Manipulation Planning with a Robotic Arm: A Tangent Configuration Space Approach

Julia López Gómez



MInf Project (Part 1) Report
Master of Informatics
School of Informatics
University of Edinburgh

2024

Abstract

Efficiently describing robot displacement is a complex task due to the immense number of possibilities involved, making motion planning a fundamental challenge in robotics. Defining the collision-free configuration space (C-free) has become essential to formalise this problem, with existing methods imposing limiting assumptions on the task or only offering probabilistic guarantees of success. The C_IRIS algorithm was introduced in 2023, which described collision-free motion for the first time in a rational, bijective parametrisation of C-free, called the tangent configuration space (TC-space). This new approach allowed assumptions to be made solely on the convexity of obstacles in the 3D Cartesian space while providing rigorous guarantees of non-collision.

This dissertation explores the implications of utilising the TC-space to describe manipulation planning tasks. In this work, we define the fundamental mathematical tools necessary for manipulation applications, formalising their scalability within the context of the tangent configuration space. We describe the forward and inverse kinematics in this new parametrisation, formalise grasping constraints for standard geometries, and showcase the implementation of these concepts using the Drake robotics toolbox. In addition, we demonstrate the extent of C_IRIS to robots with universal joints. Ultimately, our research aims to establish a foundation for future work in manipulation planning by presenting current tools within the framework of this parametrisation.

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, for his help and guidance throughout the last year.

This dissertation is dedicated to my mother, my wonderful flatmates, my friends, and my family. Without them and their continuous support, this would not be possible.

Last but not least, I wanted to thank all of the outstanding teachers I've had throughout my life, with a big shoutout to Leo, who introduced me to robotics. Without them, not only would this work not be possible, but I probably would not be where I am today.

Table of Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Project Aims and Limitations . . . 2
- 1.3 Achievements and Contributions . . . 2
- 1.4 Report Structure . . . 3

2 Background **4**
- 2.1 Definitions . . . 4
  - 2.1.1 Dealing with Complexity . . . 5
- 2.2 Previous Work . . . 5
  - 2.2.1 Motion Planning and C-free Decomposition . . . 5
  - 2.2.2 Manipulation Planning . . . 7
- 2.3 Drake . . . 8

3 Assumptions and Problem Statement **10**
- 3.1 Joint Requirements . . . 11
- 3.2 Notation . . . 11
- 3.3 Decomposition of Universal Joints . . . 12

4 Manipulation Planning Pipeline **14**
- 4.1 Basic Pick-and-Place . . . 14
  - 4.1.1 Problem Formalisation . . . 14
  - 4.1.2 Pick-and-Place Pipeline . . . 17
- 4.2 The Big Picture: Manipulation Trajectories in C-space . . . 19
  - 4.2.1 From Intermediate Frames to Configuration Space . . . 19
  - 4.2.2 Movable Objects and Constraints . . . 19
  - 4.2.3 Manipulation in TC-space . . . 20

5 Rational Forward and Inverse Kinematics **21**
- 5.1 Forward Kinematics . . . 22
  - 5.1.1 Rational Forward Kinematics . . . 26
- 5.2 Inverse Kinematics . . . 28
  - 5.2.1 "Rational" Inverse Kinematics . . . 29
  - 5.2.2 Scalability to Higher DOF or 3D Robots . . . 29

6 Manipulation Constraints **30**

6.1 Constraints for Standard Geometries 30
6.1.1 Cubes 31
6.1.2 Prisms 32
6.1.3 Cylinders 34
6.1.4 Spheres 34

7 Simulations, Experiments and Discussion **36**
7.1 Visualisation of Grasping Constraints in C-space 36
7.2 Growing Polygons with C_IRIS 37

8 Conclusion **39**
8.1 Future Work and MInf 2 39
8.1.1 MInf 2 40

Bibliography **41**

A Requirements for Non-Collision Certificates **45**
A.1 Archimedean Property 45
A.2 Polytopes and Convexity 46

B Joint Decomposition and Algebraic Kinematics **47**

C Spatial Algebra **50**
C.1 Arithmetic Rules 50
C.2 Representing Rotations 51

D Computation of Gripper Frames **52**
D.1 Definition of Pseudoinverse 52
D.2 Drake Code Example 52

E Robot Kinematics **54**
E.1 Stereographic Projection 54
E.2 Derivation of IK for 3R Planar Robot 55

Chapter 1

Introduction

1.1 Motivation

Imagine a simple task, such as reaching a door handle or picking up your toothbrush. Now, think about every step that your arm must take to achieve it: every time you move a finger, twist your elbow, wrist, or shoulder; every direction or speed choice; how you decide where to grasp the handle, or how you avoid touching the glass in which your toothbrush is. Motion planning has been a fundamental challenge in robotics over the last 50 years, and it presents itself as a question: How can a robot determine the sequence of movements required to achieve a task?

In the mid-1980s, Lozano-Perez et al. (1987) described a robot system "Handey" already capable of locating, grasping and displacing objects. The authors established then a fundamental challenge that the field of robotics still struggles with today: dealing with the number of options available while constructing a plan. This translates into handling complex, high-dimensional tasks. We are interested in searching for more efficient and generalisable approaches adaptable to diverse, sophisticated robots, as the versatility of planning directly impacts the extent of robotics applications. This ranges from household chores to industrial automation, medical surgery, prosthetics, or space exploration.

The concept of configuration space (C-space) has become fundamental to formalising motion planning, introducing a mathematical space where each point represents one possible robot posture. Since its proposal (Lozano-Perez, 1983), this space has been broadly used to describe trajectories. Previous approaches have tried to tackle the complexity issue by constraining the tasks, dimensionality or obstacles, introducing limiting assumptions that do not apply to realistic, complex scenarios. Alternatively, other methods can achieve less constrained tasks via sampling, although sacrificing guarantees of non-collision due to their probabilistic nature.

Dai et al. (2023) proposed an alternative, parametrised version of the C-space to describe collision-free motion. As a result, this approach allowed assumptions to be made only on the convexity of obstacles, while providing guarantees of non-collision and working relatively fast in arbitrary dimensions. However, these advancements have not yet been

explored in manipulation applications, which is the goal of this dissertation.

1.2 Project Aims and Limitations

This dissertation aims to reimplement and understand the advancements of the current state-of-the-art in motion planning (Dai et al., 2023) while exploring the set-up of Dai's contributions extended to manipulation planning applications. These are concerned with describing grasping constraints in a bijective parametrisation of the C-space, the tangent configuration space, that Dai introduced.

We give a walkthrough on how to formalise manipulation planning tasks, setting a foundation for adapting current approaches to using the tangent configuration space. The main goal is to provide a baseline for future research to develop algorithms capable of generating optimal trajectories for manipulation tasks in TC-space (see Section 8.1).

In terms of limitations, this work describes motion and manipulation without being concerned with the proper physics and dynamics implications, focusing on planning. Furthermore, similar to ongoing research, we provide examples restricted to one manipulation object. As mentioned above, we are interested in replicating the state-of-the-art in motion planning and providing a baseline for future manipulation applications.

1.3 Achievements and Contributions

For the purpose of this report, we have achieved the following goals and contributions:

- Gathered a deep understanding of the mathematical formulations of motion and manipulation planning problems, their limitations, and the current state-of-the-art.
- Delved into the advancements proposed by Dai et al. (2023) on motion planning, understanding the formulation of the tangent configuration space as a rational parametrisation of the configuration space and its implications.
- Proved the scalability of Dai et al. (2023)'s contributions to robots with universal joints, an extent that was not considered in the original paper.
- Implemented experiments carried by Dai et al. (2023) in the most recent version of the open-source robotics toolbox Drake (Tedrake and the Drake Development Team, 2019).
- Described the forward and inverse kinematics of a robot in the context of the TC-space.
- Derived constraints for grasping standard geometries such as cubes, prisms, cylinders, and spheres, offering visualisations in Drake.
- Ran experiments on Drake providing quantitative analysis on the results proposed by Dai et al. (2023).

The code implementation for these contributions is provided in this GitHub repository*.

*https://github.com/julialopezgomez/MInf1—Dissertation.git

1.4 Report Structure

The content of each chapter has been distributed as follows:

Chapter 2 discusses previous relevant research and introduces fundamental concepts in robotics to understand the planning problem formalisation, as well as the implications of this work. We also introduce the Drake software and its relevance in the context of manipulation planning and this work.

Chapter 3 explains how to formalise the description of robots, their decomposition, and mathematical representation. We introduce the monogram notation, how to describe a robotic system in terms of its joints and links, and the assumptions made about the robot in the context of this work. Additionally, we introduce our first contribution: proof of the feasibility of a robotic system with universal joints in the problem statement via the decomposition of this joint type into revolute joints.

Chapter 4 formulates a basic pick-and-place problem and introduces the pipeline of a simple manipulation task, using the notation described in the previous chapter. We also introduce the representation of a manipulation problem in C-space, and discuss its translation to TC-space.

Chapter 5 describes a robot's forward and inverse kinematics parametrised in TC-space. These result in rational functions and are thus called rational forward and inverse kinematics.

Chapter 6 presents the formulation of grasping constraints generalised for common geometries, such as cubes, prisms, cylinders, and spheres, using the widely used robotic manipulator Schunk WSG as an example.

Chapter 7 develops experiments to visualise the implications of the TC-space in planning applications.

Chapter 8 draws conclusions from the previous sections, summarises the project's implications and contributions, and discusses the next steps in continuing this work.

Chapter 2

Background

2.1 Definitions

Motion and manipulation planning are the fields in robotics that deal with the challenges described in Section 1.1, and thus the areas of focus of this dissertation. Motion planning encompasses the challenge of describing how a robot can move from one location to another in a given workspace, while manipulation planning extends this challenge to the grasping and displacement of objects. Both problems entail avoiding collisions with obstacles.

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

- The **degrees of freedom (DOF)** 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 2.1a, the pendulum has two DOF because its position is defined by θ_1 and θ_2.
- 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 2.1b.
- 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 2.1, the C-space is 2-dimensional because the pendulum has two DOF. Each point in the C-space in 2.1c represents a different value for θ_1 and θ_2, and thus, a different configuration.

  We denominate **collision-free configuration space** (**C-free**) to the space of all configurations in which a robot is not in collision.
- 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 2.1a, 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.



Figure 2.1: 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 $[0, 2\pi)$.

With these concepts, we can visualise a motion planning problem in Figure 2.2.

A more extensive description of these concepts can be found in Lynch and Park (2017, Chapter 2).

2.1.1 Dealing with Complexity

The complexity of motion and manipulation planning relies on the number of degrees of freedom of the robot in question. The higher the number of DOF, the higher the dimension of the robot's C-space, and thus the more complex the planning problem becomes (Lynch and Park, 2017, Chapter 2).

Robots with many joints, such as robotic arms, can have dozens of DOF, making computational complexity a fundamental challenge and prompting many limitations amongst proposed solutions, as discussed in Section 1.1.

2.2 Previous Work

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 the collision-free configuration space (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.

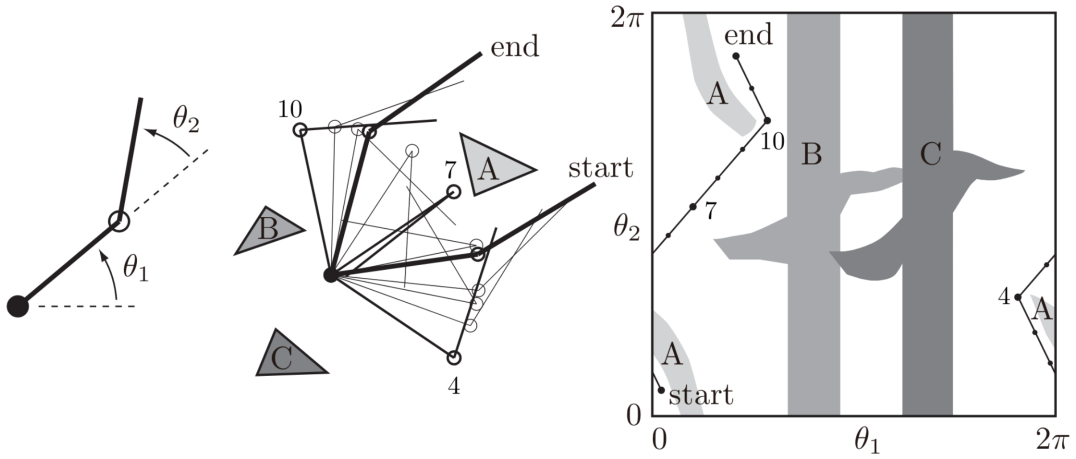


Figure 2.2: Adapted from Lynch and Park (2017). On the left, a double pendulum with two DOF θ_1 and θ_2. In the centred picture, the pendulum taking its "end-effector" (extreme) from the `start` to the `end` position in task space, while avoiding obstacles A, B and C. On the right, the C-space of the double pendulum, with θ_1 in the horizontal axis and θ_2 in the vertical one. C-space obstacles are the greyed areas A, B and C, and the white background represents C-free. The motion from `start` to `end`, with the intermediate configurations, is represented as a path in C-free.

2.2.1.1 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, Kavraki (1995) only allowed robots to translate and not rotate in the task space, while Branicky and Newman (1990)'s research was restricted to two and three DOF robots. Hubbard (1996) attempted to describe obstacles as sets of spheres, but it did not consistently provide an acceptable level of accuracy. A review conducted in 2012 by Latombe (2012) revealed that task space obstacles described with semi-algebraic sets could be transferred to semi-algebraic obstacles in the C-space. Despite being a significant breakthrough, dealing with complex geometries and computational complexity remained a challenge.

