A Geometric Approach for Generating Feasible Configurations of Robotic Manipulators

by

Filip Marić

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy:
Graduate Department of Institute for Aerospace Studies
University of Toronto

Abstract

A Geometric Approach for Generating
Feasible Configurations of Robotic Manipulators

Filip Marić
Doctor of Philosophy
Graduate Department of Institute for Aerospace Studies
University of Toronto
2023

Most robotic manipulators, and especially those designed with autonomous operation in mind, consist of a series of joints that rotate about a single axis, also known as revolute joints. These mechanisms give robotic manipulators the degrees of freedom and versatility similar to that of the human arm, which they are designed to outperform. However, this results in a geometry of motion or *kinematics* that makes all aspects of robotic manipulation challenging from a computational perspective. A major part of this challenge lies in the fact that computing joint configurations adhering to a specific set of constraints (i.e., gripper pose) is a non-trivial problem. The procedure of finding feasible joint configurations and the mathematical problem associated with it are known as *inverse kinematics* — a core part of motion planning, trajectory optimization, calibration and other important challenges in successfully performing robotic manipulation. In recent years, the overall decrease of computation time required to perceive and process environmental and proprioceptive information has helped realize the potential of robotic manipulation in dynamic environments. Concurrently, a new standard in manipulator design has emerged, where additional degrees of freedom are added in order to increase their overall dexterity and capacity for motion. These two developments have vastly increased the requirements for inverse kinematics algorithms, which are now expected to deal with infinite solution spaces and difficult, nonlinear constraints. On the other hand, the addition of degrees of freedom in recent robot designs has enabled algorithms to search for locally optimal configurations with respect to some performance criteria in an infinitely large solution space. This property has motivated approaches that leverage non-Euclidean geometries to replace conventional constraints and optimization criteria, thereby overcoming computational bottlenecks and common failure modes. The contributions presented in this thesis propose three such approaches, that aim to develop new ways of looking at the problems associated with inverse kinematics through the use of geometric representations that are not widely utilized in robotic manipulation.

Epigraph

Let no man ignorant of geometry enter here.

Inscription above Plato's Academy, Athens.

Acknowledgements

The research presented in this thesis has been made possible by the efforts and support of my family and friends. I feel indebted to all of you for being in my corner in both the worst and best of times.

Everyone in STARS Lab, past and present, thank you for making me feel welcome when I first visited the lab and continuing to do so every time I came back. Trevor, Oliver, Brandon, Valentin, Emmet, Lee and Olivier, thank you for all the formative discussions, intramural football and the late night debates in the Green Room. Special thanks to Matt — our joint research has been the most rewarding experience of my PhD and I truly feel fortunate to have worked with you. All my colleagues in LAMOR, especially Antea, Petki and Marta, thank you for all the laughs and everyday hijinks in C9-09 that made work a joy. Jura and Luka, thank you for the research discussions, outings, hundreds of football games and everything else. I would like to thank my DEC committee for providing valuable feedback throughout the program and helping me present my work in the best possible way.

Finally, I want to thank my supervisors, Ivan Petrović and Jonathan Kelly, for believing in me and making all of this happen. Organizing a joint PhD involved plenty of effort and flexibility on both sides and I hope I have made your efforts worthwhile. Hvala!

Contents

Notation

a	:	Symbols in this font are real scalars.
$\mathbf{a}$	:	Symbols in this font are real column vectors.
$\mathbf{A}$	:	Symbols in this font are real matrices.
$\mathcal{N}(\boldsymbol{\mu}, \mathbf{R})$	:	Normally distributed with mean $\boldsymbol{\mu}$ and covariance $\mathbf{R}$.
$E[\cdot]$	:	The expectation operator.
$\underrightarrow{\mathcal{F}}_a$	:	A reference frame in three dimensions.
$(\cdot)^\wedge$	:	An operator associated with the Lie algebra for rotations and poses. It produces a matrix from a column vector.
$(\cdot)^\vee$	:	The inverse operation of $(\cdot)^\wedge$.
$\mathbf{I}$	:	The identity matrix.
$\mathbf{0}$	:	The zero matrix.
$\mathbf{p}_a^b$	:	Point b (denoted by the superscript) and expressed in $\underrightarrow{\mathcal{F}}_a$ (denoted by the subscript).
$\mathbf{R}_{ab}$	:	The 3×3 rotation matrix that transforms vectors from $\underrightarrow{\mathcal{F}}_b$ to $\underrightarrow{\mathcal{F}}_a$: $\mathbf{p}_a = \mathbf{R}_{ab}\mathbf{p}_b$.
$\mathbf{T}_{ab}$	:	The 4×4 transformation matrix that transforms homogeneous points from $\underrightarrow{\mathcal{F}}_b$ to $\underrightarrow{\mathcal{F}}_a$: $\boldsymbol{p}_a = \mathbf{T}_{ab}\boldsymbol{p}_b$.

Chapter 1

Introduction

> One is not idle because one is absorbed. There is both visible and invisible labor. To contemplate is to toil, to think is to do. The crossed arms work, the clasped hands act. The eyes upturned to Heaven are an act of creation.
>
> Victor Hugo

As a starting point for discussing the central problem in this thesis, the reader is invited to consider that a significant portion of actions performed by living organisms involve some form of manipulation. Studies of ancient cytoskeletons suggest that flagella[1] evolved before eukaryotes, showing that interaction with the environment had developed prior to the last common ancestor of all plants, animals, and fungi. Further up the evolutionary tree, familiar insects such as bees, wasps and spiders have inspired countless research and engineering efforts through their evolved ability to construct intricate dwellings and traps for their prey. Perhaps even more impressive is the ants' ability to individually or collectively carry and manipulate objects whose weight and size is a multiple of their own. Chimpanzees, our close relatives, have demonstrated the ability of using rudimentary tools to hunt for insects, and gorillas have been observed using sticks to measure the depth of rivers and streams. Eventually, anatomically modern humans specialized in moving or altering their environments to create shelter, cultivate food and hunt wildlife on unprecedented scales. The humans' ability to efficiently manipulate resulted in a surplus of resources, triggering a positive-feedback loop leading to the first sedentary societies, increase in population and the rise of modern civilization [Harari, 2014].

While considering manipulation through an evolutionary lens gives a notion of its importance, it leaves the meaning of the word vague and context-dependent. In order to discuss the contributions proposed in this thesis, a working definition of what it means to manipulate is required. For example, an etymological definition states that manipulation refers to activities performed by hands. This definition is clearly too narrow when considering that the action itself may be performed by tools and other types of *end-effectors* (i.e., pincers, tentacles). The review by Mason [2018] discusses a broader definition of manipulation obtained by observing its ends and means — stating that manipulation is any action that takes place when an agent moves things other than itself through selective contact. However, even this may be too specific considering that contact may not always be necessary—a conductor may trigger motions of an entire orchestra just by moving her arms. Ultimately, we follow [Mason, 2018] in borrowing a suitable definition from the NASA robotics and autonomous systems roadmap

[1] a slender threadlike structure, especially a microscopic appendage that enables many protozoa, bacteria, etc. to swim.

effort [Miller, 2015], where manipulation is defined as "making an intentional change to the environment or the objects being manipulated". In this thesis, we explore manipulation from the perspective of motion and posture associated with the intentional change taking place, without specifying the means by which it is performed [2]. This general viewpoint allows us to develop tools and perspectives that can be utilized for more than one specific type of manipulation task.

Remark: Robotic Manipulators

Figure 1.1: Left: Depiction of serfdom in medival Europe, c. 1310. Right: a scene from the 1920 play 'Rossum's Universal Robots' by Karel Čapek.

Regardless of the means, it is well-established that difficult and repetitive manipulation tasks have historically been left to and performed by the lowest rungs of society. In exchange for essential resources such as food and shelter, medieval central European peasantry would perform menial field work, known as "rabota" in old Slavonic. It is exactly this word that forms the root of the word *robot*, introduced in the 1920 play "Rossum's Universal Robots" [Čapek, 1923], written by the Czech novelist and playwright Karel Čapek (1880-1938). Inspired by the Yiddish legend of the Golem and Mary Shelley's Frankenstein, the robots in Karel's play were servant workers created from organic matter, who resembled humans in every aspect other than the lack of a soul. For better or worse, it seems unlikely that robots such as those imagined by Karel are to become reality in the near future, as their development is barred by the limits of technology and ethical considerations. Despite this, generations of scientists, inventors and engineers have worked on creating *robotic manipulators* — machines designed to facilitate, and eventually eliminate, the modern-day "rabota".

Robotic manipulators were first utilized in controlled industrial environments, where they performed simple, repetitive tasks by following predefined joint trajectories [Mandfield, 1989]. For example, large industrial manipulators have completely replaced human workers in performing some of the most arduous tasks in the automotive industry, such as spray painting, welding and transporting parts. An advantage of industrial manipulators over traditional heavy machinery lies in their capacity for being reprogrammed to adapt to changes in task specifications, at higher cost efficiency in comparison to bespoke machines. Trained operators can manually reprogram these large robots, which generally involves changing key joint configurations (e.g., grasping or home states) and connecting them with an appropriate trajectory of joint velocities or accelerations computed offline [Sciavicco and Siciliano, 2012]. The fixed nature of these trajectories result in additional workplace safety considerations for human workers, which are usually addressed by placing robots in closed-off environments such as cages. Consequently, the overall research and engineering efforts in robotic manipulation have been focused on expanding the adaptation capabilities of these robots in order to enable safe cooperation and working

[2] e.g., whether the subject is moving freely or in contact with an object

with humans, with the ultimate goal of making them fully *autonomous*.

Most robotic manipulators, and especially those designed with autonomous operation in mind, consist of a series of joints that rotate about a single axis (i.e., revolute) joints. These mechanisms give robotic manipulators the degrees of freedom and versatility similar to that of the human arm, which they are designed to outperform. However, this results in a geometry of motion or *kinematics* that makes all aspects of robotic manipulation challenging from a computational perspective [Lynch and Park, 2017]. A major part of this challenge lies in the fact that computing joint configurations adhering to a specific set of constraints (i.e., gripper pose) is a non-trivial problem. The procedure of finding feasible joint configurations and the mathematical problem associated with it are known as *inverse kinematics*. Inverse kinematics is a core part of motion planning, trajectory optimization, calibration and other important challenges in successfully performing robotic manipulation. In recent years, the overall decrease of computation time required to perceive and process environmental and proprioceptive information has helped realize the potential of robotic manipulation in dynamic environments. Concurrently, a new standard in manipulator design has emerged, where additional degrees of freedom are added in order to increase their overall dexterity and capacity for motion. These two developments have vastly increased the requirements for inverse kinematics algorithms, which were now expected to deal with infinite solution spaces and difficult workspace constraints. For example, robots performing manipulation tasks alongside humans must now account for the time-varying geometry of the environment and agent locations in order to avoid collisions and other accidents. Moreover, the motion of these robots needs to be predictable and safe, while also ensuring there exists a capacity for rapid changes in trajectory by avoiding kinematic singularities.

The requirement of real-time environmental awareness and reactivity has introduced additional nonlinear and time-varying constraints and criteria for inverse kinematics algoithms to deal with, making the problem substantially more difficult. On the other hand, the addition of degrees of freedom in recent robot designs has enabled algorithms to search for locally optimal configurations with respect to some performance criteria in an infinitely large solution space. This property in particular has motivated approaches that leverage non-Euclidean geometries to replace conventional constraints and optimization criteria, thereby overcoming computational bottlenecks and common failure modes. The non-Euclidean nature of data in robotics can often be leveraged to improve performance of various algorithms. For example, it has been shown that using a Kalman filter that accounts for the Lie group geometry of rotation matrices improves the quality of state estimates for rigid object motion [Solà et al., 2020]. As a more relevant example, Lie group characterizations of rigid transformations expose an elegant procedure for constructing kinematic models without the need for bespoke parameterizations [Murray et al., 1994]. It has even been shown that assigning a Riemannian geometry to learned motion policies in tree-structured manipulation tasks allows for their seamless combination and transfer to the configuration space [Cheng et al., 2018]. The contributions presented in this thesis propose three such approaches, that aim to develop new ways of looking at the problems associated with finding feasible configurations (i.e., solving inverse kinematics) through the use of geometric representations that are not widely utilized in robotic manipulation.

Structure and Contributions

This dissertation is written with the goal of describing the proposed contributions in a (largely) self-contained manner, assuming the reader possess elementary knowledge of linear algebra and calculus. Chapter 2 provides an introduction to the mathematical concepts that serve as foundations for describing the core problems that the contributions proposed in this thesis address. Concepts related to characterizing robot motion and configurations are introduced in Chapter 3, where conventional methods for deriving mathematical models of robot

kinematics are described. Crucially, this chapter also formally defines and categorizes inverse kinematics—the problem of finding feasible configurations or configuration changes, that serves as a thread connecting all proposed contributions.

The first contribution of this thesis is proposed in Chapter 4, where we derive an index for maintaining numerical stability in optimization formulations of inverse kinematics problems. This index relies on Riemannian manfiolds to give a geometric interpretation of numerical stability that helps avoid failure modes characteristic of conventional approaches. The theme of geometric interpretations is continued in the second contribution proposed in Chapter 5. We derive an alternative kinematic model applicable to a large class of robotic manipulators using distance geometry, as opposed to the conventionally used joint angles. Using this model, we are able to define inverse kinematics as low-rank distance matrix completion, allowing the use of Riemannian optimization to outperform standard inverse kinematics approaches in highly constrained settings. Finally, in Chapter 6, we leverage a graphical description of our distance-geometric robot kinematics model to develop a learned generalizable inverse kinematics solver based on graph neural networks. Moreover, we use a generative variational autoencoder architecture to model the entire solution set of a given inverse kinematics problem as a probability distribution.

Remark: Contributions

The three main contributions proposed in this thesis can be formally summarized as follows:

- A measure of proximity to kinematic singularities inherent to robotic manipulators, derived from Riemannian geometry;
- A method for solving the inverse kinematics problem through a distance-based characterization that enables the use of novel optimization methods for finding solutions in constrained workspaces; and
- A method for designing learned models based on graph neural networks for solving the inverse kinematics problem for a wide class of robotic manipulators.

Chapter 2

Mathematical Foundations

> The geometry of the place was all wrong. One could not be sure that the sea and the ground were horizontal.
>
> H.P. Lovecraft

This chapter provides an overview the mathematical material required to derive and understand the contributions of the thesis. The first section gives an introduction to the theory of differential geometry, which forms the basis for the novel geometric approaches described in later chapters. The second section reviews spatial geometry that lies at the core of rigid-body kinematics used in modeling the motion of robotic manipulators. The third section introduces methods for reasoning about spatial configurations using distances and describes the distance geometry problem. Finally, the fourth section discusses essential concepts in deep learning that are relevant to the final contribution.

2.1 Differential Geometry

Differential geometry is the mathematical discipline that studies the geometric properties of differentiable curves and surfaces, otherwise known as differentiable manifolds. In this thesis, we study and employ manifolds that are *smooth*, meaning that they are infinitely differentiable at all points. Informally, an important property of smooth manifolds as topological spaces is that they locally (i.e., within the neighbourhood of each point) 'look like' a Euclidean space. An often-cited example of a manifold is the 2-sphere, which resembles a plane from the perspective of a surface-dwelling observer.

2.1.1 Differentiable and Smooth Manifolds

Analysis on manifolds is analogous in some ways to navigation over the surface of the Earth (an oblate spheroid) and uses similar notions such as charts and atlases, examples of which are shown in Fig. 2.1.

Definition 1 (Chart and Atlas)**.** *A chart on the d-dimensional manifold $\mathcal{M}$ is a diffeomorphic mapping $\phi : U \to \tilde{U}$ from an open set $U \subset \mathcal{M}$ to an open set $\tilde{U} \subseteq \mathbb{R}^d$. A C^k-atlas of $\mathcal{M}$ is a family of charts $(\phi_i)_{i=1,\cdots,N}$ with $\phi_i : U_i \to \tilde{U}_i$ such that*

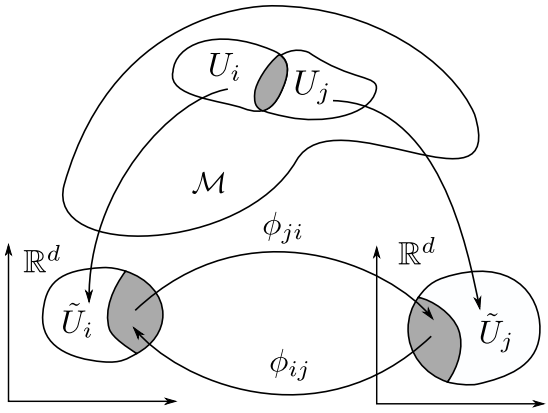


Figure 2.1: Visualization of charts ϕ_i and ϕ_j on a d-dimensional manifold $\mathcal{M}$. Note that $\phi_{ij} = \phi_i \circ \phi_j^{-1}$ and $\phi_{ji} = \phi_j \circ \phi_i^{-1}$.

- *For each point $\boldsymbol{\Sigma} \in \mathcal{M}$ there exists $i \in 1, \cdots, N$ such that $\boldsymbol{\Sigma} \in U_i$,*
- *For each pair $i, j \in 1, \ldots, N$ where $U_i \cap U_j \neq \emptyset$, the composition $\phi_i \circ \phi_j^{-1} : \phi_j(U_i \cap U_j) \to \mathbb{R}^d$ belongs to the class of differentiability C^k.*

Defining a chart over a (sub-)space allows one to locally 'navigate' such a space as if it were Euclidean. This allows us to formally define a d-dimensional differentiable manifold $\mathcal{M}$ of class C^k as a set with at least one C^k-atlas $(\phi_i)_{i=1,\cdots,N}$, where $\phi_i : U_i \to \tilde{U}_i, \tilde{U}_i \subseteq \mathbb{R}^d$. Crucially, this differentiable structure enables computation of derivatives of curves defined on the manifold itself. Using this definition, we can also formally define smooth manifolds.

Definition 2 (Smooth Manifold)**.** *A d-dimensional smooth manifold $\mathcal{M}$ is a set with at least one C^∞-atlas $(\phi_i)_{i=1,\cdots,N}$, where $\phi_i : U_i \to \tilde{U}_i, \tilde{U}_i \subseteq \mathbb{R}^d$. In other words, a smooth manifold is a differentiable manifold of class C^∞.*

In this thesis we employ a variety of smooth manifolds to represent sets describing the structure and properties of robotic manipulators, allowing us to perform calculus in a geometrically appropriate manner.

Tangent Vectors and the Tangent Space

Informally, a tangent vector $\boldsymbol{v}$ at some point $\boldsymbol{\Sigma}$ on the manifold (Fig. 2.2) can be thought of as the *velocity* along a curve passing through that point [1]. There may be an infinite number of curves with identical velocities (i.e., tangent vectors) at a point on the manifold, which can be disambiguated by defining a formal equivalence between them. Specifically, two curves $\gamma_1, \gamma_2 : t \in [-1, 1] \to \mathcal{M}$ with $\gamma_1(0) = \gamma_2(0) = \boldsymbol{\Sigma} \in \mathcal{M}$ are said to be equivalent at 0 if and only if $\frac{d}{dt}(\phi \circ \gamma_1) = \frac{d}{dt}(\phi \circ \gamma_2)$ at 0 for a chart ϕ defined on $\boldsymbol{\Sigma}$.

A tangent vector is formally defined as an *equivalence class* of curves at a given point on the manifold.

Definition 3 (Tangent vector)**.** *A tangent vector $\boldsymbol{v}$ of $\mathcal{M}$ at $\boldsymbol{\Sigma} \in \mathcal{M}$ is an equivalence class of curves initialized at $\boldsymbol{\Sigma}$.*

[1] Thinking in terms of velocities of time-parameterized curves makes the concepts in this section more intuitive. In general, curve parameterizations do not need to carry any physical meaning.

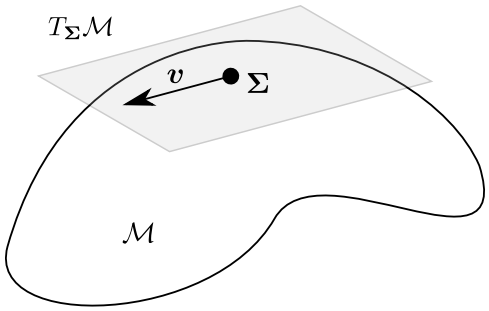


Figure 2.2: An example of a tangent vector v belonging to a tangent space $\mathcal{T}_{\Sigma}\mathcal{M}$ at Σ on the manifold $\mathcal{M}$.

Note that tangent vectors are by definition independent of the particular chart used to define the equivalence class of curves. The set of all tangent vectors to all curves at a given point spans a d-dimensional subspace, known as a *tangent space*, that approximates $\mathcal{M}$ to the first order.

Definition 4 (Tangent Space)**.** *The tangent space of $\mathcal{M}$ at Σ, denoted by $\mathcal{T}_{\Sigma}\mathcal{M}$, is the set of all tangent vectors at Σ.*

Unlike curves in Euclidean space, the vectors tangent to curves on manifolds do not necessarily share a global frame of reference, and are only meaningful in the context of the particular point on the manifold they are associated with.

Geodesics

A smooth manifold may be equipped with an *affine connection* that joins nearby tangent spaces, enabling the differentiation of tangent vectors with respect to parameterized curves on the manifold. The existence of an affine connection allows for the *parallel transport* of tangent vectors along curves such that the vectors remain parallel with respect to the connection. Crucially, this property makes it possible to find curves, known as *geodesics*, that generalize the notion of straight lines to differentiable manifolds. Geodesics are of special interest

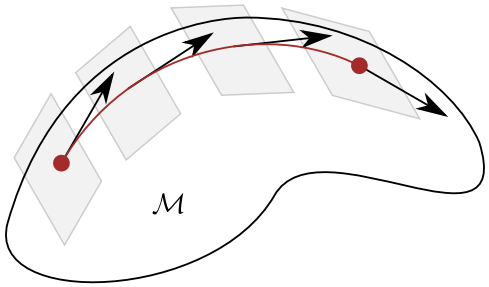


Figure 2.3: The red curve on the manifold $\mathcal{M}$ is known as a geodesic, as its tangent vetors are parallel with respect to the affine connection.

because they may[2] also define the shortest path between two points on the manifold, much like straight lines do in Euclidean space.

