Optimisation-Based Manipulation Planning in Convex Decompositions of C-free

Julia López Gómez

MInf Project (Part 2) Report
Master of Informatics
School of Informatics
University of Edinburgh

2025

Abstract

Manipulation planning involves computing sequences of robot-object interactions that allow a robot to move and reorient objects in environments with obstacles. Traditional approaches to this problem rely heavily on sampling-based methods, which provide limited guarantees of optimality, completeness and non-collision. This project proposes a novel optimisation-based manipulation planner that leverages recent advances in convex decomposition of the configuration space to enable planning over certified collision-free regions.

The planner formulates a Mixed-Integer Quadratic Program (MIQP) to generate a sequence of grasp and release actions on the object that solve the manipulation task, integrating manipulation constraints and collision avoidance into a unified optimisation framework. To compute optimal paths between these configurations, it uses Graphs of Convex Sets (GCS). We demonstrate the feasibility of this approach on a simplified bottle cap opening task, which requires multiple regrasp actions. The full pipeline is implemented in Drake and visualised in a 3D configuration space. The result is a complete system that demonstrates how recent geometric tools can be combined to enable practical and certifiable prehensile manipulation planning.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Julia López Gómez*)

Acknowledgements

Firstly, I would like to thank my supervisor, Steve Tonneau. He has not only been a great supervisor over the last two years, but he has helped me immensely in many ways beyond. Thanks to him, I have discovered the field that I want to work in, and I cannot thank him enough. Steve, it has been a pleasure.

To my flatmates, Aloia, Isa and Patri, who have managed to make the word *family* have a different meaning to me. To Luisa and María as well, who have been here since the beginning. Estos cinco años no hubieran sido lo mismo sin vosotras.

A mi madre, que la quiero con locura.

Table of Contents

Chapter 1

Introduction

Imagine a simple everyday task, such as opening a bottle of water. While this is a trivial action for humans, it is very challenging for robots: We must determine how the robot grasps the cap, where to re-grasp it if necessary, and how it avoids collisions with itself and its surroundings. Motion and manipulation planning are foundational challenges in robotics, allowing autonomous systems to navigate and interact with their environments. While motion planning focuses on finding collision-free paths for a robot, manipulation planning extends the task to interacting with movable objects, requiring sequences of grasps, placements, and constrained motions. Solving this problem is crucial for enabling robots to perform complex manipulation tasks in real-world environments, such as household assistance, industrial automation, and healthcare.

In essence, manipulation planning is a combinatorial problem in which we must find a sequence of actions that allow a robot to interact with an object while avoiding collisions. We want our robot to navigate the space of possible grasps, placements, and paths; more specifically, the collision-free submanifold of this space, known as C-free. The description of C-free has been a long-standing challenge in robotics for two main reasons: (1) the space of possible combinations grows exponentially with the robot's degrees of freedom and the number of object interactions, significantly increasing complexity, and, most importantly, (2) we do not know how to describe obstacles in this space. Without properly knowing what C-free looks like, a precise decomposition is challenging – especially in high dimensions – and analytical descriptions are intractable.

Current manipulation planners opt for sampling-based approaches to characterise C-free, such as probabilistic roadmaps (PRMs) or rapidly-exploring random trees (RRTs). While effective in high-dimensions, sampling methods cannot guarantee optimal results, solely reaching asymptotic optimality through specific variants (e.g. RRT*), and only providing probabilistic guarantees of completeness and non-collision. Optimisation-based planning could offer globally optimal solutions, but it requires a proper decomposition of C-free, preferably into convex regions. Previous approaches fail to achieve this in real-world scenarios because they make limiting assumptions to overcome the challenges outlined above, such as simplified geometries, unrealistic convexity requirements, or restricted dimensions, which do not hold in practice. Recent advances in convex decomposition (Dai et al., 2023; Petersen and Tedrake, 2023) allow for the first time a practical way to decompose C-free that works in high dimensions, does not make unrealistic assumptions and enables guarantees of non-collision. Despite still relying on probabilistic coverage of C-free, and depending on seeding for initialisation, these methods have shown promise in motion planning and are sufficient to enable optimisation techniques. However, they have not been explored in manipulation planning. The missing link is a framework that integrates these decomposition algorithms with the geometric constraints of manipulation tasks, a gap this dissertation fills.

We propose a novel manipulation planner that enables optimisation-based trajectory generation by incorporating state-of-the-art algorithms for convex decomposition of C-free. Our approach requires minimal task-specific input about grasps and placements (e.g. what to consider a grasp, or the range of orientations and positions on a surface where the object can be placed at rest) and can scale to higher dimensions. By integrating optimisation techniques with manipulation constraints, our method generates globally optimal trajectories and is capable of providing guarantees of non-collision.

1.1 MInf 1: Previous Work

Part 1 of this project (López Gómez, 2024) focused on understanding how recent advances in the decomposition of the collision-free configuration space (C-free) could be applied to manipulation planning. In particular, we studied the C-IRIS algorithm, which decomposes C-free into convex regions while providing guarantees of non-collision. Unlike prior decomposition methods, which relied on restrictive assumptions such as rare convex requirements or low-dimensional settings, C-IRIS presented a practical alternative capable of scaling to high dimensions. The main objective of Part 1 was to understand the theoretical implications of this algorithm, to outline the pipeline required to solve a manipulation planning problem, and discuss the integration of the two approaches. Our contributions included:

- We proved that C-IRIS could be expanded to support robots with universal joints by decomposing them into combinations of revolute joints.
- We presented the manipulation planning problem using standard mathematical robotics notation and identified the key subspaces involved in manipulation. We also discussed how the classical subspace-based structure introduced by Siméon et al. (2002) could be leveraged within the C-IRIS framework.
- Since C-IRIS operates in the tangent configuration space or TC-space (an alternative representation of configuration space that was newly introduced), we defined and implemented rational forward and inverse kinematics examples in this space (mappings between the joint values of the robot and the position and orientation of the end-effector).
- We proposed formulations for grasping constraints using relative frame transforms for objects with common shapes (cubes, cylinders, spheres, prisms).
- We reproduced the original C-IRIS implementation and ran experiments to study the seeding process of C-IRIS during initialisation, showing that performance was highly dependent on seed location.

The primary goal of Part 1 was to understand the decomposition approach offered by C-IRIS and study how it could be extended to cover realistic manipulation tasks. In this second part of the project, we build upon these insights to design and implement a complete manipulation planner.

1.2 Project Aims and Limitations

The aim of this dissertation is to formulate a complete manipulation planner that leverages convex optimisation to produce globally optimal, collision-free trajectories. Building on the theoretical foundation developed in Part 1, we now combine recent advances in C-free decomposition with the structure of manipulation planning to generate full plans that require minimal manual input and provide practical guarantees of feasibility and efficiency. Our approach demonstrates that convex decomposition methods such as C-IRIS, when integrated into a structured manipulation pipeline, can be applied to realistic manipulation tasks and offer guarantees of optimality, which was previously unattainable as previous manipulation planners have relied on sampling.

This work focuses specifically on *prehensile manipulation*, where the object is grasped and lifted, as opposed to other contact-rich tasks such as pushing or pivoting. The key limitations of our work lie in the decomposition of C-free. Although algorithms like C-IRIS enable certified convex decomposition without strong assumptions, they still rely on approximate coverage of C-free. This is particularly challenging in manipulation tasks, where grasping actions take place very close to collisions, and insufficient coverage may prevent feasible plans from being found, even when a valid solution exists.

1.3 Achievements and Contributions

This project presents a new manipulation planner that uses convex optimisation to compute feasible trajectories capable of displacing an object while providing non-collision guarantees. This work addresses the limitations of traditional sampling-based methods by leveraging recent advances in certified convex decomposition of the collision-free configuration space.

The two central contributions of this project are:

- **A novel optimisation-based manipulation planner:** We propose an algorithm that formulates the manipulation planning problem as a Mixed-Integer Quadratic Program (MIQP), only requiring input about the specific task constraints and the scene, and automatically generating the minimal required grasps and placements to achieve the goal. (Chapters 4 and 6).
- **A full working pipeline and experimental demonstration:** We demonstrate the feasibility of the planner through a complete pipeline applied to a representative manipulation task: opening a bottle by rotating its cap, involving multiple grasps and collision avoidance. (Chapter 5).

In addition to the main contributions, the following supporting work was developed:

- **A novel formulation of the grasp and placement spaces (*CG* and *CP*)** using simple geometric bounds and their integration with certified free-space regions to define CG_{free} and CP_{free}. This formulation enables the use of optimisation-based methods for reasoning about contact and collision together. (Section 5.3.1).
- **A custom polytope conversion method** that maps configuration-space polytopes into the tangent configuration space (TC-space) and vice versa, enabling certification of collision-freedom using C-IRIS. (Section 5.3.2 and Algorithm 2).
- **C-free decomposition using Clique Covers and trajectory generation using Graphs of Convex Sets (GSC):** We integrated modern state-of-the-art methods for addressing the limitations of C-free decomposition in terms of coverage and seeding (using Clique Covers), and to generate optimal trajectories within regions of C-free (using GCS). (Sections 5.3.2 and 5.5).
- **Experimental analysis of limitations in C-free coverage and performance:** We conducted a series of experiments to evaluate the performance of our planner, focusing on the limitations of C-free coverage and the impact of dimensionality. We also showed the scalability of our method to systems with 3 and 4 DOF. (Chapter 7).

Finally, a key achievement of this project has been the ability to understand and critically evaluate recent state-of-the-art methods such as C-IRIS IRIS-NP, Clique Covers and GCS, and to adapt and integrate them into our planning framework. These methods are recent, and their application to manipulation planning had not been demonstrated prior to this work. This project shows how they can be combined to enable optimisation-based manipulation planning with strong guarantees of feasibility and completeness. A full implementation can be found in this GitHub repository*.

1.4 Report Structure

The content of each chapter has been distributed as follows:

Chapter 2 discusses relevant literature and key notions in robotics to understand the planning problem formalisation, as well as the implications of this work. **Chapter 3** introduces the theoretical background of (1) the formulation of manipulation planning as a constraint motion planning problem, (2) the decomposition of the C-free into convex regions through different methods, (3) Mixed-Integer Programming (MIP) for optimisation-based planning, and (4) the use of the Drake Robotics Toolbox for visualisation and simulation. **Chapter 4** presents the problem statement and assumptions made in this work. **Chapter 5** introduces the task and scene description for opening a bottle cap through rotation to showcase the pipeline of our algorithm and the constraint definition process. **Chapter 6** describes the implementation of the manipulation planning algorithm, following the pipeline built in Chapter 5. **Chapter 7** presents an experimental analysis of the limitations of our method, focusing on C-free coverage and performance in 3 and 4 degrees of freedom. Finally, **Chapter 8** concludes the report, summarising the contributions and discussing future research directions.

*`https://github.com/julialopezgomez/minf2-repo`.

Chapter 2

Definitions and Previous Work

2.1 Key Concepts in Motion and Manipulation Planning

Motion planning encompasses the challenge of describing how a robot can move from one location to another in a given workspace. In contrast, **manipulation planning** extends this challenge to the grasping and displacement of objects. Both problems entail avoiding collisions with obstacles.

To describe how a robot moves, we need to determine where exactly it is in space: we need to define its configuration. The **configuration** of a robot captures the precise position and orientation of all its components, i.e. the full geometric state of the robot. A configuration is typically represented by a vector of its joint values,

$$\mathbf{q} = [q_1, q_2, \dots, q_n],$$

where n is the number of **degrees of freedom** (DOF) of the robot system, and each q_i represents the value of a joint. Depending on how the system is modelled, the configuration vector may lie in a higher-dimensional space $\mathbb{R}^m$, with $m \geq n$.

The space of all possible configurations is called the **configuration space** (C-space), and the dimension of the C-space corresponds to the number of elements in the configuration vector ($m \geq n$). Each point in the C-space represents a unique configuration of the robot. However, not all configurations are feasible in practice: some may lead to collisions with obstacles. The **collision-free configuration space** (C-free) is the submanifold of C-space where the robot can move without colliding with obstacles, with C-free $\subseteq$ C-space. The remaining configurations are in C-obs, representing collisions. Figure 2.1 illustrates this concept. C-free and C-obs are complementary spaces, i.e. C-space $=$ C-free $\cup$ C-obs and C-free $\cap$ C-obs $= \emptyset$. In contrast to the C-space, the **task space** is the space where the robot operates, typically defined by the position and orientation of the robot's **end-effector**. In Figure 2.1, (A) and (B) represent the robots in task space, while (C) and (D) show an abstract representation of the corresponding C-spaces.

Definitions of these concepts can be found in Appendix B. We recommend Sections 2.0, 2.2.0, 2.2.1, 2.3.2, and 2.5 of Lynch and Park (2017) for more detailed explanations.

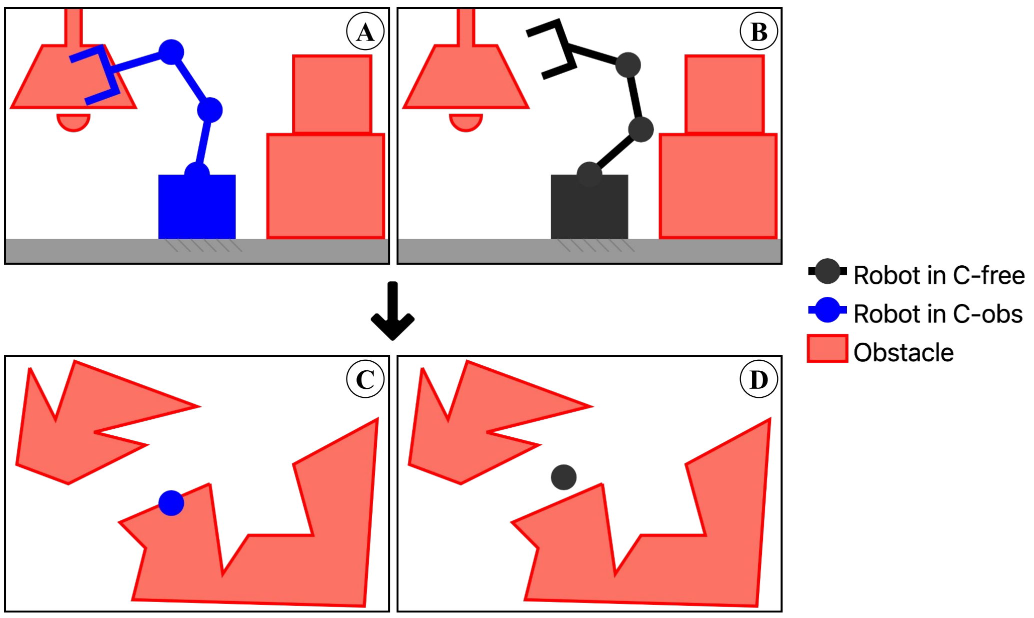

Figure 2.1: (A) The configuration of the blue robot belongs to C-obs because it is in collision with the obstacles. (B) The configuration of the black robot belongs to C-free because it is not in collision. (C) and (D) represent an abstract 3D representation of the configuration space of the robot, which in reality would have a higher dimension.

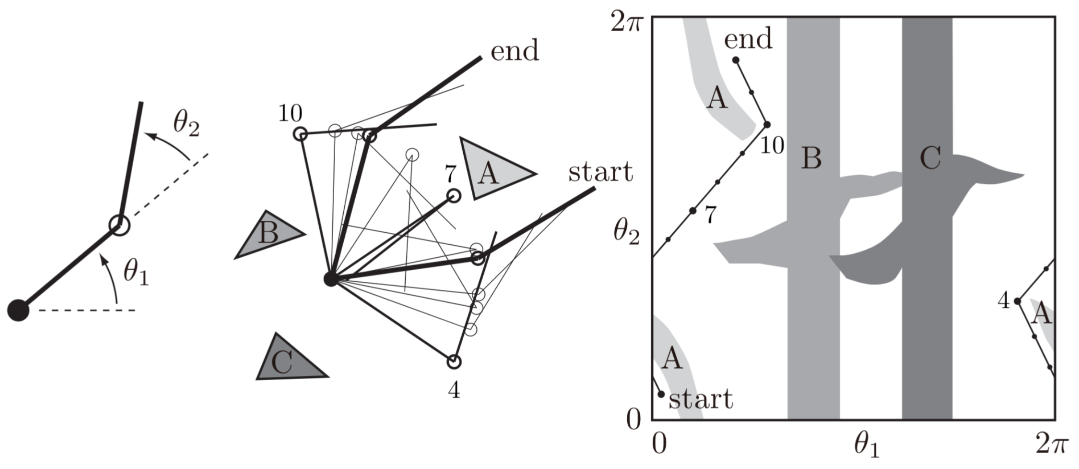

Figure 2.2: Illustration of motion planning for a two-DOF double pendulum. (**Left**) The pendulum with θ_1 and θ_2 representing its joint angles. (**Middle**) Its motion in task space, avoiding obstacles A, B, and C. (**Right**) The corresponding trajectory in C-space, where C-free (white) represents valid configurations and obstacles (grey) indicate collisions. Adapted from Lynch and Park (2017).