2.2.1.2 The Positive Approach

The positive approach breaks down C-free into smaller, more manageable sets, for which efficient algorithms already exist. This is especially true when the simpler sets are convex, including the algorithms developed by Schouwenaars et al. (2001), Deits and Tedrake (2015b) and Marcucci et al. (2022). A variety of other methods have been utilised for the positive approach, including Rapidly-exploring Random Trees (RRT) (LaValle, 1998), Probabilistic Roadmaps (PRM) (Kavraki et al., 1996), and their variations. However, due to their stochastic nature, these methods only offer probabilistic guarantees of non-collision, as they determine non-collision by sampling along the paths. Schwartz and Sharir (1983) and Lozano-Perez (1983) attempted a method called the "cell decomposition" planner approach, which involves dividing the configuration

space into discrete, manageable cells to simplify path planning. However, this method proved to be computationally infeasible, even for modestly complex scenarios.

Describing C-free rigorously when C-space obstacles are convex is possible but challenging. It has been proven that finding a minimal decomposition is NP-hard (Lingas, 1982) and approximating it is APX-hard* (Eidenbenz and Widmayer, 2003). Works such as Lien and Amato (2007) and Ghosh et al. (2013) overcome these hardness results by finding decompositions that are unions of approximately convex sets.

The IRIS algorithm (Deits and Tedrake, 2015a) accomplished to decompose C-free into convex polytopes† while being relatively fast and working in arbitrary dimensions. However, the assumption of convex obstacles in C-space presented the challenge that convex obstacles in task space are rarely convex in C-space (see Figure 2.2. Obstacles *A*, *B*, and *C* are convex in task space - centred image -, but not in C-space - rightmost image).

To overcome the challenge of dealing with convex C-space obstacles, Dai et al. (2023) proposed the C_IRIS algorithm, compounding the current state-of-the-art in motion planning. This work aims to describe C-free in a rational, bijective parametrisation of the C-space‡, known as the tangent configuration space (TC-space). Contrary to the C-space, the TC-space guarantees that convex objects in task space are convex in TC-space too.

C_IRIS is the first known method to rigorously approximate the decomposition of the collision-free TC-space into certified convex polyhedra. Altogether, Dai's contributions allow relaxing the assumptions made by previous rigorous methods on the convexity of obstacles in C-space, thus prompting the motion planning field with a baseline to compute trajectories *only* making assumptions in the convexity of obstacles in task space.

2.2.2 Manipulation Planning

In manipulation planning, research distinguishes between movable objects and obstacles. Siméon et al. (2004b) defined **movable objects** as those that can only move when grasped by a robot. Alternatively, **obstacles** refer to static or stationary bodies in the task space, meant to be avoided.

Over three decades ago, Alami et al. (1990) presented manipulation planning as a constrained version of a coordinated motion planning problem for the first time. This approach led to a more general and computationally complex instance of the classical problem defined by Lozano-Perez (1983), and has been explored by numerous works (Siméon et al., 2002, 2004a,b; Latombe, 2012) ever since.

*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 $P = NP$.

†A polytope is a generalisation of polygons and polyhedra to higher dimensions.

‡This parametrisation is achieved by using the stereographic projection substitution: $t = \tan\frac{\theta}{2}$ (Spivak, 1994). In the context of this work, θ corresponds to the joint angles of a robot system.

Alami presented the solution to a manipulation problem as a sequence of subpaths that satisfy the restrictions of grasping and ungrasping actions. These are categorised into:

- **Transit paths** - when the robot moves alone, or equivalently, the movable objects are placed at rest.
- **Transfer paths** - when the robot holds the object.

In this context, the biggest challenge in manipulation planning is automating the decomposition of motion into elementary transfer and transit paths. Previous works (Alami et al., 1990; Barraquand and Ferbach, 1994; Koga and Latombe, 1994; Ahuactzin et al., 1998; Nielsen and Kavraki, 2000) assumed that the grasping and placement configurations were given in the problem description in the form of discrete sets. This did not only rely a significant part of the task in the user, but limited its extent.

During the early 2000s, a group of researchers - Simeon, Cortés, Sahbani, and Laumond - approached this challenge in a series of four papers (Siméon et al., 2002; Sahbani et al., 2002; Siméon et al., 2004a,b), each one building upon the advancements of the previous. These works presented an approach to manipulation planning by representing grasping and placement constraints as subspaces of C-free, connected through transfer and transit paths. This approach broadened the scope of the problem by allowing the computation of continuous configurations directly in this decomposition of C-free. The last paper in the series (Siméon et al., 2004b) depended on the task by exploring the use of probabilistic roadmaps to refine the previous results.

While the advancements presented in these papers have laid a solid foundation for modern manipulation planning, they still present limitations in scalability to robots with a higher number of DOF, and more complex manipulation scenarios.

Several works have conducted further research in manipulation planning. For instance, Diankov et al. (2008) proposed a method of caging grasps that ensured object security without the need for precise control. Berenson (2011) delved into the definition of constraints for robust manipulation strategies, while other researchers such as Saut et al. (2007) or Haustein (2020) explored manipulation tasks with fingertip motions.

What these methods have in common is that they aim to represent grasping and placement constraints in the C-space or task space, but none has yet attempted this task in the tangent configuration space.

2.3 Drake

The experiments and implementation presented in this dissertation are run using the open-source robotics toolbox Drake (Tedrake and the Drake Development Team, 2019). Drake is a C++ library with Python bindings (pydrake library). It started to be developed in 2016 to provide an interface to simulate complex manipulation applications capable of generalising to real-world dynamics (Sherman, 2022). In this work, we make use of the Python bindings, with functional code in any Drake stable or nightly release version from February 2024 to the date this report was published.

Released in 2019, Drake is still under development, with regular updates, deprecations and new features in line with the state-of-the-art advancements. Although this presents challenges such as adapting to new releases and navigating still limited online resources, we have chosen this software because it is growing widely in the robotics community, and it is unique to providing mature implementations of state-of-the-art research and a simulation framework for manipulation tasks accurately scalable to real-world problems.

Chapter 3

Assumptions and Problem Statement

This dissertation assumes the same environment description as the one presented in (Dai et al., 2023, Section 2), except for considering systems with universal joints. Such environment is described in this chapter.

We are given a known task-space environment in which our robot, obstacles and movable objects have been decomposed as a union of compact, convex bodies*. Robots can be decomposed into links interconnected via joints (Figure 3.1), and we consider robotic mechanisms with N joints and $N+1$ links (including the ground).

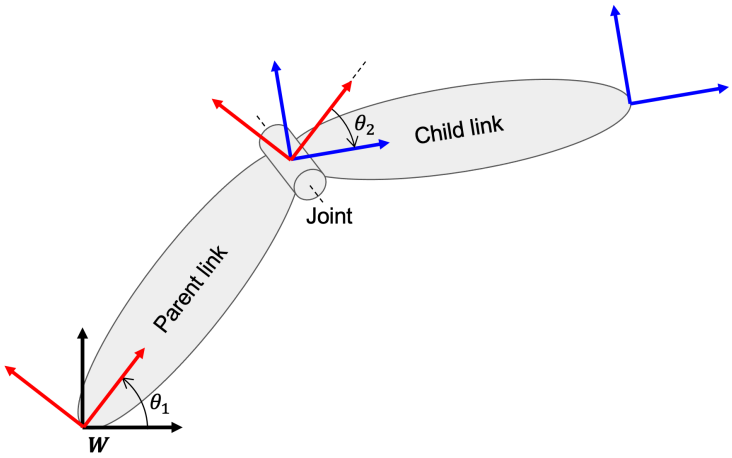


Figure 3.1: A mechanism decomposed into two links (parent and child) + ground, connected by two revolute joints θ_1 and θ_2. The world frame W is represented in black, reference frames related to the parent link are in red, and frames related to the child are in blue. θ_1 represents the rotation of the parent link w.r.t. W, and θ_2 is the joint angle between the parent and the child links.

*These requirements are selected because of their role in decomposing C-free into polytopic regions with non-collision certificates. See Appendix A for details.

3.1 Joint Requirements

The methods discussed in this work can handle robots with revolute, prismatic, planar, cylindrical, spherical and universal joints. Throughout this report, we limit ourselves to the use of revolute and prismatic joints, as the rest can be decomposed into these two (Wampler and Sommese, 2011). In Appendix B, we include the derivation of these decompositions. The decomposition of the universal joint is provided in Section 3.3, as an expansion to the problem statement considered originally by Dai et al. (2023).

- **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.

θ and z are the C-space variables associated with the revolute and prismatic joints, respectively. We assume that θ is constrained to a complete rotation

$$-\pi < \theta_l \leq \theta \leq \theta_r < \pi, \tag{3.1}$$

and z is such that the displacement is finite:

$$z_l \leq z \leq z_r \tag{3.2}$$

where $\theta_l, \theta_r, z_l, z_r \in \mathbb{R}$ are fixed bounds for each individual joint.

We define our configuration space variable as the $\mathbb{R}^N$ vector

$$q = \bigcup_i \{\theta_i, z_i\} \quad \text{for } i \in \{1, ..., N\} \tag{3.3}$$

3.2 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:

${}^{B}p_{C}^{A}$	${}^{B}R^{A}$	${}^{B}X^{A}$
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

(3.4)

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.

[‡]There are various methods to represent rotations, the most common in robotics being rotation matrices, quaternions, Euler angles, or axis angles. For explicit definitions, see Appendix C.2.

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

A thorough description of this notation can be found in Tedrake (2023, 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:

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

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

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

A brief description of elementary Spatial Algebra operations represented with this notation is given in Appendix C.1.

3.3 Decomposition of Universal Joints

This subsection constitutes our first contribution. Dai et al. (2023) stated that their method was compatible with robotic mechanisms featuring revolute (R), prismatic, planar, cylindrical, and spherical joints. We have identified an additional known algebraic joint, the universal joint (U), that can be deconstructed into two R joints and, thus, be incorporated into Dai et al. (2023)'s approach. The derivation follows:

> The **universal joint (U)** is a 2-DOF joint that rotates about two orthogonal axes. That is, composed of two revolute joints (Lynch and Park, 2017, p. 16).
>
> Let ${}^{P_i}X^{C_i}(q_i)$ be a transform that describes the relative motion between a child C and parent P links of a robotic mechanism, constraint by the i^{th} joint q_i in the system.
>
> When q_i is U, ${}^{P_i}X^{C_i}(q_i)$ takes the form
>
> $$ {}^{P_i}X^{C_i}(q_i) = \begin{bmatrix} M(\psi_i) & 0_{3\times 1} \\ 0_{1\times 3} & 0 \end{bmatrix} $$
>
> where M belongs to the $SO(3)$[‡] and is parametrised with Euler angles such that

[‡]The $SO(3)$ group, also known as the special orthogonal group, is the group of all rotation matrices. A full definition can be found in (Lynch and Park, 2017, Chapter 3).

$$M(\psi_i) = \underbrace{\begin{bmatrix} \cos(\psi_{i,x}) & -\sin(\psi_{i,x}) & 0 \\ \sin(\psi_{i,x}) & \cos(\psi_{i,x}) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_x(\psi_{i,x})} \cdot \underbrace{\begin{bmatrix} \cos(\psi_{i,y}) & 0 & -\sin(\psi_{i,y}) \\ 0 & 1 & 0 \\ \sin(\psi_{i,y}) & 0 & \cos(\psi_{i,y}) \end{bmatrix}}_{R_y(\psi_{i,y})} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\psi_{i,z}) & -\sin(\psi_{i,z}) \\ 0 & \sin(\psi_{i,z}) & \cos(\psi_{i,z}) \end{bmatrix}}_{R_z(\psi_{i,z})}$$

where one of the rotations $R_x(\psi_{i,x}), R_y(\psi_{i,y})$ and $R_x(\psi_{i,z})$ is the identity matrix I^3 - that is, respectively, where one of the Euler angles $\psi_{i,x}, \psi_{i,y}$ or $\psi_{i,z}$ is 0. This ensures that U rotates along two axes, orthogonal to each other.

More specifically, M can take one of the following three forms:

$$M^{\backslash x}(\psi_i) = R_y(\psi_{i,y}) \cdot R_z(\psi_{i,z})$$
$$M^{\backslash y}(\psi_i) = R_x(\psi_{i,x}) \cdot R_z(\psi_{i,z})$$
$$M^{\backslash z}(\psi_i) = R_x(\psi_{i,x}) \cdot R_y(\psi_{i,y})$$

Thus, U is the composition of two R joints expressed as Euler angles.

Hence, we have shown the scalability of the C_IRIS algorithm, the rest of Dai et al. (2023)'s contributions, and this work to a wider extent of robotic manipulators, including the universal joint.

Wampler and Sommese (2011) provides a detailed explanation of algebraic kinematics; however, it does not touch on universal joints. In Appendix B, we include the decomposition of the remaining algebraic joints handled by Dai's work.

Chapter 4

Manipulation Planning Pipeline

In this chapter, we break down the manipulation planning problem. We formalise it using a simple pick-and-place example (inspired from Tedrake (2023, Chapter 3)), and introduce the setup of the problem in the context of the tangent configuration space.

4.1 Basic Pick-and-Place

Consider a simple pick-and-place problem. We have a robotic arm with a gripper (end-effector), and we want to pick an object, place it in a different location, and finally retract our gripper.

4.1.1 Problem Formalisation

We formalise the problem as follows. We are given

$$\underbrace{{}^{W}X^{O_{\text{initial}}}}_{\text{Initial object pose}} \qquad \underbrace{{}^{W}X^{G}}_{\text{Initial gripper pose}} \qquad \underbrace{{}^{W}X^{O_{\text{goal}}}}_{\text{Goal object pose}} \tag{4.1}$$

These are the initial position and orientation (pose) of the object and gripper, and the goal pose where we want to place the object with respect to the world frame W.