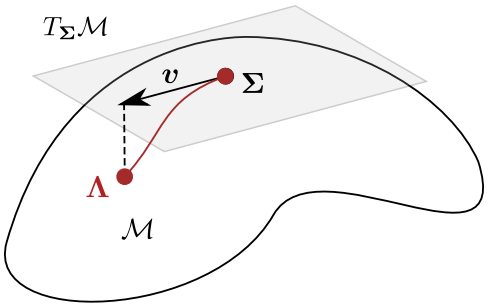


Figure 2.4: The exponential map maps a tangent space vector $\boldsymbol{v} \in T_{\boldsymbol{\Sigma}}\mathcal{M}$ to $\boldsymbol{\Lambda} = \mathrm{Exp}_{\boldsymbol{\Sigma}}\boldsymbol{v}$.

Exponential and Logarithmic Maps

Let $\boldsymbol{\Sigma}$ be a point on the manifold and $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{\Sigma}}\mathcal{M}$. It can be shown that there exists one and only one geodesic $\gamma_{(\boldsymbol{\Sigma},\boldsymbol{v})}(t)$, which is a curve γ on the manifold starting at $\boldsymbol{\Sigma}$ with the direction $\dot{\gamma}(0) = \boldsymbol{v}$. Under the assumption that the manifold is *geodesically complete*[3], the vector $\boldsymbol{v}$ can be mapped to a unique point on the manifold reached after unit time $t = 1$ by the geodesic $\gamma_{(\boldsymbol{\Sigma},\boldsymbol{v})}(t)$. This mapping is known as the *exponential map*

$$\mathrm{Exp}_{\boldsymbol{\Sigma}} : \mathcal{T}_{\boldsymbol{\Sigma}}\mathcal{M} \to \mathcal{M}\,. \tag{2.1}$$

In addition, we may also define an inverse mapping, known as the *logarithmic map*

$$\mathrm{Log}_{\boldsymbol{\Sigma}} : \mathcal{M} \to \mathcal{T}_{\boldsymbol{\Sigma}}\mathcal{M} \tag{2.2}$$

that takes a point on the manifold to a tangent vector $\boldsymbol{v} \in \mathcal{T}_{\boldsymbol{\Sigma}}\mathcal{M}$ corresponding to a geodesic that reaches that point in a unit time $t = 1$ starting from the point $\boldsymbol{\Sigma} \in \mathcal{M}$. The logarithmic map need not be unique, since there may be multiple geodesics reaching a given point on a manifold (e.g., geodesics connecting antipodal points on a 2-sphere.)

Remark: Notation
The exponential and logarithmic maps $\mathrm{Exp}_{\boldsymbol{\Sigma}}$ and $\mathrm{Log}_{\boldsymbol{\Sigma}}$ correspond to the matrix functions $\exp$ and $\log$ in some groups. However, this is not always the case (e.g., in Chapter 4) and thus the notation enforces this distinction.

2.1.2 Riemannian Manifolds

Deriving a variety of geometric concepts (e.g., length of a curve) on a smooth manifold requires a particular type of inner product, known as a *Riemannian metric*, to be defined on its tangent space. Manifolds equipped with such a metric are known as *Riemannian manifolds*.

Definition 5 (Riemannian Manifold [Lee, 2018]). *A Riemannian manifold $\mathcal{M}$ is a smooth manifold equipped with a Riemannian metric, that is, a positive-definite inner product $\langle \boldsymbol{v}_1, \boldsymbol{v}_2 \rangle_{\boldsymbol{\Sigma}}$ on the tangent space $T_{\boldsymbol{\Sigma}}\mathcal{M}$ at each point $\boldsymbol{\Sigma} \in \mathcal{M}$ that varies smoothly from point to point.*

[2]Going the "long way round" on a great circle between two points on a sphere is a geodesic but not the shortest path between the points.
[3]The property where any geodesic on the manifold can be followed indefinitely. For example, a 2-sphere is geodesically complete, whereas a punctured plane $\mathbb{R}^2 \setminus \{0\}$ is not.

Notably, every smooth manifold carries multiple Riemannian metrics, which allow the assignment of a notion of magnitude to the velocity along curves.

Length

A Riemannian metric naturally gives rise to a norm $\|\boldsymbol{v}\|_{\boldsymbol{\Sigma}} = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle_{\boldsymbol{\Sigma}}} : T_{\boldsymbol{\Sigma}}\mathcal{M} \to \mathbb{R}$. It follows that we can compute the length of a curve $\gamma(t)$ on the manifold by integrating the norm of its tangent vectors

$$\mathcal{L}(\gamma) = \int_0^1 \|\dot{\gamma}(t)\|_{\gamma(t)} dt = \int_0^1 \langle \dot{\gamma}(t), \dot{\gamma}(t) \rangle_{\gamma(t)}^{\frac{1}{2}} dt \,. \tag{2.3}$$

This is analogous to integrating velocities in order to compute the total length of a path through Euclidean space.

Riemannian Distance

The ability to compute the lengths of curves on smooth manifolds also opens the possibility of defining a measure of distance. Intuitively, the distance between two points $\boldsymbol{\Sigma}, \boldsymbol{\Lambda} \in \mathcal{M}$ is simply the length of the shortest curve connecting them (e.g., a straight line in Euclidean space)

$$\text{dist}(\boldsymbol{\Sigma}, \boldsymbol{\Lambda}) = \min_{\gamma(0)=\boldsymbol{\Sigma}, \gamma(1)=\boldsymbol{\Lambda}} \mathcal{L}(\gamma) \,.$$

On smooth manifolds, this is exactly the length of the geodesic with these elements as endpoints, also known as the *Riemannian distance*.

The fundamental theorem of Riemannian geometry states that all Riemannian manifolds admit a Levi-Cvita affine connection, where all geodesics γ are constant velocity curves (i.e., $\nabla_{\dot{\gamma}} \dot{\gamma} = 0$). Therefore, a geodesic connecting $\gamma(0) = \boldsymbol{\Sigma}$ and $\gamma(1) = \boldsymbol{\Lambda}$ on a manifold may be uniquely defined by a tangent vector $\dot{\gamma}(0) = \boldsymbol{v} \in T_{\boldsymbol{\Sigma}}\mathcal{M}$. This gives us the Riemannian distance

$$\text{dist}(\boldsymbol{\Sigma}, \boldsymbol{\Lambda}) = \int_0^1 \|\dot{\gamma}(t)\|_{\gamma(t)} dt = \int_0^1 \|\dot{\gamma}(0)\|_{\gamma(0)} dt = \|\boldsymbol{v}\|_{\boldsymbol{\Sigma}}, \tag{2.4}$$

where the norm $\|\cdot\|_{\boldsymbol{\Sigma}}$ is derived from the Riemannian metric and exactly matches the lenght of the geodesic [4]. Note that Riemannian manifolds are not neccesarily geodesically complete, in which case the exponential map only locally approximates the manifold near $\boldsymbol{\Sigma}$. Consequently, it cannot be assumed that the Riemannian distance in Eq. (2.4) is differentiable with respect to $\boldsymbol{\Lambda}$ for all Riemannian manifolds.

2.1.3 Matrix Lie Groups

Spatial transforms used to describe rigid-body motions are commonly represented as elements of several key *Matrix Lie groups*.

Definition 6 (Matrix Lie group)**.** *A matrix Lie group is a differentiable manifold $\mathcal{M}$ with an operation $\cdot$ that satisfies the four group axioms:*

- *Closure:* $(\boldsymbol{\Sigma} \cdot \boldsymbol{\Lambda}) \in \mathcal{M} \; \forall \, \boldsymbol{\Sigma}, \, \boldsymbol{\Lambda} \in \mathcal{M}$

[4] Intuitively, moving at a constant velocity for a unit time traces a line with a lenght equal to that velocity.

- *Associativity:* $(\boldsymbol{\Sigma} \cdot \boldsymbol{\Lambda}) \cdot \boldsymbol{\Upsilon} = \boldsymbol{\Sigma} \cdot (\boldsymbol{\Lambda} \cdot \boldsymbol{\Upsilon}) \; \forall \boldsymbol{\Sigma}, \boldsymbol{\Lambda}, \boldsymbol{\Upsilon} \in \mathcal{M}$
- *Identity:* $\exists \mathbf{I} \in \mathcal{M} : \boldsymbol{\Sigma} \cdot \mathbf{I} = \mathbf{I} \cdot \boldsymbol{\Sigma} = \boldsymbol{\Sigma} \; \forall \boldsymbol{\Sigma} \in \mathcal{M}$
- *Invertibility:* $\forall \boldsymbol{\Sigma} \; \exists \boldsymbol{\Sigma}^{-1} : \boldsymbol{\Sigma} \cdot \boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}^{-1} \cdot \boldsymbol{\Sigma} = \mathbf{I}$

Each matrix Lie group has an associated *Lie algebra*; a vector space whose elements are identified with vectors in $\mathbb{R}^m$, where m is the number of degrees of freedom of the differentiable manifold $\mathcal{M}$.

Definition 7 (Lie algebra)**.** *The tangent space at the identity $T_{\mathbf{I}}\mathcal{M}$ of the Lie group $\mathcal{M}$ is known as the Lie algebra $\mathfrak{m} \triangleq T_{\mathbf{I}}\mathcal{M}$.*

A member of the tangent space at any element in the group may be transformed to the tangent space at the identity element using a linear map known as the *adjoint*. The resulting Lie algebra elements can then be exactly converted to elements of the Lie group using the exponential map $\mathrm{Exp} : \mathfrak{m} \to \mathcal{M}$. The structure of the tangent space of multiplicative Lie groups (e.g., rotation matrices) is revealed by the identity

$$\boldsymbol{\Sigma}^{-1}\dot{\boldsymbol{\Sigma}} = -\dot{\boldsymbol{\Sigma}^{-1}}\boldsymbol{\Sigma}\,, \tag{2.5}$$

which follows from taking the time derivative of the invertability property of groups.

2.2 Spatial Geometry

The position and orientation of a rigid body is defined using at least two coordinate frames, such as those shown in Fig. 2.5. Following the naming convention in [Lynch and Park, 2017], the *space frame* $\underrightarrow{\mathcal{F}}_s$ is attached to a stationary reference point in the operating area and serves as a reference frame. The *body frame* $\underrightarrow{\mathcal{F}}_b$ is stationary relative to the rigid body and it usually coincides with some important point, such as the center of mass. In robotics applications it is generally assumed both frames are right-handed.

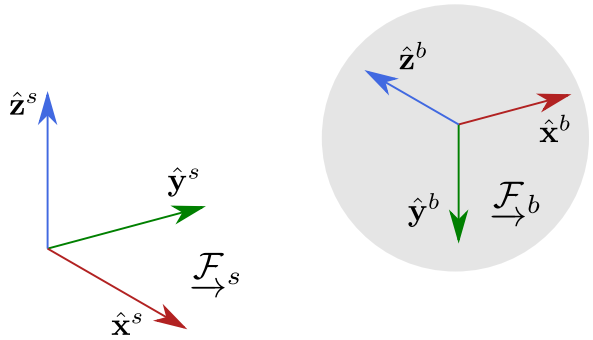


Figure 2.5: The space and body coordinate frames. The vectors $\hat{\mathbf{x}}^*, \hat{\mathbf{y}}^*, \hat{\mathbf{z}}^*$ of each frame form the basis $(\mathbf{e}_1^*, \mathbf{e}_2^*, \mathbf{e}_3^*)$ of the frames denoted by the superscripts s and b.

Any point $\mathbf{p}$ in $\underrightarrow{\mathcal{F}}_b$ can be expressed in $\underrightarrow{\mathcal{F}}_s$ using the linear identity

$$\mathbf{p}_s = \underbrace{\begin{bmatrix} \hat{\mathbf{x}}_s^b & \hat{\mathbf{y}}_s^b & \hat{\mathbf{z}}_s^b \end{bmatrix}}_{\mathbf{R}_{sb}} \mathbf{p}_b + \mathbf{p}_s^b\,, \tag{2.6}$$

where $\hat{\mathbf{x}}_s^b, \hat{\mathbf{y}}_s^b, \hat{\mathbf{z}}_s^b$ are the axes of $\underrightarrow{\mathcal{F}}_b$ expressed in $\underrightarrow{\mathcal{F}}_s$, as denoted by the subscript. Together with the translation vector $\mathbf{p}_s^b$ that represents the position of the origin of $\underrightarrow{\mathcal{F}}_b$, the 3×3 *rotation matrix* $\mathbf{R}_{sb}$ completes a full description of the pose (i.e., position and orientation) of $\underrightarrow{\mathcal{F}}_b$ relative to $\underrightarrow{\mathcal{F}}_s$.

Remark: Notation
In the remainder of this thesis, when the frame or point label is not relevant, the related subscript and/or superscript will be dropped. For example, in Eq. (2.6), $\mathbf{p}_s$ and $\mathbf{p}_b$ have no superscript, implying that they refer to the same point. On the other hand, the superscript in $\mathbf{p}_s^b$ refers to the root of the coordinate frame $\underrightarrow{\mathcal{F}}_b$, with the subscript denoting that it is expressed in $\underrightarrow{\mathcal{F}}_s$.

From Eq. (2.6) it follows that vectors in a third coordinate frame $\underrightarrow{\mathcal{F}}_c$ defined in $\underrightarrow{\mathcal{F}}_b$ can be expressed in $\underrightarrow{\mathcal{F}}_s$ using the translation and rotation matrices obtained using a pair of linear identities

$$\begin{aligned} \mathbf{p}_s^c &= \mathbf{R}_{sb}\mathbf{p}_b^c + \mathbf{p}_s^b\,, \\ \mathbf{R}_{sc} &= \mathbf{R}_{sb}\underbrace{\begin{bmatrix}\hat{\mathbf{x}}_b^c & \hat{\mathbf{y}}_b^c & \hat{\mathbf{z}}_b^c\end{bmatrix}}_{\mathbf{R}_{bc}}. \end{aligned} \tag{2.7}$$

Generally, any pair $(\mathbf{R}, \mathbf{p})$ can represent both the pose of a coordinate frame in space and an arbitrary spatial transformation of points expressed in that frame to a parent frame. Moreover, any pair $(\mathbf{R}, \mathbf{p})$ also admits a matrix representation, whereby Eq. (2.7) can be compactly written as

$$\underbrace{\begin{bmatrix}\mathbf{R}_{sc} & \mathbf{p}_s^c \\ \mathbf{0}^\intercal & 1\end{bmatrix}}_{\mathbf{T}_{sc}} = \underbrace{\begin{bmatrix}\mathbf{R}_{sb} & \mathbf{p}_s^b \\ \mathbf{0}^\intercal & 1\end{bmatrix}}_{\mathbf{T}_{sb}}\underbrace{\begin{bmatrix}\mathbf{R}_{bc} & \mathbf{p}_b^c \\ \mathbf{0}^\intercal & 1\end{bmatrix}}_{\mathbf{T}_{bc}}, \tag{2.8}$$

where $\mathbf{T}_{bc}, \mathbf{T}_{sb}, \mathbf{T}_{sc}$ are known as *homogeneous transformation matrices*. This representation is often used in robotics due to its convenient properties.

2.2.1 Rotations

Rotation matrices are elements of the *special orthogonal group* $\mathrm{SO}(3)$, a matrix Lie group defined as

$$\mathrm{SO}(3) = \left\{\mathbf{R} \in \mathbb{R}^{3\times 3} |\ \mathbf{R}^\intercal\mathbf{R} = \mathbf{I}, \det(\mathbf{R}) = 1\right\}, \tag{2.9}$$

with the standard matrix product as the group operation. The time derivative of the group invertability property in Eq. (2.5) for rotations shows that the tangent space of $\mathrm{SO}(3)$ consists of matrices with a skew symmetric structure

$$\boldsymbol{\omega}^\wedge = \begin{bmatrix}0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0\end{bmatrix}, \tag{2.10}$$

associated with the vector space of angular velocities $\boldsymbol{\omega} \in \mathbb{R}^3$ via the linear $\wedge$ (wedge) operator. The tangent space at the identity element $\mathbf{R} = \mathbf{I}$ is also the Lie algebra $\mathfrak{so}(3) \triangleq \mathcal{T}_{\mathbf{I}}\mathrm{SO}(3)$ associated with $\mathrm{SO}(3)$.

Exponential Map

Generalized $\mathfrak{so}(3)$ Lie algebra elements are identified with *rotation vectors* $\boldsymbol{\phi} \triangleq \boldsymbol{\omega} t$, associated with rotations achieved by moving at an angular velocity $\boldsymbol{\omega}$ for a time t. From Eq. (2.5) it follows that $\boldsymbol{\omega}^\wedge = \mathbf{R}^{-1}\dot{\mathbf{R}} = -\dot{\mathbf{R}^{-1}}\mathbf{R}$. By solving the differential equation $\dot{\mathbf{R}} = \mathbf{R}\boldsymbol{\omega}^\wedge$, we arrive at the mapping

$$\mathbf{R}(\boldsymbol{\phi}) = \mathrm{Exp}(\boldsymbol{\phi}) = \exp\left(\boldsymbol{\phi}^\wedge\right) = \sum_{n=0}^{\infty} \boldsymbol{\phi}^{\wedge n} \frac{1}{n!} \in \mathrm{SO}(3)\,, \tag{2.11}$$

which is the surjective exponential map $\mathrm{Exp} : \mathfrak{so}(3) \to \mathrm{SO}(3)$ of Lie algebra elements to the manifold of rotation matrices, whereby the standard matrix exponential denoted by $\exp$.

Alternatively, the rotation vector $\boldsymbol{\phi} \triangleq \hat{\mathbf{s}}\theta$ can be parameterized as a rotation by an angle $\theta \in \mathbb{R}$ about an axis $\hat{\mathbf{s}} = \left\{\hat{\mathbf{s}} \in \mathbb{R}^3 \middle| \|\hat{\mathbf{s}}\| = 1\right\}$. The exponential map in Eq. (2.11) then simplifies to

$$\mathbf{R}(\theta, \hat{\mathbf{s}}) = \mathrm{Exp}(\theta\hat{\mathbf{s}}) = \mathbf{I} + \sin\theta\hat{\mathbf{s}}^\wedge + (1 - \cos\theta)(\hat{\mathbf{s}}^\wedge)^2 \in \mathrm{SO}(3)\,, \tag{2.12}$$

which is also known as Rodrigues' rotation formula. This *angle-axis* parameterization of rotations is often used in robotics due to its relative simplicity and interpretability.

Logarithmic Map

For a rotation matrix generated by rotating about some axis by an angle $\theta \in [0, \pi)$, the associated rotation vector can be retrieved using the logarithmic map,

$$\boldsymbol{\phi}^\wedge = \mathrm{Log}(\mathbf{R})^\wedge = \frac{\mathbf{R} - \mathbf{R}^\mathsf{T}}{2\sin\theta}\,, \tag{2.13}$$

where the angle θ is obtained by taking

$$\theta = \arccos\frac{\mathrm{tr}\,(\mathbf{R}) - 1}{2}\,. \tag{2.14}$$

Note that the logarithmic map is not an inverse of the exponential map for angles larger than π, as it will not return a rotation vector representing the minimal rotation to the identity.

2.2.2 Transforms and Twists

Homogeneous transformation matrices or *transforms* are elements of the *Special Euclidean Group* $\mathrm{SE}(3)$, a matrix Lie group defined as

$$\mathrm{SE}(3) = \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0}^\mathsf{T} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \middle| \mathbf{R} \in \mathrm{SO}(3), \mathbf{p} \in \mathbb{R}^3 \right\}. \tag{2.15}$$

The properties of $\mathrm{SE}(3)$ are derived by combining properties of the Euclidean element $\mathbf{p}$ with those of $\mathrm{SO}(3)$ special orthogonal group. Observing the time derivative identity in Eq. (2.5) for this Lie group, it can be shown that the tangent space of $\mathrm{SE}(3)$ consists of matrices with a fixed structure

$$\dot{\boldsymbol{\xi}}^\wedge = \begin{bmatrix} \boldsymbol{\omega}^\wedge & \boldsymbol{v} - \boldsymbol{\omega} \times \mathbf{p} \\ \mathbf{0}^\mathsf{T} & 0 \end{bmatrix}\,, \tag{2.16}$$

also known as *twists*. Twists identify with the vector space of generalized velocities $\dot{\boldsymbol{\xi}} = \begin{bmatrix}\boldsymbol{\omega}^\intercal & \dot{\boldsymbol{\rho}}^\intercal\end{bmatrix}^\intercal \in \mathbb{R}^6$ through the linear $\wedge$ operator [5]. The tangent space at the identity element $\mathbf{T} = \mathbf{I}_4$ is also the Lie algebra $\mathfrak{se}(3) \triangleq \mathcal{T}_{\mathbf{I}}\mathrm{SE}(3)$ associated with $\mathrm{SE}(3)$.

Exponential Map

Lie algebra elements $\boldsymbol{\xi}^\wedge \in \mathfrak{se}(3)$ are identified with *exponential coordinates* $\boldsymbol{\xi} \triangleq \begin{bmatrix}\boldsymbol{\phi}^\intercal & \boldsymbol{\rho}^\intercal\end{bmatrix}^\intercal$. Interpreting these vectors as representations of transforms resulting from movement at a constant generalized velocity $\dot{\boldsymbol{\xi}}$, we arrive at

$$\mathbf{T} = \mathrm{Exp}(\boldsymbol{\xi}) = \begin{bmatrix}\mathrm{Exp}(\boldsymbol{\phi}) & \mathbf{J}_l(\boldsymbol{\phi})\boldsymbol{\rho} \\ \mathbf{0}^\intercal & 1\end{bmatrix} \in \mathrm{SE}(3) \tag{2.17}$$

which is the surjective exponential map $\mathrm{Exp} : \mathfrak{se}(3) \to \mathrm{SE}(3)$ of Lie algebra elements to the manifold of homogeneous transformation matrices. Note that $\mathbf{J}_l$ is the $\mathrm{SO}(3)$ *left-Jacobian* matrix

$$\mathbf{J}_l(\boldsymbol{\phi}) = \left(\mathbf{I}_3 + \frac{1-\cos\theta}{\theta^2}\hat{\mathbf{s}}^\wedge + \frac{\theta - \sin\theta}{\theta^3}(\hat{\mathbf{s}}^\wedge)^2\right), \tag{2.18}$$

where the axis $\hat{\mathbf{s}}$ and angle θ correspond to the rotation vector $\boldsymbol{\phi}$.