Motion planning requires navigating through all possible robot postures and avoiding obstacles, and therefore it occurs in C-space. Understanding and describing C-free is as essential as challenging to find collision-free trajectories. We do not know how to describe obstacles in C-space directly; therefore, we need to map them from task space (known as inverse kinematics). Mapping often results in complex, unintuitive regions, and we usually only have access to a collision checker that tells us whether a configuration is valid or not. Likewise, task-specific constraints (e.g. end-effector orientation) and goals are defined in task space and must be mapped into C-space.

To illustrate motion planning in a real example, Figure 2.2 shows a 2-DOF double pendulum navigating around obstacles. The **left** image depicts the pendulum with joint angles θ_1 and θ_2. The **centred** image shows the pendulum moving from the `start` to `end` position in task space, avoiding obstacles A, B, and C. The **right** image shows the same path in C-space, where each point in the path represents a configuration (θ_1, θ_2). In this C-space representation, the white background represents C-free, and the greyed areas represent C-obs. The motion from `start` to `end` follows a path within C-free.

A glossary of key terms can be found in Appendix A.

2.2 Previous Work

This section reviews the literature on motion and manipulation planning, focusing on two main themes: (1) the decomposition of the collision-free configuration space (C-free) and (2) the evolution of manipulation planning from discrete, user-defined grasps to more general, modern planners.

We highlight the recent advancements in motion planning, particularly the development of methods that decompose C-free into simpler sets, such as convex regions, as these approaches enable scalability to higher dimensions and the use of optimisation-based techniques for trajectory generation. In manipulation planning, we discuss the evolution of geometric approaches and why the recent advancements in motion planning are relevant to this field and remain unexplored.

2.2.1 Motion Planning and C-free Decomposition

In the early 1980s, Lozano-Perez (1983) presented the notion of configuration space as an approach to motion planning for the first time. Ever since, it has been foundational in robotics to describe C-free to facilitate the later computation of paths or trajectories within this non-collision area. The complexity of describing C-space obstacles and the corresponding C-free has led to the use of two distinct approaches:

1. **The negative approach** - describes the C-space obstacles directly from their task space description. Then, C-free is described as their complement.
2. **The positive approach** - directly describes C-free as a union of simpler sets.

For the **negative approach**, Canny (1988) demonstrated that describing obstacles in task space for motion planning is an intractable (NP-hard) problem. Consequently, researchers were forced to make limiting assumptions about the robot's capabilities. For instance, only allowing robots to translate and not rotate (Kavraki, 1995), restricting the robot to two or three DOF (Branicky and Newman, 1990), or sacrificing accuracy by describing the obstacles as sets of spheres (Hubbard, 1996). A full review on describing C-space obstacles can be found in (Latombe, 2012, Chapter 3).

The **positive approach** breaks down C-free directly into simpler, more manageable sets. This approach is particularly appealing because it enables the use of optimisation-based motion planning methods, which are especially efficient when the simpler sets are convex (Schouwenaars et al., 2001; Deits and Tedrake, 2015b; Marcucci et al., 2022). By decomposing C-free into convex or approximately convex regions, planners can take advantage of the properties of convex sets to compute paths more efficiently.

A variety of methods have been developed to decompose C-free into simpler sets:

- **Sampling-Based Methods**: Rapidly-exploring Random Trees (RRT) (LaValle, 1998) and Probabilistic Roadmaps (PRM) (Kavraki et al., 1996) are widely used in motion planning. These methods describe C-free by sampling configurations in C-space and connecting them through linear paths. While considered examples of the positive approach –they ”decompose” C-free into a connected graph of linear paths– they do not represent C-free as a union of simpler sets. Thus, they cannot directly integrate with optimisation algorithms. Sampling methods are highly effective in high-dimensional spaces, but they only offer probabilistic guarantees of non-collision and completeness[*]. Additionally, they do not inherently generate optimal trajectories, although variants like RRT* (Karaman and Frazzoli, 2011) asymptotically approach optimality with additional computation.
- **Exact Decomposition**: Under the assumption of convex obstacles in C-space (rare in practice), Lingas (1982) proved that finding a minimal decomposition of C-free is NP-hard even for only two and three dimensions, and approximating it is APX-hard[†] (Eidenbenz and Widmayer, 2003).
- **Convex Decomposition**: To overcome the hardness results of exact and approximate decomposition, works like Lien and Amato (2007) or Ghosh et al. (2013) achieved a C-free decomposition into unions of approximately convex sets. The IRIS algorithm (Deits and Tedrake, 2015a) represents a significant advancement in this area, as it can decompose C-free into convex polytopes and work in arbitrary dimensions while providing certificates of non-collision. However, IRIS assumed that obstacles in C-space are convex, which presented a challenge as this is rarely the case in practice (see Figure 2.2. Obstacles A, B, and C are convex in task space –centred image–, but not in C-space –rightmost image).

Recent Advancements in Convex Decomposition Recent works have addressed the limitations of IRIS by relaxing the assumption of convex obstacles in C-space. Dai et al. (2023) proposed the C-IRIS algorithm, capable of decomposing C-free in convex regions guaranteed to be collision-free. This novel work *only* makes assumptions on the convexity of obstacles in task space, which is attainable (Mesadi and Tasdizen, 2016), and works in relatively high dimensions. C-IRIS provides a baseline in motion planning to compute trajectories that are guaranteed to be collision-free and can scale to real-world scenarios.

[*] A *probabilistically complete* algorithm is such that, if a feasible path exists, it will eventually find it regardless of the dimension or complexity of the C-space (Kleinbort et al., 2019).

[†] An APX-hard problem is such that no polynomial-time algorithm can approximate its solution within a factor of $1+\delta$ of the optimum, for some $\delta > 0$, unless $P = NP$.

Building on this foundation, newer variants of IRIS have been developed to further improve scalability and applicability:

- IRIS-NP (Petersen and Tedrake, 2023): Extends IRIS to handle non-convex obstacles. Developed in parallel to C-IRIS, IRIS-NP trades collision-free guarantees for computational efficiency, making it suitable for high-dimensional systems.
- IRIS-NP2 (Werner et al., 2024b): A refined version of IRIS-NP, significantly faster and capable of building larger polytopes.
- IRIS-ZO (Werner et al., 2024b): Focuses on zero-order optimisation techniques, also improving the efficiency of IRIS-NP.

C-IRIS is the only one of these methods that explicitly decomposes C-free into convex polytopes that are *guaranteed to be collision-free*. Nonetheless, C-IRIS's certificates of non-collision can be used to certify the collision-free nature of the other IRIS variants. While effective, these methods rely on manual seeding, requiring the user to provide initial small collision-free regions to start the decomposition process and grow the polytopes. Werner et al. (2024a) proposed a method to automate this seeding process using Clique Covers, further improving the efficiency of the decomposition. In addition, to complement the development of the IRIS variants, Marcucci et al. (2022) introduced a method to generate optimal trajectories within the certified convex decomposition of C-free enabling the use of efficient solvers to compute globally optimal solutions.

These advancements represent the current state-of-the-art in motion planning, providing efficient methods for decomposing and exploring C-free while working in arbitrary dimensions and relaxing the limiting assumption of convex obstacles in C-space, making them suitable for real-world applications. In summary, these methods still achieve probabilistic coverage of the C-space, are complete and globally optimal within the convex decomposition, and can leverage efficient solutions (Tedrake, 2023b, Minute 12:00).

2.2.2 Manipulation Planning

While motion planning focuses on planning collision-free paths for robots amongst obstacles (Lozano-Perez, 1983), manipulation planning extends this problem by introducing **movable objects**, i.e. objects that can only be moved by a robot. That is, either the movable objects are being transported by the robot (**transfer paths**) or they are at rest at a stable placement (**transit paths**) (Alami et al., 1990). The solution to a manipulation planning problem involves generating a sequence of subpaths that satisfy these motion restrictions, resulting in a constrained version of the classical motion planning problem (Latombe, 2012).

Early Methods and Discrete Set Limitation In 1990, Alami et al. (1990) formulated manipulation planning as an alternating sequence of transit and transfer paths separated by grasping and ungrasping actions. Ever since, this framework has been explored by numerous works. Early approaches in the 1990s relied on a discrete set of grasp and placement configurations, which had to be specified manually by the user (Koga and Latombe, 1994; Barraquand and Ferbach, 1994; Ahuactzin et al., 1998). This was a serious limitation, as the solver required prior knowledge of every configuration in which the object must be grasped from or placed on a stable surface, and missing any one of them meant the planner would fail to find a solution. In complex scenarios, these intermediate configurations could be numerous, or complicated to predict. An essential part of the manipulation task was essentially being solved by the user.

Sampling-Based Methods for Continuous Grasps and Placements By the late 1990s, randomised motion planning had proven effective for high-dimensional robots (LaValle, 1998; Kavraki et al., 1996). This led to probabilistic roadmap (PRM) planners being applied to manipulation, although still requiring predefined grasp and placement configurations as input (Nielsen and Kavraki, 2000). A significant breakthrough occurred when researchers began to treat grasp and placement constraints as continuous spaces instead of discrete sets (Siméon et al., 2002; Sahbani et al., 2002; Siméon et al., 2004b). Siméon et al. (2004a) introduced a general manipulation planner that represented all possible grasps and placements as continuous domains. This formulation presented a significant shift, enabling the planner to generate the necessary intermediate grasps and placements automatically. Siméon et al. (2004a)'s work essentially automated the task decomposition that earlier methods left to the user. Despite its success, this approach still relied on sampling-based planners, which only offer probabilistic guarantees of non-collision and completeness and do not inherently generate optimal trajectories or allow for the use of optimisation-based methods.

Recent Developments: Learning-Based Methods Recent work has explored learning-based approaches to manipulation. Reinforcement learning (RL) and learning from demonstration (LfD) techniques have shown promise in enabling robots to acquire manipulation skills in unstructured environments (Suomalainen et al., 2022; Elguea-Aguinaco et al., 2023; Ravichandar et al., 2020). However, these methods typically focus on policy learning rather than planning, and often require large amounts of training data, careful reward definition, and are difficult to generalise or guarantee safety. Therefore, this dissertation focuses on geometric and optimisation-based planning, which remains more reliable for tasks with structured constraints and safety requirements.

Manipulation Planning in a Decomposed C-free Previous to the recent contributions in convex C-free decomposition described in the motion planning section (Section 2.2.1), for most of the past decades, manipulation planning has relied primarily on sampling-based representations of C-free. To the best of our knowledge, manipulation frameworks like Siméon et al. (2004a)'s or the aforementioned have never been used in approximate or convex decompositions of C-free, certainly not in the certified convex decomposition enabled by C-IRIS or its variants. As these techniques have begun to demonstrate applicability in higher DOF, for the first time they have the potential to integrate their benefits into practical manipulation scenarios.

Siméon et al. (2004a)'s contributions remain fundamental, as they were the first to demonstrate continuous grasp-based planning. Our work builds upon this approach by formulating an optimisation-based manipulation planner that generates optimal trajectories within convex decompositions of C-free, integrating modern motion planning techniques with manipulation planning.

Chapter 3

Theoretical Overview

In this chapter, we present the theoretical background behind our proposed manipulation planner. We begin with the formulation of manipulation as constrained motion planning from Siméon et al. (2004a), which we later reinterpret through convex optimisation in Chapter 5. We then introduce methods to decompose the collision-free configuration space (C-free) into convex regions, focusing on the IRIS algorithm and its variants C-IRIS and IRIS-NP, how they provide certified or probabilistic guarantees of non-collision, and how to address their limitations with seeding through recent techniques (Werner et al., 2024a) that automate this step. We follow with an introduction to the optimisation techniques we use to generate feasible trajectories within this convex decomposition, namely Mixed-Integer Programming (MIP) and Graphs of Convex Sets (GCS) to construct optimal transit and transfer motions. We finish by introducing Drake, the robotics toolbox used to implement and test our planner, which integrates the convex decomposition and trajectory generation methods described in this chapter.

3.1 Manipulation as Constrained Motion Planning

Manipulation planning is a fundamental problem in robotics that involves generating sequences of motions for a robot to displace objects in an environment with obstacles. Unlike classical motion planning, where the robot navigates its configuration space (C-space) to reach a goal, manipulation planning introduces additional complexity due to the presence of *movable objects*. These objects can only be moved when grasped by the robot, leading to a constrained version of the motion planning problem.

This section formalises the constrained problem, presenting the mathematical formulation used in Siméon et al. (2004a) and its predecessor works.

3.1.1 Problem Formulation

Consider a robot $\mathcal{R}$ with N degrees of freedom (DOF) operating in a 3-dimensional workspace. The robot has to manipulate a movable object $\mathcal{M}$ with $1 \leq M \leq 6$ DOF (3D ridig body free motion has 6 DOF, or less if constrained) that can only be moved when grasped by the robot. The environment also contains static obstacles that the robot and the movable object must avoid. Let $C_{\text{rob}} \subset \mathbb{R}^N$ and $C_{\text{obj}} \subset \mathbb{R}^M$* be the configuration spaces of the robot and the movable object, respectively. The composite configuration space of the system is given by:

$$CS = C_{\text{rob}} \times C_{\text{obj}} \subset \mathbb{R}^{N+M}$$

The submanifold of CS that corresponds to collision-free configurations is denoted as CS_{free}. A configuration in CS_{free} is one where the robot and the movable object do not intersect with any static obstacles or with each other. The manipulation planning problem is thus defined as finding a collision-free path in CS_{free} that enables the robot to move the object from an initial to a goal configuration.

In this report, we use C-space and C-free to refer to the general concepts of configuration space and its collision-free subset. In the context of manipulation planning, CS and CS_{free} refer to the composite configuration space of the robot and the movable object.

3.1.2 Manipulation Constraints

A solution to a manipulation planning problem is a path in CS_{free} that satisfies specific constraints related to the manipulation task. These constraints ensure that the movable object $\mathcal{M}$ can only move when it is grasped by the robot. The solution path is an alternating sequence of two types of sub-paths, separated by grasp and ungrasp actions:

- **Transit Paths**: Paths where the robot moves alone while the movable object $\mathcal{M}$ remains stationary at a stable position. During a transit path, the configuration parameters of $\mathcal{M}$ remain constant. Transit paths are used to reposition the robot so that it can grasp the object or change its grasp.
- **Transfer Paths**: Paths where the robot moves while holding the movable object $\mathcal{M}$ with a fixed grasp. During a transfer path, the configuration parameters of $\mathcal{M}$ change with the robot. That is, the relative pose of $\mathcal{M}$ with respect to the robot's end-effector remains constant. Transfer paths move the object to a new position.

Because of these constraints, manipulation planning has a state-based aspect, illustrated in Figure 3.1. A valid configuration in CS_{free} can be in one of two states: either the object is grasped by the robot (G), or it is placed at a stable position (P). The transition between these states happens through grasping-ungrasping actions, which represent configurations where the object is both grasped and placed in a stable position. Likewise, the transition between two grasping configurations is given by transfer paths, while the transition between two placement configurations is given by transit paths.

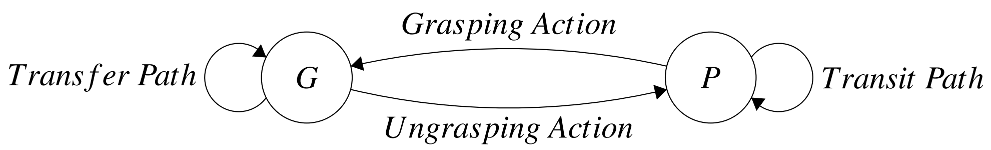

Figure 3.1: Finite State Machines (FSMs) representing the state nature of manipulation planning, showing the transitions between grasping (G) and placing (P) configurations.

*Siméon et al. (2004a) assume the dimensions of C_{rob} and C_{obj} are equal to the DOF of $\mathcal{R}$ and $\mathcal{M}$.

