming*. John Wiley & Sons, 2013.

[135] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B Tenenbaum, and Shuran Song. Densephysnet: Learning dense physical object representations via multi-step dynamic interactions. In *RSS*, 2019.

[136] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, et al. Transporter networks: Rearranging the visual world for robotic manipulation. *CoRL*, 2020.

[137] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *T-RO*, 2020.

[138] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In *IROS*. IEEE, 2018.

[139] Jiaji Zhou, Matthew T Mason, Robert Paolini, and Drew Bagnell. A convex polynomial model for planar sliding mechanics: theory, application, and experimental validation. *IJRR*, 2018.

[140] Jiaji Zhou, Robert Paolini, J Andrew Bagnell, and Matthew T Mason. A convex polynomial force-motion model for planar sliding: Identification and application. In *ICRA*. IEEE, 2016.