The Mozzi-Chasels theorem states that any transform may be parameterized as a rotation about and translation along a *screw* axis

$$\boldsymbol{\mathcal{S}} = \begin{bmatrix}\hat{\mathbf{s}} \\ -\hat{\mathbf{s}} \times \mathbf{q} + h\hat{\mathbf{s}}\end{bmatrix}, \tag{2.19}$$

at position $\mathbf{q}$ with a direction $\mathbf{R}$ and pitch h [Selig, 2005]. Analogously to the angle-axis parameterization of rotations, exponential coordinates may also be parameterized with $\boldsymbol{\xi} \triangleq \boldsymbol{\mathcal{S}}\theta$ representing a screw motion by an angle θ about this axis. The exponential map in Eq. (2.17) is then expressed as

$$\mathbf{T}(\theta, \boldsymbol{\mathcal{S}}) = \mathrm{Exp}(\boldsymbol{\mathcal{S}}\theta) = \begin{bmatrix}\mathrm{Exp}(\hat{\mathbf{s}}\theta) & \mathbf{J}_l(\boldsymbol{\phi})(-\hat{\mathbf{s}} \times \mathbf{q} + h\hat{\mathbf{s}}) \\ \mathbf{0}^\intercal & 1\end{bmatrix}. \tag{2.20}$$

This identity will be important when modeling degrees of freedom of robotic mechanisms.

Logarithmic Map

For $\theta \in [0, \pi)$, exponential coordinates $\boldsymbol{\xi}$ can be recovered using the logarithmic map

$$\boldsymbol{\xi}^\wedge = \mathrm{Log}(\mathbf{T})^\wedge = \begin{bmatrix}\boldsymbol{\phi}^\wedge & {\mathbf{J}_l}^{-1}(\boldsymbol{\phi})\mathbf{p} \\ \mathbf{0}^\intercal & 0\end{bmatrix}, \tag{2.21}$$

where $\boldsymbol{\phi}^\wedge$ is computed via Eq. (2.13) and Eq. (2.14).

Adjoint

Control and estimation applications often require exponential coordinates to be represented in different frames. This equates to transporting tangent space vectors to a different point on the manifold using the adjoint operator.

[5] Note that $\dot{\boldsymbol{\rho}} = \boldsymbol{v} - \boldsymbol{\omega} \times \mathbf{p}$ corresponds to the velocity of a point at the space frame origin on an infinitely large moving body, expressed in the space frame. Further, $\boldsymbol{\omega}$ is the angular velocity expressed in the space frame.

For elements of $\mathrm{SE}(3)$ this is a linear operation

$$\mathrm{Ad}\left(\mathbf{T}\right) = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{p}^{\wedge}\mathbf{R} & \mathbf{R} \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{2.22}$$

that transforms exponential coordinates $\boldsymbol{\xi}$ to a frame $\mathbf{T}$

$$\boldsymbol{\xi}_a = \mathrm{Ad}\left(\mathbf{T}_{ab}\right)\boldsymbol{\xi}_b\,.$$

This identity can also be used to transform generalized velocities.

Linear and Angular Velocities

In some cases, it is necessary to transform twists $\dot{\boldsymbol{\xi}}$ into the more intuitive angular and linear velocities $\boldsymbol{\omega}, \boldsymbol{v}$. From Eq. (2.16) it follows that these velocities are related in the space frame $\underrightarrow{\mathcal{F}}_s$ via the identity

$$\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_{3\times 3} \\ \mathbf{p}^{\wedge} & \mathbf{I}_3 \end{bmatrix}^{-1} \dot{\boldsymbol{\xi}}\,. \tag{2.23}$$

where $\mathbf{t}$ is the position of the body frame $\underrightarrow{\mathcal{F}}_b$ origin. This transformation is useful in cases where velocities are defined by an external source for applications such as teleoperation or control.

2.3 Distance Geometry

The theory of distance geometry plays an important role in the development of computational methods for analyzing problems defined using inter-point distances [Liberti et al., 2014]. This elegant theoretical framework is often applied to solve a diverse set of problems spanning computational chemistry [Havel, 2002], signal processing [Ding et al., 2010], and acoustics [Dokmanic et al., 2015]. Liberti et al. [Liberti et al., 2014] present a detailed taxonomy of distance geometry problems, which can be collectively stated as follows:

Problem 1 (Distance Geometry Problem)**.** *Given an integer $K > 0$, a set of vertices V, and a simple undirected graph $G = (V, E)$ whose edges $\{u, v\} \in E$ (where $u, v \in V$) are assigned nonnegative weights*

$$\{u, v\} \mapsto d_{u,v} \in \mathbb{R}_+\,,$$

find a function $p : V \to \mathbb{R}^K$ such that the Euclidean distances between pairs match the assigned weights

$$\forall\,\{u, v\} \in E\,, \quad \|p(u) - p(v)\| = d_{u,v}. \tag{2.24}$$

The function $p : V \to \mathbb{R}^K$ is also known as a *realization* of the graph G. Any realization p of G maps all the vertices in V to a collection of points $\mathbf{P} \in \mathbb{R}^{|V|\times K}$, where each row is the position $\mathbf{p}^u = p(u) \in \mathbb{R}^K$ of the point corresponding to vertex $u \in V$.

In some cases, we may wish to reduce the number of possible realizations by constraining a subset of inter-point distances (i.e., edge weights) to some interval. Consequently, we can extend Problem 1 such that the edges in E are weighted by positive intervals, resulting in the more general *interval distance geometry problem* [Liberti et al., 2014]:

Problem 2 (Interval Distance Geometry Problem). *Given an integer $K > 0$ and a simple undirected graph $G = (V, E)$ whose edges $\{u, v\} \in E$ are weighted by intervals*

$$\{u, v\} \mapsto [d^-_{u,v}, d^+_{u,v}] \subseteq \mathbb{R}_+,$$

find a realization in $\mathbb{R}^K$ such that Euclidean distances between pairs belong to the edge intervals

$$\forall \{u, v\} \in E\,, \quad \|p(u) - p(v)\| \in [d^-_{u,v}, d^+_{u,v}]. \tag{2.25}$$

Note that for all $e = \{u, v\} \in E$, the notation for Problem 2 supports unconstrained or missing distances ($d^-_e = 0$, $d^+_e \to \infty$), as well as the equality constraints found in Problem 1 ($d^-_e = d^+_e$).

In this thesis, we refer to both Problem 1 and 2 as the DGP, where the specific formulation can be inferred from the presence or absence of distance intervals on the edge weights of G. If a realization that satisfies an instance of the DGP exists, the corresponding collection of points $\mathbf{P}$ can be arbitrarily translated, rotated, and reflected such that the distance constraints still hold [Dokmanic et al., 2015]. This defines the equivalence class $[\mathbf{P}]$ of Eq. (5.8). Additionally, there may exist a set (finite or infinite) of equivalence classes $[\mathbf{P}]$ that correspond to distinct solutions to Problem 1 or 2 — a distinction crucial for the contribution proposed in Chapter 5.

2.3.1 Euclidean Distance Matrices

$$\begin{bmatrix} 0 & (d_{1,2})^2 & (d_{1,3})^2 & (d_{1,4})^2 \\ (d_{2,1})^2 & 0 & (d_{2,3})^2 & (d_{2,4})^2 \\ (d_{3,1})^2 & (d_{3,2})^2 & 0 & (d_{3,4})^2 \\ (d_{4,1})^2 & (d_{4,2})^2 & (d_{4,3})^2 & 0 \end{bmatrix}$$

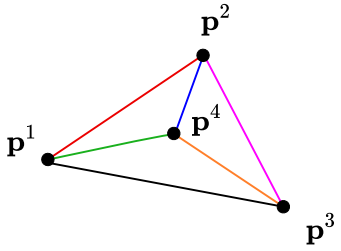


Figure 2.6: A visualization of how the Euclidean distance matrix (EDM) is constructed for a set of four points $\{\mathbf{p}\}_1^4$. Note that the identity in Eq. (2.27) populates the EDM with squared distances, as it is linear in the Gram matrix, but quadratic in positions.

Consider a realization of a graph G, obtained by solving Problem 1. By arranging the resulting points in a matrix $\mathbf{P} = [\mathbf{p}^0, \mathbf{p}^1, \ldots, \mathbf{p}^{N-1}]^\mathsf{T} \in \mathbb{R}^{N \times K}$, all interpoint distances $d_{u,v}$ can be determined via the Euclidean norm:

$$d_{u,v} = \|\mathbf{p}^u - \mathbf{p}^v\|\,.$$

Inner products $\mathbf{x}_{u,v} \triangleq \mathbf{p}^u\,(\mathbf{p}^v)^\mathsf{T}$ can be arranged into a *Gram matrix* $\mathbf{X} \triangleq \mathbf{P}\mathbf{P}^\mathsf{T}$, which belongs to the set of $N \times N$ symmetric positive semidefinite matrices $\mathcal{S}^N_+$. Elements of the Gram matrix can conveniently express squared interpoint distances

$$d^2_{u,v} = \mathbf{x}_{u,u} - 2\mathbf{x}_{u,v} + \mathbf{x}_{v,v}\,, \tag{2.26}$$

and the full set of squared interpoint distances in Eq. (2.26) can be efficiently calculated using the matrix identity

$$\left[d^2_{u,v}\right] = \mathcal{K}(\mathbf{X}) = \operatorname{diag}(\mathbf{X})\mathbf{1}^\mathsf{T} + \mathbf{1}\operatorname{diag}(\mathbf{X})^\mathsf{T} - 2\mathbf{X}, \tag{2.27}$$

where $\operatorname{diag}(\mathbf{X})$ is the vector formed by the main diagonal of the Gram matrix [Dokmanic et al., 2015], and $\mathbf{1}$ is

a column vector of ones. The resulting matrix $\mathbf{D} = \mathcal{K}(\mathbf{X})$ is known as a *Euclidean Distance Matrix* (EDM). We use $\mathcal{K}(\mathbf{X})$ to denote the linear operator mapping $\mathbf{X}$ to $\mathbf{D}$ as defined by Eq. (2.27).

Recovering Points

Consider the problem of recovering the original collection of points $\mathbf{P}$ from squared inter-point distances in the matrix $\mathbf{D}$. Necessary and sufficient conditions for a matrix to be an EDM can be found in [Sippl and Scheraga, 1986]. If $\mathbf{D}$ is an EDM, a Gram matrix that satisfies Eq. (2.27) can be recovered by taking

$$\mathbf{X} = -\frac{1}{2}\mathbf{J}\mathbf{D}\mathbf{J}, \tag{2.28}$$

where $\mathbf{J} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}^\mathsf{T}$ is the so-called geometric centering matrix [Dokmanic et al., 2015]. Once $\mathbf{X}$ has been recovered, a collection of points $\hat{\mathbf{P}} \in \mathbb{R}^{N\times K}$ can be obtained through the eigenvalue decomposition $\mathbf{X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathsf{T}$ by taking the first K eigenvalues λ_i[6]:

$$\hat{\mathbf{P}}^\mathsf{T} = \left[\operatorname{diag}\sqrt{\lambda_0}, \sqrt{\lambda_1}, \ldots, \sqrt{\lambda_{K-1}}, \mathbf{0}_{K\times N-K}\right] \mathbf{U}^\mathsf{T}. \tag{2.29}$$

While the squared distances of points in $\hat{\mathbf{P}}$ recovered using this procedure match those defined in $\mathbf{D}$ exactly, they are in general not equal to the original $\mathbf{P}$. This is due to the fact that inter-point distances are preserved under rigid transformations. In order to recover the absolute positions of the points, at least K points, known as *anchors*, need to have their positions defined a priori. These anchors are used to formulate the orthogonal Procrustes problem [Dokmanic et al., 2015], whose solution is the rotation (or reflection) $\mathbf{R} \in \mathrm{O}(K)$ and translation $\mathbf{p} \in \mathbb{R}^K$ that transform the positions of anchors in $\hat{\mathbf{P}}$ to their predefined positions. This transformation can then be applied to all the points in $\hat{\mathbf{P}}$ to yield the desired set of points $\mathbf{P}$.

2.4 Geometric Deep Learning

Deep learning is a subset of machine learning where multiple connected layers of data representations are built in an automated manner. Most deep learning models are based on *artificial neural networks* (ANNs), whose parameters are modified according to some performance criteria, resulting in better data representations for a given task. Owing to the majority of input data being high dimensional at first glance (e.g., images, point clouds), estimating parameters of deep learning models presents a significant challenge. Moreover, most tasks based in the physical world induce regularities in data that drastically reduce their dimensionality. The concept of geometric deep learning [Bronstein et al., 2017] refers to the endavor of exposing such regularities through a set of common geometric principles. These principles can be used to incorporate prior physical knowledge into neural architectures and provide a principled and unified way for constructing models.

This section will provide a brief overview of deep learning concepts pertaining to the final contribution presented in this doctoral thesis, where we use a *generative* model to obtain a distribution over inverse kinematics solutions. Further, graph neural networks (GNNs) described in this section are representative of the geometric deep learning paradigm, and are used to imbue our IK architecture with relational inductive biases characteristic of geometric problems.

[6] Assuming the recovered realization is K-dimensional, only the first K eigenvalues are nonzero.

2.4.1 Feedforward Neural Networks

Neural networks are mathematical models composed of connected artificial neurons. Each artificial neuron takes a sum of multiple inputs scaled by *weights* together with a *bias* term, which is then passed through a nonlinear *activation function* to produce an output. Neurons are typically composed into *layers* that map an input vector $\mathbf{x} \in \mathbb{R}^{N_x}$ on to a latent representation

$$\mathbf{z} = \mathbf{f}(\mathbf{x}) = \boldsymbol{\phi}(\mathbf{W}\mathbf{x} + \mathbf{b}) \,,$$

where $\mathbf{W} \in \mathbb{R}^{N_z \times N_x}$ is the matrix of weights, $\mathbf{b} \in \mathbb{R}^{N_z}$ are the bias terms and $\boldsymbol{\phi}$ is the activation function.

Layers of neurons are typically composed in series such that the vectorized outputs

$$\mathbf{z}_{i+1} = \mathbf{f}_i(\mathbf{z}) = \boldsymbol{\phi}_i(\mathbf{W}_i\mathbf{z}_i + \mathbf{b}_i) \,, \tag{2.30}$$

of each neuron in the previous layer have directed connections to each neuron in the subsequent layer, forming an acyclic computation graph. This architecture is known as a *feedforward neural network* or *multilayer perceptron* (MLP), which parameterizes a model composed of L layers of neurons

$$\mathbf{y} = \mathrm{NN}(\mathbf{x}; \boldsymbol{\alpha}) = \mathbf{f}_1 \circ \mathbf{f}_2 \circ \cdots \circ \mathbf{f}_L \,, \tag{2.31}$$

whose outputs $\mathbf{y}$ are determined by parameters $\boldsymbol{\alpha}$, where $\circ$ denotes a composition operation $\mathbf{f} \circ \mathbf{h} \triangleq \mathbf{f}(\mathbf{h}(\cdot))$. The overall number of layers L is also known as the *depth*, while the number of neurons in a given layer is its *width*. The universal approximation theorem for neural networks states that every continuous function can be approximated arbitrarily closely with just one *hidden* layer (i.e., $L = 3$).

2.4.2 Learning

Training or *learning* of a neural network involves changing the weights $\mathbf{W}$ and biases $\mathbf{b}$ after each data point is processed, with the goal of reducing the error of the output compared to the expected result. The learning error is expressed via a *loss* function $\mathcal{L}$, which makes it possible to formulate learning as a numerical optimization problem. Crucially, a neural network may be trained efficiently using variants of gradient descent, where the gradient of the loss with respect to the weights and biases is computed using the *backpropagation* algorithm. Learning over large amounts of data is made tractable by using stochastic gradient descent algorithms that use a gradient estimation based on a subset of the available data. A distinction is made between *supervised* and *unsupervised* learning, which is explained in more detail in Section A.1 and Section A.2.

2.4.3 Graph Neural Networks

Neural networks that operate on graphs instead of vectors, and structure their computations accordingly, are known as *graph neural networks* (GNNs). The relational inductive biases induced by the specific architectural assumptions of such networks have shown to result in better generalization compared to classical feedforward neural networks described in Section 2.4.1.

Remark: Graph Networks

In discussing GNN architectures, this dissertation adopts the formalism of *graph networks* introduced by Battaglia et al. [2018b]. This framework provides the theory and notation required to describe a wide variety of GNN architectures, including the EGNN architecture used in Chapter 6.

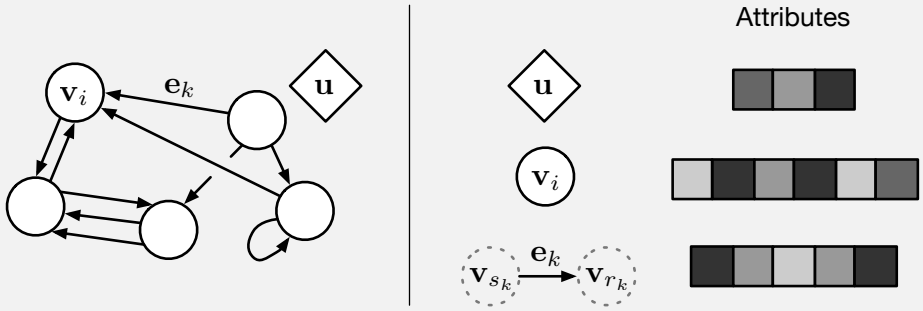


Figure 2.7: Visualization of the graph network formalism, image curtesy of Battaglia et al..

The working definition of graph in this thesis is therefore (unless otherwise specified), a directed, attributed graph with a global attribute. In this terminology, a node is denoted as $\mathbf{v}_i$, an edge as $\mathbf{e}_k$, and the global attributes as $\mathbf{u}$. Further, s_k and r_k to indicate the indices of the sender and receiver nodes (see below), respectively, for edge k.

Graph neural networks require the data to be structured as a graph defined by the tuple $G = (\mathbf{u}, V, E)$ The vector $\mathbf{u}$ is a global attribute containing data that is somehow representative of the graph globally. The set $V = \{\mathbf{v}_i\}_1^{N_v}$ is the set of nodes (of cardinality N_v), where each vector $\mathbf{v}_i$ is an attribute associated with that node. The set $E = \{(\mathbf{e}_k, r_k, s_k)\}_1^{N_e}$ is the set of edges (of cardinality N_e), where each vector $\mathbf{e}_k$ is the attribute of an edge, r_k is the index of the receiver node, and s_k is the index of the sender node. The GNN modifies input graphs using the generalized propagation equations

$$\begin{aligned} \mathbf{e}'_k &= \mathrm{NN}^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right) & \bar{\mathbf{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \\ \mathbf{v}'_i &= \mathrm{NN}^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) & \bar{\mathbf{e}}' &= \rho^{e\to u}\left(E'\right) \\ \mathbf{u}' &= \mathrm{NN}^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right) & \bar{\mathbf{v}}' &= \rho^{v\to u}\left(V'\right) \end{aligned} \tag{2.32}$$

where feedforward neural networks $\mathrm{NN}^e, \mathrm{NN}^v, \mathrm{NN}^u$ and aggregation operations $\rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$ (e.g., summation, maximum, minimum) update the node, edge and global attributes as shown in Algorithm 1. Note that $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$, $V' = \{\mathbf{v}'_i\}_{i=1:N^v}$, and $E' = \bigcup_i E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$ are sets of all or neighbouring nodes and edges used in particular attribute updates. The resulting updated node, edge and global attributes form the graph $G' = (E', V', \mathbf{u}')$, which now contains information on the neighborhoods of each node and edge. This process may be repeated multiple times such that the data from more distant nodes and edges is propagated fully.

Algorithm 1: Graph Neural Network (GNN)

Input: $E, V, \mathbf{u}$
Parameters : $\mathrm{NN}^v, \mathrm{NN}^e, \mathrm{NN}^u, \rho^{e\to v}, \rho^{e\to u}, \rho^{v\to u}$
for $k \in \{1 \dots N^e\}$ **do**
  $\mathbf{e}'_k \leftarrow \mathrm{NN}^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}\right)$ ▷ 1. Compute updated edge attributes
end
for $i \in \{1 \dots N^n\}$ **do**
  let $E'_i = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{r_k=i,\ k=1:N^e}$
  $\bar{\mathbf{e}}'_i \leftarrow \rho^{e\to v}\left(E'_i\right)$ ▷ 2. Aggregate edge attributes per node
  $\mathbf{v}'_i \leftarrow \mathrm{NN}^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$ ▷ 3. Compute updated node attributes
end
let $V' = \left\{\mathbf{v}'\right\}_{i=1:N^v}$
let $E' = \left\{\left(\mathbf{e}'_k, r_k, s_k\right)\right\}_{k=1:N^e}$
$\bar{\mathbf{e}}' \leftarrow \rho^{e\to u}\left(E'\right)$ ▷ 4. Aggregate edge attributes globally
$\bar{\mathbf{v}}' \leftarrow \rho^{v\to u}\left(V'\right)$ ▷ 5. Aggregate node attributes globally
$\mathbf{u}' \leftarrow \mathrm{NN}^u\left(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \mathbf{u}\right)$ ▷ 6. Compute updated global attribute
return $\left(E', V', \mathbf{u}'\right)$

Chapter 3

Robot Kinematics

We're functioning automatic
And we are dancing mechanic
We are the robots

Kraftwerk - The Robots

The contributions of this thesis are based on analyzing robotic manipulators from the perspective of kinematics, a subfield of classical mechanics that deals with motions of objects in space without considering the forces that cause their movement. Kinematic analysis provides insight into the underlying geometry of a robot's motion, allowing for statements to be made regarding the robot's dexterity, accuracy and overall posture in the workspace. These insights are a crucial part of motion planning, control, and trajectory optimization algorithms that are ubiquitous in most robotic manipulation applications. This chapter introduces concepts and problems in robot kinematics that are key to understanding the challenges faced by practitioners and researchers when designing robotic systems for real-world applications.