Our objective is to take the object O to the goal location. That is, to make:

$$X^{O} = X^{O_{\text{goal}}}$$

4.1.1.1 Assumptions

To take O to the goal location, we make the following assumptions:

1. When the *gripper is open*, the object does not move *relative to the world*:

$$ {}^{W}X^{O} \equiv X^{O} \quad \text{is constant.} \tag{4.2}$$

2. When the *gripper is closed*, the object does not move *relative to the gripper*:

$$ {}^{G}X^{O} \quad \text{is constant.} \tag{4.3}$$

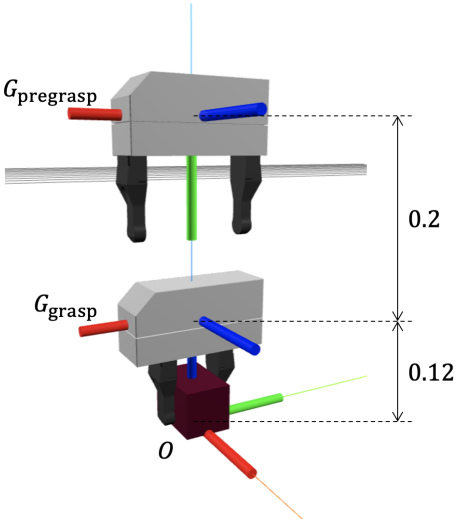


Figure 4.1: Simulation in Drake. The gripper above is at pregrasping pose ($X^{G_{\text{pregrasp}}}$), and the one below at grasping pose ($X^{G_{\text{grasp}}}$), touching the object. The frame of each body is represented, with the x axis in red, the y axis in green, and the z axis in blue. The distance of the frame origins from G_{pregrasp} to G_{grasp} is 0.2, and from G_{grasp} to O, 0.12. Notice the y axis of G_{grasp} is hidden as it overlaps with the z axis of O.

4.1.1.2 Grasping and Pregrasping

Pick-and-place problems require the setup of special grasping and placement frames. These frames define the gripper's pose when picking up or dropping the movable object.

Primarily, we need to consider:

- G_{grasp}: The frame associated with the gripper at the moment of touching the object for grasping or placement.
- G_{pregrasp}: The frame associated with the gripper at the step previous to picking the object. This step is defined to prevent the gripper from colliding with the object when approaching it.

These frames can be visualised in Figure 4.1. Generally, we are interested in defining:

- The pose of the object frame O relative to the grasping gripper:

$$^{G_{\text{grasp}}}X^{O} \tag{4.4}$$

- The pose of the grasping gripper relative to the pregrasping gripper:

$$^{G_{\text{pregrasp}}}X^{G_{\text{grasp}}} \tag{4.5}$$

Example 1 *Notice how, in Figure 4.1, the object frame O is slightly ahead the y axis of G_{grasp}. Likewise, G_{grasp} is ahead the y axis of $G_{pregrasp}$. Considering the distances provided in the figure, we can define the relative positions* by assigning these distances to the y component of the displacement p:*

$$^{G_{grasp}}p^{O} = [0, 0.12, 0]^T, \quad ^{G_{pregrasp}}p^{G_{grasp}} = [0, 0.2, 0]^T$$

*The position/displacement in monogram notation is in the form $^{B}p^{A} = [x, y, z]^T$.

In this example, we have only dealt with displacement, not rotation or pose. We provide a full example of how to compute the poses ${}^{G_{\text{grasp}}}X^{O}$ and ${}^{G_{\text{pregrasp}}}X^{G_{\text{grasp}}}$ in Section 4.1.2.

Remark 1 *Gripper grasping and pregrasping frames are used in most generic cases; however, additional auxiliary frames can be predefined depending on the task. For instance, common choices are $G_{postgrasp}$, if different from pregrasp, or $G_{clearance}$, if a further step is necessary to avoid collisions.*

4.1.1.3 Simplified Pick-and-Place Pipeline

The grasping and pregrasping frames, together with the problem formalisation and assumptions aforementioned, allow us to define a simplified pipeline of the manipulation problem:

1. The gripper is open (Assumption 4.2 holds).
2. Move the gripper in the world to the pregrasp pose measured from the object:

$$X^{G} \rightarrow {}^{O}X^{G_{\text{pregrasp}}}$$

3. Do the same to the appropriate grasp pose:

$$X^{G} \rightarrow {}^{O}X^{G_{\text{grasp}}}$$

4. Close the gripper (Assumption 4.3 holds).
5. Move gripper + object to the goal pose:

$$X^{O} \rightarrow X^{O_{\text{goal}}}$$

   The gripper moves with the object because of Assumption 4.3
6. Open the gripper.
7. Retract hand.

While this summarises the steps of a simple pick-and-place problem, there are still intermediate poses that the problem needs to take into account. A thorough description of the manipulation pipeline and the required computations is given in Section 4.1.2

4.1.1.4 Computed Special Frames

In Section 4.1.1.2, we mentioned about the G_{grasp} and G_{pregrasp} frames that:

1. We were interested in representing their pose with respect to the movable object.
2. The pose of these frames with respect to the object needed to be predefined in the problem description.

G_{grasp} and G_{pregrasp} are abstract: being defined relative to the object, they can take different poses in the world frame, depending on the location of the object.

For this purpose, we use other special frames to define the exact poses of the gripper *in the world frame*, and we compute them using G_{grasp} and G_{pregrasp}[†].

When we are picking the movable object, we refer to these frames as G_{pick}, G_{prepick} or G_{postpick}. Likewise, when we are dropping the object, we call the frames G_{place}, G_{preplace} or $G_{\text{postplace}}$. Again, we are interested in defining the location of these frames with respect to the world frame:

$$\begin{array}{cccc} G_{\text{pick}} & G_{\text{prepick}} & G_{\text{postpick}} & \Leftrightarrow \quad X^{G_{\text{pick}}} \quad X^{G_{\text{prepick}}} \quad X^{G_{\text{postpick}}} \\ G_{\text{place}} & G_{\text{preplace}} & G_{\text{postplace}} & \Leftrightarrow \quad X^{G_{\text{place}}} \quad X^{G_{\text{preplace}}} \quad X^{G_{\text{postplace}}} \end{array}$$

Remark 2 *These specific denominations and auxiliary frames are optional, and their definition is up to the user and the task at hand.*

For instance, generally, we are interested in *pick*, *place*, *prepick* and *postplace* frames. This group handles the poses where the gripper approaches and retracts from the object; hence, where we are concerned about collision with the object itself.

4.1.2 Pick-and-Place Pipeline

In this subsection, we describe how to compute relevant intermediate poses of the gripper (with respect to the world) in a basic pick-and-place task, to develop an understanding of the mathematical symbols and concepts and how they can be applied in practical scenarios.

We consider the scene from Figure 4.1, and want to return the *pick, place, prepick* and *postplace* poses of the gripper.

As stated in Section 4.1.1, ${}^{W}X^{O_{\text{initial}}}$, ${}^{W}X^{G}$ and ${}^{W}X^{O_{\text{goal}}}$ are given in the problem description, and we can assume are known.

4.1.2.1 Determining Grasping and Pregrasping Poses

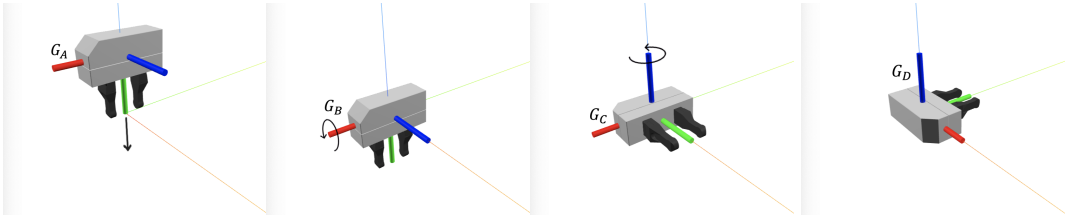
To compute the *pick*, *place*, *prepick* and *postplace* poses, we use the corresponding grasp and pregrasp poses with respect to the object: ${}^{G_{\text{grasp}}}X^{O}$ and ${}^{G_{\text{pregrasp}}}X^{G_{\text{grasp}}}$.

Recall from Section 3.2 that the pose X is the position p and orientation R of a frame relative to another. Since we are taking the scenario from Figure 4.1, we can infere ${}^{G_{\text{grasp}}}p^{O}$, ${}^{G_{\text{grasp}}}R^{O}$, ${}^{G_{\text{pregrasp}}}p^{G_{\text{grasp}}}$ and ${}^{G_{\text{pregrasp}}}R^{G_{\text{grasp}}}$ from the information in the figure:

- From Example 1, we determined that:

$$ {}^{G_{\text{grasp}}}p^{O} = \begin{bmatrix} 0 \\ 0.12 \\ 0 \end{bmatrix} \quad \text{and} \quad {}^{G_{\text{pregrasp}}}p^{G_{\text{grasp}}} = \begin{bmatrix} 0 \\ 0.2 \\ 0 \end{bmatrix} $$

[†]This also includes other predefined frames as mentioned in Remark 1.



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

Figure 4.2: Simulation in Drake. The transforms of the gripper frame necessary to locate it in the world frame W. From left to right, we refer to the gripper frames in each of the four figures as G_A, G_B, G_C and G_D. (a) G_A is above W, in the direction of G_A's y axis. (b)(c) G_B, G_C and W share origin but have distinct orientations. (d) $G_D = W$.

- In Figure 4.1, we can observe that frames G_{grasp} and G_{pregrasp} have the same orientation. We can represent this using the 3×3 identity rotation matrix:

$$^{G_{\text{pregrasp}}}R^{G_{\text{grasp}}} = I^3$$

- For $^{G_{\text{grasp}}}R^{O}$, notice that visualisations (b), (c) and (d) in Figure 4.2 represent the rotations necessary to take the grasping gripper G_{grasp} in Figure 4.1 to the object frame O. We observe that we need two rotations: 90° rotation along the x axis[‡], and 90° rotation along the z axis:

$$^{G_{\text{grasp}}}R^{O} = R_x(90^\circ) \cdot R_z(90^\circ),$$

where $R_w(\theta)$ represents a rotation of θ degrees around axis w.

4.1.2.2 Computation of the Gripper Poses

With the results obtained in the previous section, we proceed as follows using the pertinent rules of spatial algebra (Appendix C.1):

1. $^{G_{\text{grasp}}}X^{O} = \left[\begin{array}{c|c} ^{G_{\text{grasp}}}R^{O} & ^{G_{\text{grasp}}}p^{O} \\ \hline 0_{1\times 3} & 1 \end{array}\right]$
2. $^{O}X^{G_{\text{grasp}}} = \left[^{G_{\text{grasp}}}X^{O}\right]^{+}$§
3. $^{G_{\text{pregrasp}}}X^{G_{\text{grasp}}} = \left[\begin{array}{c|c} I^3 & ^{G_{\text{pregrasp}}}p^{G_{\text{grasp}}} \\ \hline 0_{1\times 3} & 1 \end{array}\right]$
4. $^{G_{\text{grasp}}}X^{G_{\text{pregrasp}}} = \left[^{G_{\text{pregrasp}}}X^{G_{\text{grasp}}}\right]^{+}$
5. $^{W}X^{G_{\text{pick}}} = {}^{W}X^{O_{\text{initial}}} \,.\, {}^{O}X^{G_{\text{grasp}}}$
6. $^{W}X^{G_{\text{prepick}}} = {}^{W}X^{G_{\text{pick}}} \,.\, {}^{G_{\text{grasp}}}X^{G_{\text{pregrasp}}}$
7. $^{W}X^{G_{\text{place}}} = {}^{W}X^{O_{\text{goal}}} \,.\, {}^{O}X^{G_{\text{grasp}}}$

[‡]The direction of rotation is clockwise, following the well-known right-hand rule (Wikipedia, 2022).

[§]Moore–Penrose Pseudoinverse: the inverse generalisation for non-square matrices. The full definition is given in Appendix D.1. X is usually squared, represented in homogeneous coordinates.

8. ${}^{W}X^{G_{\text{postplace}}} = {}^{W}X^{G_{\text{goal}}} \cdot {}^{G_{\text{grasp}}}X^{G_{\text{pregrasp}}}$

Lines $5, 6, 7$ and 8 compound the solution to our basic pick-and-place problem. Pseudocode showcasing the implementation of this computation in Drake's notation is given in Appendix D.2.

Remark 3 *We showed how to compute pertinent intermediate frames. Advanced software toolkits like Drake provide tools for computing pose transitions between such frames using techniques like SLERP (spherical linear interpolation).*

4.2 The Big Picture: Manipulation Trajectories in C-space

4.2.1 From Intermediate Frames to Configuration Space

In the previous section, we showed how to describe *steps* in the displacement of a robot attempting a manipulation task. But how do we plan for entire manipulation trajectories?

In motion planning, trajectories are computed by finding optimal paths in C-free from the initial to the goal location, navigating through robot postures guaranteed to be collision-free. In manipulation planning, the problem becomes slightly more complex. We not only seek to avoid obstacles but need to handle and displace objects. How should we plan a path in C-space for a robot to achieve a manipulation?

We have already defined intermediate gripper positions in task space to accomplish a manipulation task. However, what does the gripper's position tell us about the robot's exact configuration? Inverse kinematics is the field that determines the different ways in which a robot can define its angles to reach a specific position.

Inverse kinematics allows us to define in C-space which configurations achieve specific manipulation poses stages like prepick, pick, place, or postplace. Given multiple grasping options, we can visualize valid grasping poses as regions in C-space. This approach expresses manipulation planning as a series of subpaths in C-free: moving the robot to a "picking" or "prepicking" region, grabbing the object, and then transitioning to a "placing" region in C-free.