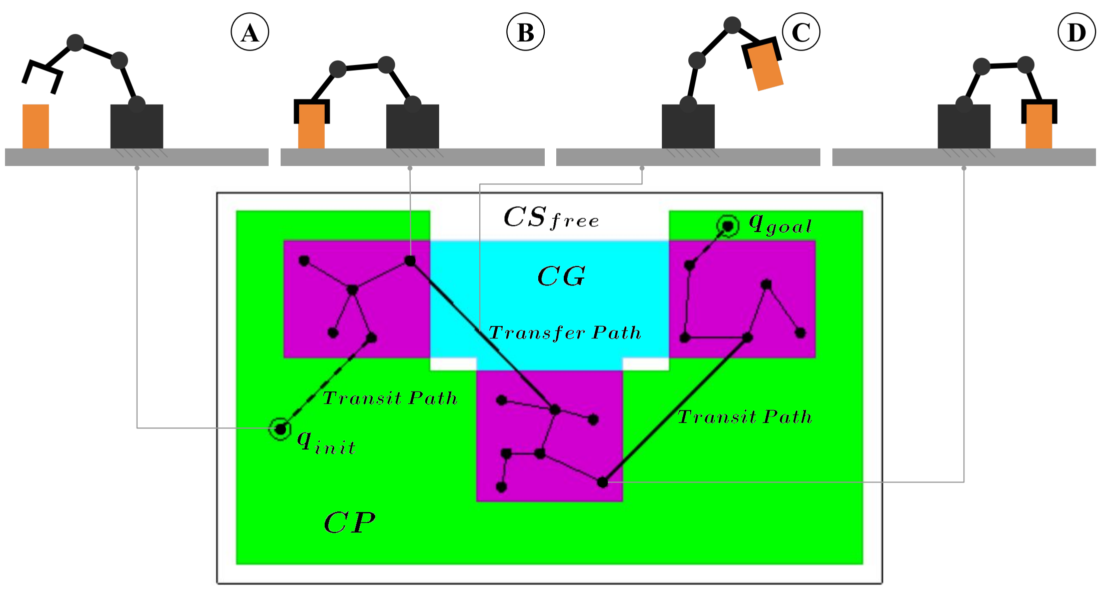

Figure 3.2: Adapted from Siméon et al. (2004a). A manipulation task in CS_{free}. The green region represents CP, the blue region CG, and the purple region is the intersection of the two ($CG \cap CP$). A path is represented from the initial to the final configurations (q_{init} to q_{goal}), decomposed in transfer and transit paths. The overlaid images depict task-space scenes at selected configurations along a manipulation path. Image A shows the object at rest and the robot in a transit path. Images B and D lie in the intersection $CG \cap CP$, representing valid grasping postures. Image C depicts a transfer path, where the robot holds and displaces the object.

3.1.3 Placement and Grasp Spaces

To formalise the manipulation planning problem, we define two subspaces of CS_{free} that are essential for defining the transitions between transit and transfer paths:

- **Placement Space** (CP): The subset of CS_{free} where the object $\mathcal{M}$ is placed at a stable position. A stable position is where the object remains at rest without requiring the robot to hold it. Each **transit path** lies entirely within CP.
- **Grasp Space** (CG): The subset of CS_{free} where the robot grasps $\mathcal{M}$ with a fixed grasp. Each **transfer path** lies entirely within CG.

Because of the manipulation constraints, in a manipulation task the object is either grasped (in CG) or at rest (in CP). Therefore, a solution to the manipulation problem lies entirely within $CG \cup CP$. The intersection of CG and CP ($CG \cap CP$), represents the set of configurations where the object is both grasped by the robot and placed at a stable position. This submanifold contains all possible configurations where the object is to be picked or left at a surface, representing critical transition points between transit and transfer paths: when the object is picked, the robot transitions from a transit path to a transfer path, and when the object is placed, the robot transitions from a transfer path to a transit path. Figure 3.2 illustrates a manipulation task in CS_{free}.

By introducing CP and CG, Siméon et al. (2004a) was able to automate the exploration of grasping and ungrasping actions within $CG \cap CP$, instead of relying on manual specification like previous works.

Remark 1 *While all transit (reciprocally, transfer) paths lie within CP (CG), not all paths in CP (CG) are transit (transfer) paths. This is because the robot can move within CP (CG) without necessarily meeting the manipulation constraints and, therefore, not following a transit (transfer) path.*

3.1.4 The Manipulation Graph

To efficiently search for valid manipulation paths, Siméon et al. (2004a) proposed the **manipulation graph**, which encodes how different transit and transfer paths connect grasp-placement configurations.

- **Nodes**: Represent connected components of $CG \cap CP$.
- **Edges**: Represent valid transit and transfer paths that allow movement between these connected components.

The term *connected component* comes from a mathematical concept in topology. According to (Adams and Franzosa, 2008, p. 186):

Definition 1 (Connectedness) *A topological space is connected if it cannot be broken down into two distinct pieces that are separated from each other.*

In robotics, Siméon et al. (2002) referred to a connected component in $CG \cap CP$ as any set of configurations within $CG \cap CP$ that could be connected without colliding with obstacles, breaking the grasp or lifting the object.

With the manipulation graph, the planning problem is reduced to identifying connected components within $CG \cap CP$ (nodes), representing configurations where the object is to be grasped or placed. Once these connected components are found, the planner can focus on finding transit and transfer paths that connect them (edges).

Our proposed manipulation planner generalises the concept of manipulation graph through convex optimisation.

3.1.5 Manipulation Planning Problem

With the concepts described in this section, now we can formally define the manipulation planning problem as:

Definition 2 (Manipulation Planning Problem) *Given known sets of placements (CP) and grasps (CG), the manipulation planning problem is to find a path in CS_{free} that connects two configurations q_i and q_f in $CG \cup CP$. The solution must alternate between transit and transfer paths, where each transition is at a configuration in $CG \cap CP$.*

3.2 C-free Decomposition

To enable trajectory optimisation in manipulation planning, we need a decomposition of the collision-free configuration space (C-free) into simpler subspaces. Efficient optimisation is possible when these subspaces are convex, as this allows the use of efficient solvers such as `Mosek` or `Gurobi` (MOSEK, 2024; Gurobi Optimization, 2024) to find trajectories both within and between these subspaces.

However, C-free is typically non-convex, high-dimensional, and intractable to decompose analytically. Prior works made strong, limiting assumptions to simplify the problem. The IRIS algorithm (Deits and Tedrake, 2015a) is an example, as it requires the obstacles to be convex in C-space, which is rarely the case in practice. However, this method was capable of scaling to high dimensions and providing rigorous guarantees of non-collision, laying the foundation for future work. Recent advancements have introduced practical methods that decompose C-free into convex regions, while working in high dimensions and not requiring convex C-space obstacles. These methods provide for the first time the ability to generate large regions within C-free in realistic scenarios, achieving either certified or probabilistic guarantees of non-collision. Examples include C-IRIS or IRIS-NP, both variants of IRIS.

This section explores the decomposition of C-free into convex regions, discussing the requirements for collision-free guarantees and how to achieve high coverage of C-free.

3.2.1 The IRIS Algorithm

IRIS (Iterative Regional Inflation by Semidefinite programming) is a decomposition algorithm that generates convex regions –polytopes[†]– within C-free. The algorithm makes the assumption that obstacles in C-space are convex. While not realistic in practice, this assumption allows IRIS to provide rigorous guarantees of non-collision.

IRIS builds on the idea of *separating hyperplanes*. The *Separating Hyperplane Theorem* is a known concept in convex optimisation (Boyd and Vandenberghe, 2004, Section 2.5), and states that two convex bodies $\mathcal{A}$ and $\mathcal{B}$ can be separated by a hyperplane if and only if they do not intersect (Figure 3.3).

Find a, b such that:

$$\begin{cases} a^\top x + b > 0, & \forall x \in \mathcal{A} \\ a^\top y + b < 0, & \forall y \in \mathcal{B} \end{cases} \tag{3.1}$$

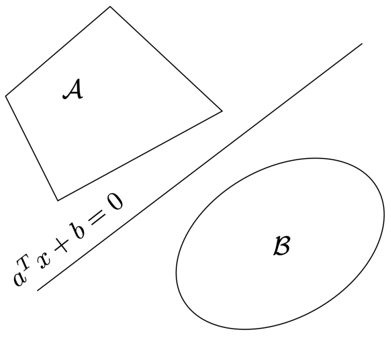

Figure 3.3: The separating hyperplane conditions (left) and illustration of two convex bodies $\mathcal{A}$ and $\mathcal{B}$ separated by a hyperplane (right). Adapted from Dai et al. (2023).

[†]A polytope is a generalisation of polyhedra to higher dimensions.

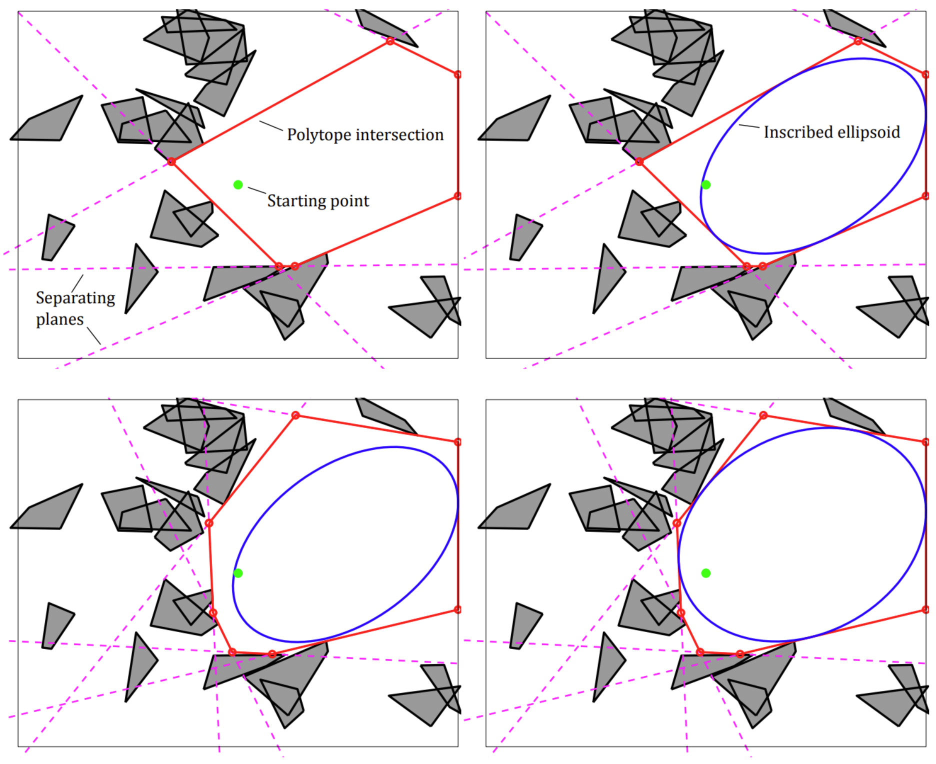

Figure 3.4: Iris pipeline. Each row represents an iteration. The figures in the left column show the certification step, and the figures in the right the inflation step. Adapted from Deits and Tedrake (2015a).

By applying this theorem, Iris can certify that convex regions in C-space are not in collision with obstacles *only if the obstacles are convex in C-space*.

The algorithm starts with an initial seed point q_0 in C-free and iterates two steps:

1. **Certification step:** The algorithm finds a family of separating hyperplanes –a certificate of non-collision $\mathcal{C}_{\mathcal{P}i}$– between the current region $\mathcal{P}_i$ (or q_0) and the obstacles $\mathcal{O}$ in C-space.
2. **Inflation step:** The algorithm expands $\mathcal{P}_i$ in C-space by finding the maximal ellipsoid that does not violate the non-collision certificates $\mathcal{C}_{\mathcal{P}i}$ from the previous step and generating an inflated new polytope $\mathcal{P}_{i+1}$.

The region generated by the inflation step passes to the next iteration, and a new certificate of non-collision is generated. This process continues until the region stops growing. A visualisation of the process is depicted in Figure 3.4.

3.2.2 Certified Decomposition with C-Iris

To overcome the limitations of Iris (convex obstacles in C-space), C-Iris (Dai et al., 2023) introduces certificates of non-collision under the only assumption of convexity of the obstacles in task space. The method is based on the concept of *tangent configuration space* (TC-space). The TC-space is a bijective parametrisation of the classical configuration space (C-space), i.e. every configuration in C-space corresponds to a unique configuration in TC-space. The conversion from C-space to TC-space variables is achieved using the stereographic projection substitution $t = \tan(\theta/2)$[‡], where θ is the angle value of a revolute joint in C-space. If we define the C-space variable as

$$\mathbf{q} = \bigcup_{i=1}^{N} \{\theta_i, z_i\} \in \mathbb{R}^N,$$

where θ_i is the angle of the i-th revolute joint and z_i is the value of the i-th prismatic joint of a system with N joints. The TC-space variable is defined as

$$\mathbf{s} = \bigcup_{i=1}^{N} \{t_i, z_i\} \in \mathbb{R}^N,$$

The interest of TC-space lies in its implications in formulating forward kinematics (f_{kin}). f_{kin} is the process of mapping the joint values of a robot to the position and orientation of its end-effector in task space. Usually, f_{kin} results in a non-linear, trigonometric function of the joint values. However, when expressed in TC-space, the *rational* forward kinematics return a rational polynomial, which can be leveraged in optimisation through *Sums of Squares* (SOS) programming[§]. This property allows C-IRIS to provide guarantees of collision-free regions by mapping TC-space variables to task space and generating the non-collision certificates (like IRIS through separating hyperplanes[¶]) *in task space*. Thus, the requirement of task space convex obstacles.

C-IRIS follows the same iteration steps as IRIS, incorporating the TC-space mapping. It initialises with a seed point s_0 in TC-free and proceeds iteratively with the certification and inflation steps.

This method provides rigorous guarantees that the resulting convex regions lie entirely within C-free. However, C-IRIS still relies on certification steps that are computationally expensive, and makes mild assumptions such as revolute joints bounded within $(-\pi, \pi)$ and convexity of task space obstacles. Moreover, working in TC-space distortions distances compared to C-space, and entails manual conversion between C-space and TC-space variables.

3.2.3 Probabilistically Certified Decomposition with IRIS Variants

Several faster variants have been developed to address the computational inefficiency of C-IRIS. IRIS-NP (Petersen and Tedrake, 2023) generalises the original IRIS to operate in non-convex environments, eliminating the need for convex C-space obstacles, specific assumptions about joint types and the TC-space mapping. It sacrifices the guarantees of non-collision in exchange for generating larger regions faster, making it highly scalable to higher dimensions. In late 2024, further improvements were introduced with IRIS-NP2 and IRIS-ZO (Werner et al., 2024b), achieving even faster results.

[‡]Appendix C provides a visualisation of the stereographic projection.

[§]For the purpose of this report, we will not delve into the specifics of SOS programming. Refer to Dai et al. (2023) for details.

[¶]For C-IRIS, Dai et al. (2023) also proposes an alternative certification program, consisting of proving the infeasibility of finding a point in the intersection of two convex sets.

IRIS-NP, IRIS-NP2, and IRIS-ZO all offer probabilistic guarantees of non-collision, in contrast to the rigorous guarantees provided by C-IRIS. These algorithms can be used directly to decompose C-free or, alternatively, they can be used so seed C-IRIS, which can then provide certificates of non-collision. In addition, C-IRIS's certificates can also be used to find the smallest shrinking of a region grown with IRIS-NP, IRIS-NP2, or IRIS-ZO that is guaranteed to be collision-free.

3.2.4 Seeding Challenges and Clique Cover Initialisation

All IRIS, variants require an initial, collision-free region $\mathcal{P}_0$ containing a seed point $q_0, s_0 \in \mathcal{P}_0$ to start the growth process. These seeds require manual specification, which is tedious and task-specific, especially in complex, high-dimensional scenes.

To automate this process, Werner et al. (2024a) introduced a method that approximately covers C-free with a small number of polytopes, using *clique covers* of visibility graphs. Appendix D includes an overview of the method.

The method has been implemented within the Drake Robotics Toolbox (Tedrake and the Drake Development Team, 2019), integrated with IRIS-NP. This allows for automatic seeding, providing a scalable, efficient method to decompose C-free into convex regions, that previously required tedious manual specification.

3.3 Trajectory Generation

To formulate an optimisation-based manipulation planner, we need to generate feasible trajectories that satisfy the constraints introduced in Section 3.1, while remaining in the collision-free space C-free approximated through the methods in Section 3.2.

In this section, we introduce a background on the mathematical programs we have utilised to compute manipulation trajectories.

3.3.1 Mixed-Integer Programming

Mixed-Integer Programming (MIP) is an optimisation technique that combines discrete and continuous decision variables to solve complex problems (mix).

Definition 3 (Mixed-Integer Program) *A Mixed-Integer Program (MIP) is an optimisation problem of the form:*

$$\begin{aligned}
\min_{x,z} \quad & f(x,z) && \textit{(objective function)} \\
\textit{s.t.} \quad & g_i(x,z) \leq 0 \quad \forall i \in \{1,\dots,m\} && \textit{(inequality and equality constraints)} \\
& x \in \mathbb{R}^n && \textit{(continuous variables)} \\
& z \in \mathbb{Z}^p && \textit{(integer variables)}
\end{aligned}$$

where f is typically a linear or quadratic objective, and g_i represents affine or convex constraints.

In our context, the continuous variables x represent the robot's configuration or trajectory parameters, while the integer variables z encode decisions such as which convex region of C-free a configuration should belong to. This allows us to express constraints like:

"The robot's configuration must lie within at least one collision-free region."