Remark: Associated literature
The notation and structure of this chapter is inherited from the excellent introductory textbook by Lynch and Park [2017]. A more advanced treatment from a group-theoretic perspective may be found in the textbook by Selig [2005], while Sciavicco and Siciliano [2012] give a practical overview with an emphasis on control and planning applications.

3.1 Robot Structure

We focus on a particular subset of robots that includes most common commercial manipulators designed to operate (semi-)autonomously. These robotic manipulators are comprised of a series of N joints with a single axis of rotation (i.e., *revolute* joints) connected by $N-1$ rigid links, with an end-effector (i.e., gripper or a tool) represented as an additional link attached to the final joint. As the example in Fig. 3.1a and Fig. 3.1b illustrates, the coordinate frame $\underrightarrow{\mathcal{F}}_{i-1}$ is attached to the i-th joint, which is the *parent* of the link L_i. The space frame $\underrightarrow{\mathcal{F}}_{s}$ may be placed anywhere, but it is often positioned such that one of its axes coincides with the rotation axis of the first joint (in this case it is referred to as the *base frame*). Finally, the *end-effector* is often modeled as an additional *child* link of the final joint, with the body frame $\underrightarrow{\mathcal{F}}_{b}$ attached to some relevant point (e.g., the middle of the gripper).

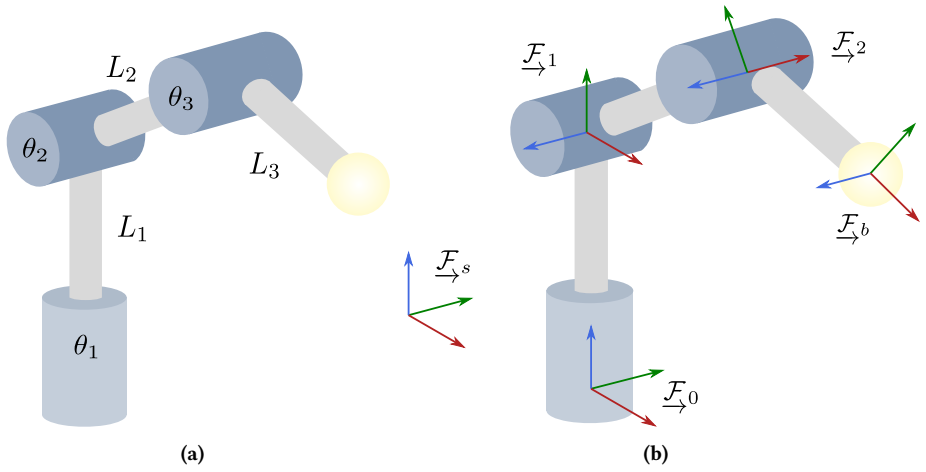


Figure 3.1: a) Simplified sketch of the robot structure, with links denoted by L and joints denoted by θ. The yellow sphere represents an end-effector (such as a tool or a gripper). b) Sketch of the coordinate systems commonly defined and used in computations related to manipulation tasks.

3.1.1 Configuration Space

As a starting point for kinematic analysis, it is important to find a way to express the positions of all points on the robot, also known as the robot *configuration*. Instead of keeping track of all such points, the configuration of a robot can be specified using coordinates θ_i arranged into a vector $\boldsymbol{\theta} \in \mathcal{C}$, where $\mathcal{C}$ is the *configuration space* or *C-space*. The minimum number of coordinates required to represent the configuration of the robot is equal to its *degrees of freedom* (DOF). Given that robots are constructed using a large variety of different joint and link types specialized for particular tasks, the geometry of their configuration spaces can vary significantly and therefore require careful analysis.

The configuration space of robotic manipulators of the type shown in Fig. 3.1b admits a relatively simple characterization. Given their unchanging joint and link geometry, the configuration of such robots can be fully represented in a vector space joint angles $\theta_i \in [-\pi, \pi)$ arranged in a vector $\boldsymbol{\theta}$. Note that this is a simplification, since removing the angle range constraints (i.e. joint limits) and adding 2π to all angles results in the same robot posture, meaning that these two configurations define the same point in C-space. This observation reveals that the true shape of the configuration space is that of an N-dimensional torus. While this characterization is inconsequential for the methods and approaches discussed in the thesis, the underlying geometry of the configuration spaces needs to be kept in mind when discussing the proximity of different configurations.

3.1.2 Task Space

A class of tasks performed by the robot is generally parameterized using coordinates τ_i arranged into a vector $\boldsymbol{\tau} \in \mathcal{T}$, where $\mathcal{T}$ is the *task space* defined by this parameterization. The task space is specified by the user and generally coincides with the pose or position of the end-effector (e.g., a gripper used to manipulate objects), but may also be learned (e.g., from demonstration data).

A point in the task space may be reachable by multiple distinct robot configurations. Moreover, if the number of degrees of freedom of a robot exceeds the dimension of the task space (i.e., $m = \dim(\mathcal{C}) - \dim(\mathcal{T}) > 0$), any $\boldsymbol{\tau}$ may be reachable by an m-dimensional set of configurations. Such sets are often more accurately defined as algebraic varieties — sets of solutions to systems of polynomial equations. For example, robots with five degrees

of freedom may reach any end-effector position with an infinite number of configurations on a two-dimensional variety, while robots with seven degrees of freedom can reach any end-effector pose with a one-dimensional variety of configurations. These robots are often said to be *redundant* in the context of a given task space.

3.2 Forward Kinematics

Having fully defined the spaces and coordinates specifying the robot's configuration and task, the next step required for most applications involves finding a mapping between the two. The procedure of mapping a robot's configuration to the associated task space coordinates is known as *forward kinematics*.

Definition 8 (Forward kinematics)**.** *The mapping* $\mathrm{FK} : \mathcal{C} \to \mathcal{T}$ *of a configuration* $\boldsymbol{\theta} \in \mathcal{C}$ *to a vector of task space coordinates* $\boldsymbol{\tau} \in \mathcal{T}$ *is known as the forward kinematic mapping. The procedure for computing this mapping is known as the robot's forward kinematics.*

Generally, FK is a nonlinear function defined by the robots structural properties (e.g., link geometry, joint types, etc.). Further, FK is commonly defined to be injective, meaning that every configuration $\boldsymbol{\theta}$ maps to a single point in the task space (e.g., one configuration cannot map to two different end-effector poses). For clarity, a distinction is made between the standard forward kinematics in Definition 8 and *differential forward kinematics*, which relates velocities the configuration space to those in the task space.

Remark: End-effector Poses and Velocities
The contributions of this thesis pertain to tasks involving either:

- tracking reference end-effector velocities
- reaching a reference end-effector pose

In both of these cases, the task space corresponds to elements $\mathbf{T} \in \mathrm{SE}(3)$ of the special Euclidean group representing the space of poses. As discussed in Section 2.2.2, transforms $\mathbf{T}$ can be represented by the more compact task space of six-dimensional exponential coordinates $\boldsymbol{\xi}$, associated with Lie algebra elements $\boldsymbol{\xi}^{\wedge} \in \mathfrak{se}(3)$. Likewise, the time derivatives $\dot{\boldsymbol{\xi}}$ of these coordinates give a compact representation of generalized end-effector velocities.

3.2.1 Motion of Rigid Links

The motion of links, as governed by joint rotations, can be described using a variety of different parameterizations. From the Mozzi-Chasels theorem in Section 2.2.2 it follows that the actuation of a rigid link by its parent joint may be described as a screw motion. By taking Eq. (2.19) and setting the pitch to $h = 0$ in order to reflect a lack of translation along the rotation axis, the screw axis for revolute joints is expressed as

$$\boldsymbol{\mathcal{S}} = \begin{bmatrix} \hat{\mathbf{s}} \\ -\hat{\mathbf{s}} \times \mathbf{q} \end{bmatrix}, \tag{3.1}$$

where $\hat{\mathbf{s}}$ is the rotation axis of a particular joint and $\mathbf{q}$ is the position of the joint in $\underrightarrow{\mathcal{F}}_s$ at $\boldsymbol{\theta} = \mathbf{0}$, as shown in Fig. 3.2. Multiplying these screws with joint angles results in exponential coordinates describing the pose

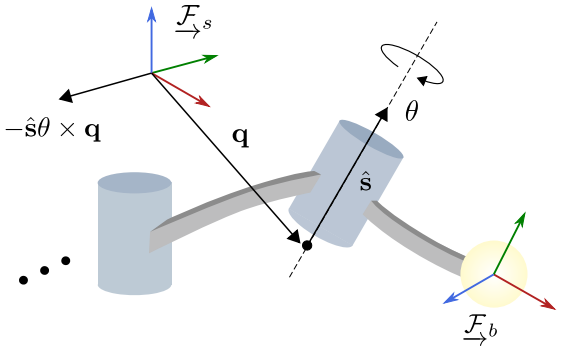


Figure 3.2: Sketch of the parameterization of link motions for revolute joints using screws. The vector $\hat{\mathbf{s}}$ is the joint's rotation axis, while $\mathbf{q}$ is the position of some point on the rotation axis in the robot's home configuration (i.e., $\boldsymbol{\theta} = \mathbf{0}$).

change of the body frame $\underrightarrow{\mathcal{F}}_b$ attached to the end-effector

$$\boldsymbol{\xi} = \boldsymbol{\mathcal{S}}\theta = \begin{bmatrix} \hat{\mathbf{s}}\theta \\ -\hat{\mathbf{s}}\theta \times \mathbf{q} \end{bmatrix} . \tag{3.2}$$

Finally, the exponential map in Eq. (2.17) gives the $\mathrm{SE}(3)$ transform

$$\mathbf{T}_{sb} = \mathrm{Exp}(\boldsymbol{\mathcal{S}}\theta)\,\mathbf{M}\,, \tag{3.3}$$

describing how $\underrightarrow{\mathcal{F}}_b$ with the pose $\mathbf{M} = \mathbf{T}_{sb}|_{\theta=0}$ at a zero configuration is positioned and oriented after rotating its parent joint to by an angle θ.

3.2.2 Pose Forward Kinematics

Once the basic relationship of joint rotations to link movement is specified in the form of parameterized $\mathrm{SE}(3)$ transforms, the full motion manifold of the robot may be obtained by composition. Specifically, the pose of a frame whose motion is determined by multiple revolute joints connected by rigid links is given by composing Eq. (3.3) through left multiplication. This results in the forward kinematic mapping

$$\mathrm{FK}(\boldsymbol{\theta}) = \prod_{i=1}^{n} \mathrm{Exp}(\boldsymbol{\mathcal{S}}_i\theta_i)\mathbf{M}\,, \tag{3.4}$$

where $\boldsymbol{\mathcal{S}}_i$ is the screw axis of joint i expressed in $\underrightarrow{\mathcal{F}}_s$ and the matrix $\mathbf{M} = \mathbf{T}_{sb}|_{\boldsymbol{\theta}=\mathbf{0}}$ is the pose of the end-effector (i.e., body) frame $\underrightarrow{\mathcal{F}}_b$ when the configuration $\boldsymbol{\theta}$ is set to zero (i.e., the robot is in a "home" configuration).

This *product of exponentials* formulation of forward kinematics was introduced by Murray et al. [1994] and uses a non-minimal parameterization of structure based on joint screws. An important advantage of this approach is that the mapping in Eq. (3.4) may be differentiated using Lie algebra in a computationally efficient manner. Note that there exist minimal parameterizations such as the one introduced by Hartenberg and Denavit [1955], which are often used in applications such as kinematic calibration. The coordinate systems generated by these parameterization may not directly correspond to the physical locations of joints and links.

3.2.3 Differential Forward Kinematics

Other than knowing where the end-effector is, many applications require a characterization of its velocity. The relationship between joint velocities and end-effector twists[1] takes the form of a locally linear mapping

$$\dot{\boldsymbol{\xi}} = \mathbf{J}\left(\boldsymbol{\theta}\right)\dot{\boldsymbol{\theta}}\,, \tag{3.5}$$

where $\mathbf{J}$ is the Jacobian of forward kinematic mapping in Eq. (3.4) with respect to configuration space coordinates

$$\mathbf{J}\left(\boldsymbol{\theta}\right) = \frac{\partial\, FK}{\partial\, \boldsymbol{\theta}} \in \mathbb{R}^{6\times n} \tag{3.6}$$

for $n = \dim(\mathcal{C})$. To simplify notation, we drop the $\boldsymbol{\theta}$ as the argument of $\mathbf{J}$ in the remainder of this text. For the forward kinematics parameterization in Eq. (3.4) we have

$$\mathbf{J} = \begin{bmatrix}\mathbf{J}_{\boldsymbol{\mathcal{S}}_1} & \ldots & \mathbf{J}_{\boldsymbol{\mathcal{S}}_n}\end{bmatrix}\,, \tag{3.7}$$

where each column represents the contribution of the corresponding joint to the motion of the target frame and $\dot{\boldsymbol{\xi}}$ is resulting twist. This contribution is computed by representing each screw in $\underrightarrow{\mathcal{F}}_s$ using

$$\mathbf{J}_{\boldsymbol{\mathcal{S}}_i} = \mathrm{Ad}\left(\prod_{j=1}^{i-1}\mathrm{Exp}(\boldsymbol{\mathcal{S}}_j\theta_j)\right)\boldsymbol{\mathcal{S}}_i \tag{3.8}$$

where $\mathrm{Ad} : \mathrm{SE}(3) \mapsto \mathbb{R}^{6\times 6}$ is the adjoint operator defined in Eq. (2.22). Similar expressions can be found for higher order derivatives (e.g, acceleration) by further differentiating the forward kinematic mapping [Müller, 2021].

As a consequence of the definition of forward kinematics in Eq. (3.4), the Jacobian defined by Eq. (3.7) and Eq. (3.8) relates joint velocities to twists in the space frame $\underrightarrow{\mathcal{F}}_s$. This twist can be expressed in the body frame $\underrightarrow{\mathcal{F}}_b$ via

$$\mathbf{J}_b = \mathrm{Ad}\left(\mathbf{T}_{bs}\right)\mathbf{J}\,, \tag{3.9}$$

where $\mathbf{T}_{bs} = \mathbf{T}_{sb}{}^{-1}$ is the pose of $\underrightarrow{\mathcal{F}}_s$ in $\underrightarrow{\mathcal{F}}_b$. This identity is useful in many practical cases where velocities are defined in the body frame.

Linear and Angular Velocities

By definition, twists entangle rotational and linear velocities. This may present difficulties when controlling or interpreting the end-effector velocity in applications such as teleoperation or visual servoing. In such cases, the Jacobian in Eq. (3.7) can be transformed via the identity

$$\mathbf{J}_l = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_{3\times 3}\\ \mathbf{p}^{\wedge} & \mathbf{I}_3\end{bmatrix}^{-1}\mathbf{J}\,, \tag{3.10}$$

[1] Technically, twists are the Lie algebra elements $\hat{\boldsymbol{\xi}}$, but the term is often used for the associated vector of generalized velocities as well.

where $\mathbf{p}$ is the position of the coordinate frame the velocities are expressed in. This separates the angular and linear velocities $\boldsymbol{\omega}$ and $\boldsymbol{v}$ such that

$$\begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{v} \end{bmatrix} = \mathbf{J}_l \dot{\boldsymbol{\theta}} \,. \tag{3.11}$$

This is often refered to as the *geometric* or *analytical* Jacobian. In addition to the above identity, the geometric Jacobian may also be computed using a geometric constructive procedure [Sciavicco and Siciliano, 2012].

3.3 Inverse Kinematics

Most applications of robotic manipulation in the real world, such as grasping objects or interacting with the environment, are difficult to define solely in terms of joint angles (i.e., in the configuration space). Instead, most tasks are defined in terms of end-effector poses relative to a static coordinate frame (i.e. in the task space). In order to compute the joint configurations and velocities necessary to achieve a desired motion in the task space, an inverse of Eq. (3.4) or Eq. (3.5) needs to be found. The procedure of mapping a vector of task space coordinates to a robot's joint configurations is known as *inverse kinematics*.

Definition 9 (Inverse Kinematics)**.** *The mapping $IK : \mathcal{T} \to \mathcal{C}$ of a task space coordinates $\boldsymbol{\tau} \in \mathcal{T}$ to a set of joint configurations $IK(\boldsymbol{\tau}) = \{\boldsymbol{\theta} \in \mathcal{C} \mid FK(\boldsymbol{\theta}) = \boldsymbol{\tau}\}$ is known as the inverse kinematic mapping. The procedure for fully or partially computing this mapping is known as a robot's inverse kinematics.*

Unlike forward kinematics, IK is generally not unique. In other words, IK is not injective and therefore multiple feasible configurations exist for a single set of task coordinates $\boldsymbol{\tau}$. Consequently, there is no single best approach to solving the inverse kinematics problem, and many solution approaches have been developed for particular applications.

Owing to its widespread use in robotics [Angeles et al., 2013] and computer graphics [Aristidou et al., 2018], IK remains and active research area with an abundance of relevant literature. However, it is important to note that the difficulty of the IK problem varies depending on the underlying assumptions imposed by a given application (e.g., mechanism, task space ...) and that these assumptions are often not explicitly stated. To avoid this ambiguity, this thesis focuses on and stresses a distinction between two instances of the IK problem common to robotics applications: *differential inverse kinematics* and *pose inverse kinematics*.

3.3.1 Differential Inverse Kinematics

Differential inverse kinematics is the problem of finding joint velocities that generate a desired twist[2] of the end-effector. This instance of IK also occurs in task space control, where the goal end-effector pose is defined in terms of an incremental motion of the end-effector and carries the assumption of the solution being close to the starting configuration.

When $\dim(\mathcal{C}) = \dim(\mathcal{T})$, joint velocities can be computed by simply inverting the linear Jacobian identity in Eq. (3.5). However this does not generally hold for $\mathcal{T} \triangleq \mathrm{SE}(3)$, since many commercial manipulators have $\dim(\mathcal{C}) > 6$. In this case Eq. (3.5) is solved for $\dot{\boldsymbol{\theta}}$ as

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^{\dagger} \dot{\boldsymbol{\xi}} \,, \tag{3.12}$$

[2] or linear and angular velocity.

where $\mathbf{J}^\dagger = \mathbf{J}^\intercal \left(\mathbf{J}\,\mathbf{J}^\intercal\right)^{-1}$ is the Moore-Penrose pseudoinverse. Note that Eq. (3.12) is the *least-squares* solution to the optimization problem

$$\begin{aligned} \min_{\dot{\boldsymbol{\theta}}} \quad & \frac{1}{2}\dot{\boldsymbol{\theta}}^T \mathbf{W} \dot{\boldsymbol{\theta}} + \mathbf{c}^\intercal \dot{\boldsymbol{\theta}} \\ \text{s.t.} \quad & \mathbf{A}\dot{\boldsymbol{\theta}} = \mathbf{b} \end{aligned} , \tag{3.13}$$

where $\mathbf{W} = \mathbf{I}, \mathbf{c} = \mathbf{0}, \mathbf{A} = \mathbf{J}$ and $\mathbf{b} = \dot{\boldsymbol{\xi}}$. Crucially, a solution to any such quadratic program [Boyd and Vandenberghe, 2004] reduces to solving a linear system.

Redundancy Resolution

Aside from reaching a goal pose, redundant degrees of freedom may be used to track reference velocities $\dot{\boldsymbol{\tau}}_s$ in a secondary task space by setting the cost function of Eq. (3.13) to $\|\mathbf{J}_s^\dagger \dot{\boldsymbol{\tau}}_s - \dot{\boldsymbol{\theta}}\|^2$. The closed-form solution then becomes

$$\dot{\boldsymbol{\theta}} = \mathbf{J}^\dagger \dot{\boldsymbol{\xi}} + \left(\mathbf{J}^\dagger \mathbf{J} - \mathbf{I}\right) \mathbf{J}_s^\dagger \dot{\boldsymbol{\tau}}_s \,. \tag{3.14}$$

These *redundancy resolution* techniques [Nakamura et al., 1987] demonstrate the utility of redundant degrees of freedom, for they make it possible to choose solutions that fit a certain criteria. With some modification, this framework can be extended to multi-level task hierarchies [Siciliano and Slotine, 1991] and to non-Euclidean [Jaquier et al., 2020] or learned [Cheng et al., 2018] task spaces.

3.3.2 Pose Inverse Kinematics

The problem of pose inverse kinematics exactly fits Definition 9 in that no assumptions exist regarding the proximity of the solution manifold to the initial configuration, making solutions difficult to find using the first-order approximation in Eq. (3.12). This problem is instead formulated as a nonlinear program of the form

$$\begin{aligned} \boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} \quad & f(\boldsymbol{\theta}) \\ \text{s.t.} \quad & h_i(\boldsymbol{\theta}) = 0 \,, i = 1, \ldots, N_h \,, \\ & g_j(\boldsymbol{\theta}) \leq 0 \,, j = 1, \ldots, N_g \,, \end{aligned} \tag{3.15}$$

where an error $e(\boldsymbol{\theta}, \mathbf{T}_{goal})$ between the current and goal end-effector poses, or its norm, is encoded as an objective f or as the (in)equality constraints $h_i \,, g_j$. While this formulation admits a more general set of constraints and optimality criteria, it usually requires the use of optimization approaches with more computational overhead.

Closed-Loop IK

A straightforward formulation of the problem in Eq. (3.15) involves setting the objective to the squared magnitude of the space frame transform between the current and goal poses

$$f(\boldsymbol{\theta}) = \left\| \text{Log}(\mathbf{T}(\boldsymbol{\theta}) \mathbf{T}_{goal}^{-1}) \right\|^2 ,$$

where $\text{Log}(\cdot)$ is the $\text{SE}(3)$ logarithmic map in Eq. (2.21). When no additional constraints are added, the iteration of the Gauss-Newton method is exactly equal to Eq. (3.12) with $\Delta\boldsymbol{\theta} = \alpha\dot{\boldsymbol{\theta}}$, where the step size α is chosen using a line search algorithm.