4.2.2 Movable Objects and Constraints

A question emerges: how do we represent in C-space that the robot is moving the object? Siméon et al. (2002) approached this problem by representing the configuration space of the robot and the movable object in conjunction:

$$\text{C-space} = \text{C-space}_{robot} \times \text{C-space}_{object} \tag{4.6}$$

This way, each point in C-space described the location of the robot and object simultaneously. To represent manipulations in this C-space, Siméon et al. (2002) proposed the the description of two subspaces:

- C-space$_{grasp}$, representing the configurations in which the robot is compliantly grasping the object - at a valid grasping pose -, and
- C-space$_{placement}$, representing the configurations in which the robot laid at rest without the need of a robot holding it.

These subspaces of C-space can be visualised in Figure 4.3.

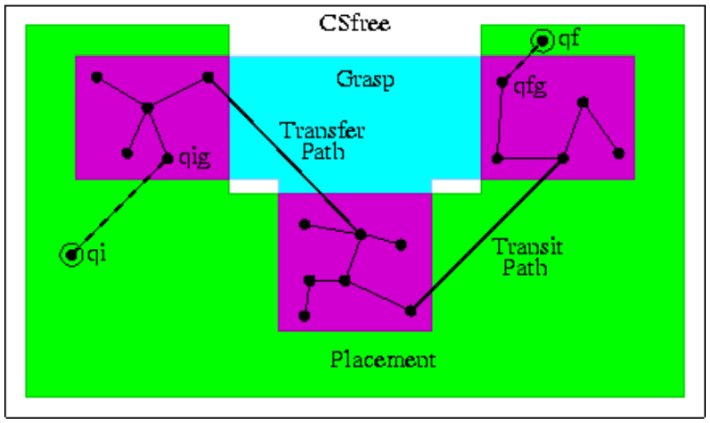


Figure 4.3: Adapted from Siméon et al. (2004a). A manipulation task in C-free. The green region represents C-space$_{placement}$, the blue region C-space$_{grasp}$, and the purple region is the intersection of the two. A path is represented from the initial to the final configurations, decomposed in transfer and transit.

Recall from Section 2.2.2 that manipulation trajectories can be decomposed into transfer and transit paths, where in transfer paths, the robot carries the movable object, and in transit paths, the object is at rest. With Siméon et al. (2002)'s approach, transfer paths always lie in C-space$_{grasp}$, while transit paths in C-space$_{placement}$, where the object is in rest. The intersection of C-space$_{grasp}$ and C-space$_{placement}$ represents the configurations where the object could be picked or dropped.

Remark 4 *To describe the subspaces described by Siméon et al. (2002), it is crucial to effectively represent the grasping constraints: that is, the constraints that define the regions where the robot can grasp the object. Chapter 6 exemplarises how to define constraints for standard geometries.*

4.2.3 Manipulation in TC-space

This study establishes a foundation for computing manipulation trajectories, similar to the methods used by Siméon et al. (2002), but representing them in a bijective parametrisation of the configuration spaces described (TC-space). Representing manipulation within the TC-space involves transcribing the constraints and paths of manipulation tasks into a bijective parametrization of the configuration space (C-space) which has been found to be beneficial in motion planning applications (Dai et al., 2023). In TC-space, each point no longer represents a simple configuration but rather a one-to-one mapping to other parameters. The bijective property ensures that every point in C-space has one exact mapping in TC-space, thus allowing us to search for trajectories equivalent to those in C-space.

Chapter 5

Rational Forward and Inverse Kinematics

In the field of robotics, having a solid understanding of forward and inverse kinematics is crucial when it comes to motion and manipulation planning. In the previous chapter, we explained how to identify suitable poses or transforms for an end-effector (the gripper) in the context of a manipulation task. But how do we associate gripper poses with robot configurations? Two questions arise:

1. If we have information about the joint values of our robot, can we determine the pose of its gripper?
2. Conversely, if we know the location of the gripper, can we find the joint values that achieve that position?

Forward and inverse kinematics are methods used to address these questions. Essentially, they enable us to map robot configurations to end-effector task-space locations and vice versa. We formalise these concepts as follows:

Definition 1 (Forward kinematics) *The **forward kinematics (FK)** of a robot refer to computing the position and orientation of the end-effector frame - relative to the world frame - from the robot joint coordinates q:*

$$X^G = f_{kin}^G(q) \tag{5.1}$$

where $q \in \mathbb{R}^N$ is the vector of joint positions of the robot and G is the end-effector frame. In Figure 5.1, the forward kinematics consist of determining the transform $X^{C_3} = {}^{P_1}X^{C_3}$ from the joint angles θ_1, θ_2 and θ_3.

Definition 2 (Inverse kinematics) *The **inverse kinematics (IK)** of a robot refer to calculating all possible sets of joint variables q that could be used to attain a given position and orientation of the end-effector:*

$$q = f_{kin}^{-1}(X^G) \tag{5.2}$$

where $q \in \mathbb{R}^N$ is the vector of joint positions of the robot and G is the end-effector frame. In Figure 5.1, the inverse kinematics consist of determining the joint angles θ_1, θ_2 and θ_3 knowing only the pose of the gripper X^{C_3}.

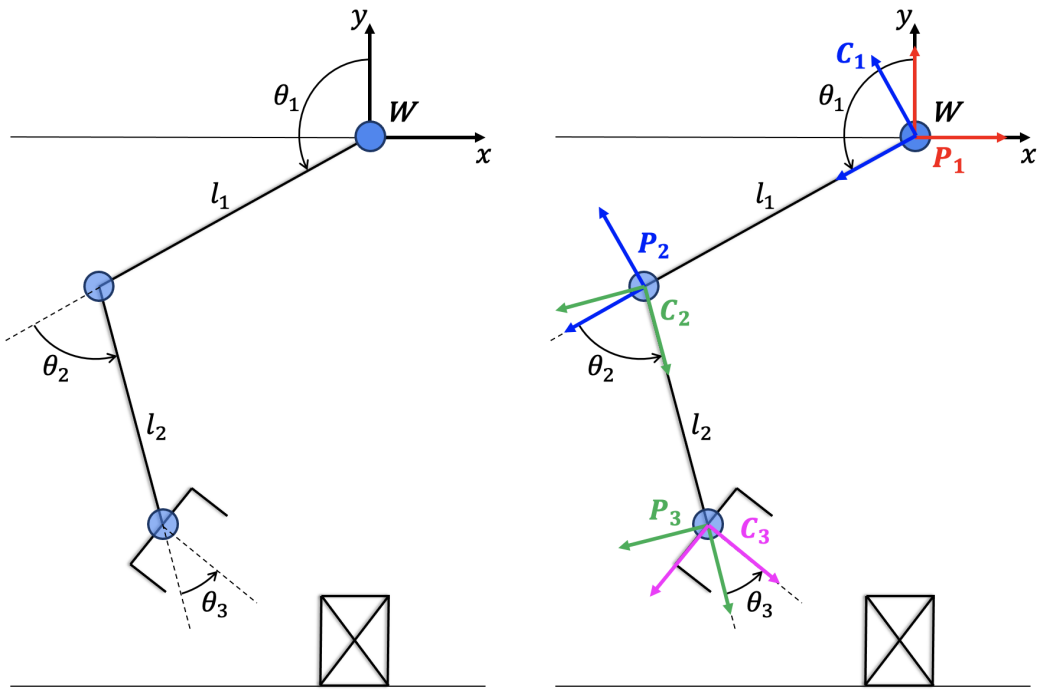


Figure 5.1: (a) 3R planar robotic manipulator with three links (l_1, l_2 and gripper) + ground, and three R joints θ_1, θ_2 and θ_3. The movable object is represented as a crossed box. (b) visualises the parent and child frames associated with each joint from (a). The world frame W aligns with P_1, and the end-effector frame is C_3.

Thorough definitions of these concepts can be found in most robotics textbooks, such as Craig (2005), Lynch and Park (2017) or Tedrake (2023).

In this chapter, we introduce the mathematical formalisation of the forward and inverse kinematics, describing their scalability to the tangent configuration space.

5.1 Forward Kinematics

A description of the forward kinematics parametrisation in TC-space is presented in Dai et al. (2023, Section 3.3). In this section, we delve into this description and provide relevant examples that will be useful in describing the parametrised version of the inverse kinematics problem.

In Definition 1, we established that forward kinematics aim to determine the transform* of our gripper X^G relative to the world frame W - or a reference frame F - in task space, derived from the joint variables q.

We can represent this as a composition of rigid transforms†, expressed in homogeneous

*In the context of robot kinematics, poses are usually referred to as transforms. Pose and transform are often used interchangeably in research.

†Type of transforms that occurs in Euclidean space and preserves the Euclidean distance between every pair of points (Bottema and Roth, 1990).

coordinates:

$$ {}^{F}X^{G} = \begin{bmatrix} {}^{F}R^{G}(q) & {}^{F}p^{G}(q) \\ 0_{1\times 3} & 1 \end{bmatrix} = \prod_{i\in[N]} {}^{P_i}X^{C_i}(q_i) \cdot {}^{C_i}X^{P_{i+1}} \tag{5.3} $$

where $q \in \mathbb{R}^N$ is the vector of joints of the corresponding robotic system or kinematic chain[‡] (from frames F to G), N is the number of joints and $[N] = \{1,\ldots,N\}$. Let's break this expression down:

- Firstly, we associate a parent P_i and a child C_i frame to each joint q_i. This can be observed in Figure 5.1b. Frame P_i is rigidly fixed to the parent link of joint q_i and, likewise, C_i is rigidly attached to the child link of q_i.
- ${}^{F}R^{G}(q)$ and ${}^{F}p^{G}(q)$ are the relative rotation and position, respectively, of frame G measured from F, when each joint variable in q moves by its corresponding q_i.
- Likewise, ${}^{P_i}X^{C_i}(q_i)$ is the motion of the child frame C_i of joint q_i, relative to the parent frame P_i of the same joint, when the joint moves by q_i. Intuitively, this expression describes how each joint moves.
- ${}^{C_i}X^{P_{i+1}}$ describes the motion from the child frame of a joint C_i to the parent frame of the next joint P_{i+1}. This usually represents some physical property of the i^{th} link of the robot, such as its length. For instance, notice in Figure 5.1b how the motion from frame C_1 to P_2 is just a displacement given the distance of the first link l_1.
- Expression 5.3 assumes that the reference frame F is the parent frame of the first joint P_1, and that the frame of the gripper is the child frame of the last joint C_N. Given that q_N is the last joint in the chain, we can assume that ${}^{C_N}X^{P_{N+1}}$ is the identity I[§].

Wampler and Sommese (2011, Chapter 4) gives a thorough introduction to algebraic kinematics. They stated that the transforms ${}^{P_i}X^{C_i}(q_i)$ can be written in the form:

$$ {}^{P_i}X^{C_i}(\theta_i) = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & 0 \\ \sin(\theta_i) & \cos(\theta_i) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{if the joint } q_i \text{ is Revolute } (\theta_i) \tag{5.4} $$

$$ {}^{P_i}X^{C_i}(z_i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{if the joint } q_i \text{ is Prismatic } (z_i) \tag{5.5} $$

[‡]A kinematic chain is a sequence of links connected by 1-DOF joints. Our robot (as per described in Section 3) and any subsystem of it are kinematic chains.

[§]Exceptions exist, for instance, when we need to account for the length of the last link.

Likewise, when the transforms ${}^{C_i}X^{P_{i+1}}$ represent the link length, they can be written as:

$$ {}^{C_i}X^{P_{i+1}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.6} $$

Remark 5 *Notice that the expressions 5.4, 5.5 and 5.6 are presented for motion in the 3D Cartesian space. For planar (2D) motions, the transforms would take the form of* 3×3 *matrices instead of* 4×4*. This can be visualised in the following example.*

Dai et al. (2023) provided an example derivation of the forward kinematics of a 2R planar robot (a double pendulum), to compute the position of the end-effector. As a contribution, we provide an example with a 3R planar robot manipulator, further describing the FK behaviour in rotation as well.

Example 2 *We formulate the FK of the robot in Figure 5.1 in the form of expression 5.3:*

$$ {}^{W}X^{G} = {}^{P_1}X^{C_1}(\theta_1) \cdot {}^{C_1}X^{P_2} \cdot {}^{P_2}X^{C_2}(\theta_2) \cdot {}^{C_2}X^{P_3} \cdot {}^{P_3}X^{C_3}(\theta_3) \cdot I \tag{5.7} $$

Given expressions 5.4, 5.5 and 5.6, we can expand Eq. 5.7 as follows:

$$ \begin{aligned} {}^{F}X^{G} = & \underbrace{\begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{{}^{P_1}X^{C_1}(\theta_1)} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & l_1 \\ 0 & 0 & 1 \end{bmatrix}}_{{}^{C_1}X^{P_2}} \cdot \\ & \underbrace{\begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{{}^{P_2}X^{C_2}(\theta_2)} \cdot \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & l_2 \\ 0 & 0 & 1 \end{bmatrix}}_{{}^{C_2}X^{P_3}} \cdot \\ & \underbrace{\begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 \\ \sin(\theta_3) & \cos(\theta_3) & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{{}^{P_3}X^{C_3}(\theta_3)} \end{aligned} \tag{5.8} $$

By deriving expression 5.8, we can write the ***position*** *of the end-effector as:*

$$ {}^{F}p^{G}(q) = {}^{P_1}p^{C_3}(\theta_1,\theta_2,\theta_3) = \begin{bmatrix} {}^{P_1}p_x^{C_3}(\theta_1,\theta_2,\theta_3) \\ {}^{P_1}p_y^{C_3}(\theta_1,\theta_2,\theta_3) \end{bmatrix} = \begin{bmatrix} -l_2\sin(\theta_1+\theta_2) - l_1\sin(\theta_1) \\ l_2\cos(\theta_1+\theta_2) + l_1\cos(\theta_1) \end{bmatrix} \tag{5.9} $$

which can be expanded to:

$$ {}^{F}p_x^{G}(\theta_1,\theta_2,\theta_3) = -l_2\sin\theta_1\cos\theta_2 - l_2\cos\theta_1\sin\theta_2 - l_1\sin\theta_1 \tag{5.10} $$

$$ {}^{F}p_y^{G}(\theta_1,\theta_2,\theta_3) = l_2\cos\theta_1\cos\theta_2 - l_2\sin\theta_1\sin\theta_2 + l_1\cos\theta_1 \tag{5.11} $$

In terms of ***rotation****, in our example, we can derive it intuitively by adding the angle joints:* $\theta_1+\theta_2+\theta_3$ *(see Figure 5.2). If we expand expression 5.8:*

$$ {}^{F}R^{G}(q) = {}^{P_1}R^{C_3}(\theta_1,\theta_2,\theta_3) = \begin{bmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) \end{bmatrix} \tag{5.12} $$

we see that this rotation matrix generates a rotation of $\theta_1+\theta_2+\theta_3$*, validating our initial intuition*[¶].