Thanks to the convexity of each region, the optimisation over x remains efficient, while z allows us to handle decisions on different regions.

3.3.2 Graphs of Convex Sets (GCS)

To implement trajectory optimisation over convex decompositions, we use the *Graphs of Convex Sets* (GCS) framework introduced by Marcucci et al. (2022) and implemented in Drake (Tedrake and the Drake Development Team, 2019).

GCS encodes the planning problem as a directed graph where:

- **Nodes** represent convex regions (e.g., polytopes) in C-free.
- **Edges** represent feasible trajectory segments between regions.

GCS works directly over convex decompositions of C-free, integrating perfectly with the outputs of IRIS-NP and C-IRIS algorithms. This method provides a straightforward framework to generate trajectories within the provided convex regions. This makes GCS well-suited for computing *optimal* transit and transfer paths.

3.4 Drake

All experiments and implementations in this dissertation are carried out using the open-source robotics toolbox Drake (Tedrake and the Drake Development Team, 2019). Drake is a C++ library with Python bindings (pydrake) designed to simulate and plan complex manipulation tasks that generalise to real-world dynamics (Sherman, 2022). In this work, we use the Python API via the stable release `1.35.0`, running in a Docker container built from the official image[‖].

Drake is actively developed and regularly updated with state-of-the-art features, which can introduce challenges such as version changes and limited online support. Despite this, it remains one of the most comprehensive frameworks for manipulation and robotics, offering native support for convex decomposition methods (IRIS, C-IRIS, and their variants) and trajectory optimisation tools. This makes it uniquely suited for implementing the planner proposed in this work.

[‖]`https://hub.docker.com/layers/robotlocomotion/drake/1.35.0/images/sha256-20d3de2bee61ce41a850824e4af1678662682869b2ae3d74bde4c639e3bec106`

Chapter 4

Assumptions and Problem Statement

This chapter introduces the manipulation planning problem we address in this work. Our goal is to compute an optimal sequence of robot motions that move a grasped object from an initial to a goal configuration, while avoiding collisions with the environment, respecting manipulation constraints. The inputs to our problem are:

- A robot with a known structure and joint limits.
- A movable object the robot needs to manipulate.
- A known static environment with obstacles.
- A set of possible grasps and placements for the object.
- An initial and a goal configuration for the robot and the object.

The desired output is an optimal, collision-free trajectory that the robot can follow to complete the task.

We discuss the mathematical formulation of the problem, the assumptions made in our approach, and the requirements for collision-free guarantees.

A note on Monogram Notation

To describe spatial relations between the robot, object, and the environment, we use the Monogram Notation (Tedrake, 2023a, Chapter 3.1), which allows us to describe the relative position, orientation, and pose of bodies using the frames attached to them:

- ${}^{B}p^{A}_{C}$ denotes the position of frame A relative to frame B expressed in frame C.
- ${}^{B}R^{A}$ denotes the orientation of frame A measured from frame B.
- ${}^{B}X^{A}$ denotes the pose of frame A measured frame B.

The pose –or transform– represents both the position and orientation of a frame.

A more thorough summary of the notation and its integration with arithmetic operations is included in Appendix E. This report does not require a deep understanding of the notation to follow the problem statement.

4.1 Problem Statement

We address the problem of generating optimal, collision-free manipulation trajectories for a robot operating in a known task-space environment. The task involves planning a sequence of motions for a robotic manipulator $\mathcal{R}$ to move an object $\mathcal{M}$ from an initial to a goal configuration through a series of grasping and ungrasping actions, subject to constraints imposed by the environment (e.g. obstacles and joint limits) or the manipulation task (e.g. grasping and placement constraints).

To represent the state of the full system, we define a composite configuration space CS that includes both the configuration variables of $\mathcal{R}$ and $\mathcal{M}$:

$$CS = C_{\text{rob}} \times C_{\text{obj}} \subset \mathbb{R}^{N+M},$$

where C_{rob} and C_{obj} are the configuration spaces of $\mathcal{R}$ and $\mathcal{M}$, respectively, and N and M their dimensions. The collision-free subspace of CS is denoted as $CS_{\text{free}} \subset CS$.

The inputs to our problem are:

- A **robotic manipulator** $\mathcal{R}$ with a known kinematic structure and joint limits, with configurations $\mathbf{q} \in C_{\text{rob}} \subset \mathbb{R}^N$.
- A **movable object** $\mathcal{M}$ (a rigid body) with configurations $\mathbf{q} \in C_{\text{obj}} \subseteq SE(3) \subset \mathbb{R}^M$*.
- A set of **static obstacles** O in the task space.
- A continuous specification of **valid grasps** CG **and placements** CP for $\mathcal{M}$ such that $CG, CP \subseteq CS \subset \mathbb{R}^{N+M}$, where CG and CP are convex sets.
- A valid **initial and goal configuration** $\mathbf{q}_{\text{init}}, \mathbf{q}_{\text{goal}} \in CG \cup CP \subseteq CS \subset \mathbb{R}^{N+M}$.
- A set of **manipulation constraints** $\mathcal{C}$ that fix the posture of $\mathcal{M}$ when $\mathcal{M}$ is at rest, and the relative posture of $\mathcal{R}$'s end-effector and $\mathcal{M}$ when $\mathcal{M}$ is grasped by $\mathcal{R}$.

The desired output is an optimal, collision-free trajectory $\tau \in CS_{\text{free}}$ that transports $\mathcal{R}$ and $\mathcal{M}$ from $\mathbf{q}_{\text{init}}$ to $\mathbf{q}_{\text{goal}}$ with the minimum number of grasps, satisfying that $\mathcal{M}$ can only be displaced when it is held by $\mathcal{R}$ (at a valid grasp in CG), and that if $\mathcal{M}$ is not being grasped, it must remain at rest (at a valid placement in CP), while satisfying the manipulation constraints $\mathcal{C}$. That is, τ must consist of an alternating sequence of:

- **Transit Paths:** $\mathcal{R}$ moves alone while $\mathcal{M}$ is at rest.
- **Transfer Paths:** $\mathcal{R}$ moves while holding $\mathcal{M}$ at a fixed grasp.

Remark 2 *By requiring grasps (CG) and placements (CP) to be defined in CS, and leveraging C-free decomposition algorithms like* IRIS-NP *and* C-IRIS*, we make use of convex optimisation techniques to achieve a globally optimal manipulation trajectory in CS_{free}. To the best of our knowledge, this is the first work to propose a manipulation planner that can handle optimisation-based methods to compute trajectories, and therefore the first to provide globally optimal solutions (within the convex decomposition) to the manipulation problem.*

*$SE(3)$ denotes the Special Euclidean group of rigid body transformations, that consists in 3D rotations, translations, or arbitrary combinations of them.

Joint requirements

$\mathcal{R}$ and $\mathcal{M}$ can be described by any combination of revolute, prismatic, planar, cylindrical, spherical, and universal joints. In this work, we limit ourselves to revolute and prismatic joints, as these can represent the other joint types (Wampler and Sommese, 2011) (See proof in Appendix F).

We associate the revolute joint with the rotation angle θ and the prismatic joint with the displacement z. We then define our configuration space variable as the $\mathbb{R}^{N+M}$ vector

$$\mathbf{q} = \bigcup_i \{\theta_i, z_i\} \quad \text{for } i \in \{1, ..., N+M\}. \tag{4.1}$$

Requirements for Collision-Free Guarantees

Our method supports the use of any of the IRIS variants to decompose CS_{free} into convex regions, inherently providing probabilistic guarantees of non-collision. When rigorous certification is needed, the regions can be verified using C-IRIS. However, this requires additional (still attainable) assumptions.

C-IRIS relies on the use of the tangent configuration space (TC-space) to provide certified collision-free guarantees. Under the need for collision-free certification, we define the TC-space variable as the $\mathbb{R}^{N+M}$ vector

$$\mathbf{s} = \bigcup_i \{t_i, z_i\} \quad \text{for } i \in \{1, ..., N+M\} \tag{4.2}$$

where t_i is the stereographic projection of θ_i, i.e., $t_i = \tan(\theta_i/2)$. In addition, we assume that revolute joints θ_i are constrained to a complete rotation

$$-\pi < \theta_{li} \leq \theta_i \leq \theta_{ri} < \pi \quad \forall i, \tag{4.3}$$

and z is such that the displacement is finite:

$$z_{li} \leq z_i \leq z_{ri} \quad \forall i, \tag{4.4}$$

where $\theta_{li}, \theta_{ri}, z_{li}, z_{ri} \in \mathbb{R}$ are fixed bounds for each individual joint.

Lastly, to guarantee non-collision, we assume that the robot $\mathcal{R}$, obstacles O, and movable object $\mathcal{M}$ have been decomposed as unions of convex, rigid bodies in task space.

Chapter 5

Task-Specific Pipeline - Bottle Cap Opening

This chapter presents the first of our two principal contributions: demonstrating the feasibility of our proposed planner through a practical example. Before formulating our manipulation planning algorithm in Chapter 6, we describe in this section the full pipeline that our method follows to generate optimal, collision-free manipulation plans by solving the (simplified) task of opening a bottle by rotating its cap.

We first define the task and environment setup, introducing the system's objectives, constraints, and configuration variables. We then provide a high-level overview of the pipeline, which is composed of two main stages: (1) planning a sequence of grasp and release configurations that solve the task, and (2) generating optimal, collision-free trajectories that connect them. These stages are preceded by a preprocessing step that decomposes the collision-free configuration space (CS_{free}) into convex regions and integrates them with the grasp and placement spaces (CG and CP), enabling the use of convex optimisation tools throughout the pipeline.

The remainder of this chapter walks through each stage, illustrating how we determine where to grasp and release the object, how we ensure these configurations are reachable via feasible motions, and how we compute the corresponding optimal trajectories to complete the task.

5.1 Task Definition

We demonstrate the feasibility of our proposed manipulation planner by applying it to a simplified task: opening the cap of a bottle by rotating it. This task captures the key structure of a manipulation problem: interacting with a movable object under constraints and requiring a combination of transit and transfer motions to achieve the goal.

Our objective is not to solve the full real-world bottle opening task, but rather to model a minimal example with both prismatic and revolute joints where the full pipeline can be executed and visualised. This example highlights the core challenges of manipulation planning and allows us to validate each component of our method in a controlled setting.

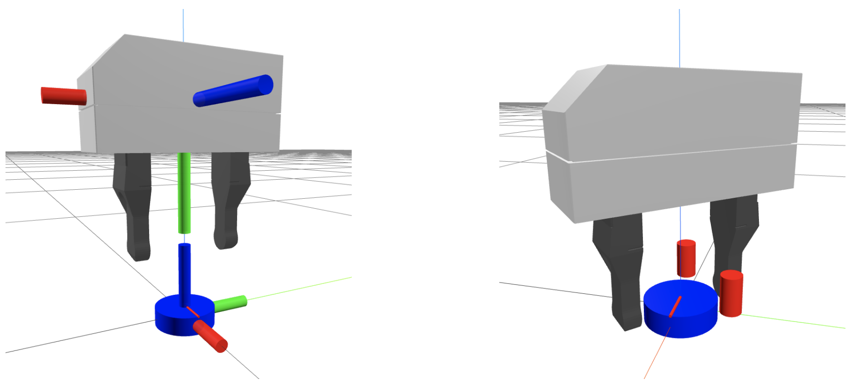

Figure 5.1: Example task scene with a gripper (robot $\mathcal{R}$) and bottle cap (movable object $\mathcal{M}$). 5.1a shows the frames of the gripper and bottle cap (x-axis in red, y-axis in green, z-axis in blue). 5.1b shows the scene with two obstacles in red.

We define the task environment as follows (Figure 7.1):

- **Bottle cap $\mathcal{M}$ (one revolute joint):** Represented as a rigid cylinder with one revolute joint around its z-axis. The base of the cap is fixed at the origin of the world frame, meaning it cannot translate, nor rotate around the x and y axes.
- **Gripper $\mathcal{R}$ (one prismatic and one revolute joint):** A simplified model of the Schunk WSG gripper (Schunk, n.d.) with two joints:
  - A revolute joint that rotates the gripper around its y-axis, aligned with the bottle cap's rotation axis.
  - A prismatic joint that actuates one finger along the gripper's x-axis, allowing it to slide towards the bottle cap.

  The gripper is welded at a fixed distance from the bottle cap, meaning that it can only rotate around the cap and slide its finger towards it, but the gripper's base cannot translate or rotate in other directions.
- **Obstacles O:** Two fixed obstacles are placed in the scene, represented as rigid cylinders. We must avoid collisions with them.

This setup defines a 3-DOF system with the following configuration space variable:

$$\mathbf{q} = \{\theta^{\text{cap}}, \theta^{\text{gripper}}, z^{\text{finger}}\} \in CS \subset \mathbb{R}^3, \tag{5.1}$$

where $\mathbf{q} = \{\theta^{\text{cap}}\} \in C_{\text{obj}} \subset \mathbb{R}$ and $\mathbf{q} = \{\theta^{\text{gripper}}, z^{\text{finger}}\} \in C_{\text{rob}} \subset \mathbb{R}^2$. θ^{cap} represents the angle of rotation of the bottle cap, θ^{gripper} the angle of the gripper, and z^{finger} the displacement of the gripper finger along its x-axis.

Remark 3 *The axes of rotation of θ^{cap} and $\theta^{gripper}$ are inverted, meaning that the gripper rotates in the opposite direction to the bottle cap. There is a transformation that aligns them; therefore, for simplicity and without loss of generality, during the following sections we assume they are aligned.*

Joint limits To satisfy the collision-free requirements defined in Chapter 4, bottle cap rotation has been constrained to the range $-\pi < \theta^{\text{cap}} < \pi$, gripper rotation to $-1 \leq \theta^{\text{gripper}} \leq 1$ and finger displacement to the range $z^{\text{finger}} \in [-0.055, 0]$. Rotation limits are defined in radians, and the finger displacement is defined in meters.

Manipulation Constraints For this task, we consider a valid grasp configuration as one where the gripper finger is in contact with the cap. In our scene, this is when $z^{\text{finger}} = -0.025$.

The cap is fixed in the world and considered placed at all times. Thus, every configuration is a valid placement and belongs to *CP*.

Initialisation and Objective We determine the cap to be fully closed when $\theta^{\text{cap}} = -3.14$ and fully open when $\theta^{\text{cap}} = 3.14$. We define the general objective of the task as opening the cap by achieving a 360° rotation, i.e., from $\theta^{\text{cap}} = -3.14$ to $\theta^{\text{cap}} = 3.14$, with the minimum number of grasps and releases.

Justification for the Task While low-dimensional, this setup is non-trivial: the limited rotation range of the gripper means that the cap cannot be opened in a single grasp. Regrasping is necessary to complete the task, making it a minimal yet representative example for evaluating our planner. Moreover, the 3D dimension of *CS* in this task allows us to visualise *CG*, *CP*, CS_{free}, and planned trajectories, a unique benefit when explaining and validating the method.

5.2 Pipeline Overview

Our approach builds on the classical notion of a manipulation graph that we introduced in Section 3.1.4, where nodes correspond to connected components of grasping and placement configurations (i.e., connected regions of $CG \cap CP$), and edges represent feasible transfer and transit motions connecting them. We reinterpret this structure to incorporate optimal trajectory generation, but the core idea remains the same: to solve a manipulation task, we must determine *where* the robot should grasp and release the object (nodes) and *how* to move between these configurations safely and efficiently (edges).

We address these questions through a pipeline that consists of two main stages:

1. **Grasp and Placement Planning:** We compute a sequence of required grasp and release configurations that solve the task. This stage involves enforcing the manipulation constraints, and ensuring that the planned configurations are collision-free and can be connected by feasible transfer and transit paths.
2. **Trajectory Generation:** We generate optimal collision-free trajectories that connect the planned grasp and release configurations through transfer and transfer paths.

A prior preprocessing step is necessary to define the grasp and placement spaces (*CG* and *CP*) for the task, decompose CS_{free} into convex regions, and integrate these spaces to compute connected components and the collision-free submanifolds of CG and CP, CG_{free} and CP_{free}.

The next sections describe each stage of the pipeline in detail, starting with the preprocessing step (Section 5.3), followed by grasp and placement planning (Section 5.4), and concluding with trajectory generation (Section 5.5).

5.3 Preprocessing Stage: CG, CP, CS_{free}, and Connected Components

Before solving the planning problem, we perform a preprocessing step to describe the structure of the configuration space and prepare the necessary inputs for our optimisation-based planner. This stage defines the geometric subspaces associated with the task and identifies the collision-free regions where planning can take place. Specifically, we describe how to formulate grasp and placement spaces (CG and CP), decompose CS_{free} into convex regions using clique covers and IRIS variants, compute the CS_{free} regions that intersect with CG and CP, and identify the connected components of these intersections.