Some literature refers to this first-order approach as *closed-loop IK* (CLIK) [Siciliano et al., 2010], as it emulates a feedback control problem [Sciavicco and Siciliano, 1986]. Major advantages of CLIK methods include their ease of implementation relative to other nonlinear optimization algortihms. Owing to the connection of IK with control literature, there also exist a variety of extensions providing numerical robustness [Buss and Kim, 2005] and efficient incorporation of secondary objectives through redundancy resolution [Nakamura et al., 1987]. However, alongside the convergence issues commonly encountered with first-order local optimization, CLIK methods will often trade numerical stability for accuracy [Spong et al., 2005].

Nonlinear Programming

It has been shown that variants of Eq. (3.15) may be solved with a higher success rate using a variety of unconstrained or bounded nonlinear programming methods such as L-BFGS-B [Zhu et al., 1997] or SQP [Schulman et al., 2014]. These methods have robust theoretical underpinnings [Boyd and Vandenberghe, 2004] and can approximately support a wide range of constraints through the addition of penalties to the cost function [Beeson and Ames, 2015]. However, the highly nonlinear nature of the problem makes them susceptible to local minima, often requiring multiple initial guesses before returning a global minimum, if at all.

Chapter 4

Geometry-Aware Singularity Avoidance

> Success depends upon previous preparation, and without such preparation there is sure to be failure.
>
> — Confucius

Articulated robots such as manipulators increasingly must operate in uncertain and dynamic environments where interaction (with human coworkers, for example) is necessary. In these situations, the capacity to quickly adapt to unexpected changes in task space constraints is essential. At certain points in a manipulator's configuration space, termed *singularities*, a robot loses one or more degrees of freedom (DoF) and is unable to move in specific task space directions. The inability to move in arbitrary directions compromises adaptivity and, potentially, safety. In this chapter we introduce a *geometry-aware singularity index*, defined using a Riemannian metric on the manifold of symmetric positive definite matrices, to provide a measure of proximity to singular configurations. We demonstrate that our index avoids some of the failure modes and difficulties inherent to other indices previously introduced in literature. Further, we show that our index can be differentiated easily, making it compatible with local optimization approaches for task space control. Our experimental results establish that, for reaching and path following tasks, optimization based on our index achieves a higher degree of numerical stability compared to a common manipulability maximization technique and ensures singularity-robust motions.

4.1 Motivation and Related Work

Articulated robots are often required to perform tasks in which workspace movement is constrained, due to safety considerations or for other reasons. The constraints may also be altered during task execution as a result of unexpected changes in the environment (such as a human coworker pushing the robot, for example). Depending on the link and joint geometry, certain joint configurations can lead to a loss of task space mobility or to hazardous joint movements, potentially resulting in task failure or worse. Such configurations are known as *singularities* [Duffy, 1980] and singularity avoidance is an important part of most control and motion planning algorithms for articulated robots. Identifying and avoiding singularities has thus been the focus of significant research efforts within the robotics community [Tourassis and Ang Jr, 1992]. Moreover, geometrically intuitive optimization criteria that encode the proximity of a configuration to one or more singular or near-singular

regions have found a variety of applications, ranging from control and motion planning to kinematic synthesis.

A majority of robot manipulators are comprised of revolute joints, and nearly all relevant tasks can be described in terms of sets of nonlinear constraints that are functions of the joint configurations. When these constraints are defined in task space, robots are especially vulnerable to kinematic singularities [Beiner, 1992, 1997, Buss, 2004]. Kinematic singularities inhibit the robot's ability to generate end-effector velocities in certain directions in the task space. These singularities can be identified by observing the conditioning of the Jacobian matrix of the robot, which maps configuration space velocities to task space velocities [Sciavicco and Siciliano, 2012]. This relationship forms the basis of several kinematic sensitivity indices proposed in literature [Cardou et al., 2010, Patel and Sobh, 2015]. Many such indices can be interpreted geometrically through the notion of the *manipulability ellipsoid* [Yoshikawa, 1985], whose axis lengths correspond to the singular values of the Jacobian matrix and indicate the overall sensitivity of actuators to link displacements. Perhaps the most common index is the *manipulability index* [Yoshikawa, 1985] proposed by Yoshikawa in 1985, which is proportional to the volume of the manipulability ellipsoid [Maciejewski and Klein, 1989]. Salisbury and Craig suggest a *dexterity index* in [Salisbury and Craig, 1982] that provides an upper bound for the relative error amplification, which is a function of the ratio between the longest and shortest manipulability ellipsoid axes. A geometry-aware similarity measure between two manipulability ellipsoids is formulated by Rozo et al. [2017] using the Stein divergence—a statistical divergence between two probability measures.

In this chapter, we introduce a *geometry-aware singularity index* based on a differential geometric characterization of the manipulability ellipsoid described by Jaquier et al. [2020], which can be made robust to the failure modes of the manipulability and dexterity indices. We base our index on the Riemannian metric in [Pennec et al., 2006], enabling us to compute the length of the geodesic between the manipulability ellipsoid and a sufficiently "non-singular" reference ellipsoid. By determining the gradient of this length with respect to the joint values, we are able to augment common pose and differential IK methods with an effective singularity avoidance criterion.

Most task space reference tracking schemes for robotic manipulators are closed-loop IK formulations described in Section 3.3.1, that use a linearized kinematic model. This approach has been successfully applied for both control and planning [Sciavicco and Siciliano, 2012], where the local mapping of joint motions to spatial displacements (i.e., the Jacobian) of the robot is used to produce the desired end-effector movement [Xian et al., 2004, Pham et al., 2010]. Redundancy resolution schemes based on null-space optimization techniqes [Nakamura et al., 1987, Marani et al., 2002] have long been relied upon for singularity avoidance, however they are themselves subject to algorithmic singularities [Chiaverini, 1997] caused by contradictory objectives and hence are commonly used only for simple task hierarchies. Alternatively, methods based on nonlinear programming can easily be extended to include a variety of additional constraints and objectives [Cheng et al., 1993, Schulman et al., 2013]. Recently, quadratic programming (QP) [Frank et al., 1956] formulations have been explored as an efficient method for singularity avoidance in constrained inverse kinematics solvers [Zhang et al., 2012, Dufour and Suleiman, 2017, Jin et al., 2017].

4.2 The Manipulability Ellipsoid

Consider an n-dimensional unit sphere in the space of configuration velocities $\|\dot{\boldsymbol{\theta}}\|^2 = 1$. Observing that $\dot{\boldsymbol{\tau}} = \mathbf{J}\dot{\boldsymbol{\theta}}$, we obtain a mapping of this sphere to velocities in the p-dimensional task space

$$\dot{\boldsymbol{\theta}}^\mathsf{T}\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\tau}}^\mathsf{T}(\mathbf{J}\,\mathbf{J}^\mathsf{T})^{-1}\dot{\boldsymbol{\tau}}\,. \tag{4.1}$$

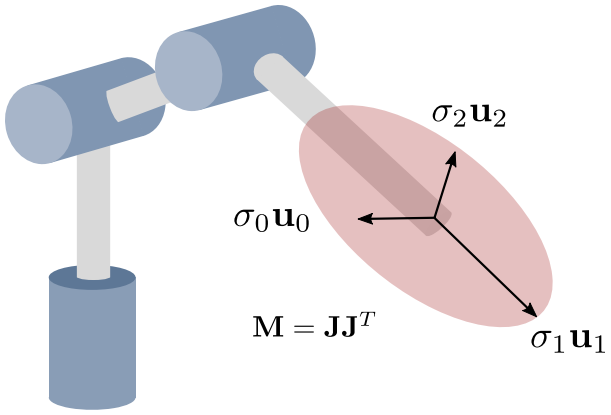


Figure 4.1: A three DoF manipulator and the manipulability ellipsoid associated with a task space defined by the end-effector position. Note that the axes of the ellipsoid correspond to the singular values σ and vectors $\mathbf{u}$ of the Jacobian $\mathbf{J}$. If we were to include the end-effector orientation, the ellipsoid would be six-dimensional.

Given the task and configuration characterizations discussed in Chapter 3, the identity in Eq. (4.1) shows that the scaling of end-effector velocities to the configurations space is reflected by the matrix

$$\mathbf{M}(\boldsymbol{\theta}) = \mathbf{J}\,\mathbf{J}^{\intercal}\,, \tag{4.2}$$

whose eigenvalues correspond to the squared singular values σ^2 of the Jacobian matrix $\mathbf{J}$. Consequently, configurations in which one or more eigenvalues of $\mathbf{M}$ become zero correspond to cases where the Jacobian matrix is poorly conditioned and non-invertible. That is, for any configuration $\boldsymbol{\theta}$, we can use Eq. (4.2) to compute a symmetric positive semidefinite matrix $\mathbf{M}$ that contains information about the task space mobility of the manipulator.

Notably, there exists an isomorphism between the set of $p \times p$ symmetric positive semidefinite matrices and the set of ellipsoids of dimension $\leq p$ centred at the origin. For this reason, the matrix $\mathbf{M}$ is also known as the *manipulability ellipsoid* of the end-effector [Yoshikawa, 1985]. The principal axes $\sigma_0\mathbf{u}_0, \sigma_1\mathbf{u}_1, \ldots, \sigma_{p-1}\mathbf{u}_{p-1}$ of this ellipsoid can be determined through singular value decomposition of $\mathbf{J} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. The lengths and orientations of these axes indicate directions in which larger task space velocities can be generated, as illustrated in Fig. 4.1, where the task space consists of the end-effector position of a three DoF manipulator. Conversely, directions admitting higher mobility are also directions in which the manipulator is more sensitive to perturbations.

4.3 Singularities

Using the manipulability ellipsoid to geometrically interpret the Jacobian, we can conclude that $\dot{\boldsymbol{\tau}} = \mathbf{J}\dot{\boldsymbol{\theta}}$ has a numerically unstable solution for all configurations associated with one or more degenerate (i.e., zero-length or near zero-length) axes. The detection and avoidance of these configurations, known as *singularities*, requires careful interpretation of the Jacobian's singular values. Moreover, end-effector movements that begin from configurations in close proximity to singularities also tend to result in high joint velocities and undesirable dynamic characteristics [Cardou et al., 2010, Patel and Sobh, 2015]. Consequently, many indicators have been developed that are used for singularity avoidance and to improve the kinematic sensitivity of task space control and inverse kinematics algorithms.

4.3.1 Manipulability index

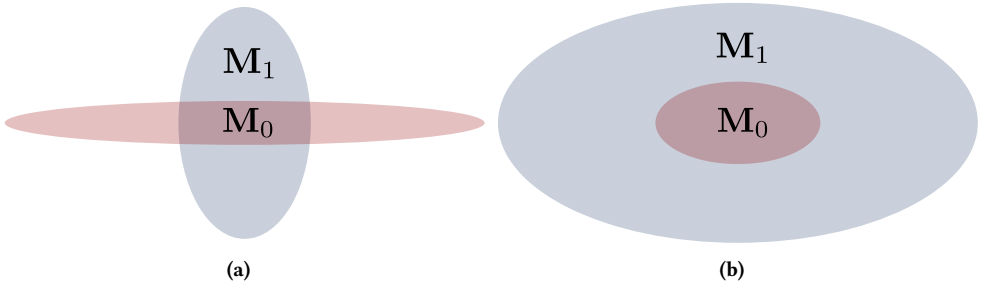


Figure 4.2: An example of failure modes for common singularity indices. (a) The manipulability index fails for configurations involving elongated ellipsoids with one or more axes of near zero length. (b) The dexterity index fails when the ellipsoids are of uniformly small scale.

A common indicator used to detect the proximity of a configuration to a singularity is known as the *manipulability index*, expressed as

$$m = \sqrt{\det(\mathbf{J}\ \mathbf{J}^{\mathsf{T}})}\,. \tag{4.3}$$

The manipulability index also admits a geometric interpretation, since the index value is proportional to the volume of the manipulability ellipsoid. Because singularities involve manipulability ellipsoids with one or more axes of zero length, and therefore zero volume, the manipulability index can be used to detect such configurations. Moreover, the differentiability of Eq. (4.3) has resulted in many singularity avoidance methods that maximize the volume of the manipulability ellipsoid by optimizing the manipulability index [Dufour and Suleiman, 2017, Marić et al., 2016].

The manipulability index is not as effective in detecting configurations that are in close proximity to singularities. The scenario in Fig. 4.2a shows an ellipsoid of relatively large volume with an axis length close to zero. This also presents a challenge for singularity avoidance methods, since optimizing for configurations with a higher manipulability ellipsoid volume does not guarantee that they are further from singular regions.

4.3.2 Dexterity index

Another common index used to detect and avoid singularities is the *dexterity index* [Salisbury and Craig, 1982],

$$\kappa = \frac{\sigma_{max}}{\sigma_{min}}\,, \tag{4.4}$$

where σ_{max} and σ_{min} are the maximal and minimal singular values of the Jacobian, also known as the condition number. The value of the dexterity index for a given configuration is associated with the distortion of task space sensitivity. This value can be interpreted geometrically as a ratio between the longest and shortest axis lengths of the manipulability ellipsoid.

The geometric interpretation again reveals that an important drawback of this approach lies in the inability to encode the scale of the manipulability ellipsoid. Since the ratio of axis lengths becomes infinite only when the configuration is exactly singular and gives no information about the ellipsoid size, it is impossible to use the dexterity index to provide a measure of proximity to a singularity, as shown in Fig. 4.2b. Similarly, optimizing for configurations with a dexterity index close to one results in homogeneous task space sensitivity, without providing any idea of its magnitude. Note that there exist other measures of kinematic sensitivity used in

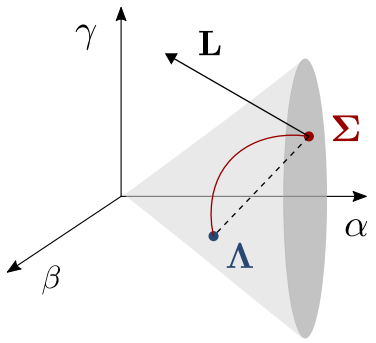


Figure 4.3: Visualization of the convex cone formed by the set $\mathcal{S}^2_{++}$ of matrices of the form $\begin{pmatrix} \alpha & \beta \\ \beta & \gamma \end{pmatrix}$. The matrices $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$ lie inside the cone, and the matrix $\mathbf{L} = \mathrm{Log}_{\mathbf{\Sigma}}(\mathbf{\Lambda})$ lies in the tangent space of $\mathbf{\Sigma}$. The shortest path connecting $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$ is the geodesic shown in red. Note that the length of the geodesic differs from the length of the dashed straight line in Euclidean space.

singularity avoidance that are tailored for specific problem instances such as parallel manipulators or walking robots [Cardou et al., 2010].

4.4 A Geometry-Aware Singularity Index

The manipulability and dexterity indices described in Section 4.3 are commonly used for detecting and avoiding singular configurations. In this section, we show that the differential-geometric characterization of manipulability ellipsoids introduced in [Jaquier et al., 2020] induces a Riemannian metric that naturally defines a distance between manipulability ellipsoids. We use this distance to specify a family of geometrically intuitive singularity indices, parameterized by a choice of a specific reference ellipsoid.

4.4.1 The Riemannian Manifold of SPD Matrices

Manipulability ellipsoids of non-singular configurations correspond to the set of symmetric matrices with strictly positive eigenvalues. This set is known as the set of symmetric positive definite (SPD) matrices,

$$\mathcal{S}^p_{++} = \{\mathbf{\Sigma} \,|\, \mathbf{\Sigma} = \mathbf{\Sigma}^T,\, \mathbf{x}^T \mathbf{\Sigma} \mathbf{x} > 0 \,\forall \mathbf{x} \in \mathbb{R}^p \setminus \{\mathbf{0}\}\}\,, \tag{4.5}$$

which forms a convex cone in $\mathbb{R}^K$, where $K = p\,(p+1)/2$. As shown in Fig. 4.3, a straight line in $\mathbb{R}^K$ does not represent the shortest path between points on the $\mathcal{S}^p_{++}$ manifold. This means that we cannot rely on the Euclidean metric to induce a distance that is useful when reasoning about the similarity of ellipsoids. Fortunately, Riemannian geometry equips us with the tools necessary to establish an alternative, Riemannian metric on the set $\mathcal{S}^p_{++}$, forming a Riemannian manifold (as described in Section 2.1.2). This manifold characterization makes it possible to define a geometrically-appropriate distance on this manifold.

As shown in Fig. 4.3, the tangent space of elements in $\mathcal{M} \equiv \mathcal{S}^p_{++}$ is the space of symmetric matrices $T_{\mathbf{\Sigma}}\mathcal{M} \equiv \mathrm{Sym}_p$, which is a vector space in $\mathbb{R}^K$. A Riemannian metric on $T_{\mathbf{\Sigma}}\mathcal{M}$ with a particular set of properties can then be chosen.

Definition 10 (Riemannian metric on $\mathcal{S}^p_{++}$ [Pennec et al., 2006]). *For some $\mathbf{\Sigma} \in \mathcal{M}$, a positive-definite*

inner product of two elements $\mathbf{Z}_1, \mathbf{Z}_2 \in T_{\mathbf{\Sigma}}\mathcal{M}$ *can be defined as*

$$\langle \mathbf{Z}_1, \mathbf{Z}_2 \rangle_{\mathbf{\Sigma}} = \mathrm{Tr}(\mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{Z}_1 \, \mathbf{\Sigma}^{-1} \, \mathbf{Z}_2 \, \mathbf{\Sigma}^{-\frac{1}{2}}) \,, \tag{4.6}$$

and is called the Riemannian metric on $\mathcal{M}$.

This Riemannian metric is invariant to affine transformations [Pennec et al., 2006]: for any element of the degree-p general linear group $\mathbf{A} \in GL_p$ with the group action $\mathbf{A} \circ \mathbf{\Sigma} = \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T$ on the space Sym_p, we have

$$\langle \mathbf{A} \circ \mathbf{Z}_1, \mathbf{A} \circ \mathbf{Z}_2 \rangle_{\mathbf{A}\circ\mathbf{\Sigma}} = \langle \mathbf{Z}_1, \mathbf{Z}_2 \rangle_{\mathbf{\Sigma}}. \tag{4.7}$$

This property will be useful when considering how the manipulability ellipsoid varies depending on configuration.

The Riemannian metric defined in Eq. (4.6) allows us to determine the lengths of curves on $\mathcal{S}^p_{++}$. Of particular interest is the length of geodesics connecting two points on the manifold, known as the Riemannian distance.

Definition 11 (Riemannian distance on $\mathcal{S}^p_{++}$ [Pennec et al., 2006])**.** *For* $\mathcal{M}$ *equipped with the Riemannian metric of Eq.* (5.11)*, the Riemannian distance between* $\mathbf{\Sigma}, \mathbf{\Lambda} \in \mathcal{M}$ *is defined as*

$$d(\mathbf{\Sigma}, \mathbf{\Lambda}) = \left\| \mathrm{Log}_{\mathbf{\Sigma}}(\mathbf{\Lambda}) \right\|_{\mathrm{F}} \,, \tag{4.8}$$

where F *denotes the Frobenius norm and*

$$\mathrm{Log}_{\mathbf{\Sigma}}(\mathbf{\Lambda}) = \mathbf{\Sigma}^{\frac{1}{2}} \log\left(\mathbf{\Sigma}^{-\frac{1}{2}} \mathbf{\Lambda} \mathbf{\Sigma}^{-\frac{1}{2}}\right) \mathbf{\Sigma}^{\frac{1}{2}} \tag{4.9}$$

is the logarithmic map on defined on the $\mathcal{S}^p_{++}$ *manifold.*

In fact, this is the length of the geodesic connecting two non-singular manipulability ellipsoids [Jaquier et al., 2020]. It is important to note that the exponential map covering $\mathcal{S}^p_{++}$ is uniquely defined everywhere on the manifold, meaning that the logarithmic map can always be computed.

4.4.2 The Riemannian Distance as a Singularity Index

The Riemannian distance in Eq. (4.8) gives an affine-invariant distance between the manipulability ellipsoid $\mathbf{M}$ associated with some configuration and a reference ellipsoid $\mathbf{\Sigma}$. By choosing a geometrically appropriate $\mathbf{\Sigma}$, this distance can be used to retrieve a measure of proximity of a given configuration to a singularity.

Consider the manipulability ellipsoid $\mathbf{M}(\boldsymbol{\theta}) = \mathbf{J}\,\mathbf{J}^\intercal$ of a robot in some configuration $\boldsymbol{\theta} \in \mathcal{C}$ and some reference ellipsoid $\mathbf{\Sigma}$. We define the squared length of the geodesic (i.e., the squared Riemannian distance) connecting $\mathbf{M}$ and $\mathbf{\Sigma}$ as

$$\lambda = \left\| \mathrm{Log}_{\mathbf{\Sigma}}(\mathbf{M}) \right\|^2_{\mathrm{F}} \,, \tag{4.10}$$

which follows directly from Eq. (4.8). In the context of singularity detection and avoidance, we refer to the scalar λ in Eq. (4.10) as the geometry-aware singularity index.