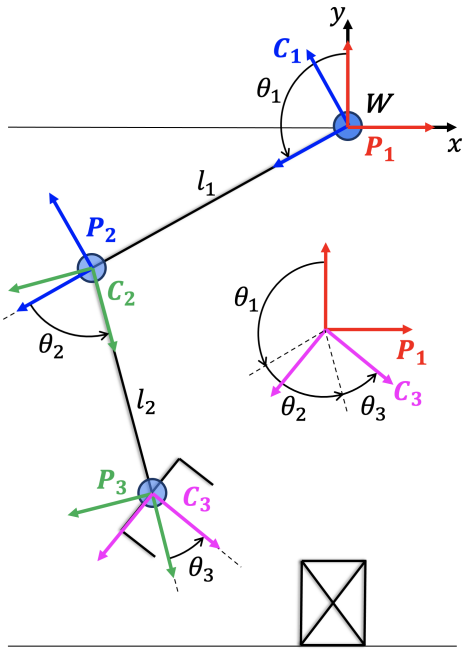


Figure 5.2: 3R planar robot manipulator. The figure visualizes how the orientation of the end-effector can be computed by adding the revolute joint angles θ_i.

The position equations 5.10 and 5.11 are *multilinear trigonometric polynomials* [‖]. Expression 5.3 produces position vectors ${}^{F}p^{G}(q)$ whose elements are multilinear trigonometric polynomials. Thus, each component ${}^{F}p_w{}^{G}(q)$ assumes the form:

$$ {}^{F}p_w^{G}(q) = \sum_j c_{jw} \prod_{i\in[N]} \xi_{ij,w}(q_i) \tag{5.13} $$

where $w \in \{x,y,z\}$[**], $\xi_{ij,w} \in \{\cos(\theta_i), \sin(\theta_i), z_i\}$, and $[N] = \{1,\ldots,N\}$, with N being the number of joints in the kinematic chain from F to G. $c_{jw} \in \mathbb{R}$ is a constant that represents some physical parameter of the robot, such as link length or joint axis.

Essentially, expression 5.13 formally describes that the elements of the position vector p are multilinear trigonometric polynomials. The *multilinear* property is intrinsic in the

[¶]It is important to handle modifications to the rotation matrices with care, as even minor changes, such as axis flips, can produce a different outcome.

[‖]A multilinear polynomial, in the context of trigonometric functions, is such that each term of the polynomial contains each variable (θ_i, z_i) at most once, to a power of one or zero.

[**]w can be z because we have defined expression 5.13 for the general 3D case, where the position vector has three components x, y, z.

product $\prod_{i\in[N]} \xi_{ij,w}(q_i)$, as it ensures that each joint variable q_i appears at most once in each term of the polynomial, to a power of one or zero. j is just a variable to index each term in the polynomial.

5.1.1 Rational Forward Kinematics

While classic forward kinematics allow us to map robot configurations to task space locations, rational forward kinematics deal with representing TC-space robot configurations in task space. Rational FK consist of introducing a bijective parametrization of C-space variables, which results in rational equations to describe the task space position of the end-effector.

Recall from Section 3.1 that we defined the **configuration space variable** q as a set of revolute and prismatic joint values:

$$q = \bigcup_i \{\theta_i, z_i\}$$

for $i \in \{1,\dots,N\}$, with N being the number of joints of the robot.

In the previous section, we determined that FK described positions as multilinear trigonometric polynomial functions. The properties of these functions allow us to introduce the **stereographic projection**[††] (Spivak, 1994) to the revolute joint variables. This consists of the following substitution:

$$t_i := \tan\left(\frac{\theta_i}{2}\right) \tag{5.14}$$

By applying basic trigonometry, this allows us to write

$$\cos(\theta_i) = \frac{1-t_i^2}{1+t_i^2}, \quad \sin(\theta_i) = \frac{2t_i}{1+t_i^2} \tag{5.15}$$

Remark 6 *The **stereographic projection** is the mathematical tool that generates the tangent configuration space. This parametrisation is bijective when $\theta_i \in (-\pi,\pi)$, which is one of the constraints that we assumed in the problem statement (Expression 3.1). As observed in expression 5.15, this substitution generates fractional functions, hence is considered a rational bijective parametrisation.*

With this substitution, we introduce the **tangent configuration space (TC-space) variable** s:

$$s = \bigcup_i \{t_i, z_i\} \tag{5.16}$$

We can then formulate the rational forward kinematics as:

$$X^G = f^G_{rat_kin}(s) \tag{5.17}$$

where $s = \bigcup_i \{t_i, z_i\} \in \mathbb{R}^N$ is the vector of TC-space joint variables, N is the number of joints and $i \in [N]$.

[††]A visualisation of the stereographic projection is provided in Appendix E.1.

Example 2 (continued) *Following with our example, the formalisation of the TC-space variable allows us to describe our end-effector task-space* ***position*** *(from expressions 5.10 and 5.11) as:*

$$ {}^{F}p_x^G(t_1,t_2,t_3) = -l_2\left(\frac{2t_1}{1+t_1^2}\right)\left(\frac{1-t_2^2}{1+t_2^2}\right) - l_2\left(\frac{1-t_1^2}{1+t_1^2}\right)\left(\frac{2t_2}{1+t_2^2}\right) - l_1\left(\frac{2t_1}{1+t_1^2}\right) \tag{5.18} $$

$$ {}^{F}p_y^G(t_1,t_2,t_3) = l_2\left(\frac{1-t_1^2}{1+t_1^2}\right)\left(\frac{1-t_2^2}{1+t_2^2}\right) - l_2\left(\frac{2t_1}{1+t_1^2}\right)\left(\frac{2t_2}{1+t_2^2}\right) + l_1\left(\frac{1-t_1^2}{1+t_1^2}\right) \tag{5.19} $$

which simplifies to:

$$ {}^{F}p_x^G(t_1,t_2,t_3) = \frac{-2l_2t_1(1-t_2^2) - 2l_2t_2(1-t_1^2) - 2l_1t_1(1+t_2^2)}{(1+t_1^2)(1+t_2^2)} \tag{5.20} $$

$$ {}^{F}p_y^G(t_1,t_2,t_3) = \frac{l_2(1-t_1^2)(1-t_2^2) - 4l_2t_1t_2 + l_1(1-t_1^2)(1+t_2^2)}{(1+t_1^2)(1+t_2^2)} \tag{5.21} $$

For ***rotations****, we determined that the orientation of our end-effector could be easily computed by adding the angles of the revolute joints (in our 2D example). This can be parametrised using a simple substitution according to the stereographic projection:*

$$ t_i := \tan\left(\frac{\theta_i}{2}\right) \Leftrightarrow \theta_i := 2\arctan(t_i) \tag{5.22} $$

such that the orientation $\theta_1+\theta_2+\theta_3 = 2\arctan(t_1)+2\arctan(t_2)+2\arctan(t_3)$.

Similar to expression 5.13, we can generalise the position of the end-effector with rational forward kinematics as a rational function:

$$ {}^{F}p_w^G(s) = \sum_j c_{jw} \prod_{i\in[N]} \frac{{}^{F}f_{ij,w}^G(s_i)}{{}^{F}g_{ij,w}^G(s_i)} = \frac{{}^{F}f_w^G(s)}{{}^{F}g_w^G(s)}, \quad w \in \{x,y,z\} \tag{5.23} $$

where w represents each component x,y,z of the position vector p, N is the number of joints in the system with $[N] = \{1,\ldots,N\}$, and

$$ \frac{{}^{F}f_{ij,w}^G(s_i)}{{}^{F}g_{ij,w}^G(s_i)} \in \left\{ \underbrace{\frac{1-t_i^2}{1+t_i^2}}_{\cos\theta_i}, \underbrace{\frac{2t_i}{1+t_i^2}}_{\sin\theta_i}, \underbrace{\frac{z_i}{1}}_{z_i} \right\} $$

Like in equation 5.13, j refers to each term of the polynomial, and c_{jw} represents physical properties of the robot links.

Expression 5.23 essentially formalises that when using TC-space variables in FK, each coordinate of the position of our end-effector is a *rational function* with a polynomial numerator ${}^{F}f_w^G(s)$ and a positive, polynomial denominator ${}^{F}g_w^G(s)$ [‡‡].

[‡‡] ${}^{F}g_w^G(s)$ is positive since each denominator term ${}^{F}g_{ij,w}^G(s_i)$ is either $1+t_i^2$ or 1, which are strictly positive.

We can simplify expression 5.23 as

$$^{F}p^{G}(s) = \frac{^{F}f^{G}(s)}{^{F}g^{G}(s)} \tag{5.24}$$

where $^{F}f^{G}(s)$ is a vector of polynomials (with vector components x, y, z) and $^{F}g^{G}(s)$ is a single, positive polynomial. Since in our problem statement assumptions (Section 3.1) we constrained θ and z, we can constrain the TC-space variable as well such that

$$s_l \leq s \leq s_r. \tag{5.25}$$

5.2 Inverse Kinematics

Inverse kinematics (IK) present a more complex problem than FK in open-chain robots, as more than one robot configuration may be able to achieve one task space location of the end-effector. Thus, the IK problem can have multiple (even infinity) or no solutions. In this section, we present as contribution the formalisation of the inverse kinematics of 3R planar robots following Example 2. We discuss the generalisation of the approach and computations to higher DOF and the TC-space.

We continue with the example scenario from Figure 5.1. We want to determine the joint angles θ_1, θ_2 and θ_3 from a given end-effector position and orientation. We treat the position and orientation as constants represented by:

$$X = {}^{F}p_x^G \rightarrow \text{The } x \text{ component of the position vector } {}^{F}p^G.$$
$$Y = {}^{F}p_y^G \rightarrow \text{The } y \text{ component of the position vector } {}^{F}p^G.$$
$$\Phi = \text{The orientation of the end-effector frame } C_3.$$

We further assume that the link lengths of our robot l_1, l_2 are given. From the derivations presented in the last section (Eq. 5.9, 5.12), we can formulate a system of equations:

$$\begin{cases} (1)\ X = -l_2 \sin(\theta_1 + \theta_2) - l_1 \sin\theta_1 \\ (2)\ Y = l_2 \cos(\theta_1 + \theta_2) + l_1 \cos\theta_1 \\ (3)\ \Phi = \theta_1 + \theta_2 + \theta_3 \end{cases} \tag{5.26}$$

where we have three equations and three unknowns θ_1, θ_2 and θ_3. We can derive these equations using the corresponding arithmetics. A step-by-step computation is provided in Appendix E.2. We arrived at the following results:

$$\theta_1 = -\arctan\left(\frac{X}{Y}\right) \pm \arccos\left(\frac{l_2^2 - l_1^2 - X^2 - Y^2}{-2l_1\sqrt{X^2+Y^2}}\right) \tag{5.27}$$

$$\theta_2 = \arctan\left(\frac{-X - l_1\sin\theta_1}{Y - l_1\cos\theta_1}\right) - \theta_1 \tag{5.28}$$

$$\theta_3 = \Phi - \theta_1 - \theta_2 \tag{5.29}$$

With this formalisation, we can infer the joint angles of any 3R planar robot similar to the one in Figure 5.1[§§].

[§§]Notice that we must be careful with the use of the arctangent. In this situation, if $Y < 0$ or $Y - l_1\cos\theta_1 < 0$, we need to add or subtract 180° to the computations of θ_1 and θ_2, whatever adapts to the range $-\pi < \theta_i < \pi$.

Example 3 *We consider a practical example in which our 3R planar robot has link lengths* $l_1 = 3, l_2 = 2$*, and we want to place the gripper at position* $[-1,-4]$*, with an orientation of* 180°*. That is:* $X = -1, Y = -4$ *and* $\Phi = 180$*. Substituting these values in Equations 5.27, 5.28, 5.29, we obtain the following results:*

$$\boxed{\theta_{1a} \approx 193.18^\circ, \quad \theta_{2a} \approx -70.53^\circ, \quad \theta_{3a} \approx 57.35^\circ}$$

$$\boxed{\theta_{1b} \approx 139.75^\circ, \quad \theta_{2b} \approx 70.53^\circ, \quad \theta_{3b} \approx -30.28}$$

For θ_{1a} *to be in the range* $(-\pi, \pi)$*, we subtract 360 and obtain* -166.82°.

5.2.1 "Rational" Inverse Kinematics

In the context of the TC-space, the "rational" inverse kinematics problem consists of mapping end-effector locations to TC-space variables $s = \bigcup_i \{t_i, z_i\}$, such that

$$s = f^{-1}_{rat_kin}(X^G)$$

This can be done in multiple ways. For instance, we could simply calculate the classic IK for C-space variables, and then apply the stereographic projection to the revolute joint angles θ_i obtained.