5.3.1 Grasp and Placement Polytopes: CG and CP

The first step in our preprocessing pipeline is to define the subspaces of the configuration space where the object is either grasped by the robot (CG) or placed stably (CP).

In our example, we define these subspaces directly in terms of bounds on the configuration space variables:

- **Placement space (CP):** Since every configuration of the bottle cap is a valid placement, CP covers the full CS, bounded by the joint limits $\theta^{\text{cap}} \in [-3.14, 3.14]$, $\theta^{\text{gripper}} \in [-1, 1]$, and $z^{\text{finger}} \in [-0.055, 0]$.
- **Grasp space (CG):** A configuration is considered a valid grasp when the gripper finger is in contact with the bottle cap. In our scene, this is when $z^{\text{finger}} = -0.025$. To give the resulting polytope a non-zero volume, we define a small band of width 1 mm: $z^{\text{finger}} \in [-0.025, -0.024]$, while keeping the other joint variables within their standard limits.

The convex hull of all possible corner points defined by these bounds is then computed. Since there are two bounds for each of the three configuration variables, this results in $2^3 = 8$ corner points, which form the vertices of the polytope. CG and CP are illustrated in Figure 5.2a.

Alternatively, grasp constraints can be defined using relative transforms between the gripper and object frames (e.g., fixing ${}^{G}X^{\mathcal{M}}$ to a desired range of grasp poses). While this provides greater flexibility and generality, it typically involves more complex constraint formulations, which we do not delve into for the scope of this dissertation.

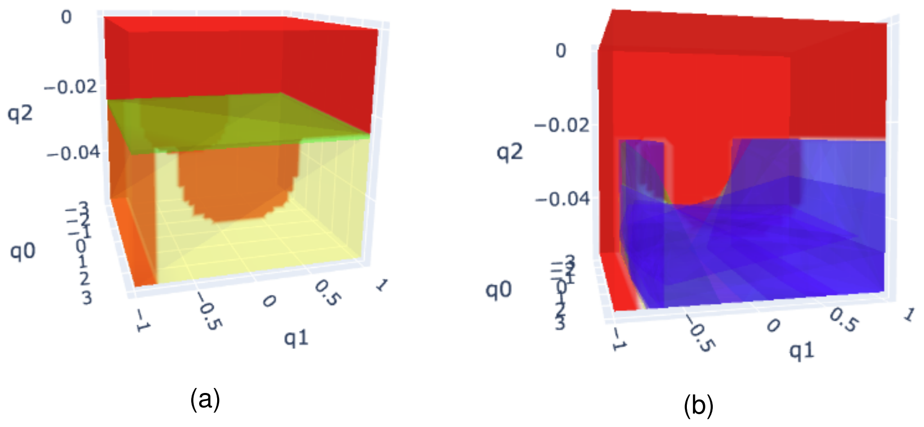

Figure 5.2: CS 3D visualisation for the bottle cap task with two obstacles. In red, colliding configurations. q0 represents θ^{cap}, q1 represents θ^{gripper}, and q2 represents z^{finger}. 5.2a illustrates CG and CP, in green and yellow respectively; we can observe how CG covers the plane where q2 $= z^{\text{finger}} = -0.025$. 5.2b shows the generated collision-free space CS_{free} as a union of blue regions.

5.3.2 Convex Decomposition of CS_{free}

To enable optimisation-based planning, we approximate the collision-free configuration space CS_{free} with a union of convex polytopes. This is done using a two-step process: initial growth using clique covers, and subsequent certification using C-IRIS.

We first generate an approximate convex decomposition of CS_{free} using the clique cover method (Werner et al., 2024a), that we presented in Section 3.2.4. This algorithm samples collision-free configurations and builds a visibility graph, which is then decomposed into cliques. Each clique is inflated into a convex polytope using IRIS-NP (in the current version of Drake, which we use). Alternative growing algorithms like C-IRIS, IRIS-NP2 or IRIS-ZO could be substituted in manual implementations. Using this method, we achieved over 99% coverage of CS_{free}.

Collision-free Certification Since IRIS-NP operates in C-space, while C-IRIS requires polytopes in the tangent configuration space (TC-space), we convert each polytope by mapping its vertices into TC-space using the stereographic projection $t = \tan(\theta/2)$. We then compute the convex hull in TC-space and extract its centre via its maximum volume ellipsoid to define a seed s_0 for C-IRIS.

C-IRIS shrinks each region by iteratively adjusting its faces to find the largest inner polytope that can be rigorously certified as collision-free. The resulting certified polytope is guaranteed to lie within CS_{free}. Finally, we map these certified polytopes back to C-space for use in planning.

This yields a decomposition of CS_{free} into convex regions, each with rigorous collision-free guarantees (Figure 5.2b).

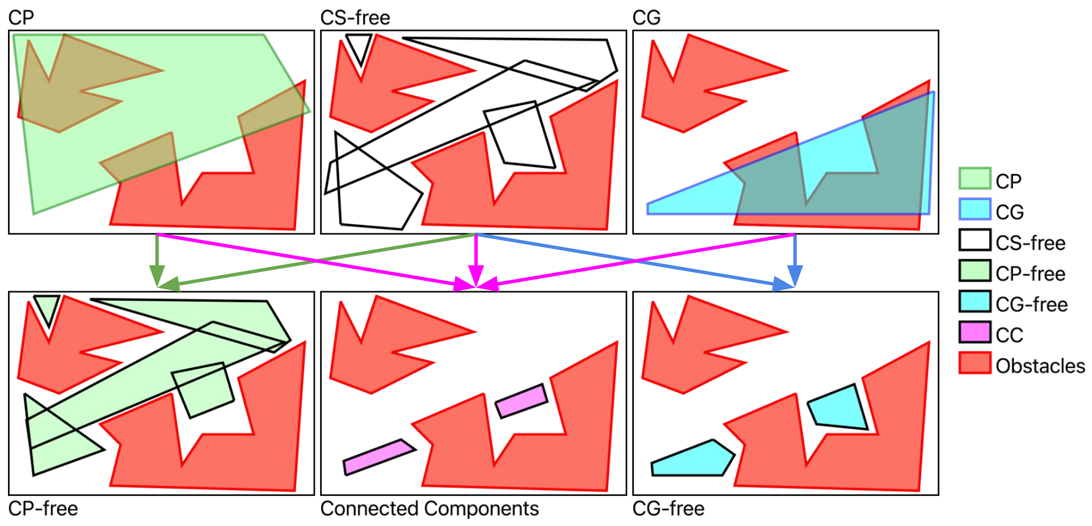

Figure 5.3: Construction of CG_{free}, CP_{free}, and connected components of $CG \cap CP \cap CS_{\text{free}}$ in an abstract 2D representation of CS. Top row: original sets CP (placement), CS_{free}, and CG (grasp); bottom row: resulting CP_{free} (green), CG_{free} (blue), and connected components –CC– (pink) obtained by intersecting CP, CG, and CS_{free} in the relations established by the arrows. Red denotes obstacles.

5.3.3 CG_{free}, CP_{free} and Connected Components

Once CG and CP are defined and CS_{free} has been decomposed into convex regions, we compute their intersection to determine the regions of the grasp and placement spaces that are also collision-free, namely CG_{free} and CP_{free}.

$$CG_{\text{free}} = \bigcup_i CG \cap \mathcal{P}_i, \text{ and } CP_{\text{free}} = \bigcup_i CP \cap \mathcal{P}_i, \quad \forall \mathcal{P}_i \in CS_{\text{free}}. \tag{5.2}$$

CG and CP result in a (possibly disconnected) collection of convex polytopes.

Moreover, we identify the connected components (CC) by intersecting CG, CP and CS_{free}:

$$CC = \bigcup_i CG \cap CP \cap \mathcal{P}_i = CG_{\text{free}} \cap CP_{\text{free}}, \quad \forall \mathcal{P}_i \in CS_{\text{free}}. \tag{5.3}$$

An intuitive illustration of computations 5.2 and 5.3 is shown in Figure 5.3.

Recall from Section 3.1.4 that connected components represent regions of $CG \cap CP$, i.e. grasping and releasing key configurations, that can be connected without lifting the object or changing grasp. They represent the *nodes* of our manipulation graph. Therefore, by formulating these components as unions of convex sets, we can integrate them into our optimisation program to ensure that key grasping and releasing configurations we want to find lie within $CG \cap CP$. Figure 5.4 shows an example of the connected components of our bottle cap task.

Likewise, by having formulated CG_{free} and CP_{free} as unions of convex polytopes, we can use them in our optimisation program to ensure that configurations in $CG \cap CP$ that connect transfer paths lie within the same connected regions of CG_{free}, and

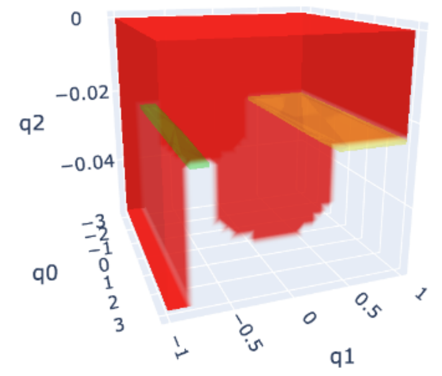

Figure 5.4: Visualisation of the two connected components of $CG \cap CP \cap CS_{\text{free}}$ in our Drake implementation of the bottle cap task. There are two components, depicted in green and yellow.

configurations in $CG \cap CP$ that connect transit paths lie within the same connected regions of CP_{free}. We thoroughly describe this process in the next section.

5.4 Planning Grasp-Release Configurations

The goal of this stage is to compute a discrete sequence of grasp and release configurations that take the system from the initial configuration $\mathbf{q}_{\text{init}}$ to the goal configuration $\mathbf{q}_{\text{goal}}$, while satisfying all manipulation constraints. To do this, we formulate the problem as a *Mixed-Integer Quadratic Program (MIQP)*, where the integer variables enforce logical constraints (such as if a configuration belongs to a free region), and the continuous variables define the actual configurations in the robot's configuration space.

5.4.1 Mixed-Integer QP Formulation

We formulate the problem of planning a grasp-release sequence as a Mixed-Integer Quadratic Program (MIQP). The objective is to compute a discrete sequence of configurations $\{\mathbf{x}_i\}_{i=0}^{2\ell+1}$ that connects an initial configuration $\mathbf{x}_0 = \mathbf{q}_{\text{init}}$ to a goal configuration $\mathbf{x}_{2\ell+1} = \mathbf{q}_{\text{goal}}$, while satisfying manipulation constraints, collision-avoidance, and connectedness conditions. Each $\mathbf{x}_i \in \mathbb{R}^d$ is a configuration of the robot and object in the configuration space, with $d = 3$ in our case.

The planner assumes a fixed number of grasps ℓ, and alternates between transfer and transit modes. The sequence takes the form:

$$\mathbf{x}_0 = \mathbf{q}_{\text{init}},\ \mathbf{x}_1 = g_1,\ \mathbf{x}_2 = r_1,\ \mathbf{x}_3 = g_2,\ \ldots,\ \mathbf{x}_{2\ell} = r_\ell,\ \mathbf{x}_{2\ell+1} = \mathbf{q}_{\text{goal}},$$

where grasp configurations g_i and release configurations r_i alternate. This ensures that the final resulting path conforms an alternate sequence of transit and transfer paths, such that:

$$\begin{tikzcd} {} \arrow[r] & \mathbf{q}_{\text{init}} \arrow[r, "Transit\ Path"'] & g \arrow[r, bend right, "Transfer\ Path"'] & r \arrow[l, bend right, "Transit\ Path"'] \arrow[r, "Transit\ Path"'] & \mathbf{q}_{\text{goal}} \end{tikzcd}$$

To find the minimum number of grasps required to solve the task, we solve this MIQP iteratively for increasing values of ℓ, starting from 1. The first value of ℓ for which a feasible solution exists gives us a solution with the minimal number of grasping actions.

Decision Variables The optimisation program includes:

- **Continuous variables** $\mathbf{x} \in \mathbb{R}^{(2\ell+2)d}$, representing the sequence of configurations.
- **Binary variables**:
  - $z_{i,j} \in \{0,1\}$: indicates if $\mathbf{x}_i$ lies in convex polytope $j \in \{1,\dots,J\}$ of CS_{free}
  - $y_{i,k} \in \{0,1\}$: indicates if $\mathbf{x}_i$ lies in connected region $k \in \{1,\dots,K\}$ of CP_{free}
  - $w_{i,r} \in \{0,1\}$: indicates if $\mathbf{x}_i$ lies in connected region $r \in \{1,\dots,R\}$ of CG_{free}

where J is the number of convex polytopes in CS_{free}, K is the number of connected regions in CP_{free}, and R is the number of connected regions in CG_{free}. Recall from Section 5.3.3 that CG_{free} and CP_{free} are unions of convex polytopes that might not be fully connected, forming *connected regions of* CG_{free} *and* CP_{free}, each containing one or more polytopes. The binary variables $z_{i,j}$, $y_{i,k}$, and $w_{i,r}$ are used to enforce logical constraints on the configurations, ensuring that they lie within the defined regions of CS_{free}, CP_{free}, and CG_{free} respectively. We thoroughly describe the constraints enforced by these variables in Section 5.4.2.

5.4.2 Constraints

To ensure the feasibility and correctness of the planned grasp-release sequence, our MIQP formulation includes three types of constraints: (1) joint limits, (2) manipulation constraints to encode the physical meaning of transfer and transit paths, and (3) connectivity constraints to ensure that consecutive configurations can be joined by a feasible path within CG_{free} or CP_{free}.

1. Joint Limit Constraints Each configuration $\mathbf{x}_i$ must lie within the joint limits of the robot and object, forming inequality constraints:

$$\mathbf{q}_{\min} \leq \mathbf{x}_i \leq \mathbf{q}_{\max} \quad \forall i \in \{0,\dots,2\ell+1\}, \tag{5.4}$$

where $\mathbf{q}_{\min}, \mathbf{q}_{\max} \in \mathbb{R}^d$ are the lower and upper bounds of the configuration space, defined by the joint limits of the system. In our task, $\mathbf{q}_{\min} = [-3.14, -1, -0.055]$ and $\mathbf{q}_{\max} = [3.14, 1, 0]$.

2. Manipulation Constraints To ensure that grasp and release configurations correspond to valid transfer and transit motions, we define the following equality constraints:

- **Transfer constraints:** For each pair $(\mathbf{x}_{2i+1}, \mathbf{x}_{2i+2})$ corresponding to a grasp and its subsequent release, the relative transformation between gripper and object must be preserved. In our task, this translates into:

$$(\theta_{2i+1}^{\text{gripper}} - \theta_{2i+1}^{\text{cap}}) = (\theta_{2i+2}^{\text{gripper}} - \theta_{2i+2}^{\text{cap}}), \tag{5.5a}$$

$$z_{2i+1}^{\text{finger}} = z_{2i+2}^{\text{finger}}, \tag{5.5b}$$

for all $i \in \{0, \dots, \ell - 1\}$, enforcing that the grasp is maintained throughout the transfer motion.

- **Transit constraints:** For each pair $(\mathbf{x}_{2i}, \mathbf{x}_{2i+1})$ corresponding to transitions between release and the next grasp, $\mathbf{q}_{\text{init}}$ and the first grasp, or the last release and $\mathbf{q}_{\text{goal}}$, we require the object configuration to remain fixed:

$$\theta_{2i}^{\text{cap}} = \theta_{2i+1}^{\text{cap}}, \tag{5.5c}$$

for all $i \in \{0, \dots, \ell\}$.

These constraints guarantee that the sequence of configurations conforms a feasible manipulation strategy composed of alternating transit and transfer paths.

3. Connectivity Constraints Finally, we must ensure that each pair of consecutive configurations can be connected through a feasible, collision-free motion within the appropriate subspace:

- **Transfer motions:** For all $i \in \{0, \dots, \ell - 1\}$, the grasp and release pair $(\mathbf{x}_{2i+1}, \mathbf{x}_{2i+2})$ must both lie within the same connected region of CG_{free}. This ensures that a continuous, collision-free transfer motion exists between them.
- **Transit motions:** For all $i \in \{0, \dots, \ell\}$, the release and subsequent grasp (or goal) pair $(\mathbf{x}_{2i}, \mathbf{x}_{2i+1})$ must lie within the same connected component of CP_{free}, ensuring the feasibility of the corresponding transit motion.

We describe here the formulation of these constraints for CP_{free}, which applies to all transit segments in the plan. An analogous formulation is applied to enforce transfer motions within CG_{free}.

Recall from Section 5.3.3 that CP_{free} is a union of disjoint connected regions, each of which consists of one or more convex polytopes. Let $\{\mathcal{R}_k\}_{k=1}^{K}$ denote the connected regions of CP_{free}, where each region $\mathcal{R}_k$ is a union of convex polytopes. Likewise, let $\{\mathcal{P}_j\}_{j=1}^{J}$ denote each of the convex polytopes of CS_{free}.