As a robot moves and changes its configuration, the manipulability ellipsoid varies in shape, size, and orientation. In Section 4.3 we noted that conventional indices such as *dexterity* and *manipulability* generally represent only one particular property of the manipulability ellipsoid (i.e., volume or axis length ratio). In contrast, λ is

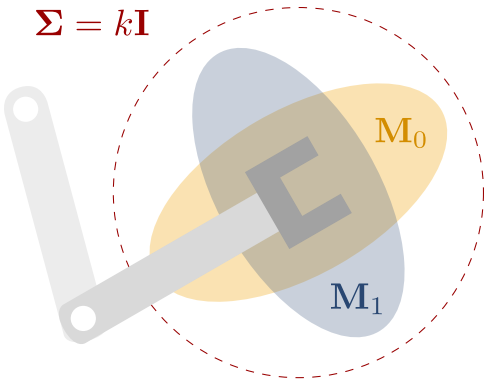


Figure 4.4: Singularity avoidance formulation s-IK, in which the distance between a sphere $\boldsymbol{\Sigma} = k\,\mathbf{I}$ and the manipulability ellipsoid $\mathbf{M}$ is minimized. Note that both $\mathbf{M}_0$ and $\mathbf{M}_1$ are the same distance from $\boldsymbol{\Sigma}$, since the squared metric λ is independent of orientation.

a function of all axis lengths, as well as the ellipsoids' shape and orientation. This presents an opportunity to avoid many of the issues that are common when applying the conventional indices; the problems arise because one property (e.g., the ellipsoid volume) may remain constant while the ellipsoid itself changes. Importantly, the affine-invariance property of the underlying Riemannian metric described in Eq. (5.11) guarantees that the value of our index is always determined by the relative difference between two ellipsoids.

4.4.3 Choosing the Reference Ellipsoid $\boldsymbol{\Sigma}$

The utility of the geometry-aware singularity index in Eq. (4.10) clearly depends on an appropriate choice of the reference ellipsoid $\boldsymbol{\Sigma}$, which must be selected with the goal of singularity avoidance in mind. In this section, we propose two possible choices that help to avoid some of the degeneracies that may occur when using other common indices.

Choosing $\boldsymbol{\Sigma} = k\,\mathbf{I}$

Consider selecting a reference ellipsoid $\boldsymbol{\Sigma}$ that has a spherical shape with a radius greater or equal to the length of the longest possible axis of the manipulability ellipsoid $\mathbf{M}$, as shown in Fig. 4.4. Formally, this class of ellipsoids can be expressed as

$$\boldsymbol{\Sigma} = k\mathbf{I},\; k \geq \sigma_{max}^2\,, \tag{4.11}$$

where σ_{max} is the largest eigenvalue of $\mathbf{M}$. The scaling factor k in Eq. (4.11) is chosen to be larger than the largest manipulability ellipsoid eigenvalue, forming a sphere that encapsulates the ellipsoid. By choosing

$$k \geq \mathrm{Tr}\left(\mathbf{M}(\boldsymbol{\theta})\right)\;, \forall \boldsymbol{\theta} \in \mathcal{C}, \tag{4.12}$$

we ensure that the geometry-aware singularity index (denoted by λ) decreases with the increase of the singular values of the Jacobian, since the manipulability ellipsoid will always be contained within the reference ellipsoid $\boldsymbol{\Sigma}$. Because the maximum volume of the manipulability ellipsoid for any manipulator is bounded, k can also be found empirically by moving the robot and increasing k whenever Eq. (4.12) fails to hold.

Since the sphere is symmetric, λ is invariant to the orientation of the manipulability ellipsoid. This result follows directly from the affine-invariance property described in Section 4.4.2 and can be proven easily by inserting

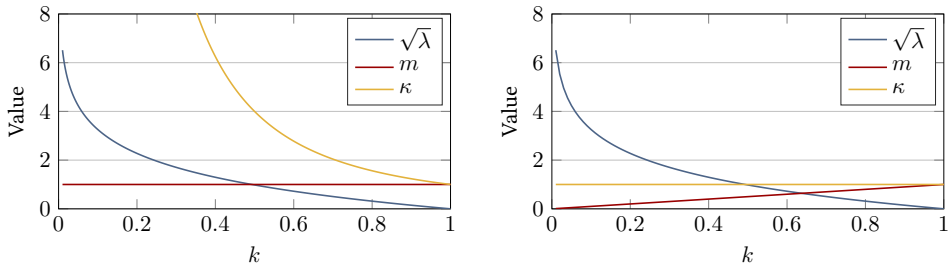


Figure 4.5: Performance of our geometry-aware singularity index λ for $\mathbf{\Sigma} = \mathrm{Tr}(\mathbf{M})\,\mathbf{I}$ in cases where the manipulability index m (left) and dexterity index κ (right) are ambiguous. The left plot corresponds to the constant volume scenario shown in Fig. 4.2a, while the right plot represents the constant shape scenario from Fig. 4.2b.

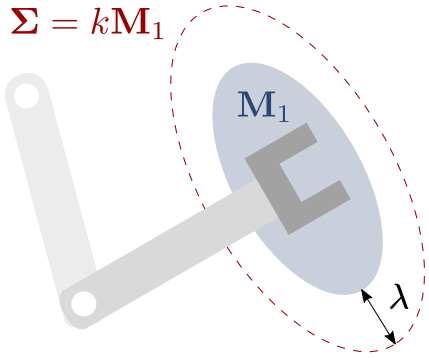


Figure 4.6: Singularity avoidance formulation s-IK2, in which the reference ellipsoid $\mathbf{\Sigma}$ produced at each iteration is a scaled variant of the original.

$\mathbf{\Sigma} = k\,\mathbf{I}$ into Eq. (4.7). Such a property is desirable in a singularity avoidance context because the orientation of the manipulability ellipsoid does not change the singular values of the Jacobian.

We can study the behavior of the geometry-aware singularity index with this choice of $\mathbf{\Sigma}$ in scenarios where the more common manipulability and dexterity indices fail, as shown in Fig. 4.2a and Fig. 4.2b. We begin with an arbitrary ellipse that has two axes of equal length; the two axes are then progressively scaled by the factors k and $1/k$, respectively, effectively 'squeezing' the ellipse in a way that maintains the overall area. In the top plot of Fig. 4.5, we see that the manipulability index remains unchanged because the area is constant, making it impossible to differentiate between the nearly singular 'squeezed' ellipse and a circle. The dexterity index increases with k and reaches a value of 1 for the circular shape. Our geometry-aware singularity index decreases as $k \to 1$, since the ellipse moves closer to the circular $\mathbf{\Sigma}$ on the manifold. Next, we again consider an arbitrary ellipse with axes of equal length that are both progressively scaled by k, uniformly inflating the ellipse. The bottom of plot of Fig. 4.5 shows that the dexterity index does not differentiate between the smaller and larger ellipses because the ratio of the maximal and minimal axis lengths remains unchanged. The manipulability index is proportional to the area of the ellipse and thus increases as $k \to 1$, while our geometry-aware singularity index decreases as the ellipse expands towards a circle of radius of 1. These results demonstrate that our proposed index avoids some notable 'blind spots' of the two commonly-used indices.

Choosing $\boldsymbol{\Sigma} = k\mathbf{M}$

As described in Section 3.3.1, control and planning algorithms of the form in Eq. (3.13) commonly integrate criteria such as singularity avoidance by directly making use of the gradient of the relevant indices. This reveals an interesting instance of our proposed index, where the reference ellipsoid is a scaled version of the manipulability ellipsoid at the current configuration of the robot. We begin by choosing

$$\boldsymbol{\Sigma} = k\mathbf{M}_0, \;\; k \geq 1, \tag{4.13}$$

where $\mathbf{M}_0$ is the manipulability ellipsoid evaluated at each time step of the tracking algorithm. Inserting $\boldsymbol{\Sigma}$ into Eq. (4.10), the inside of the matrix logarithm evaluates to

$$(k\mathbf{M}_0)^{-\frac{1}{2}} \, \mathbf{M} \, (k\mathbf{M}_0)^{-\frac{1}{2}} \Big|_{\mathbf{M}=\mathbf{M}_0} = k^{-1}\mathbf{I} \tag{4.14}$$

at each operating point. While it is clear that this instance of the index evaluates identically at every operating point, we can gain additional insight by considering the gradient of the index.

In Section 4.5.1, we explain that computing the gradient analytically for our index is generally nontrivial because of the requirement that the element-wise derivative of $\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{M}\boldsymbol{\Sigma}^{-\frac{1}{2}}$ be commutative with its inverse. However, it follows from Eq. (4.14) that the identity in Eq. (4.20) holds at the operating point, and we are able to obtain the partial derivative

$$\frac{\partial \lambda}{\partial \theta_i} = -2 \log(k) \, \mathrm{Tr}\left(\frac{\partial \mathbf{J}}{\partial \theta_i} \mathbf{J}^{\dagger} \right), \tag{4.15}$$

where $\mathbf{J}^{\dagger}$ is the Jacobian pseudo-inverse. As shown in Fig. 4.6, this gradient, constructed from partial derivatives in Eq. (4.15), gives a direction in which $\mathbf{M}$ expands along all of its axes. Interestingly, Eq. (4.15) is proportional to the gradient of the manipulability index [Marić et al., 2019], revealing a differential-geometric generalization of manipulability maximization approaches. In fact, our results in Section 4.6.1 confirm that this instance of the proposed index performs similarly to manipulability maximization in compatible task space tracking algorithms. Performance for this choice of $\boldsymbol{\Sigma}$ hinges on the rate at which $\mathbf{M}_0$ is updated, as all the above properties are lost outside the neighborhood of the operating point.

4.5 Singularity Avoidance

This section describes how the geometry-aware singularity index defined by Eq. (4.8) can be used for singularity avoidance in a common family of task space control algorithms. We achieve this by integrating the proposed index into velocity inverse kinematics problem described in Section 3.3.2 as well as the pose inverse kinematics problem described in Section 3.3.1. The formulation applied herein can be seen as an extension of that in [Jaquier et al., 2020], where a Jacobian-based approach is employed to follow reference directions in the tangent space of SPD matrices in order to reach a specific, desired ellipsoid.

Reaching a specific orientation of the manipulability ellipsoid is generally not important for ensuring singularity avoidance, which is the task of maintaining sufficient numerical conditioning of the Jacobian. Therefore, we directly optimize the squared affine-invariant distance between the current manipulability ellipsoid $\mathbf{M}$ and a reference ellipsoid $\boldsymbol{\Sigma}$, with the overall goal of changing the ellipsoid's shape to extend "sensitive" axes. This distance is reflected in the geometry-aware singularity index λ, which can be added to the cost function of a

common nonlinear programming formulation of the inverse kinematics problem

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \mathbf{W} (\boldsymbol{\theta} - \boldsymbol{\theta}_0) + \alpha\, \lambda(\boldsymbol{\theta}) \\ \text{s.t.} \quad & \mathbf{f}(\boldsymbol{\theta}) = \mathbf{T}_{goal} \in \mathrm{SE}(3)\ , \end{aligned} \tag{4.16}$$

where $\mathbf{W}$ is a weighting matrix used to prioritize certain joints, $\mathbf{T}_{goal}$ is the goal end-effector pose, and α is a gain parameter. A solution to Eq. (4.16) can be found by iteratively solving a sequence of quadratic programs obtained by linearizing the cost and constraints. This sequential quadratic programming (SQP) approach has previously been shown to be effective when designing singularity-robust kinematic controllers [Dufour and Suleiman, 2017, Zhang et al., 2016]. By adding a velocity-minimizing term to the cost and joint velocity constraints, we arrive at the following QP

$$\begin{aligned} \min_{\dot{\boldsymbol{\theta}}} \quad & \dot{\boldsymbol{\theta}}^T \mathbf{W} \dot{\boldsymbol{\theta}} + \alpha (\nabla \lambda(\boldsymbol{\theta}_0))\, \dot{\boldsymbol{\theta}} \\ \text{s.t.} \quad & \mathbf{J}\dot{\boldsymbol{\theta}} = \dot{\boldsymbol{\xi}} \\ & \dot{\boldsymbol{\theta}}_{min} \leq \dot{\boldsymbol{\theta}} \leq \dot{\boldsymbol{\theta}}_{max}\ , \end{aligned} \tag{4.17}$$

which is exactly the task space tracking formulation shown in Eq. (3.13) for $\mathbf{c} = \alpha(\nabla\lambda(\boldsymbol{\theta}_0))\,, \mathbf{A} = \mathbf{J}$ and $\mathbf{b} = \dot{\boldsymbol{\xi}}$, with an added inequality constraint limiting the joint velocities. In both control and inverse kinematics applications, this QP is redefined at each time instance or iteration and new $\nabla\lambda$ and $\mathbf{J}$ are calculated to reflect the current configuration. The joint velocity limits serve the additional purpose of enforcing joint position limits, since the velocity limits can be changed at each iteration to reflect the space of locally feasible joint motions. Depending on the choice of the gain parameter α, the robot will be guided in a direction in which the overall distance from singularities increases or decreases.

Problems of the form in Eq. (4.17) have been shown to allow for a wide variety of additional constraints, such as collision avoidance [Schulman et al., 2013] and manipulability maximization [Dufour and Suleiman, 2017]. Similarly, the minimization of the geometry-aware singularity index can be integrated as a secondary task in the redundancy resolution framework from Eq. (3.14). This can be done by setting the cost function to $\|(\nabla\lambda)^\dagger \dot{\lambda} - \dot{\boldsymbol{\theta}}\|^2$, where $(\nabla\lambda)^\dagger$ is the left pseudoinverse of the geometry aware singularity index gradient and $\dot{\lambda}$ is the desired rate of change of the index. However, we have empirically found that Eq. (4.17) gives better performance in observed tracking tasks.

4.5.1 Gradient Computation

Optimization methods used in control and kinematic synthesis require the gradient $\nabla\lambda$ to produce joint displacements that avoid singularities. The gradient can be expressed as a concatenation of partial derivatives of λ with respect to the joint positions θ_i,

$$\nabla\lambda = \begin{bmatrix} \dfrac{\partial\lambda}{\partial\theta_0} \cdots \dfrac{\partial\lambda}{\partial\theta_n} \end{bmatrix} \in \mathbb{R}^n\,.$$

These partial derivatives can be obtained using elementary matrix calculus as

$$\frac{\partial\lambda}{\partial\theta_i} = 2\,\mathrm{Tr}\left(\frac{\partial \log(\boldsymbol{\varpi})}{\partial\theta_i} \log(\boldsymbol{\varpi})^{\intercal} \right)\,, \tag{4.18}$$

where

$$\boldsymbol{\varpi} = \boldsymbol{\Sigma}^{-\frac{1}{2}} \mathbf{M} \boldsymbol{\Sigma}^{-\frac{1}{2}}\,. \tag{4.19}$$

The partial derivative of $\log(\boldsymbol{\varpi})$ admits the closed form solution

$$\frac{\partial \log(\boldsymbol{\varpi})}{\partial \theta_i} = \boldsymbol{\varpi}^{-1} \frac{\partial \boldsymbol{\varpi}}{\partial \theta_i} \tag{4.20}$$

only if the matrices $\boldsymbol{\varpi}^{-1}$ and $\frac{\partial \boldsymbol{\varpi}}{\partial \theta_i}$ commute. Unfortunately, the matrices in Eq. (4.20) are generally not commutative and the identity is therefore invalid when considering $\boldsymbol{\Sigma}$ of an arbitrary shape.

Instead of finding the partial derivatives analytically, we can evaluate them numerically by leveraging a result from computational matrix analysis. We begin with a lemma showing that Eq. (4.20) can be expressed using directional derivatives.

Lemma 1 ([Dattorro, 2005]). *The partial derivatives of* $\log(\boldsymbol{\varpi})$ *with respect to individual elements of* $\boldsymbol{\theta}$ *can be defined as*

$$\frac{\partial \log(\boldsymbol{\varpi})}{\partial \theta_i} = \nabla_{\frac{\partial \boldsymbol{\varpi}}{\partial \theta_i}} \log(\boldsymbol{\varpi}), \tag{4.21}$$

where $\nabla_{\mathbf{E}} \mathbf{f}(\mathbf{g})$ *is the directional (Fréchet[a]) derivative of* $\mathbf{f}$ *at* $\mathbf{g}$ *in the direction* $\mathbf{E}$.

Proof. Using the chain rule for matrix-valued functions, the partial derivative of $\mathbf{f}(\mathbf{g}(\boldsymbol{\theta})) : \mathbb{R}^n \to \mathbb{R}^{K \times K}$ with respect to $\boldsymbol{\theta} \in \mathbb{R}^n$ is expressed as

$$\frac{\partial \mathbf{f}}{\partial \theta_i} = \nabla \mathbf{f}(\mathbf{g}) \cdot \frac{\partial \mathbf{g}}{\partial \theta_i}. \tag{4.22}$$

The product definition of directional derivative of $\mathbf{f}$ at $\mathbf{g}$ in the $\mathbf{E}$ direction is given by

$$\nabla_{\mathbf{E}} \mathbf{f}(\mathbf{g}) = \nabla \mathbf{f}(\mathbf{g}) \cdot \mathbf{E}. \tag{4.23}$$

The equivalence of Eq. (4.22) and Eq. (4.23) is self-evident. □

[a] Generalization of the directional derivative to vector-valued functions.

From Lemma 1, it follows that the partial derivative of the logarithm in Eq. (4.18) can be computed using Eq. (4.21). We first compute the direction $\frac{\partial \boldsymbol{\varpi}}{\partial \theta_i}$, which is a derivative with a closed form expression

$$\frac{\partial \boldsymbol{\varpi}}{\partial \theta_i} = \boldsymbol{\Sigma}^{-\frac{1}{2}} \left(\frac{\partial \mathbf{J}}{\partial \theta_i} \mathbf{J}^T + \mathbf{J} \frac{\partial \mathbf{J}}{\partial \theta_i}^T \right) \boldsymbol{\Sigma}^{-\frac{1}{2}}. \tag{4.24}$$

Once the direction is obtained, the directional derivative in Eq. (4.18) can be accurately and efficiently computed using the identity introduced in [Higham, 2008], which we formalize in the following proposition:

Proposition 1. *The partial derivatives of* $\log(\boldsymbol{\varpi})$ *with respect to individual elements of* $\boldsymbol{\theta}$ *can be computed using the identity*

$$\log\left(\begin{bmatrix} \boldsymbol{\varpi} & \frac{\partial \boldsymbol{\varpi}}{\partial \theta_i} \\ \mathbf{0} & \boldsymbol{\varpi} \end{bmatrix} \right) = \begin{bmatrix} \log(\boldsymbol{\varpi}) & \frac{\partial \log(\boldsymbol{\varpi})}{\partial \theta_i} \\ \mathbf{0} & \log(\boldsymbol{\varpi}) \end{bmatrix}. \tag{4.25}$$

Proof. (Theorem 3.6 in [Higham, 2008]) Let $\mathbf{f}$ be a differentiable matrix function and $\mathbf{g} \in \mathbb{R}^{n \times n}$ be a symmetric matrix differentiable at $t = 0$. Let

$$\mathbf{g}(t) = \mathbf{g} + t\mathbf{E},$$

and we have the identity

$$\mathbf{f}\left(\begin{bmatrix}\mathbf{g} & \mathbf{E}\\ \mathbf{0} & \mathbf{g}\end{bmatrix}\right) = \begin{bmatrix}\mathbf{f}(\mathbf{g}) & \nabla_{\mathbf{E}}\mathbf{f}(\mathbf{g})\\ \mathbf{0} & \mathbf{f}(\boldsymbol{\tau})\end{bmatrix},$$

where $\nabla_{\mathbf{E}}\mathbf{f}(\mathbf{g})$ is the directional derivative of $\mathbf{f}$ at $\mathbf{g}$ in direction $\mathbf{E}$. From Eq. (4.19) and Eq. (4.24) it is clear that for $\mathbf{g} = \boldsymbol{\varpi}$ and $\mathbf{E} = \frac{\partial \boldsymbol{\varpi}}{\partial \theta_i}$ the symmetry assumptions hold, completing the proof. □

Since the size of the matrix representing the manipulability ellipsoid is generally at most 6×6, computing the matrix in Eq. (4.25) remains computationally tractable and the overall computation time negligible. This result allows us to explore arbitrary choices of the reference ellipsoid $\boldsymbol{\Sigma}$ in Eq. (4.19), enabling the adaptation of λ to both the structure of the robot and the task.

4.6 Experimental Results

In this section we present experimental results for the proposed geometry-aware singularity avoidance index when implemented within the QP-based task space control formulation defined by Eq. (4.17). Specifically, we consider the two index variants with reference ellipsoids $\boldsymbol{\Sigma}$ defined in Section 4.4.3 (labeled s-IK and s-IK2). In order to validate the benefits of using a Riemannian metric, we also evaluate a singularity index derived from the standard Euclidean metric

$$\lambda_E = \|\mathbf{M} - \boldsymbol{\Sigma}\|_F^2, \tag{4.26}$$

where a spherical $\boldsymbol{\Sigma}$ is selected as described in Section 4.4.3. As part of our proposed approach, the gradient of this index is integrated into the cost Eq. (4.17) in place of the linear term; the resulting formulation is labeled e-IK. We also compare our index to the manipulability maximization method from Dufour and Suleiman [2017], where a manipulability gradient term is again added as the linear component of the cost in Eq. (4.17); the resulting formulation is labeled m-IK. As a baseline, we use a standard approach to task space control obtained by setting $\alpha = 0$ in Eq. (4.17); this last formulation is simply labelled IK.

In our evaluation, we perform two benchmark experiments involving a pair of common tasks: reaching and path following. First, in Section 4.6.1 we demonstrate how our method performs in a simple reaching task, where a goal end-effector position must be attained while maximizing the overall distance from singular regions. Next, in Section 4.6.2 we demonstrate how our method can be used to guide the manipulator away from singularities while following a circular end-effector path. All experiments were performed on a laptop computer with an Intel i7-8750H CPU running at 2.20 GHz and with 16 GB of RAM.