Example 4 *We parametrise solution "a" from Example 3 using* $t_i := \tan \frac{\theta_i}{2}$*. We obtain* $t_1 \approx -8.66, t_2 \approx -0.71, t_3 \approx 0.55$*. By substituting in Eq. 5.20 and 5.21, we get that* $p_x \approx -0.999$ *and* $p_y \approx -3.999$*, matching the initial values* $X = -1, Y = -4$.

Alternatively, we could compute expressions similar to Eq. 5.27, 5.28 and 5.29, but for TC-space variables. That is, create a system of equations from Equations 5.20, 5.21, and the $\theta_1 + \theta_2 + \theta_3$ parametrised using Eq. 5.22, and solve it for each t_i. This produces a system of polynomial equations, as opposed to the multilineal system described in Eq. 5.26. Therefore, there are trade-offs with both formulations, as one needs to deal with high-degree polynomials and the other with trigonometric equations.

5.2.2 Scalability to Higher DOF or 3D Robots

In the previous examples, we discussed the inverse kinematics (IK) of a 3R planar robot. The forward kinematics (FK) and IK problems have different solutions for each robotic system. In cases where robots have more degrees of freedom (DOF), the system of linear equations described in Equation 5.26 would have a larger number of unknown variables, leading to a problem with a higher number of possible solutions. Thus, increasing the complexity of the computation. Dealing with 3D robots, as opposed to planar, involves considering more given information. In such cases, we need to be given the position with a z coordinate and the orientation of the end-effector with 3D rotations in one of the forms presented in Appendix C.2. Essentially, this is a slightly more complex instance of the problem too. However, the resources provided in this chapter are sufficient to formalise the 3D instance of the problem.

Chapter 6

Manipulation Constraints

As mentioned in Section 4.2, the definition of grasping constraints is crucial for the description of robotic manipulation in C-space and TC-space. In this chapter, we explore how to define constraints on standard 3D geometries, providing visualisations in Drake*.

All the constraints provided are simulated with the widely used robotic gripper Schunk WSG (Schunk, n.d.).

6.1 Constraints for Standard Geometries

In this section, we discuss possible grasping constraints for cubes, prisms, cylinders and spheres. We define these constraints as poses of the object relative to the gripper ${}^{G}X^{O}$, similar to in Chapter 4, but discussing the full range of plausible grasping poses for each object. The descriptions provided in this section can be later used to compute gripper poses in task space, and from there, IK can be applied to determine grasping regions in C-space and TC-space.

It is important to note that universal constraints do not exist, even for geometric objects of the same shape. When planning grasping tasks, the user must consider which grasping locations are appropriate for the specific object and circumstances, as many factors can be present, such as weight, velocity, stability preferences, etc. We provide some standard constraints as an example, to serve as ground for escalating to more complex systems.

Assumptions: For the purpose of defining the constraints, we made the following assumptions:

- Every geometry discussed has at least one dimension smaller than the gripper claw, so grasping is feasible.
- We placed the frame of the object at its centre to consider grasping constraints relative to that position.

*All the frames depicted in the visualisations provided in this report are represented with the x axis in red, the y axis in green, and the z axis in blue.

6.1.1 Cubes

Cubes can be grasped from above or the sides. We focused on representing the angle and orientation at which the gripper can approach the object.

We need to define **position** and **rotation** constraints.

For **position**, we set the distance from the Schunk frame G to the object frame O to 0.11. We can also define this in a range, depending on the size of both the manipulator and the object (See Figure 6.1). These values are chosen based on the physical properties of both the gripper and the object at hand. For cubes with side length k, and grippers of length l (from the gripper frame origin to the tip of the fingers), we generally do not want the distance from G to O to be outside the range $l \pm \frac{k}{2}$.

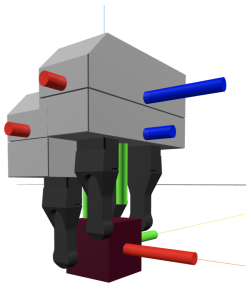


Figure 6.1: Visualisation in Drake. Two grippers grasping a cube at different heights. In this specific simulation, the distance from the gripper frame to the object frame ranges between 0.095 and 0.13.

Since O is ahead of the y axis of G, we define the position constraint as

$$ {}^{G}p^{O} = \begin{bmatrix} x \\ C_y \\ z \end{bmatrix} \quad \text{for } 0.095 \leq C_y \leq 0.13 \tag{6.1} $$

We can similarly define position constraints along the x axis.

In terms of **rotation**, we represented how the gripper can be inclined along both the x and y axes of the object. To compute the rotation constraints, we decided to set a constant distance for simplicity.

See Figure 6.2. It represents how grippers rotate about the y axis of the object. To define this constraint, we observed that this rotation happened about the x axis of the gripper.

If we look at the two gripper frames that delimit the rotation, we can see that a rotation about the x axis is necessary to align the z axis of the gripper and object frames. This rotation happens in the range $[\pi, 0]$. Once the z axis is aligned, we only need to make a rotation of $\frac{\pi}{2}$ radians about z to align the gripper and object frames. Therefore, we defined the first rotation constraints as follows:

$$ {}^{G}R^{O} = R_x(C_x) \cdot R_z\left(\frac{\pi}{2}\right) \quad \text{for } 0 \leq C_x \leq \pi \tag{6.2} $$

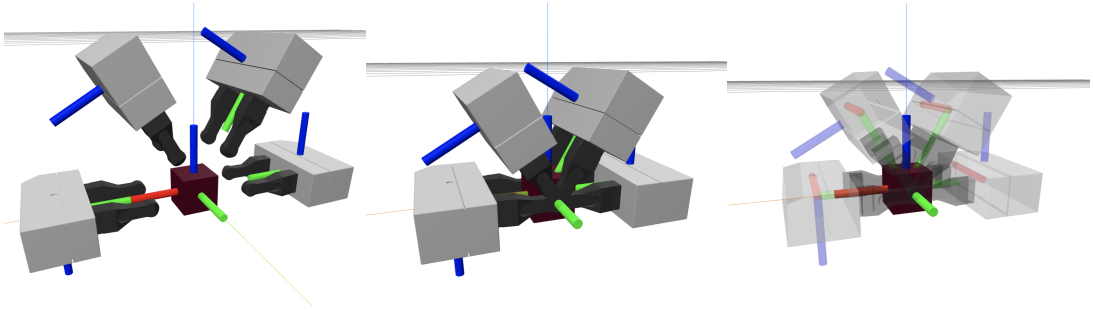


Figure 6.2: Simulation in Drake. Gripper in different rotations along the y axis approaching the cube in valid grasping poses.

We can do the same for constraining rotation about the x axis of the object (see Figure 6.3). In this case, the x axis of the object and gripper are already aligned, hence we did not need the z rotation. We defined the following rotation constraints:

$$^{G}R^{O} = R_x(C_x) \quad \text{for } 0 \leq C_x \leq \pi \tag{6.3}$$

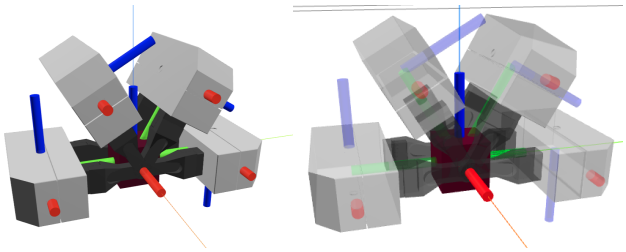


Figure 6.3: Simulation in Drake. Gripper in different rotations along the x axis approaching the cube in valid grasping poses.

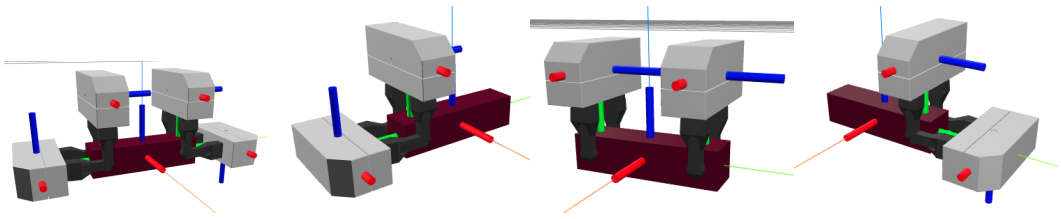
6.1.2 Prisms

In terms of prisms, we have assumed that they are lying on the surface on their long side. We can split the calculation of grasping constraints into three sections. In Figure 6.4a, we have plotted the grippers that delimit these regions.

For the first region (Figure 6.4b), we defined both specific position and rotation constraints. For rotation, given that the x axes of the gripper and object frames were aligned, we represented the constraints as

$$^{G}R^{O} = R_x(C_x) \quad \text{for } 0 \leq C_x \leq \frac{\pi}{2} \tag{6.4}$$

In terms of position, this scenario is slightly more complicated. Notice in Figure 6.4b that every grasping pose in this section places the tip of the Schunk in the same position respective to the object centre. For a prism with horizontal length k, and height m, the distance from the object frame to this position is $\frac{k-m}{2}$. If we let the length of the gripper from frame G to tip be l, by looking at Figure 6.4b, we determined:



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

Figure 6.4: Simulation in Drake. Gripper in different poses to grasp a prism. (a) visualise the grippers in the delimiting positions that separate the relevant sections for computing constraints. (b),(c),(d) represent the grippers delimiting each region.

- The horizontal gripper is at distance ${}^{G}p^{O} = \begin{bmatrix} 0 \\ l + \frac{k-m}{2} \\ 0 \end{bmatrix}$ from the object frame.
- The vertical gripper is at distance ${}^{G}p^{O} = \begin{bmatrix} 0 \\ l \\ \frac{k-m}{2} \end{bmatrix}$ from the object frame.

We formalised the geometrical considerations of the position of the gripper in Figure 6.5. We established that the position was dependent on the rotation, such as:

$$ {}^{G}p^{O} = \begin{bmatrix} 0 \\ l + \frac{k-m}{2} \cdot \cos({}^{G}R_x^O) \\ \frac{k-m}{2} \cdot \sin({}^{G}R_x^O) \end{bmatrix} \quad \text{for } 0 \leq {}^{G}R_x^O \leq \frac{\pi}{2}, \tag{6.5} $$

Where ${}^{G}R_x^O$ is the rotation about the x axis determined by expression 6.4.

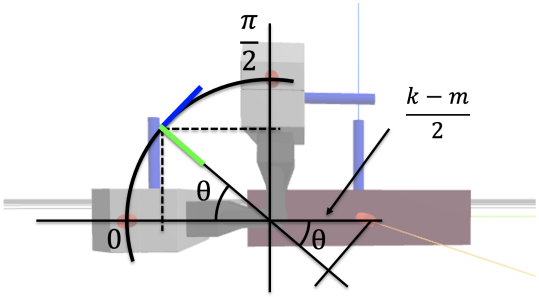


Figure 6.5: Inverted unit circle visualisation on gripper rotation. The position of the gripper frame changes relative to the angle of rotation.

The constraints for the region in Figure 6.4d can be similarly computed by adapting the signs of the expression accordingly.

For the constraints of the section represented by Figure 6.4c, we defined them as:

$$ {}^{G}p^{O} = \begin{bmatrix} 0 \\ l \\ C_z \end{bmatrix}, \quad \text{for } -\frac{k-m}{2} \leq C_z \leq \frac{k-m}{2} \tag{6.6} $$

$$ {}^{G}R^{O} = R_x\left(\frac{\pi}{2}\right) \tag{6.7} $$

6.1.3 Cylinders

To define the grasping constraints of cylinders, we assumed that the object was placed vertically and that the grasping was produced along the cylinder (See Figure 6.6).

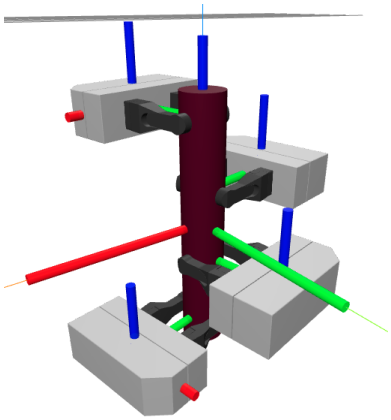


Figure 6.6: Simulation in Drake. Multiple Schunk grippers grasping a cylinder at valid grasping poses.

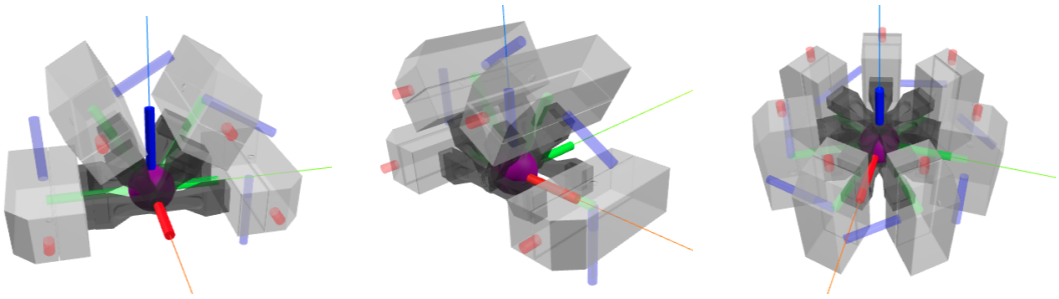
We assume the cylinder has length k and radius r, and denominate the length from the gripper frame to the tip of the fingers as l. Considering that the x axis of the object and gripper frames are aligned, we define the constraints of the cylinder as follows:

$$ {}^{G}p^{O} = \begin{bmatrix} 0 \\ l \\ C_z \end{bmatrix}, \quad \text{for } -\frac{k-r}{2} \leq C_z \leq \frac{k-r}{2} \tag{6.8} $$

$$ {}^{G}R^{O} = R_z(C_{Rz}), \quad \text{for } -\pi \leq C_{Rz} \leq \pi \tag{6.9} $$

6.1.4 Spheres

Spheres conform the geometry with less restricted constraints from the ones we have seen (see Figure 6.7).