We introduce the following binary variables:

- $z_{i,j} \in \{0, 1\}$: configuration $\mathbf{x}_i$ lies in polytope $\mathcal{P}_j$.
- $y_{i,k} \in \{0, 1\}$: configuration $\mathbf{x}_i$ lies in region $\mathcal{R}_k$.

The constraints are imposed as follows:

1. **Polytope containment:** For each configuration $\mathbf{x}_i$, we enforce that it must lie in at least one collision-free polytope:

$$\sum_{j=1}^{J} z_{i,j} \geq 1, \quad \forall i. \tag{5.6a}$$

For each polytope $\mathcal{P}_j$ defined as $A_j\mathbf{x}_i \leq b_j$, we enforce:

$$A_j\mathbf{x}_i \leq b_j + M(1 - z_{i,j}), \tag{5.6b}$$

where M is a sufficiently large constant. This activates the constraint (i.e. enforces polytope membership $A_j\mathbf{x}_i \leq b_j$) only if $z_{i,j} = 1$.

2. **Region membership:** Each configuration $\mathbf{x}_i$ must belong to exactly one connected region $\mathcal{R}_k$:

$$\sum_{k=1}^{K} y_{i,k} = 1, \quad \forall i. \tag{5.6c}$$

3. **Region-to-polytope consistency:** If $\mathbf{x}_i$ is in polytope $\mathcal{P}_j$, and $\mathcal{P}_j$ is part of connected region $\mathcal{R}_k$, then $\mathbf{x}_i$ must also be in that region:

$$z_{i,j} \leq y_{i,k} \quad \text{for all } j \text{ such that } \mathcal{P}_j \in \mathcal{R}_k. \tag{5.6d}$$

4. **Transit pair region constraint:** For each pair of consecutive configurations $(\mathbf{x}_{2i}, \mathbf{x}_{2i+1})$, we require them to lie in the same region:

$$y_{2i,k} = y_{2i+1,k}, \quad \forall k \in \{1, \ldots, K\}, \quad \forall i \in \{0, \ldots, \ell\}. \tag{5.6e}$$

This constraint structure ensures that each configuration lies in at least one certified polytope, belongs to exactly one connected region of CP_{free}, and that consecutive configurations in transit steps remain within the same connected region.

Remark 4 *By enforcing connectivity constraints for both transfer and transit motions on alternate pairs of configurations, we ensure that each configuration in our path except from $\mathbf{q}_{init}$ and $\mathbf{q}_{goal}$ lie within the intersection of CG_{free} and CP_{free}. As formulated in Equation 5.3, this guarantees that our grasp and release configuration belong to connected components of $CG \cap CP$ (our nodes). Added to the implicit guarantee of connectivity constraints (connectivity of nodes through transfer and transit motions –our edges–), this ensures that the resulting path is a valid manipulation strategy, and shows our approach as a generalisation of the manipulation graph.*

5.4.3 Cost Function

The objective of the MIQP is to minimise the total path length in configuration space, which we approximate as the sum of squared Euclidean distances between consecutive configurations:

$$\min_{\mathbf{x}_0, \ldots, \mathbf{x}_{2\ell+1}} \sum_{i=0}^{2\ell} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|_2^2. \tag{5.7}$$

While the use of Euclidean distance in configuration space does not necessarily correspond to physical trajectory length in Cartesian space, it acts as an effective surrogate for trajectory efficiency and simplicity.

Because we iterate over increasing values of ℓ until the first feasible solution is found, this cost is minimised *within* each fixed-length candidate solution. The resulting plan is therefore optimal both in terms of the number of grasps (minimal ℓ) and, within ℓ, in terms of total path length between grasp and release configurations.

5.4.4 Summary and Full Problem Statement

We now summarise the full optimisation problem as a single mixed-integer quadratic program. The goal is to compute a discrete sequence of configurations $\{\mathbf{x}_i\}_{i=0}^{2\ell+1}$ that solves the task with minimal total path length, subject to the discussed constraints:

$$\begin{aligned}
\min_{\mathbf{x}_0,\ldots,\mathbf{x}_{2\ell+1}} \quad & \sum_{i=0}^{2\ell} \|\mathbf{x}_{i+1} - \mathbf{x}_i\|_2^2 && \text{(Eq 5.7)} \\
\text{s.t.} \quad & \mathbf{q}_{\min} \leq \mathbf{x}_i \leq \mathbf{q}_{\max} && \text{(Joint Limits, Eq. 5.4)} \\
& \text{Transfer Constraints} && \text{(Eq. 5.5a, 5.5b)} \\
& \text{Transit Constraints} && \text{(Eq. 5.5c)} \\
& \text{Connectivity Constraints} && \text{(Eq. 5.6a–5.6d)}
\end{aligned} \tag{5.8}$$

This MIQP formulation captures a complete manipulation planning problem under the convex optimisation framework. It ensures that all configurations are collision-free and within joint limits, that manipulation constraints are respected across alternating grasp and release actions, and that the sequence can be interconnected by feasible paths that belong to valid grasp or placement subspaces. The missing piece is the connection of these configurations through collision-free motions that respect the manipulation constraints. This is achieved in the next stage of our pipeline.

The integration of polytope containment, connected region logic, and manipulation-specific constraints into a unified MIQP is a key contribution of this work. It transforms the high-level structure of manipulation graphs into an optimisable program, using convex geometries to guarantee both feasibility and plan efficiency.

5.5 Trajectory Generation

Given the sequence of grasp and release configurations computed in the previous stage, the final step of the pipeline is to generate feasible, collision-free trajectories that connect them. Each trajectory segment must respect the associated manipulation path –either a *transit* or *transfer*– and lie entirely within the collision-free configuration space CS_{free}. This section describes how we use Graphs of Convex Sets (GCS) to generate such trajectories efficiently and how we assemble them into the final manipulation plan.

5.5.1 Graphs of Convex Sets (GCS)

For each pair of consecutive configurations $(\mathbf{x}_i, \mathbf{x}_{i+1})$ in the grasp-release sequence, we use GCS to compute a collision-free trajectory within CS_{free}. Because we already ensured in the previous stage that each pair lies within the same connected region of CG_{free} or CP_{free}, the trajectory problem reduces to finding a path between the two configurations that lie within the collision-free polytopes that describe CS_{free}.

This is achieved by passing the collision-free convex regions (CS_{free}), the start and end configurations $\mathbf{x}_i, \mathbf{x}_{i+1}$, and a cost function to the GCS solver. The cost function is defined to minimise the Euclidean path length between the two configurations, ensuring that the resulting trajectory is globally optimal in length.

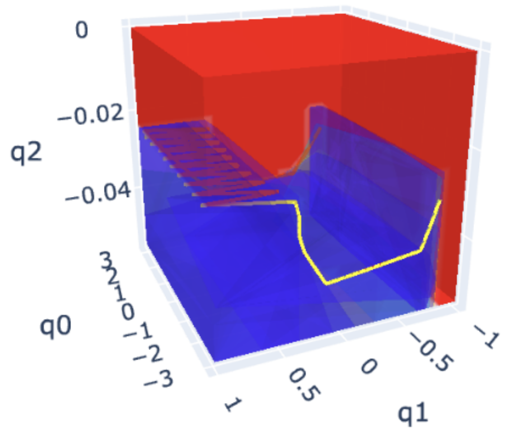

Figure 5.5: Example of GCS-generated trajectories for the bottle cap task.

Moreover, because our manipulation constraints fix certain degrees of freedom (e.g., the object pose during transit or the relative pose between gripper and object during transfer) at the start and end configurations, the minimal path length trajectory naturally preserves them, provided that a valid transit or transfer path exists and that CP_{free} and CG_{free} are not distorted within the submanifold where the constraints are satisfied. However, to guarantee completeness (i.e., that if a solution exists, it is found), we need to pass the manipulation constraints to the GCS solver to ensure they are met throughout the trajectory.

Remark 5 *As discussed in Remark 1 (Section 3.1.3), all transit and transfer paths lie within CP and CG respectively. However, not all paths in CP and CG are valid transit or transfer motions. This means that while the connectivity constraints guarantee that there is a path within CP_{free} or CG_{free} that connects our configurations, it does not ensure that a valid transit or transfer path exists. Our method guarantees that, if a feasible transit or transfer path within CP_{free} or CG_{free} exists, it will be found. However, if no such path exists, GCS will return an infeasible solution.*

This process yields *globally optimal*, collision-free trajectories for each individual transit and transfer segment, since GCS computes the shortest path over a convex decomposition of the feasible space. An example of such trajectories generated for our task is shown in Figure 5.5, where each segment lies entirely within CS_{free} and follows the optimal path between planned configurations.

5.5.2 Final Trajectory Concatenation

Once GCS sub-trajectories have been computed for each segment, we concatenate them into a full trajectory:

$$\tau = \bigcup_{i=0}^{2\ell} \tau_i \quad \text{where} \quad \tau_i = \text{GCS}(\mathbf{x}_i, \mathbf{x}_{i+1}), \tag{5.9}$$

resulting in a complete motion plan that is globally optimal within CS_{free}, and satisfies all manipulation constraints by construction.

Chapter 6

Manipulation Planner Algorithm

This chapter presents our second key contribution: the formalisation of a complete manipulation planning algorithm that integrates convex decomposition, mixed-integer optimisation, and trajectory generation into a unified pipeline. The algorithm is based on the principles of the manipulation graph, laid out in Section 3.1.4, where key grasp and placement configurations are nodes, and the edges are defined by the transfer and transit motions between them. In this chapter, we provide a concrete implementation of this idea through pseudocode, integrating the steps outlined in Chapter 5 into a single algorithmic framework.

6.1 Algorithm Description

As we defined in the problem statement of Chapter 4, the inputs of our planner are:

- A robotic manipulator $\mathcal{R}$, movable object $\mathcal{M}$, and static obstacles O.
- Valid grasp and placement spaces CP and CG in CS.
- A valid initial and goal configuration $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{goal}}$.
- A set of manipulation constraints $\mathcal{C}$.

$\mathcal{R}$, $\mathcal{M}$, and O are given by the scene. The user must define CP and CG through configuration bounds or transforms between the gripper and the object (${}^{G}X^{\mathcal{M}}$), as defined in Section 5.3.1, the manipulation constraints $\mathcal{C}$ as we demonstrated in Section 5.4.2 through equations 5.5a-5.5c, and the corresponding $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{goal}}$ for their desired trajectory. In addition, our algorithm takes as input the maximum number of grasps K that we want to consider in the planning process, and a boolean flag `certify` that indicates whether we want to certify the convex polytopes in which we decompose CS_{free}. The output of the algorithm is a trajectory τ that connects $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{goal}}$ through a sequence of grasps and releases while satisfying all constraints.

The complete algorithm is described in Algorithm 1. First, we compute a convex decomposition of the collision-free configuration space CS_{free} using the clique cover method, as defined in Section 5.3.2. Optionally, we invoke C-IRIS to rigorously certify

Algorithm 1 MANIPULATIONTRAJECTORYPLANNER

Input: Obstacles $\mathcal{O}$, Object $\mathcal{M}$, Robot $\mathcal{R}$, Constraints $\mathcal{C}$, Grasp space CG, Placement space CP, Initial and goal configs $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{goal}}$, Max grasps K, Certification flag `certify`
Output: Trajectory τ

```
1:  $CS_{\text{free}} \leftarrow$ DECOMPOSECSPACEWITHCLIQUECOVER$(\mathcal{O}, \mathcal{M}, \mathcal{R})$
2:  if certify then
3:      $CS_{\text{free}} \leftarrow$ CERTIFYREGIONSWITHCIRIS$(CS_{\text{free}})$
4:  end if
5:  $CG_{\text{free}}, CP_{\text{free}} \leftarrow$ COMPUTEGRASPPLACEMENTFREEREGIONS$(CG, CP, CS_{\text{free}})$
6:  for $\ell = 1$ to $K$ do
7:      $\pi \leftarrow$ SOLVEGRASPRELEASEMIQP$(\ell, CG_{\text{free}}, CP_{\text{free}}, \mathbf{q}_{\text{init}}, \mathbf{q}_{\text{goal}}, \mathcal{C})$
8:      if $\pi \neq \emptyset$ then
9:          break
10:     end if
11: end for
12: if $\pi = \emptyset$ then
13:     return ”No valid manipulation path found for less than $K$ grasps”
14: end if
15: $\tau \leftarrow$ GENERATETRANSFERTRANSITPATHSWITHGCS$(\pi, CS_{\text{free}}, \mathcal{C})$
16: return $\tau$
```

each region as collision-free in line 3. In line 5, we compute the grasp and placement free regions CG_{free} and CP_{free}, defined as the intersection of the convex polytopes that represent the grasp and placement spaces with the certified polytopes of CS_{free} (Section 5.3.3). CG_{free} and CP_{free} represent the feasible submanifolds of CS_{free} where the transfer and transit motions of the trajectory will occur.

Next, we search for the shortest sequence of alternating grasp and release configurations that solve the task. This is achieved by solving the MIQP formulated in Section 5.4 (Eq. 5.8), which enforces all constraints –joint limits (Eq. 5.4), manipulation constraints $\mathcal{C}$ (Eq. 5.5) and connectivity constraints (Eq. 5.6)– and uses a path-length cost (Eq. 5.7) to obtain efficient solutions. The algorithm iterates over increasing values of ℓ (the number of grasps) until a feasible plan is found or the maximum number of grasps K is reached. If a solution is found, it corresponds to a path of configurations that solves the task in the minimum number of grasps, which we denote π. These grasp-release configurations represent the nodes of the manipulation graph.

Finally, given the selected configurations, we use GCS (Section 5.5) to compute collision-free, optimal-length trajectories that connect them, within the generated collision-free polytopes CS_{free}. This constructs the actual edges of the graph, generating feasible transit and transfer motions that lie entirely in CS_{free} and, if necessary, enforce the manipulation constraints directly. These trajectories are then concatenated as explained in Eq. 5.9.

The optional certification step for polytopes is encapsulated in Algorithm 2, corresponding to the polytope shrinking process described in Section 5.3.2. We first convert the polytopes to the TC-space representation (line 1), which consists of extracting the original polytopes' vertices and computing the TC-space variable $\mathbf{s}$ of each, computing the convex hull of the set and extracting the vertices again, conforming the newly parametrised polytope. Then, we scale the polytopes down using a binary search strategy to find the minimum shrinkage that guarantees a collision-free certificate (line 2). Finally, we convert the resulting polytopes back to the original configuration space representation $\mathbf{q}$ (line 3). The shrinkage step has been retrieved from the C-IRIS implementation (Dai et al., 2023), but the method to convert polytopes to and from TC-space is a contribution of this work.

Algorithm 2 CERTIFYREGIONSWITHCIRIS

```
Input: Free regions $CS_{\text{free}}$
Output: Certified free regions $CS_{textcertified}$
1: $\mathcal{S}_{\text{regions}} \leftarrow$ CONVERTPOLYTOPESTOSVALUE($CS_{\text{free}}$)
2: $\mathcal{S}_{\text{cert}} \leftarrow$ FINDMINSHRINKAGEWITHCERTIFICATE($\mathcal{S}_{\text{regions}}$)
3: $\mathcal{Q}_{\text{certified}} \leftarrow$ CONVERTPOLYTOPESTOQVALUE($\mathcal{S}_{\text{cert}}$)
4: return $CS_{\text{certified}}$
```

6.2 Summary of Contributions

This chapter has formalised the complete planning pipeline into a concrete algorithm that generates collision-free, optimal manipulation trajectories. These trajectories are optimal in the number of grasps required, and the transfer and transit motions between grasp/release configurations are globally optimal in length within the CS_{free} representation of the scene.

Besides trivial inputs such as the initial and goal configurations of the trajectory or the maximum number of grasps to consider, the algorithm only requires the user to specify the grasp and placement spaces and the manipulation constraints required to fix the object at rest or in the robot's gripper. These requirements are task-specific and do not differ at the core from those of existing planners like Siméon et al. (2004a).

In addition to the overall integration, several components of this planner represent contributions of this work, including:

- The definition and use of collision-free grasp and placement regions (CG_{free}, CP_{free}) as subspaces of CS_{free}, maintaining convexity through intersection and making these representations suitable for convex optimisation.
- The reparametrisation of polytopes to and from TC-space to allow certification of convex regions using C-IRIS.
- The MIQP formulation to generate grasp-release sequences as a first step to manipulation planning.

We provide a full implementation of our manipulation planner in Drake, through a containarised environment, in this GitHub repository[*]

[*]https://github.com/julialopezgomez/minf2-repo.