4.6.1 Reaching Task

We begin by examining how the formulation given in Eq. (4.17) can be used to solve reaching tasks, where the end-effector needs to reach a desired goal position. In our analysis, we consider the class of planar kinematic chains with an increasing number of DoF, as well as three robotic manipulators commonly used in collaborative, assistive, and research robotics. We purposely avoid specifying a goal orientation in order to induce kinematic redundancy that can be used to optimize the singularity avoidance indices being tested. The overall performance is determined by comparing the minimal and maximal singular values of the Jacobian in the final configuration, as these values provide a definitive indicator of singularity robustness for a given configuration. We make the assumption that the joint limits and dynamic effects are accounted for by the velocity constraints at each iteration, making this problem similar to a standard inverse kinematics problem. The experiment consists of

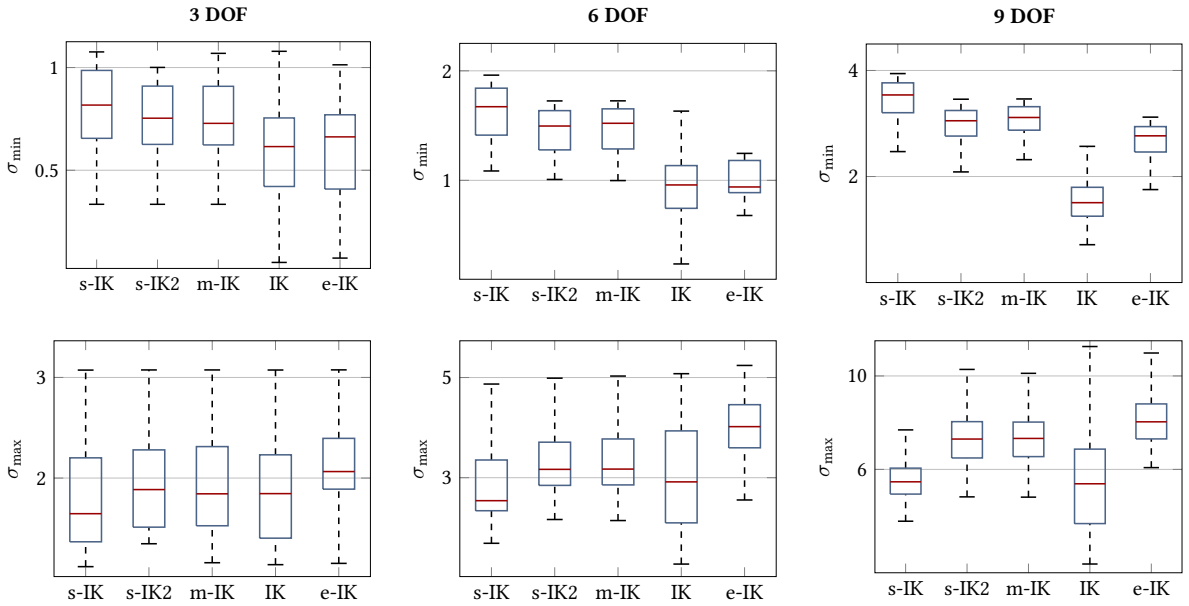


Figure 4.7: Results of solving 200 random inverse kinematics (IK) problems; each column corresponds to IK solutions for a planar manipulator with a different number of DoF. The plots in the top row show the minimal singular values σ_{min} of the manipulator Jacobian in the final configuration, while the plots in the bottom row show the maximal singular values σ_{max}. The two leftmost boxes in each plot, labeled s-IK and s-IK2, represent our method with for different choices of $\boldsymbol{\Sigma}$. The box labeled m-IK corresponds to the method in Dufour and Suleiman [2017], while the box labeled IK shows the results without optimizing for singularity avoidance. Finally, the box labeled e-IK shows the results obtained when using a Euclidean metric.

performing 200 random (and randomly initialized) reaching tasks, while respecting upper and lower limits on joint velocities.

First, we examine the results for three, six, and nine DoF planar kinematic chains with joint velocities limited to $\frac{\pi}{8}$ rad/s$(22.5°)$; the results are summarized by the box plots in Fig. 4.7. For methods s-IK, s-IK2, and m-IK, we chose $\alpha = 1$, since it produced satisfactory singularity avoidance results with a similar number of successes across the board. The gradient of Eq. (4.26), used in the e-IK formulation, generally has a larger magnitude and so we use $\alpha = 0.1$ to ensure numerical stability. Examining the top row of Fig. 4.7, we see that the method labeled s-IK, corresponding to the reference ellipsoid $\boldsymbol{\Sigma} = k\mathbf{I}$ with $k = \text{Tr}(\mathbf{M}) \geq \sigma_{\text{max}}$ (updated at each iteration), achieves the highest median minimal singular value σ_{min}. Moreover, increasing the number of DoF further amplifies this effect, as there is a larger space of solutions that can be explored due to the higher degree of redundancy. This result is highly desirable from the perspective of singularity avoidance, since we are trying to avoid situations where the Jacobian is not invertible. However, minimizing λ in this case also results in $\mathbf{M}$ adopting a more spherical shape. This can be seen in the bottom row of Fig. 4.7, where s-IK produces a lower median maximal singular value σ_{max} than s-IK2 or m-IK, reflecting the spherical shape of the reference ellipsoid. We posit that the spherical shape results in a more uniform mobility profile for the end-effector, while not affecting the proximity to singular configurations. Alternatively, by choosing $\boldsymbol{\Sigma} = k\mathbf{M}$ with $k = 2$ (s-IK2), we achieve an overall increase of both the minimal and maximal singular values, very similar to that of m-IK. Intuitively, choosing $k = 2$ means that, at every iteration, we attempt to reach an ellipsoid that is twice the size of the current ellipsoid. The results can be interpreted by observing that the gradient in this case has the form of Eq. (4.15), which is very similar to the manipulability-based gradient Marić et al. [2019] used in trajectory optimization. The m-IK method from Dufour and Suleiman [2017] maximizes $\det(\mathbf{M})$, which

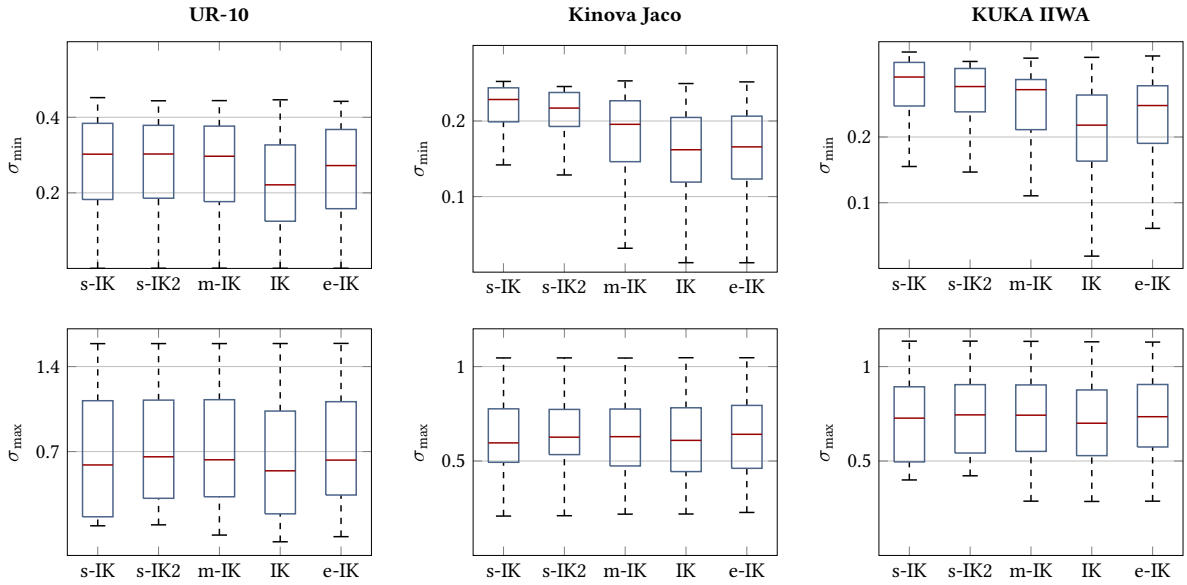


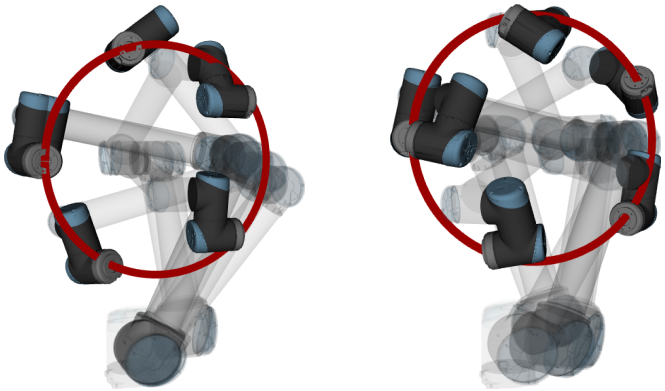
Figure 4.8: Results of solving 200 random inverse kinematics (IK) problems; each column corresponds to IK solutions for a different common manipulator. The plots in the top row show the minimal singular values $\sigma_{\min}$ of the manipulator Jacobian in the final configuration, while the plots in the bottom row show the maximal singular values $\sigma_{\max}$. The two leftmost boxes in each plot, labeled s-IK and s-IK2, represent our method with for different choices of $\boldsymbol{\Sigma}$. The box labeled m-IK corresponds to the method in Dufour and Suleiman [2017], while the box labeled IK shows the results without optimizing for singularity avoidance. Finally, the box labeled e-IK shows the results obtained when using a Euclidean metric.

translates to maximizing the overall volume of the manipulability ellipsoid. While this method outperforms the baseline IK approach, the median minimal singular value obtained using this method is noticeably smaller than that of s-IK. Finally, the e-IK method outperforms only the baseline in terms of the minimal singular value, as it appears to prioritize maximizing the largest singular value.

We have also performed the same experiment using common six and seven DoF robots: the Universal Robots UR10, the Kinova Jaco manipulator, and the KUKA IIWA 14. All joint velocities were again limited to at most $\frac{\pi}{8}$ rad/s in either direction and the gain value was increased to $\alpha = 10$ for s-IK, s-IK2, e-IK, and m-IK. Examining the results in Fig. 4.8, we note that the overall singular values are lower than that of the planar case. This is because the movement of these robots is more constrained in three dimensions than the movement of the planar mechanisms in two dimensions. Moreover, the majority of the translational mobility in these robots is produced by the first three joints, further exacerbating this phenomenon. The top row of Fig. 4.8 again shows that the s-IK method, using a spherical reference ellipsoid, produces superior results, with s-IK2 coming in as a close second. In the bottom row, we see that the maximum singular values are similar in all scenarios for all robots.

4.6.2 Circular Path Tracking

In this experiment, we evaluated the performance of our singularity avoidance formulation in an task space control scenario for several different manipulators by tracking a circular path with the end-effector. All manipulators used in this experiment have six or more DoF, while the task required only the position of the end-effector to remain on the defined path at all times. We are able to use the available kinematic redundancy to optimize the movement of each manipulator such that singular and near-singular configurations are avoided. Again, the formulation in Eq. (4.17) is used to produce a locally optimal joint displacement at each iteration, and we compare



(a) (b)

Figure 4.9: (a) Following a circular path with the UR10 manipulator without attempting to avoid singular configurations. Note the large change in the wrist configuration when the manipulator reaches the top of the circle. (b) Following a circular path with the UR10 manipulator while using the singularity avoidance formulation s-IK. Note that the wrist and base configurations change more slowly, improving the conditioning of the Jacobian.

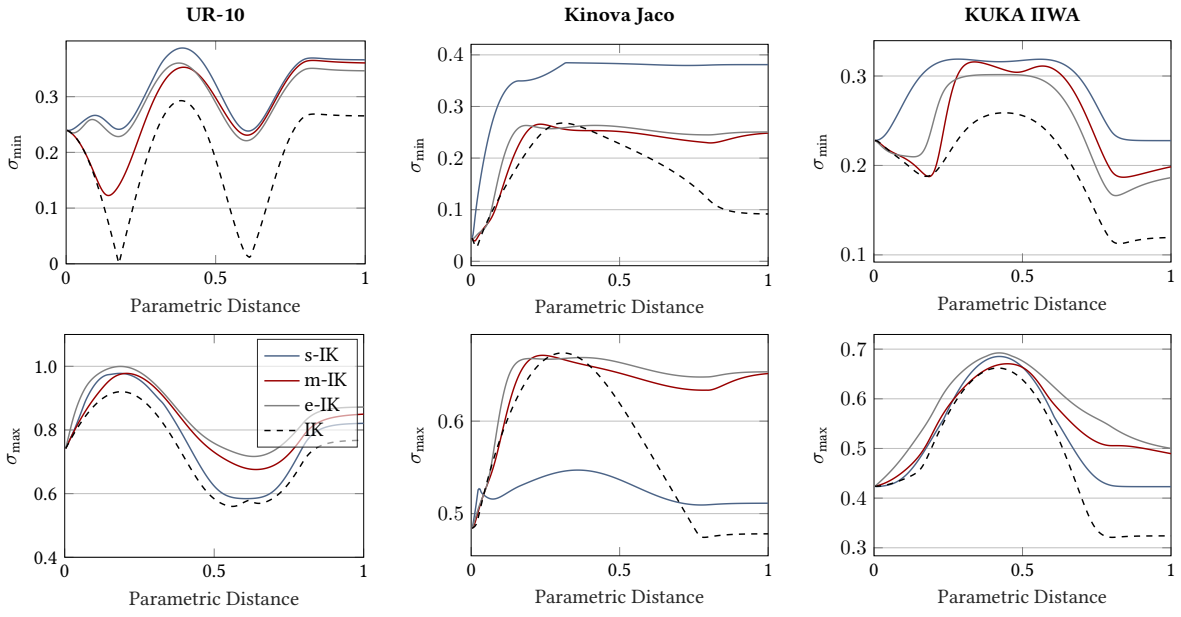


Figure 4.10: Jacobian conditioning throughout the circular trajectory, parameterized by $t = [0, 1]$. Minimal singular values σ_{min} are displayed in the top row and maximal singular value σ_{max} are displayed in the bottom row.

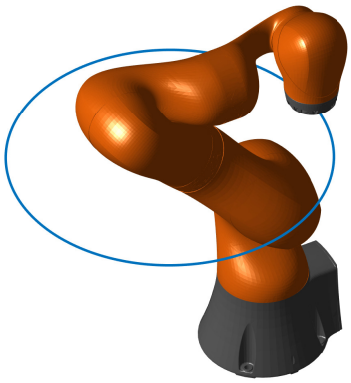


Figure 4.11: Visualization of the circular path followed by the KUKA IIWA 14 manipulator. The end effector orientation is constrained to remain constant throughout the task.

the s-IK, m-IK, e-IK, and IK methods.

In Fig. 4.10 we see that the maximal and minimal singular values of the Jacobian vary throughout the trajectory (as executed by the chosen methods). In the leftmost column, the regular IK method produces two nearly-singular configurations for the UR10, whereas all other methods avoid these singularities. The singularities are indicated by two dips in $\sigma_{\min}$ for the IK method, corresponding to the top and bottom of the circular path shown in Fig. 4.9a. At these points, the wrist configuration of the manipulator in Fig. 4.9a shifts significantly in order to continue to follow the position reference—this is a clear indicator that the arm is passing near a singularity. The m-IK method produces a trajectory closer to the first singularity than for s-IK or IK, since the m-IK maximizes the overall manipulability ellipsoid volume by prioritizing the increase of the two larger singular values. Our method outperforms both the IK and m-IK methods by maintaining a $\sigma_{\min}$ that is approximately two times larger than that produced by m-IK. The trade-off can be seen by observing the bottom row, where it is clear that the $\sigma_{\max}$ achieved by s-IK is somewhat lower than by m-IK or e-IK, while still outperforming IK. The e-IK method performs surprisingly very well in this scenario, maintaining a $\sigma_{\min}$ that is only slightly lower than that produced by s-IK, while achieving higher $\sigma_{\max}$. Fig. 4.9b shows the trajectories generated by s-IK; note the smaller variations in wrist movement, which suggests a better joint-to-task space mapping in terms of end-effector position change compared to Fig. 4.9a.

Results for the same task performed by the Kinova Jaco arm are shown in the middle column of Fig. 4.10 and offer a contrasting example. The performance of e-IK no longer matches s-IK in terms of singularity avoidance, as our method maintains a significantly higher $\sigma_{\min}$ throughout the trajectory. Interestingly, $\sigma_{\min}$ and $\sigma_{\max}$ reach a similar value for s-IK, which corresponds to the spherical shape of the reference ellipsoid used by this method. The rightmost column of Fig. 4.10 shows that results for the KUKA-IIWA confirm the observed trend. For all the robots, the baseline IK method is outperformed by all other methods, demonstrating that even using a 'geometrically improper' criterion based on the Euclidean metric helps to avoid singularities. Surprisingly, the performance of m-IK and e-IK is similar across all examples.

Finally, we have evaluated how all four methods perform when the path-following task also requires the end-effector orientation to remain fixed. We perform this task with the seven DoF KUKA IIWA 14 robot, since it has the redundancy needed to optimize for singularity avoidance when tracking the full end-effector pose reference. A visualization of the task can be seen in Fig. 4.11, while the changes of all Jacobian singular values are shown in Fig. 4.12. Our results indicate that using the baseline IK method results in a significant drop in

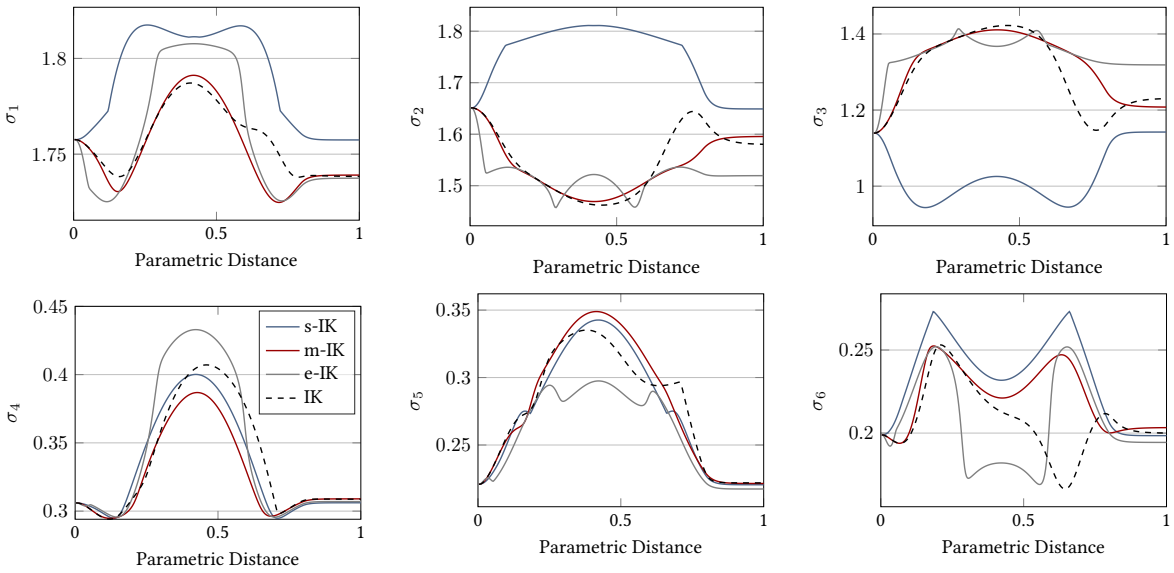


Figure 4.12: Jacobian conditioning throughout the execution of a circular trajectory with constant end-effector orientation for the KUKA IIWA 14 robot. The trajectory is parameterized by $t = [0, 1]$. Plots show how the Jacobian singular values σ_i change as the robot follows the path. Singular values are indexed as follows" $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_6$.

σ_{min} around $t = 0.6$, signaling that the robot is near a singularity. This singularity is avoided by the s-IK and m-IK methods, which produce similar changes of the lowest singular value. On the other hand, the performance of e-IK is arguably worse than that of the baseline IK method, with the manipulator configuration remaining nearly singular from $t = 0.3$ to $t = 0.55$. We see that s-IK also results in larger $\sigma_{\text{max}} = \sigma_1$ and σ_2 than all the other methods.

4.7 Summary and Conclusions

In this chapter, we described a novel method for singularity avoidance that uses a well-known Riemannian metric on the manifold of SPD matrices to formulate a computationally tractable optimization criterion based on geodesic length. We proved that our geometry-aware singularity index can be differentiated by computing directional derivatives using an identity from computational matrix analysis. Moreover, we showed that various and geometrically distinct criteria can be derived from this formulation by changing a single parameter (i.e., reference ellipsoid) and that some choices result in robustness to failure modes that are common for other indices. We demonstrated that the proposed index can be integrated into a common optimization formulation of task space reference tracking. The experimental results indicate that our index consistently achieves the largest minimal singular values among all of the methods compared. Moreover, we justified the use of the Riemannian metric by performing a comparison to an instance of our index that uses the standard Euclidean metric, which yields less consistent results and worse performance in terms of singularity avoidance. Finally, it is important to note that there may be other choices of the reference ellipsoid suitable for singularity avoidance, as well as choices specifically tailored to a given task or kinematic structure. We consider this one of the key advantages of our approach—our method can be tailored to a specific task to a greater degree than other indices such a the manipulability index.

As an avenue for future work, we note that the formulation presented herein can easily be integrated into

existing control and planning pipelines and that it would be interesting to benchmark their performance. We have previously integrated a manipulability maximization term into the trajectory optimization framework of [Marić et al., 2019] and this work was followed up by integrating the proposed index into a similar formulation that is better suited for nonlinear objectives [Petrović et al., 2020, 2021]. From a theoretical point of view, we are exploring possible equivalences between certain choices for the reference ellipsoid and existing singularity avoidance criteria. Furthermore, determining how a reference ellipsoid should be selected for a specific task may elucidate further advantages over more conventional methods.

4.7.1 Limitations

There are several limitations that should be considered when using the approach presented herein for singularity avoidance. First, it is crucial that the reference ellipsoid $\mathbf{\Sigma}$ encapsulates all manipulability ellipsoids that can be reached by the robot. This can be accomplished by defining a large ellipsoid beforehand, either empirically or analytically, as described for the spherical reference ellipsoid in Section 4.4.3. In Section 4.4.3 we have shown that the reference ellipsoid can also be redefined at each iteration and its lengths modified to ensure it remains larger than the manipulability ellipsoid. This option may provide greater numerical stability, assuming that the $\mathbf{\Sigma}$ is updated at an adequate rate. Another problematic scenario may occur when the initial manipulator configuration is itself singular, since the manifold geometry described in Section 4.4 holds only for non-singular ellipsoids. This situation is easily detected either by directly observing the singular values of the Jacobian or noting that matrix logarithm has failed—a small perturbation can be applied to the joint configuration to leave the singular region.