(a) (b) (c)

Figure 6.7: Simulation in Drake. Gripper in different rotations to grasp the sphere.

Any orientation that positions the gripper above the ground is acceptable. However, we must be cautious, as even if individual rotation axes are constrained to the hemisphere, a combination of them may not fulfil the requirement.

Hence, we decided to represent this instance differently, as a combination of constraints of the object measured from the gripper frame, and vice-versa. We therefore formalised the constraints of the sphere as:

$$ {}^{G}p^{O} = \begin{bmatrix} 0 \\ l \\ 0 \end{bmatrix} \tag{6.10} $$

$$ {}^{O}p^{G} = \begin{bmatrix} x \\ y \\ C_z \end{bmatrix}, \quad \text{for } C_z \geq 0 \tag{6.11} $$

The first component of the constraint implies that the gripper always faces the sphere centre in the y axis at a distance of l. The second term indicates that any position of the gripper relative to the object is valid, but the z component must be positive, ensuring that the gripper stays in the upper half-sphere.

Chapter 7

Simulations, Experiments and Discussion

7.1 Visualisation of Grasping Constraints in C-space

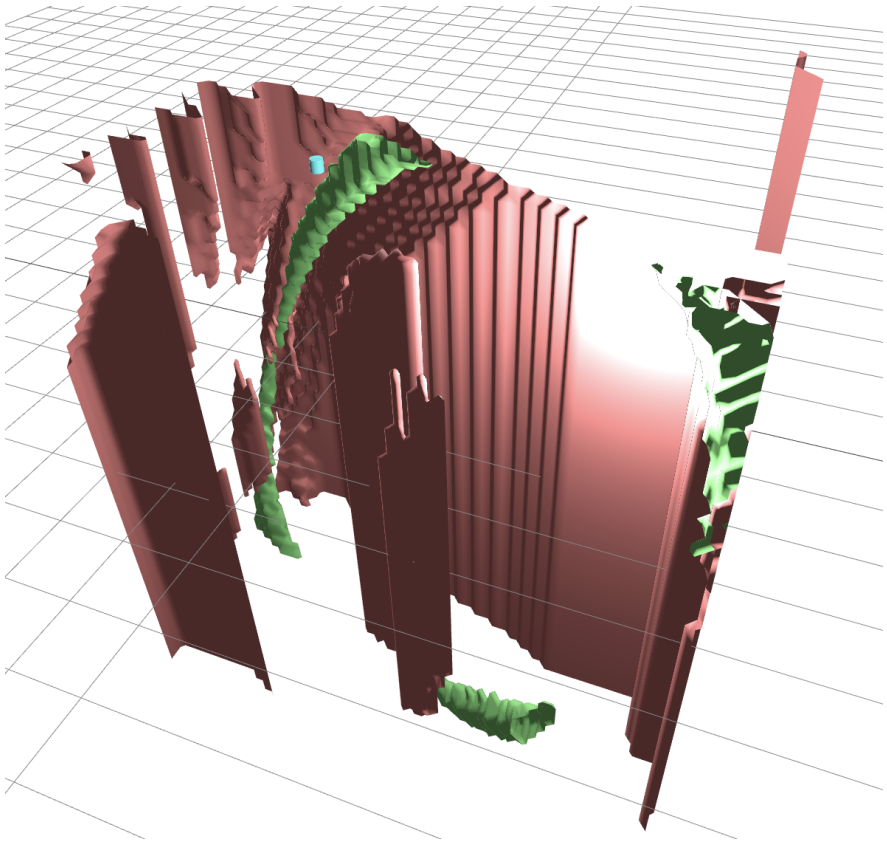


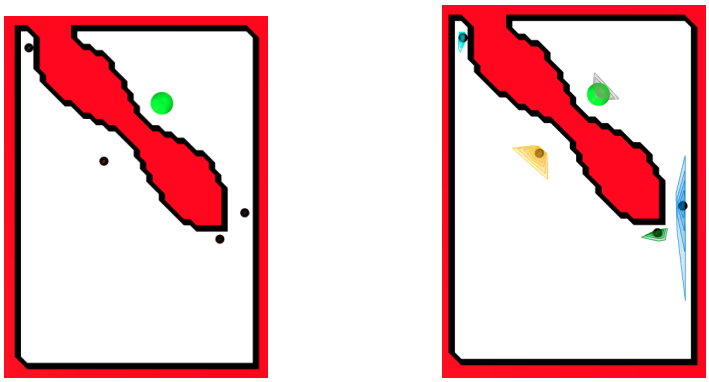
Figure 7.1: Simulation in Drake. Adapted from Tedrake (2023, Notebook 6). C-free in conjunction with the grasping space of a cylinder in the presence of other cylindrical obstacles and a set of shelves. In red, the C-free. In green, the configuration space of the grasping.

Figure 7.1 represents the configuration space of a grasping action for a cylinder in green and the C-free space in red. We adapted this simulation from Tedrake (2023, Notebook 6), modifying it to represent the grasping constraints of a cylinder.

This visualisation conceptualises the complexity of the configuration space and the abstraction that it assumes from the intuition of task-space applications.

In the context of further research, we are interested in identifying the intersection of the two subspaces and analysing its resembling with the equivalent in TC-space. Exploring the topology of these spaces can prove essential to optimise methods that utilise them.

7.2 Growing Polygons with C_IRIS



(a) (b)

Figure 7.2: Simulation in Drake. TC-spacevisualisation of the C_IRIS algorithm. (a) shows the default seeding points presented by the authors to generate the convex regions in C-free. (b) plots these regions in the first few iterations of the algorithm, using the bilinear alternation method.



(a) (b)

Figure 7.3: Simulation in Drake. TC-spacevisualisation of the C_IRIS algorithm. (a) shows newly added seeding points more central in C-free. (b) plots the regions generated in the first few iterations of the algorithm, using the bilinear alternation method.

We conducted an experiment exploring the effect of the seeding point locations at the time of generating convex polytopes. For this purpose, we defined new seeding points

more centric in C-free and ran the C_IRIS algorithm comparing the new seeds with the default ones. Figure 7.2 shows the growing regions with the default seeds, while Figure 7.3 does the same with the newly defined. We can observe how the new seeds clearly generated bigger polytopes, thus performing better. Not only that, but it did so in a runtime of 278.28 seconds, as opposed to 309.99 seconds for Figure 7.2.

It becomes apparent that the location of initial seeds plays a role in the efficiency of C_IRIS. Thus, it is a matter that should be explored in the future.

Chapter 8

Conclusion

In this report, we have explored the implications of using the TC-space in motion and manipulation planning applications. We expanded Dai et al. (2023)'s advances to accommodate robots with an additional type of joint (Universal), and presented the mathematical tools necessary to understand the formulation of the manipulation planning problem. We provided relevant examples and derived calculations to demonstrate the practical implementation of the techniques described. We have also formalised and explained the forward and inverse kinematics problem, presenting a description of the IKs in this context for the first time. Additionally, we have defined grasping constraints for standard geometries and used the Drake robotics toolbox to visualise the implications of the concepts described.

The primary objective of this work was to introduce the concept of manipulation planning and provide a detailed description of the tangent configuration space. This serves as a solid foundation for individuals new to this field to appreciate the significance of this task and explore further research opportunities. By presenting a comprehensive overview of the essential tools in robotics, we aim to facilitate a better understanding of this subject matter, leading to the expansion of knowledge and the advancement of research in this area.

8.1 Future Work and MInf 2

There are numerous ways in which the concepts described in this report could be used, redefined and extended in the future. The immediate continuation of this work would be to implement the tools described here to formulate an algorithm to compute manipulation trajectories in TC-space. This goal entails the cohesion of the contents of this work, algorithmic theory, and mathematical optimisation, together with the Drake toolbox, on generating complete manipulation simulations and comparing them with previous approaches to analyse the effects of the tangent configuration space parametrisation.

Additional work could also be carried out by doing further experiments and analysis on the methods presented by Dai et al. (2023). That is, conducting a more extensive review

of the efficiency of the algorithms proposed to gain a richer idea of the scalability of the contributions.

Further work could be developed in the definition of constraints and the rational parametrisation of the inverse kinematics. While we provided an introduction to the problem and planar computational examples, there is still a vast extent of ways in which to explore the depth and limitations of robot kinematics.

In addition, a continuation of this work could also target physical manipulations, attempting to implement the methods discussed for real-world arms and applications.

8.1.1 MInf 2

Next year, we intend to continue exploring the field of manipulation planning in the context of TC-space applications. Although a thorough plan is not yet in place, the above expansions have been presented as plausible topics for the MInf Part 2 master dissertation. Primarily, the objective is to formulate an algorithm to describe manipulation motions in TC-space, analyse its implications and compare it in the context of current research, as this has not been explored before in the robotics field.

Bibliography

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.

D. Berenson. *Constrained Manipulation Planning*. PhD thesis, Carnegie Mellon University, June 2011. URL https://www.proquest.com/openview/6d762c3ba0fb2eff9bbb6606308804e9/1?pq-origsite=gscholar&cbl=18750.

O. Bottema and B. Roth. *Theoretical Kinematics*. Dover Books on Physics. Dover Publications, 1990. ISBN 9780486663463. URL https://books.google.co.uk/books?id=AMlIAwAAQBAJ.

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.

J.J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005. ISBN 9780201543612. URL https://books.google.co.uk/books?id=MqMeAQAAIAAJ.

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

B.N. Datta. *Numerical Linear Algebra and Applications: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2010. ISBN 9780898717655. URL https://books.google.co.uk/books?id=-tW8-FUoxWwC.

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.

R. Diankov, S.S. Srinivasa, D. Ferguson, and J. Kuffner. Manipulation planning with caging grasps. In *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pages 285–292, Daejeon, Korea (South), 2008. IEEE. ISBN 978-1-4244-2821-2 978-1-4244-2822-9. doi: 10.1109/ICHR.2008.4755966. URL http://ieeexplore.ieee.org/document/4755966/.

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.

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.

B. Grünbaum. *Convex polytopes. Prepared by V. Kaibel, V. Klee, and G.M. Ziegler. 2nd ed*, volume 221. 01 2003. ISBN 978-0-387-40409-7. doi: 10.1007/978-1-4613-0019-9.

J.A. Haustein. *Robot Manipulation Planning Among Obstacles: Grasping, Placing and Rearranging*. PhD thesis, 2020. URL https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-266792. Publisher: KTH Royal Institute of Technology.

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.

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.

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.

S. Lall. Sums of squares. Lecture Slides, April 2011. URL `https://web.stanford.edu/class/ee364b/lectures/sos_slides.pdf`. EE364b Course.

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.

T. Lozano-Perez, J. Jones, E. Mazer, P. O'Donnell, W. Grimson, P. Tournassoud, and A. Lanusse. Handey: A robot system that recognizes, plans, and manipulates. In *Proceedings. 1987 IEEE International Conference on Robotics and Automation*, volume 4, pages 843–849, 1987. doi: 10.1109/ROBOT.1987.1087847.

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

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

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.

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.

J.P. Saut, A. Sahbani, S. El-Khoury, and V. Perdereau. Dexterous manipulation planning using probabilistic roadmaps in continuous grasp subspaces. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2907–2912, 2007. doi: 10.1109/IROS.2007.4399090.

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.

J.T. Schwartz and M. Sharir. On the "piano movers'" problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983. URL https://api.semanticscholar.org/CorpusID:120028872.

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. Spivak. *Calculus Third Edition*. Cambridge University Press, 1994.

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

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.

Wikipedia. Right-hand rule. https://en.wikipedia.org/wiki/Right-hand_rule, 2022.

Appendix A

Requirements for Non-Collision Certificates

We assume that the robot, obstacles and movable objects in our environment are decomposed into convex, compact bodies.

Specifically, for the purpose of decomposing C-free into certified convex polytopes, we require our robot and obstacles to be decomposed as a union compact, convex sets that can be expressed as Archimedean, basic semialgebraic sets*.

A.1 Archimedean Property

We formally define the Archimedean property as:

Definition 3 *A semialgebraic set* $\mathcal{S}_g = \{x | g_i(x) > 0, i \in [n]\}$ *is Archimedean if there exists* $N \in \mathbb{N}$ *and* $\lambda_i(x) \in \Sigma$ *such that:*

$$N - \sum_{i=1}^{n} x_i^2 = \lambda_0(x) + \sum_{i=1}^{n} \lambda_i(x) g_i(x)$$

where $[n] = \{1, \ldots, n\}$ *and* Σ *is the cone of Sums-of-Squares (SOS) polynomials*†.

Intuitively, this definition ensures that the space described by the equalities and inequalities of the semi-algebraic set do not extend out to infinity.

This requirement is necessary to provide non-collision certificates as per the definitions of Dai et al. (2023). Specifically, the Archimedean property allows to certify the positivity or infeasibility of the generalisation of these two programs:

*A basic semialgebraic set is a set defined by polynomial equalities and polynomial inequalities, and a semialgebraic set is a finite union of basic semialgebraic sets.

†An introduction to SOS polynomials can be found in Lall (2011).

1. Two bodies $\mathcal{A}$ and $\mathcal{B}$ are *not in collision* if there exists a hyperplane which separates them. (Separating Hyperplane Theorem (Boyd and Vandenberghe, 2004, Section 2.5))
2. If two bodies $\mathcal{A}$ and $\mathcal{B}$ are *in collision* then there exists a point x such that $x \in \mathcal{A}$ and $x \in \mathcal{B}$.