Chapter 7

Experiments

This chapter presents the experimental analysis to evaluate our manipulation planning algorithm. We tested the algorithm in two scenes: the 3 DOF bottle cap scenario from the task described in Chapter 5, and the 4 DOF equivalent by including the motion of the other gripper finger. We analyse the performance of the algorithm in terms of CS_{free} coverage and time efficiently in trajectory generation.

We ran all experiments using the same setup with 3 and 4 DOF, aiming to rotate a bottle cap from $-\pi$ to π, and starting from the same gripper initial orientation. The only difference in both scenes is an extra prismatic joint in the 4 DOF scene, increasing the dimensionality of CS.

Experiment 1: CS_{free} decomposition with Clique Covers

DOF	Coverage (Actual)	# Regions	Coverage Time (s)	Traj. Time (s)
3	0.70-0.90 (0.94)	4	29.8	29.5
3	0.96 (0.98)	7	54.6	34.0
3	0.99 (0.996)	11	87.7	112.6
4	0.70 (0.77)	4	16.5	*
4	0.90 (0.95)	5	31.2	**
4	0.96 (0.98)	10	78.5	29.0
4	0.99 (0.996)	14	115.9	95.9

Table 7.1: Decomposition and trajectory generation results using clique covers for different DOFs systems and coverage thresholds. Actual coverage may exceed the requested minimum. Asterisk * indicates no feasible grasp-release sequence was found; ** indicates the sequence was found but transit/transfer connectivity failed.

Table 7.1 reports the CS_{free} decomposition and trajectory generation results for various coverage thresholds. The first column indicates the coverage input to the Clique Cover algorithm, while the value in parentheses denotes the actual achieved coverage.

As expected, increasing coverage improves the likelihood of finding feasible paths and trajectories. However, the results show that coverage entails computational cost: both the number of convex regions and the decomposition time grow rapidly with the requested coverage. At 4 DOF, low coverage (e.g., 0.70) is insufficient to find a valid grasp-release path, while moderate coverage (0.95) is still insufficient to ensure transit/transfer connections. Only coverage at 0.96 and above achieve the feasibility of the planner. This highlights the dependence of manipulation tasks to CS_{free} representation: the higher the DOF, the higher the coverage required to ensure a valid path. This is expected, as valid grasps are in contact with the object (almost in collision), and thus, the more precise the CS_{free} representation needs to be.

Moreover, the results indicate that the time required for trajectory generation is directly correlated with the number of regions generated in CS_{free}, but not with the number of DOF of the system. Trajectory generation was less efficient in the 3 DOF system despite involving fewer regions, which did not align with our expectations. Although other factors can cause higher computational cost in convex optimisation methods like GCS (e.g. irregularly shaped figures), we believe this is a topic for further investigation.

Experiment 2: C-IRIS vs. Clique Cover with IRIS-NP

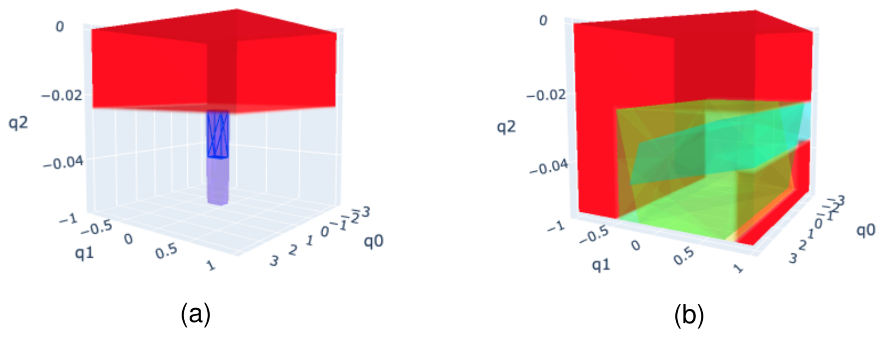

Figure 7.1: CS representation of 3 DOF task, with obstacles in red. Figure 7.1a, shows an iteration of the C-IRIS algorithm in an environment with no obstacles. Figure 7.1b shows an iteration of Clique Covers with IRIS-NP in an environment with obstacles.

Figure 7.1 illustrates the qualitative difference between the C-IRIS algorithm (a) and Clique Covers with IRIS-NP (b). In an empty environment, C-IRIS requires both manual seeding and user-defined input polytopes in the TC-space, producing limited and inefficient region growth. In contrast, Clique Covers with IRIS-NP automatically generates a diverse set of regions, even in the presence of obstacles, demonstrating greater coverage and adaptability. This highlights the limitations of C-IRIS (or IRIS-NP alone) in complex or cluttered settings, especially where visualisations are intractable.

Video Demonstration

We provide a video demonstration of our manipulation planning algorithm, showcasing the feasibility of the method and the scalability to higher DOF without further input required. The video includes examples of both the 3 DOF and 4 DOF systems*.

*3 DOF: `https://youtu.be/xkL_KB9SdUE`; 4 DOF: `https://youtu.be/XcCufms6jA0`

Chapter 8

Discussion and Future Work

Our results demonstrate the feasibility of formulating manipulation planning as a mixed-integer convex optimisation problem by leveraging recent advances in convex decomposition. While clique covers generate precise decompositions of C-free that enable solving full manipulation trajectories, coverage quality is critical to the success of this pipeline. Our experiments confirmed that even minor reductions in coverage can make the MIQP fail to produce a valid manipulation path or generate disconnected grasp/release configurations. This sensitivity highlights the main limitation of our approach: while convex decompositions like those achieved enable optimisation-based planning, they require highly precise coverage, which becomes harder to achieve as the C-space grows in dimensionality. While our experiments were limited to low-dimensional setups for clarity and tractability, extending this to more dexterous robots may require more robust constraint formulations and further improvements in decomposition methods. We also observed variability in GCS performance, particularly in timing and C-space dimensionality, suggesting that trajectory generation may benefit from exploring alternative approaches such as direct Mixed-Integer formulations.

Although we did not evaluate sampling-based planners such as RRT or PRM in this report, comparing these against our method remains an important direction for future work. Such comparisons would help quantify trade-offs between speed and guarantees. Future work should explore these comparisons in depth, along with scaling to higher-dimensional problems, improving coverage generation, and refining constraint definitions for broader manipulation tasks. Extending the approach to support contact-rich interactions like pushing or in-hand manipulation is another promising direction.

Conclusions. This project introduces a manipulation planner that operates on convex decompositions of CS_{free} and enables optimisation-based trajectory generation. We demonstrated the pipeline's viability on a representative task and highlighted the central role that precise decomposition plays in enabling feasible manipulation plans. Our contributions include the adaptation of modern decomposition algorithms to manipulation planning, a fully implemented pipeline integrating MIQP and GCS solvers, and a discussion of coverage sensitivity. This work lays the foundation for further exploration of optimisation-driven manipulation planning in increasingly complex scenarios.

Bibliography

Mixed integer programming. Online. URL https://www.gurobi.com/faqs/mixed-integer-programming/. Accessed: 2025-03-30.

C.C. Adams and R.D. Franzosa. *Introduction to Topology: Pure and Applied.* Pearson Prentice Hall, 2008. ISBN 9780131848696. URL https://books.google.co.uk/books?id=W1wnAQAAIAAJ.

J.M. Ahuactzin, K. Gupta, and E. Mazer. Manipulation planning for redundant robots: A practical approach. *The International Journal of Robotics Research*, 17(7):731–747, 1998. doi: 10.1177/027836499801700704. URL https://doi.org/10.1177/027836499801700704.

R. Alami, T. Siméon, and J.P. Laumond. A geometrical approach to planning manipulation tasks. the case of discrete placements and grasps. In Hirofumi Miura, editor, *The fifth international symposium on Robotics research* , pages 453–463. MIT Press, 1990. URL https://hal.science/hal-01309950.

J. Barraquand and P. Ferbach. A penalty function method for constrained motion planning. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 1235–1242 vol.2, 1994. doi: 10.1109/ROBOT.1994.351317.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

M.S. Branicky and W.S. Newman. Rapid computation of configuration space obstacles. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 304–310 vol.1, 1990. doi: 10.1109/ROBOT.1990.125992.

J.F. Canny. The complexity of robot motion planning. 1988. URL https://api.semanticscholar.org/CorpusID:54161536.

H. Dai, A. Amice, P. Werner, A. Zhang, and R. Tedrake. Certified polyhedral decompositions of collision-free configuration space, 2023.

R. Deits and R. Tedrake. *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*, pages 109–124. Springer International Publishing, Cham, 2015a. doi: 10.1007/978-3-319-16595-0_7. URL https://doi.org/10.1007/978-3-319-16595-0_7.

R. Deits and R. Tedrake. Efficient mixed-integer planning for uavs in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49, 2015b. doi: 10.1109/ICRA.2015.7138978.

S.J. Eidenbenz and P. Widmayer. An approximation algorithm for minimum convex cover with logarithmic performance guarantee. *SIAM Journal on Computing*, 32(3): 654–670, 2003. doi: 10.1137/S0097539702405139. URL `https://doi.org/10.1137/S0097539702405139`.

Í. Elguea-Aguinaco, A. Serrano-Muñoz, D. Chrysostomou, I. Inziarte-Hidalgo, S. Bøgh, and N. Arana-Arexolaleiba. A review on reinforcement learning for contact-rich robotic manipulation tasks. *Robotics and Computer-Integrated Manufacturing*, 81:102517, 2023. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2022.102517. URL `https://www.sciencedirect.com/science/article/pii/S0736584522001995`.

M Ghosh, N.M. Amato, Y. Lu, and J.M. Lien. Fast approximate convex decomposition using relative concavity. *Computer-Aided Design*, 45(2):494–504, 2013. ISSN 0010-4485. doi: https://doi.org/10.1016/j.cad.2012.10.032. URL `https://www.sciencedirect.com/science/article/pii/S0010448512002370`. Solid and Physical Modeling 2012.

LLC Gurobi Optimization. Gurobi optimizer reference manual, 2024. URL `https://www.gurobi.com`.

P.M. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Trans. Graph.*, 15(3):179–210, jul 1996. ISSN 0730-0301. doi: 10.1145/231731.231732. URL `https://doi.org/10.1145/231731.231732`.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning, 2011. URL `https://arxiv.org/abs/1105.1186`.

L.E. Kavraki. Computation of configuration-space obstacles using the fast fourier transform. *IEEE Transactions on Robotics and Automation*, 11(3):408–413, 1995. doi: 10.1109/70.388783.

L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996. doi: 10.1109/70.508439.

M. Kleinbort, K. Solovey, Z. Littlefield, K.E. Bekris, and D. Halperin. Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters*, 4(2):i–vii, 2019. doi: 10.1109/LRA.2018.2888947.

Y. Koga and J.C. Latombe. On multi-arm manipulation planning. In *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, pages 945–952 vol.2, 1994. doi: 10.1109/ROBOT.1994.351231.

J.C. Latombe. *Robot Motion Planning*. Springer Science & Business Media, December 2012. ISBN 978-1-4615-4022-9. Google-Books-ID: nQ7aBwAAQBAJ.

S.M. LaValle. Rapidly-exploring random trees : a new tool for path planning.

The annual research report, 1998. URL https://api.semanticscholar.org/CorpusID:14744621.

J.M. Lien and N.M. Amato. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM Symposium on Solid and Physical Modeling*, SPM '07, page 121–131, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936660. doi: 10.1145/1236246.1236265. URL https://doi.org/10.1145/1236246.1236265.

A. Lingas. The power of non-rectilinear holes. In M. Nielsen and E.M. Schmidt, editors, *Automata, Languages and Programming*, pages 369–383, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg. ISBN 978-3-540-39308-5.

T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108–120, 1983. doi: 10.1109/TC.1983.1676196.

K.M. Lynch and F.C. Park. *Modern robotics: mechanics, planning, and control*. Cambridge university press, Cambridge, 2017. ISBN 978-1-107-15630-2.

J. López Gómez. Manipulation planning with a robotic arm: A tangent configuration space approach. Minf project (part 1) report, The University of Edinburgh, Edinburgh, UK, 2024. URL https://julialopezgomez.github.io/files/MINF1.pdf.

T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake. Motion planning around obstacles with convex optimization, 2022.

F. Mesadi and T. Tasdizen. Convex decomposition and efficient shape representation using deformable convex polytopes, 2016. URL https://arxiv.org/abs/1606.07509.

ApS MOSEK. *The MOSEK optimization toolbox for MATLAB manual. Version 10.1.*, 2024. URL http://docs.mosek.com/latest/toolbox/index.html.

C.L. Nielsen and L.E. Kavraki. A two level fuzzy prm for manipulation planning. In *Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113)*, volume 3, pages 1716–1721 vol.3, 2000. doi: 10.1109/IROS.2000.895219.

M. Petersen and R. Tedrake. Growing convex collision-free regions in configuration space using nonlinear programming, 2023. URL https://arxiv.org/abs/2303.14737.

H. Ravichandar, A. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3, 05 2020. doi: 10.1146/annurev-control-100819-063206.

A. Sahbani, J. Cortés, and T. Siméon. A probabilistic algorithm for manipulation planning under continuous grasps and placements. volume 2, pages 1560 – 1565 vol.2, 02 2002. ISBN 0-7803-7398-7. doi: 10.1109/IRDS.2002.1043977.

T. Schouwenaars, B. De Moor, E. Feron, and J.P. How. Mixed integer programming for multi-vehicle path planning. *2001 European Control Conference (ECC)*, pages 2603–2608, 2001. URL https://api.semanticscholar.org/CorpusID:5379524.

Schunk. Wsg series: Universal gripper, n.d. URL `https://schunk.com/no/en/gripping-systems/parallel-gripper/wsg/c/PGR_820`.

M. Sherman. Rethinking Contact Simulation for Robot Manipulation, June 2022. URL `https://medium.com/toyotaresearch/rethinking-contact-simulation-for-robot-manipulation-434a56b5ec88`.

T. Siméon, J. Cortés, A. Sahbani, and J.P. Laumond. A manipulation planner for pick and place operations under continuous grasps and placements. volume 2, pages 2022–2027, 01 2002. doi: 10.1109/ROBOT.2002.1014838.

T. Siméon, J. Cortés, A. Sahbani, and J.P. Laumond. A General Manipulation Task Planner. In *Algorithmic Foundations of Robotics V. Springer Tracts in Advanced Robotics.*, pages 311–327. 2004a. URL `https://laas.hal.science/hal-01988619`.

T. Siméon, J.P. Laumond, J. Cortés, and A. Sahbani. Manipulation planning with probabilistic roadmaps. *I. J. Robotic Res.*, 23:729–746, 01 2004b.

M. Suomalainen, Y. Karayiannidis, and V. Kyrki. A survey of robot manipulation in contact. *Robotics and Autonomous Systems*, 156:104224, October 2022. ISSN 0921-8890. doi: 10.1016/j.robot.2022.104224. URL `http://dx.doi.org/10.1016/j.robot.2022.104224`.

R. Tedrake. *Robotic Manipulation*. 2023a. URL `http://manipulation.mit.edu`.

R. Tedrake. Motion planning around obstacles with graphs of convex sets. Presented at CMU Robotics Institute Seminar, video available at `https://www.youtube.com/watch?v=KSCC7mVJzaw`, January 2023b. URL `https://www.ri.cmu.edu/event/ri-seminar-russ-tedrake-mit-professor-2023-01-27/`. Seminar at Carnegie Mellon University, Robotics Institute.

R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL `https://drake.mit.edu`.

C.W. Wampler and A.J. Sommese. Numerical algebraic geometry and algebraic kinematics. *Acta Numerica*, 20:469–567, 2011. doi: 10.1017/S0962492911000067.

P. Werner, A. Amice, T. Marcucci, D. Rus, and R. Tedrake. Approximating robot configuration spaces with few convex sets using clique covers of visibility graphs, 2024a. URL `https://arxiv.org/abs/2310.02875`.

P. Werner, T. Cohn, R.H. Jiang, T. Seyde, M. Simchowitz, R. Tedrake, and D. Rus. Faster algorithms for growing collision-free convex polytopes in robot configuration space, 2024b. URL `https://arxiv.org/abs/2410.12649`.