The positivity of Program 1 implies that there exists a hyperplane separating the two bodies; thus, the bodies are not in collision. The infeasibility of Program 2 implies that there is no point x common to both bodies; thus, the bodies are not in collision.

A detailed explanation of these certificates can be found in Dai et al. (2023, Section 3.2)

A.2 Polytopes and Convexity

For completeness, we formalise the concept of convex polytope by introducing the definitions of convex sets and convex hulls from Boyd and Vandenberghe (2004).

Definition 4 (Convex set) *A set $\mathcal{C}$ is* convex *if the line segment between any two points in $\mathcal{C}$ lies in $\mathcal{C}$, i.e., if for any $x_1, x_2 \in \mathcal{C}$ and any θ with $0 \leq \theta \leq 1$, we have*

$$\theta x_1 + (1-\theta)x_2 \in \mathcal{C}$$

Definition 5 (Convex hull) *The* convex hull *of a set $\mathcal{C}$, denoted* $\mathbf{conv}\mathcal{C}$*, is the set of all convex combinations of points in $\mathcal{C}$:*

$$\mathbf{conv}\mathcal{C} = \{\theta_1 x_1 + \ldots + \theta_k x_k \mid x_i \in \mathcal{C},\ \theta_i \geq 0,\ i = 1, \ldots, k,\ \theta_1 + \cdots + \theta_k = 1\}$$

With these concepts, Grünbaum (2003) defined a convex polytope as follows:

Definition 6 (Convex polytope) *A* convex polytope *$\mathcal{P}$ is defined to be the convex hull of any finite set of points in $\mathbb{R}^d$.*

Appendix B

Joint Decomposition and Algebraic Kinematics

(Wampler and Sommese, 2011, Chapter 4) provides a thorough description of algebraic kinematics and low-order pairs. We summarise the different joint decompositions into revolute and prismatic, with the addition of the universal joint (U). A description of the universal joint and its restrictions and implications can be found in Lynch and Park (2017, Chapter 2.2).

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.

Recall from Chapter 5 that ${}^{P_i}X^{C_i}(q_i)$ represents the relative motion when under the joint

q_i. We can represent this transform in the form:

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

Each time you add a 3D rigid body/link to a robot system, you add 6DOF, 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 B.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 (B.4)	$q_i = \{\phi_{i,x}, \phi_{i,y}, \phi_{i,z}\}$
U	see equations (B.5, B.6, B.7)	$q_i = \{\phi_{i,x}, \phi_{i,y}\}$ or $\{\phi_{i,x}, \phi_{i,z}\}$ or $\{\phi_{i,y}, \phi_{i,z}\}$

With this, we can express the joints C, E, S decomposed into P and R:

- C joint decomposed into one R and one P:

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

- E joint decomposed into two P and one R:

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

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

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

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

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

or

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

or

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

Appendix C

Spatial Algebra

C.1 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 (2023, 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:

$$ {}^{A}p_{F}^{B} + {}^{B}p_{F}^{C} = {}^{A}p_{F}^{C} \tag{C.1} $$

- **Additive Inverse:**

$$ {}^{A}p_{F}^{B} = -{}^{B}p_{F}^{A} \tag{C.2} $$

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

$$ {}^{A}p_{C}^{B} = {}^{C}R^{F}\,{}^{A}p_{F}^{B} \tag{C.3} $$

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

$$ {}^{A}R^{B}\,{}^{B}R^{C} = {}^{A}R^{C}, \quad {}^{A}X^{B}\,{}^{B}X^{C} = {}^{A}X^{C} \tag{C.4} $$

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

$$ \left[{}^{A}R^{B}\right]^{-1} = {}^{B}R^{A}, \quad \left[{}^{A}X^{B}\right]^{-1} = {}^{B}X^{A} \tag{C.5} $$

C.2 Representing Rotations

Representing 3D rotations is crucial in various fields, such as robotics, computer graphics, and aerospace engineering. There are four standard methods to represent 3D rotations, which are discussed below:

1. **Rotation Matrices:** A 3×3 rotation matrix is an orthonormal matrix. Each column of the matrix comprises the unit vectors along a frame's x, y, and z axes in another frame. It offers a straightforward linear algebraic method for transforming coordinates and composing rotations. However, it is not the most efficient in terms of computation or storage.
2. **Euler Angles (Roll-Pitch-Yaw):** Euler angles specify rotation by sequential rotations around three principal axes, typically x, y, and z. The specific sequence of rotations (e.g., x-axis, y-axis, then z-axis) matters and can represent any rotation. However, Euler angles can suffer from gimbal lock, where two axes align, and the system loses a degree of freedom.
3. **Axis-Angle (Exponential Coordinates):** The axis-angle representation uses a vector whose direction represents the axis of rotation, and whose magnitude represents the rotation angle. This compact representation can be intuitive for single rotations, but it has singularities, such as when the rotation angle is zero.
4. **Unit Quaternions:** Quaternions provide a four-element representation, where three elements correspond to the imaginary components, and one to the real part. This unit-norm quaternion avoids singularities and gimbal lock, making it a preferred choice for interpolating rotations (SLERP) and when dealing with continuous rotation trajectories.

In practice, each representation has its use cases, and systems like Drake provide facilities to convert between them.

A review of this concepts can be fount in Tedrake (2023, Chapter 3.3).

Appendix D

Computation of Gripper Frames

D.1 Definition of Pseudoinverse

From Datta (2010, p. 245-246):

Definition 7 (Moore-Pensore Pseudoinverse) *Assume that A is a matrix $m \times n$ $(m \geq n)$ and has full rank. So, $A^T A$ is invertible. The matrix*

$$A^+ = (A^T A)^{-1} A^T$$

is called the pseudoinverse *or the* Moore-Penrose generalised inverse of *A*. *This definition of the* pseudoinverse generalises the ordinary definition of the inverse of a square matrix *A*. *Note that when A is square and invertible*

$$A^+ = (A^T A)^{-1} A^T = A^{-1} (A^T)^{-1} A^T = A^{-1}$$

D.2 Drake Code Example

Given that this dissertation's contributions include implementing manipulation simulations in Drake, we provide a snippet of code to showcase an example of how to use rigid transforms, rotation matrices, and overall manipulation actions in the software.

This method takes as input two dictionaries `X_WG` and `X_WO`, which contain information about the gripper and the object.

- `X_WO` contains poses of the object at specific steps. For instance, `X_WO["goal"]` is equivalent to ${}^{W}X^{O_{\text{goal}}}$. For the purpose of computing relevant gripper frames at grasping and pregrasping poses, this dictionary includes the initial and goal pose of the object.
- `X_WG` contains poses of the gripper at specific steps. For the purpose of this example, it is empty, so we can fill it out and return it.

This code was adapted from (Tedrake, 2023, Notebook 3).

Algorithm 1 MakeGripperFrames

Input: X_WG, X_WO
Output: X_WG

```
p_GgraspO = [0, 0.12, 0]
R_GgraspO = RotationMatrix.MakeXRotation(np.pi/2)@
      RotationMatrix.MakeZRotation(np.pi/2)
X_GgraspO = RigidTransform(R_GgraspO, p_GgraspO)
X_OGgrasp = X_GgraspO.inverse()
X_GpregraspGgrasp = RigidTransform([0, 0.2, 0])
X_GgraspGpregrasp = X_GpregraspGgrasp.inverse()
X_WG["pick"] = X_WO["initial"] @ X_OGrasp
X_WG["prepick"] = X_WG["pick"] @ X_GraspGpregrasp
X_WG["place"] = X_WO["goal"] @ X_OGrasp
X_WG["postplace"] = X_WG["place"] @ X_GraspGpregrasp
```

Appendix E

Robot Kinematics

E.1 Stereographic Projection

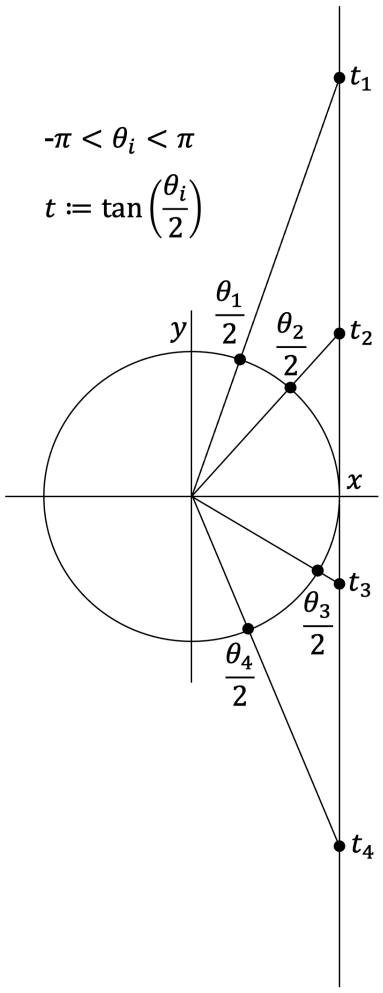


Figure E.1: Visual interpretation of the stereographic projection in the unit circle (2D), when θ is bounded to $-\pi < \theta_i < \pi$.

E.2 Derivation of IK for 3R Planar Robot

We have the following equations, where X, Y, Φ, l_1 and l_2 are given constants.

$$\begin{cases} (1)\ X = -l_2 \sin(\theta_1 + \theta_2) - l_1 \sin\theta_1 \\ (2)\ Y = l_2 \cos(\theta_1 + \theta_2) + l_1 \cos\theta_1 \\ (3)\ \Phi = \theta_1 + \theta_2 + \theta_3 \end{cases}$$

We hereby present the steps to derive the solutions for θ_1, θ_2 and θ_3:

From equations (1) and (2):

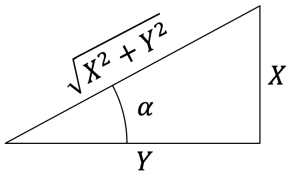
$$l_2 \sin(\theta_1 + \theta_2) = -X - l_1 \sin\theta_1 \tag{E.1}$$
$$l_2 \cos(\theta_1 + \theta_2) = Y - l_1 \cos\theta_1 \tag{E.2}$$

We square both expressions and add them together:

$$l_2^2 \sin^2(\theta_1 + \theta_2) + l_2^2 \cos^2(\theta_1 + \theta_2) = (-X - l_1 \sin\theta_1)^2 + (Y - l_1 \cos\theta_1)^2 \tag{E.3}$$
$$l_2^2(\sin^2(\theta_1 + \theta_2) + \cos^2(\theta_1 + \theta_2)) = (-X)^2 + 2Xl_1 \sin\theta_1 + l_1^2 \sin^2\theta_1 \tag{E.4}$$
$$Y^2 - 2Yl_1 \cos\theta_1 + l_1^2 \cos^2\theta_1 \tag{E.5}$$

$$l_2^2 = X^2 + Y^2 + l_1^2(\sin^2\theta_1 + \cos^2\theta_1) - 2l_1(Y\cos\theta_1 - X\sin\theta_1) \tag{E.6}$$
$$\frac{l_2^2 - l_1^2 - X^2 - Y^2}{-2l_1} = Y\cos\theta_1 - X\sin\theta_1 \tag{E.7}$$

We use the Pythagorean Theorem according to this figure:



such that

$$\alpha = \arctan\left(\frac{X}{Y}\right), \quad X = \sin\alpha \cdot \sqrt{X^2+Y^2}, \quad Y = \cos\alpha \cdot \sqrt{X^2+Y^2}$$

We divide both sides of the last expression by $\sqrt{X^2+Y^2}$ and substitute X and Y from the expressions above:

$$\frac{l_2^2 - l_1^2 - X^2 - Y^2}{-2l_1\sqrt{X^2+Y^2}} = \frac{\cos\alpha \cdot \sqrt{X^2+Y^2} \cdot \cos\theta_1 - \sin\alpha \cdot \sqrt{X^2+Y^2} \cdot \sin\theta_1}{\sqrt{X^2+Y^2}} \tag{E.8}$$
$$\frac{l_2^2 - l_1^2 - X^2 - Y^2}{-2l_1\sqrt{X^2+Y^2}} = \cos\alpha\cos\theta_1 - \sin\alpha\sin\theta_1 \tag{E.9}$$
$$\frac{l_2^2 - l_1^2 - X^2 - Y^2}{-2l_1\sqrt{X^2+Y^2}} = \cos(\theta_1 + \alpha) \tag{E.10}$$

$$\theta_1 = -\alpha \pm \arccos\left(\frac{l_2^2 - l_1^2 - X^2 - Y^2}{-2l_1\sqrt{X^2+Y^2}}\right) \tag{E.11}$$

$$\boxed{\theta_1 = -\arctan\left(\frac{X}{Y}\right) \pm \arccos\left(\frac{l_2^2 - l_1^2 - X^2 - Y^2}{-2l_1\sqrt{X^2+Y^2}}\right)} \tag{E.12}$$

We continue with θ_2. We divide equation E.1 by equation E.2:

$$\frac{l_2\sin(\theta_1+\theta_2)}{l_2\cos(\theta_1+\theta_2)} = \frac{-X - l_1\sin\theta_1}{Y - l_1\cos\theta_1} \tag{E.13}$$

$$\tan(\theta_1+\theta_2) = \frac{-X - l_1\sin\theta_1}{Y - l_1\cos\theta_1} \tag{E.14}$$

$$\boxed{\theta_2 = \arctan\left(\frac{-X - l_1\sin\theta_1}{Y - l_1\cos\theta_1}\right) - \theta_1} \tag{E.15}$$

Lastly, from equation (3):

$$\boxed{\theta_3 = \Phi - \theta_1 - \theta_2} \tag{E.16}$$