Appendix A

Glossary of terms

Term	**Definition**
Configuration	Set of joint values defining a robot's posture.
C-space	Configuration space: Space of all possible robot configurations, parametrized by joint values.
TC-space	Tangent configuration space: C-space reparametrised via stereographic projection.
C-free	Collision-free submanifold of C-space.
$C_{\mathbf{rob}}$	Robot configuration space (C_{rob}): Space of all robot configurations.
$C_{\mathbf{obj}}$	Object configuration space (C_{obj}): Space of all object configurations.
CS	Composite configuration space ($C_{\text{rob}} \times C_{\text{obj}}$) of the robot and object.
$CS_{\mathbf{free}}$	Collision-free submanifold of CS.
CG	Grasp space (CG): Valid configurations where robot grasps the object.
CP	Placement space (CP): Valid configurations where object rests stably.
Submanifold	Subspace. A is a submanifold of B if $A \subseteq B$, i.e. if A is fully contained in B (e.g., $C_{\text{rob}} \subseteq CS$).
Convex set	Set where any two points are connected by a line segment within the set.
C-IRIS	IRIS variant providing rigorous collision-free guarantees.
IRIS-NP	IRIS for non-convex obstacles: Approximate decomposition with probabilistic guarantees.
DOF	Degrees of freedom: Independent parameters defining a system's state.
Task space	Physical workspace (e.g., SE(3)) where robot operates.
Transit path	Robot motion in C_{rob} while not grasping an object.
Transfer path	Robot motion in $C_{\text{rob}} \times C_{\text{obj}}$ while maintaining grasp.
Movable object $\mathcal{M}$	Object that can only move when grasped by the robot.
End effector	Robot's tool or hand.
SE(3)	Special Euclidean group: 3D rigid body transformations.
SO(3)	Special orthogonal group: 3D rotations.

Appendix B

Key Definitions

The definitions and description of notation, arithmetics and rotations explained in this appendix are retrieved and adapted from Part 1 of this project (López Gómez, 2024).

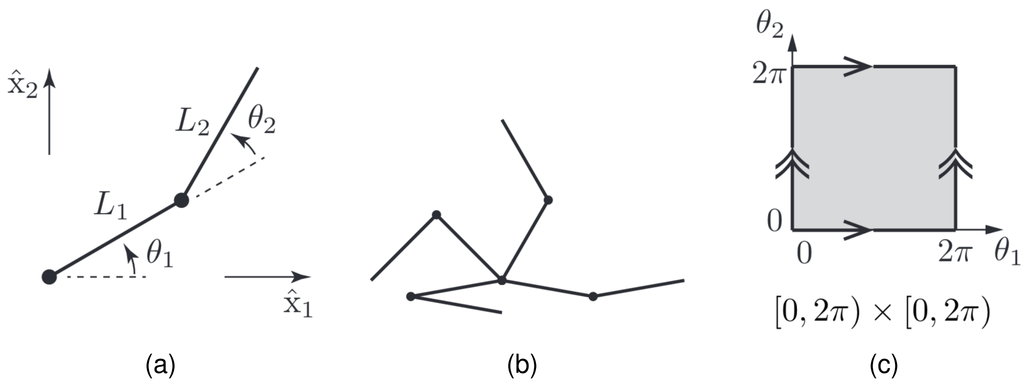

Figure B.1: Adapted from Lynch and Park (2017). (a) A double pendulum, or 2-DOF robot arm, with joint angles θ_1 and θ_2, and link lengths L_1 and L_2. (b) The arm at four different configurations. Each configuration is determined by the values of θ_1 and θ_2. (c) A sample representation of the configuration space of the pendulum, where each DOF (θ_1 and θ_2) can be any angle between $[0, 2\pi)$.

In this section, we introduce key terms to understand these problems in order to comprehend the purpose, objectives, and relevance of this work:

Definition 4 (Degrees of Freedom (DOF)) *The **degrees of freedom** of a robot are the number of independent variables that define its position and orientation in space. The DOF of a robotic arm usually come defined by the angles of its joints. In Figure B.1a, the pendulum has two DOF because its position is defined by θ_1 and θ_2.*

Definition 5 (End-effector) *The **end-effector** is the part of the robot that interacts with the environment. In Figure B.1a, the end-effector is the tip of the pendulum.*

Definition 6 (Configuration) *The **configuration of a robot** is a specification of the position of all points of the robot. Intuitively, it is a pose of the entire robot system. We* *can define a configuration by assigning values to the DOF of the robot. See Figure B.1b.*

Definition 7 (Configuration Space (C-space)) *The **configuration space** or **C-space** is the space of all configurations of a robot. Its dimension is the number of DOF of the robot. In Figure B.1, the C-space is 2-dimensional because the pendulum has two DOF. Each point in the C-space in B.1c represents a different value for* θ_1 *and* θ_2*, and thus, a different configuration.*

Definition 8 (Collision-Free Configuration Space (C-free)) *We denominate **collision-free configuration space** (**C-free**) to the space of all configurations in which a robot is not in collision.*

Definition 9 (Task Space) *The **task space** is the space in which the robot operates in the real world (e.g., the 3D-Cartesian space for a real-life 3D arm). In Figure B.1a, the task space is a 2D-Cartesian space (because the pendulum is planar, not tridimensional) represented by the axes* $\hat{x}_1, \hat{x}_2$*. The task space is distinct from the robot's C-space, as one point in task space may be achievable by more than one robot configuration.*

Appendix C

Stereographic Projection

This appendix provides a visual interpretation of the stereographic projection in the unit circle (2D), when θ is bounded to $-\pi < \theta_i < \pi$. This bound θ to $-\pi < \theta_i < \pi$ guarantees bijectivity, i.e. for each value of θ_i there is a unique t_i, and also bounds on the projection (if θ could take the value of π or $-\pi$, the stereographic projection would project to $\pm\infty$).

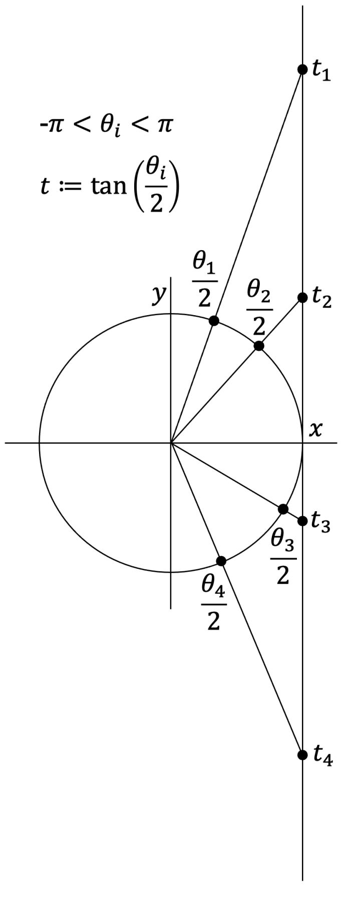

Figure C.1: Visual interpretation of the stereographic projection in the unit circle (2D), when θ is bounded to $-\pi < \theta_i < \pi$.

Appendix D

Clique Covers for Automatic Seeding in C-free Decomposition

Convex decomposition algorithms such as C-IRIS and IRIS-NP require an initial collision-free region to begin the region-growing process. Choosing these seed regions is not a trivial task. Manual specification is often tedious, unreliable, and task-specific. To address this limitation, Werner et al. (2024a) introduced an approach to automatically generate seed regions using *clique covers of visibility graphs*. This appendix provides an intuitive yet rigorous explanation of the concepts behind this technique.

Visibility Graphs

Definition 10 (Visibility Graph) *Given a robot in a workspace with obstacles, the* visibility graph *is a graph representation of the robot's collision-free configuration space. Each node in the graph corresponds to a configuration of the robot. Two nodes are connected by an edge if the straight-line path between the corresponding configurations lies entirely within the free space, i.e., the robot can move directly between them without colliding with any obstacles.*

Visibility graphs are useful for representing the connectivity of CS_{free}, and in this context, they provide a foundation for identifying regions that can be approximated as convex.

Clique Covers

Definition 11 (Clique) *Given a graph, a* clique *is a subset of vertices that are all mutually connected. In the visibility graph, a clique corresponds to a set of sampled configurations that can all directly "see" each other. That is, each configuration is reachable from any other via a straight-line path in C-free.*

A *clique cover* is a collection of cliques whose union contains all the nodes in the visibility graph. Each of these cliques can then be used to initialise a convex region, typically by computing the convex hull of the corresponding configurations. These convex hulls are then used as seed polytopes for region-growing algorithms such as IRIS-NP or C-IRIS.

The clique cover algorithm process is depicted in Figure D.1.

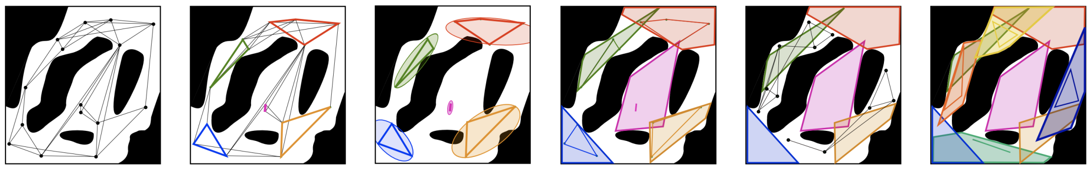

Figure D.1: Illustration of the clique cover method. Sampled configurations are grouped into cliques based on mutual visibility. The convex hull of each clique (shaded region) approximates a local convex region of CS_{free}, which is then growed using IRIS variants. Adapted from Werner et al. (2024a).

Appendix E

Monogram Notation and Arithmetic Rules

This appendix provides a brief description of the monogram notation used in this work, as well as some arithmetic rules for positions, rotations, and poses.

The contents of this appendix are adapted from Part 1 of this thesis (López Gómez, 2024, Chapter 3.2).

E.1 Monogram Notation

To describe the motion of robotic systems, we use the reference frames of their links and joints. We are interested in representing position p and orientation R. For this purpose, we use the monogram notation:

${}^{B}p^{A}_{C}$	${}^{B}R^{A}$	${}^{B}X^{A}$
Position of point or frame A measured from point or frame B expressed in frame C	Orientation of frame A measured from frame B	Pose of frame A measured from frame B

(E.1)

X represents the spatial pose (or pose), which is the position[*] and orientation of a frame relative to another. We can completely specify a frame using its pose X. The spatial transform (or transform) is the "verb form" of the pose.

A thorough description of this notation can be found in Tedrake (2023a, Chapter 3.1).

Essential remarks on this notation:

- When we want to represent the position p of a point or frame A, measured from a *frame B* and expressed in *the same frame B*, the subscript can be omitted:

$$ {}^{B}p^{A}_{B} \equiv {}^{B}p^{A} $$

[*]The position of a frame coincides with the frame origin.

- If we consider measurements relative to the world frame W, W can be omitted:

$$ {}^{W}p^{A}_{W} \equiv {}^{W}p^{A} \equiv p^{A}, \quad {}^{W}R^{A} \equiv R^{A}, \quad {}^{W}X^{A} \equiv X^{A} $$

A brief description of elementary Spatial Algebra operations and arithmetic rules represented with this notation is given below.

E.2 Arithmetic Rules

We provide common addition, inverse, and multiplication rules for positions, rotations and poses. A thorough revision of these concepts can be found in Tedrake (2023a, Section 3.3).

- **Addition of Positions:** Positions can be added if they are expressed in the same frame and their target and “measured from” frames match:

$$ {}^{A}p^{B}_{F} + {}^{B}p^{C}_{F} = {}^{A}p^{C}_{F} \tag{E.2} $$

- **Additive Inverse:**

$$ {}^{A}p^{B}_{F} = -{}^{B}p^{A}_{F} \tag{E.3} $$

- **Multiplication by a Rotation:** Multiplying positions by a rotation changes the “expressed in” frame:

$$ {}^{A}p^{B}_{C} = {}^{C}R^{F}\,{}^{A}p^{B}_{F} \tag{E.4} $$

- **Multiplication:** Rotations and transforms can be multiplied when their reference and target symbols match:

$$ {}^{A}R^{B}\,{}^{B}R^{C} = {}^{A}R^{C}, \quad {}^{A}X^{B}\,{}^{B}X^{C} = {}^{A}X^{C} \tag{E.5} $$

- **Inverse Operation:** Rotations and transforms have an inverse (when they are squared matrices):

$$ \left[{}^{A}R^{B}\right]^{-1} = {}^{B}R^{A}, \quad \left[{}^{A}X^{B}\right]^{-1} = {}^{B}X^{A} \tag{E.6} $$

Appendix F

Joint Decomposition and Algebraic Kinematics

This appendix is retrieved from our MInf 1 report López Gómez (2024). We summarise the different joint decompositions into revolute and prismatic as shown in (Wampler and Sommese, 2011, Chapter 4), with the addition of the universal joint (U), which was a contribution of López Gómez (2024).

We define six types of algebraic joints:

- **Revolute (R):** 1-DOF joint that rotates about an axis. We refer to the rotation angle as θ. An example is a simple hinge or door handle.
- **Prismatic (P):** 1-DOF joint that translates in an axis. We refer to the translation variable as z. An example is a straight linear rail.
- **Cylindrical (C):** 2-DOF joint that allows independent translation and rotation about an axis. An example is the rods of a foosball table.
- **Planar (E):** 3-DOF joint that translates in two directions, and rotates in the 2D plane defined by the translation axes. An example is an air-hockey table.
- **Spherical (S):** 3-DOF joint that allows rotation about three axes. That is, free rotation between two links. An example of the human elbow.
- **Universal joint (U):** 2-DOF joint that rotates about two orthogonal axes. They are commonly used in the steering system of vehicles.

${}^{P_i}X^{C_i}(q_i)$ represents the relative motion of frame C_i expressed in frame P_i of frame C_i expressed in frame P_i under the change of change of joint q_i. We can represent this transform in the form:

$$
{}^{P_i}X^{C_i}(q_i) = \begin{cases} \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & x_i \\ \sin(\theta_i) & \cos(\theta_i) & 0 & y_i \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} & \text{if the } i^{\text{th}} \text{ joint is R, P, C or E.} \\ \\ \begin{bmatrix} M(\psi_i) & 0_{3\times 1} \\ 0_{1\times 3} & 0 \end{bmatrix} & \text{if the } i^{\text{th}} \text{ joint is S or U.} \end{cases} \tag{F.1}
$$

Each time you add a 3D rigid body/link to a robot system, you add 6 DOF, but these get restricted by the joint type included. Here we provide a summary of the constraints applied by each joint to a transform of the form of expression F.1:

Joint	Restriction	Definition of q_i
R	$x_i = y_i = z_i = 0$	$q_i = \{\theta_i\}$
P	$\theta_i = x_i = y_i = 0$	$q_i = \{z_i\}$
C	$x_i = y_i = 0$	$q_i = \{\theta_i, z_i\}$
E	$z_i = 0$	$q_i = \{\theta_i, x_i, y_i\}$
S	see equation (F.4)	$q_i = \{\phi_{i,x}, \phi_{i,y}, \phi_{i,z}\}$
U	see equations (F.5, F.6, F.7)	$q_i = \{\phi_{i,x}, \phi_{i,y}\}$ or $\{\phi_{i,x}, \phi_{i,z}\}$ or $\{\phi_{i,y}, \phi_{i,z}\}$

We can express the joints C, E, S and U decomposed into P and R:

- C joint decomposed into one R and one P:

$$
\begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.2}
$$

- E joint decomposed into two P and one R:

$$
\begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & x_i \\ \sin(\theta_i) & \cos(\theta_i) & 0 & y_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y_i \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{F.3}
$$

- S joint decomposed into three R expressed as Euler angles:

$$M(\psi_i) = \begin{bmatrix} \cos(\psi_{i,x}) & -\sin(\psi_{i,x}) & 0 \\ \sin(\psi_{i,x}) & \cos(\psi_{i,x}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\psi_{i,y}) & 0 & -\sin(\psi_{i,y}) \\ 0 & 1 & 0 \\ \sin(\psi_{i,y}) & 0 & \cos(\psi_{i,y}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_{i,z}) & -\sin(\psi_{i,z}) \\ 0 & \sin(\psi_{i,z}) & \cos(\psi_{i,z}) \end{bmatrix} \tag{F.4}$$

- U joint decomposed into two R joints expressed as Euler angles:

$$M^{\backslash x}(\psi_i) = \begin{bmatrix} \cos(\psi_{i,y}) & 0 & -\sin(\psi_{i,y}) \\ 0 & 1 & 0 \\ \sin(\psi_{i,y}) & 0 & \cos(\psi_{i,y}) \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_{i,z}) & -\sin(\psi_{i,z}) \\ 0 & \sin(\psi_{i,z}) & \cos(\psi_{i,z}) \end{bmatrix} \tag{F.5}$$

or

$$M^{\backslash y}(\psi_i) = \begin{bmatrix} \cos(\psi_{i,x}) & -\sin(\psi_{i,x}) & 0 \\ \sin(\psi_{i,x}) & \cos(\psi_{i,x}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_{i,z}) & -\sin(\psi_{i,z}) \\ 0 & \sin(\psi_{i,z}) & \cos(\psi_{i,z}) \end{bmatrix} \tag{F.6}$$

or

$$M^{\backslash z}(\psi_i) = \begin{bmatrix} \cos(\psi_{i,x}) & -\sin(\psi_{i,x}) & 0 \\ \sin(\psi_{i,x}) & \cos(\psi_{i,x}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\psi_{i,y}) & 0 & -\sin(\psi_{i,y}) \\ 0 & 1 & 0 \\ \sin(\psi_{i,y}) & 0 & \cos(\psi_{i,y}) \end{bmatrix} \tag{F.7}$$