4.7.2 Associated Publications

Our research in singularity avoidance was initially set in the context of trajectory optimization, where several discrete states were interpolated with a continuous time-parameterized trajectory optimized for locally maximum manipulability.

1. Maric, F., Limoyo, O., Petrović, L., Petrović, I., Kelly, J. Manipulability Maximization Using Continuous-Time Gaussian Processes. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) Towards Robots that Exhibit Manipulation Intelligence Workshop

2. Marić, F., Limoyo, O., Petrović, L., Ablett, T., Petrović, I., Kelly, J. (2019, November). Fast manipulability maximization using continuous-time trajectory optimization. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (pp. 8258-8264).

This research project elucidated some fundamental drawbacks of using the manipulability index and led us to develop a novel singularity avoidance criterion based on a differential-geometric characterization of elliposids. The geometry-aware singularity index presented in this chapter is compatible with both instances of the inverse kinematics problem described in Section 3.3.

3. Marić, F., Petrović, L., Guberina, M., Kelly, J., Petrović, I. (2021). A Riemannian metric for geometry-aware singularity avoidance by articulated robots. Robotics and Autonomous Systems, 145, 103865.

In later work, we showed that this index can be used in trajectory optimization applications in place of the conventional manipulability index.

4. Petrović, L., Marić, F., Marković, I., Kelly, J., Petrović, I. (2021, October). Trajectory Optimization with Geometry-Aware Singularity Avoidance for Robot Motion Planning. In 2021 21st International Conference on Control, Automation and Systems (ICCAS) (pp. 1760-1765). IEEE.

Chapter 5

Distance-Geometric Inverse Kinematics

> One geometry cannot be more true than another; it can only be more convenient. Geometry is not true, it is advantageous.
>
> Robert M. Pirsig

Solving the inverse kinematics problem is a fundamental challenge in motion planning, control, and calibration for articulated robots. Kinematic models for these robots are typically parametrized by joint angles, generating a complicated mapping between the robot configuration and the end-effector pose. Alternatively, the kinematic model and task constraints can be represented using invariant distances between points attached to the robot. In the contribution described in this chapter, we formalize the equivalence of distance-based inverse kinematics and the distance geometry problem for a large class of articulated robots and task constraints. Unlike previous approaches, we use the connection between distance geometry and low-rank matrix completion to find inverse kinematics solutions by completing a partial Euclidean distance matrix through local optimization. Furthermore, we parametrize the space of Euclidean distance matrices with the Riemannian manifold of fixed-rank Gram matrices, allowing us to leverage a variety of mature Riemannian optimization methods. Finally, we show that bound smoothing can be used to generate informed initializations without significant computational overhead, improving convergence. We demonstrate that our inverse kinematics solver achieves higher success rates than traditional techniques, and substantially outperforms them on problems that involve many workspace constraints.

5.1 Motivation and Related Work

Various problems of a geometric nature can be expressed using points and their relative distances [Dokmanic et al., 2015]. In the case of actuated mechanisms such as robots, a geometrically intuitive alternative to joint angle parameters is obtained by considering points attached to the robot's structure [de Jalón, 2007]. The positions of these points on the robot and their relative distances can be used to describe the kinematic model of the robot [Blanchini et al., 2017, Le Naour et al., 2019b]. This approach unifies the configuration and task spaces of the kinematic model, eliminating nonlinearities that originate from forward kinematics in a variety of common workspace constraints. In contrast to a joint angle-based parametrization, searching over point positions is not restricted to feasible configurations of the robot, but to arbitrary "conformations" in Euclidean space.

A more compact representation is obtained by modelling the robot using *distances* as variables. Motion constraints of individual joint-link pairs can be modeled by defining distances between key points in the structure of the robot in a manner similar to [Porta et al., 2005b, Han and Rudolph, 2006]. Adopting this perspective allows us to set constraints on end-effector poses and joint angles by limiting the associated distances to a predetermined range. We use the distance geometry problem (DGP) described in Section 2.3 as a mathematical basis for our approach and derive simple and inclusive criteria for the compatibility of a robot mechanism with our method. Finally, we provide a free and open-source Python implementation of our algorithms and simulation experiments, which empirically show the effectiveness of our approach on a variety of robot models. Compared to typical IK formulations based on joint angles, experimental results indicate that our method often achieves higher success rates and faster convergence, and outperforms benchmark algorithms when many workspace constraints are present.

5.1.1 Distance Geometry

As stated in Section 2.3, the problem of distance geometry can be described as completing a partially-connected graph of interpoint distances. When a high degree of connectivity is present (i.e., most distances are known), the classical multidimensional scaling (MDS) algorithm [Cox and Cox, 2008] is often used. The EMBED algorithm models the problem of molecular conformation using distances, providing bounds on unknown distances using *bound smoothing* [Havel, 2002] and iteratively finding solutions for smaller problem instances. Larger problem instances that satisfy some additional assumptions can be solved with a branch-and-prune strategy [Liberti et al., 2008]. Convex relaxations of the DGP have been coupled with semidefinite programming methods in both chemistry and sensor network localization [Biswas et al., 2006, Leung and Toh, 2010].

Known distances can be arranged in an incomplete Euclidean distance matrix (EDM) [Dokmanic et al., 2015] of rank at most $K + 2$, where K is the dimension of the Euclidean space. In many applications, the DGP can be solved by determining the unknown EDM entries using a low-rank *matrix completion* approach [Nguyen et al., 2019b]. Recently, numerical optimization methods based on results from Riemannian geometry have been utilized to efficiently perform EDM completion [Vandereycken, 2012]. Mishra et al. solve the EDM completion problem using a Riemannian trust-region method by parametrizing the EDM with elements of a quotient manifold invariant to orthogonal transformations [Mishra et al., 2011]. Nguyen et al. present a similar approach in [Nguyen et al., 2019a] that finds solutions to the problem of sensor network localization using a Riemannian conjugate gradient method on a manifold representation of EDMs.

5.1.2 Euclidean Inverse Kinematics

Recently, several optimization approaches have been introduced that forgo the standard joint angle parametrization in favour of models based on Cartesian coordinates (also known as *natural coordinates* [de Jalón, 2007]). The authors of [Dai et al., 2019] use a piecewise-convex relaxation of the $\mathrm{SO}(3)$ group together with a set of points on the robot to formulate the constrained IK problem as a mixed-integer linear program (MILP). Their formulation can detect infeasible problems and provide approximate solutions to feasible problems, at the cost of a computationally intensive solution method. Yenamandra et al. [Yenamandra et al., 2019] use a similar relaxation to formulate IK as a semidefinite program. Blanchini et al. [Blanchini et al., 2015, 2017] treat points on a rigid manipulator as virtual masses in a potential field, leading to "minimum energy" solutions to convex formulations of planar and spherical inverse kinematics. Naour et al. [Le Naour et al., 2019b] formulate IK as a nonlinear program over inter-point distances, showing that solutions can be recovered for unconstrained articulated bodies.

However, the convergence of such approaches is hindered by a large and highly redundant search space, in part because the point set can be rotated and translated without affecting the distance constraints defining the IK problem. While our kinematic model is based on inter-point distances, our approach differs from previous work by encompassing a larger class of robots and allowing for constraints such as symmetric joint limits and spherical obstacle avoidance. Moreover, we provide a comparison with both heuristic and nonlinear optimization approaches, demonstrating that our proposed solution method provides a benchmark for IK problems.

Active structures such as robots also admit purely distance-based descriptions [Porta et al., 2018]. Porta et al. relate the IK problem to EDM completion [Porta et al., 2005b] for several common classes of manipulators and leverage an algebraic approach to find configurations reaching a desired end-effector pose. For general systems of kinematic and geometric constraints, Porta et al. apply a complete but computationally expensive branch-and-prune solver that iteratively eliminates regions of the solution space using geometric techniques [Porta et al., 2005a]. Our recent work [Marić et al., 2020] applies a convex *sum-of-squares* (SOS) relaxation [Parrilo, 2003, Lasserre, 2001] to a distance-geometric formulation of inverse kinematics, exploiting theoretical properties that guarantee a globally optimal solution for many problem instances. Moreover, we previously showed that kinematic constraints induce an inherently sparse structure that can be used to significantly reduce the computational burden usually associated with the SOS approach (and other convex relaxation-based methods).

In this chapter, we explore IK from a *distance geometry* perspective, revealing an equivalence between IK as defined in Section 3.3 and the general DGP defined in Section 2.3. By formalizing this equivalence for a large class of robots comprised of planar (i.e., two-dimensional), spherical, and revolute joints, we are able to connect IK to a rich literature of DGP solutions based on low-rank matrix completion [Nguyen et al., 2019b]. We find solutions using the method introduced by Mishra et. al. [Mishra et al., 2011], where a Euclidean distance matrix [Dokmanic et al., 2015] is parametrized with the manifold of fixed-rank Gram matrices [Journée et al., 2010], maintaining the advantages of a relaxed search space with fixed dimensionality and reduced redundancy. Furthermore, we show that bound smoothing [Havel, 2002] can be used as an effective initialization method that significantly improves convergence. In contrast to [Porta et al., 2005b], the use of a numerical optimization approach makes our algorithm suitable for a more general class of robots with redundant DOF, as well as the inclusion of workspace constraints.

5.2 Euclidean Distance Matrix Completion

As discussed in Section 2.3, a collection of points can be described using a graph $G = (V, E)$ that is weighted by interpoint distances. If all interpoint distances are known, the graph is *complete*, meaning that all of its edges and corresponding weights are prescribed. This graph can be compactly represented by the EDM whose elements are

$$\forall \{u, v\} \in E, \quad \mathbf{D}_{u,v} \triangleq d_{u,v}^2 \,, \tag{5.1}$$

and a realization of the graph can be obtained simply by taking the collection of points recovered via Eq. (2.28) and Eq. (2.29). It follows that the recovered collection of points is in fact a solution of the DGP defined by Problem 1. Conversely, many DGP instances are represented by graphs that only have a subset of edges defined a priori, resulting in an EDM with missing elements.

The problem of finding the missing elements in a partially defined EDM is known as the *EDM completion problem* [Dokmanic et al., 2015], which is strongly NP-hard in general [Liberti et al., 2014]. By defining the

symmetric binary matrix $\mathbf{\Omega}$ with elements

$$\mathbf{\Omega}_{u,v} \triangleq \begin{cases} 1 \text{ if } \{u,v\} \in E, \\ 0 \text{ otherwise,} \end{cases} \tag{5.2}$$

we arrive at a common statement of the EDM completion problem as low-rank matrix completion:

$$\min_{\mathbf{X} \in \mathcal{X}} f(\mathbf{X}) \triangleq \frac{1}{2} \left\| \mathbf{\Omega} \odot (\widetilde{\mathbf{D}} - \mathcal{K}(\mathbf{X})) \right\|_{\mathrm{F}}^2 , \tag{5.3}$$

where $\odot$ is the Hadamard (element-wise) matrix product, $\mathcal{K}$ is the distance matrix operator defined in Eq. (2.27), and $\widetilde{\mathbf{D}}$ is the incomplete distance matrix. Since the workspace dimension K of the robot is known, the Gram matrix $\mathbf{X}$ defined in Eq. (2.26) is constrained to the manifold

$$\mathcal{X} = \left\{ \mathbf{P}\mathbf{P}^{\mathsf{T}} : \mathbf{P} \in \mathbb{R}_*^{N \times K} \right\} , \tag{5.4}$$

where $\mathbb{R}_*^{N \times K}$ is the manifold of full-rank $N \times K$ matrices (i.e., the points in $\mathbf{P}$ don't describe a plane, straight line or a single point). It follows that $\mathcal{X}$ is the manifold of rank-K positive-semidefinite matrices. The NP-hardness of this problem originates from the non-convex constraint on the rank of $\mathbf{X}$, which can be relaxed in order to obtain a solution using Euclidean local search or *semidefinite programming* [Alfakih et al., 1999]. From Eq. (5.4), we see that relaxing the rank constraint expands the search space to collections of points with dimension greater than K. This is fundamentally incompatible with physical problems, for which the embedding dimension of points attached to bodies is at most 3. In fact, many interior point methods that solve the resulting convex semidefinite program tend to return a max-rank (and therefore potentially non-physical) solution [So and Ye, 2007].

We can avoid explicit rank constraints in Eq. (5.3) by using the Burer-Monteiro factorization [Burer and Monteiro, 2004] to define the cost function directly in terms of the points $\mathbf{P} \in \mathbb{R}^{N \times K}$. This results in a non-convex optimization problem

$$f^* = \min_{\mathbf{P} \in \mathbb{R}^{N \times K}} f(\mathbf{P}\mathbf{P}^{\mathsf{T}}), \tag{5.5}$$

which reduces the number of variables without changing the global minimum [Fang and O'Leary, 2012]. Following the derivation in [Mishra et al., 2011], the Euclidean gradient of Eq. (5.3) with respect to $\mathbf{P}$ is defined as

$$\nabla f = 4\big(\mathbf{S} - \operatorname{diag}(\mathbf{S}\mathbf{1})\big)\,\mathbf{P}, \tag{5.6}$$

where $\mathbf{S} = \mathbf{\Omega} \odot (\widetilde{\mathbf{D}} - \mathcal{K}(\mathbf{X}))$ and $\operatorname{diag}(\mathbf{S}\mathbf{1})$ is a diagonal matrix formed by the product $\mathbf{S}\mathbf{1}$.

Second-order optimization methods benefit from an exact analytical expression of the Hessian $\mathbf{H}(f)$. In [Chu et al., 2003], an analytic expression for the full Hessian of Eq. (5.5) is obtained in an element-wise fashion. However, this expensive computation can be avoided by observing that many optimization methods only require the Hessian-vector product [Pearlmutter, 1994] and by making use of the identity

$$\nabla_{\mathbf{Z}}(\nabla f) \triangleq \mathbf{H}(f) \cdot \mathbf{Z} \,,$$

where $\nabla_{\mathbf{Z}}(\nabla f) = \frac{d\nabla f\left((\mathbf{P}+t\mathbf{Z})(\mathbf{P}+t\mathbf{Z})^{\mathsf{T}}\right)}{dt}$ is the directional derivative [1] of the gradient in the direction $\mathbf{Z}$. We

[1] Denoted by $\nabla_{\mathbf{Z}}$.

then obtain

$$\begin{aligned}\mathbf{H}(f)\cdot\mathbf{Z} = 4\,\nabla_{\mathbf{Z}}\big(\mathbf{S}-\operatorname{diag}(\mathbf{S1})\big)\,\mathbf{P}\\ +4\,\big(\mathbf{S}-\operatorname{diag}(\mathbf{S1})\big)\,\mathbf{Z}\,.\end{aligned} \tag{5.7}$$

Unlike gradient descent, second-order optimization methods feature a superlinear convergence rate, which is useful when highly accurate solutions of Eq. (5.5) are required.

5.2.1 Optimization on the Manifold

As stated in Section 2.3.1, inter-point distances are invariant to rigid transformations of the underlying point set. It follows that the problem in Eq. (5.5) is invariant to right-multiplication of the variable $\mathbf{P}$ with orthogonal matrices $\mathbf{Q} \in \mathrm{O}(K)$. This results in nonisolated minima, which have been shown to cause step evaluation issues for second-order methods [Absil et al., 2009] when close to a solution, rendering the classical result of superlinear convergence void. This issue is circumvented in [Journée et al., 2010] by considering the set of all equivalence classes of the form

$$[\mathbf{P}] = \left\{\mathbf{PQ}|\,\mathbf{Q}\in\mathbb{R}^{K\times K}\,,\mathbf{Q}^{\mathsf{T}}\mathbf{Q}=\mathbf{I}\right\}. \tag{5.8}$$

Elements of this set constitute a manifold $\mathcal{M}$ which is the quotient of the set of full-rank $N\times K$ matrices by the orthogonal group $\mathrm{O}(K)$:

$$\mathcal{M} \triangleq \mathbb{R}_*^{N\times K}/\,\mathrm{O}(K). \tag{5.9}$$

It follows that the overall search space is reduced by reformulating Eq. (5.5) on the manifold $\mathcal{M}$ as

$$\phi^* = \min_{[\mathbf{P}]\in\mathcal{M}} \phi([\mathbf{P}]), \tag{5.10}$$

where $\phi([\mathbf{P}]) = f(\mathbf{P}\mathbf{P}^T)$. Moreover, it can be shown that the quotient manifold $\mathcal{M}$ has the structure of a Riemannian manifold [Absil et al., 2009] as defined in Section 2.1.2. Next, we provide an overview of how the EDM completion problem in Eq. (5.10) can be adapted to the Riemannian setting, as described by Mishra et. al. [Mishra et al., 2011]. We refer the reader to [Absil et al., 2009] for a detailed treatment of quotient manifolds and their geometry.

Formally, the tangent space $T_{\mathbf{P}}\mathcal{M}$ of a point $\mathbf{P}$ in $\mathcal{M}$ is the space of all tangent vectors $\gamma'(0)$ to curves $\gamma:\mathbb{R}\to\mathcal{M}$, where $\gamma(0)=\mathbf{P}$. The tangent space $T_{\mathbf{P}}\mathcal{M}$ is endowed with the inner product, also known as the Riemannian metric

$$g_{\mathbf{P}}(\mathbf{Z}_1,\mathbf{Z}_2) = \operatorname{Tr}(\mathbf{Z}_1^{\mathsf{T}}\mathbf{Z}_2), \quad \mathbf{Z}_1,\mathbf{Z}_2\in T_{\mathbf{P}}\mathcal{M}, \tag{5.11}$$

which is the usual metric on $\mathbb{R}^{N\times K}$. We can divide the tangent space into two orthogonal subspaces [Journée et al., 2010]

$$T_{\mathbf{P}}\mathcal{M} = \mathcal{V}_P\mathcal{M}\oplus\mathcal{H}_P\mathcal{M}\,,$$

where the tangent space to the equivalence classes in Eq. (5.8) is known as the *vertical subspace*

$$\mathcal{V}_P\mathcal{M} = \left\{\mathbf{PQ}|\,\mathbf{Q}\in\mathbb{R}^{K\times K}\,,\mathbf{Q}^{\mathsf{T}}+\mathbf{Q}=0\right\}, \tag{5.12}$$

and the orthogonal complement of $\mathcal{V}_P\mathcal{M}$ in $T_P\mathcal{M}$ is known as the *horizontal subspace*

$$\mathcal{H}_{\mathbf{P}}\mathcal{M} = \left\{\mathbf{Z}\in T_{\mathbf{P}}\mathbb{R}_*^{N\times K}|\,\mathbf{Z}^{\mathsf{T}}\mathbf{P}=\mathbf{P}^{\mathsf{T}}\mathbf{Z}\right\}. \tag{5.13}$$

Given a tangent vector $\mathbf{Z} \in T_{\mathbf{P}}\mathcal{M}$ at a point $\mathbf{P} \in \mathcal{M}$, we can recover the horizontal component from Eq. (5.13) using the *horizontal projection* operator.

Definition 12 (Horizontal projection)**.** *The horizontal projection $P_{\mathcal{H}_{\mathbf{P}}\mathcal{M}} : T_{\mathbf{P}}\mathcal{M} \to \mathcal{H}_{\mathbf{P}}\mathcal{M}$, that recovers the horizontal lift $\mathbf{Z}_{\mathcal{H}}$ of the tangent vector $\mathbf{Z} \in T_{\mathbf{P}}\mathcal{M}$ corresponding to the horizontal subspace in Eq. (5.13) is defined as*

$$P_{\mathcal{H}_{\mathbf{P}}\mathcal{M}}(\mathbf{Z}) = \mathbf{Z} - \mathbf{P}\mathbf{C}\,, \tag{5.14}$$

where $\mathbf{C}$ is a skew-symmetric matrix solving the Sylvester equation:

$$\mathbf{C}\mathbf{P}^{\mathsf{T}}\mathbf{P} + \mathbf{P}^{\mathsf{T}}\mathbf{P}\mathbf{C} = \mathbf{P}^{\mathsf{T}}\mathbf{Z} - \mathbf{Z}^{\mathsf{T}}\mathbf{P}.$$

Using the projection operator $P_{\mathcal{H}_{\mathbf{P}}\mathcal{M}}$, derivatives of the function ϕ (defined on the manifold) are computed from the derivatives of the function f (defined in Euclidean space) [Absil et al., 2009] by producing the horizontal lift of the Euclidean gradient of f at point $\mathbf{P}$:

$$\nabla\phi = P_{\mathcal{H}_{\mathbf{P}}\mathcal{M}}(\nabla f). \tag{5.15}$$

Similarly, by projecting the directional derivative of the gradient defined in Eq. (5.7), we compute the Hessian-vector product of ϕ from that of f as

$$\mathbf{H}(\phi)\,[\mathbf{Z}] = P_{\mathcal{H}_{\mathbf{P}}\mathcal{M}}(\nabla_{\mathbf{Z}}(\nabla\phi)). \tag{5.16}$$

Once the geometrically-correct derivatives have been produced, the step size is calculated and the point is moved along a descent direction on the manifold. To ensure the resulting point remains on the manifold, we use the *retraction* operator.

Definition 13 (Retraction)**.** *In order to apply a direction of movement in $\mathcal{H}_{\mathbf{P}}\mathcal{M}$ while staying on the manifold $\mathcal{M}$, we use the retraction operator, which is defined as*

$$R_{\mathbf{P}}(\mathbf{W}) = \mathbf{P} + \mathbf{W}. \tag{5.17}$$

The projection and retraction operators allow for the adaptation of classic local optimization algorithms to the Riemannian setting [Boumal and Absil, 2015, Wei et al., 2016].

5.3 Distance-Geometric Inverse Kinematics

Unlike most approaches that attempt to directly solve IK as stated in Definition 9 in terms of joint variables $\boldsymbol{\theta}$, we adopt an alternative formulation based on inter-point distances [Porta et al., 2005b, Le Naour et al., 2019a, Marić et al., 2020]. Our approach, which allows us to trivially recover $\boldsymbol{\theta}$ from our distance-based solution, is developed in detail in Sections 5.3.1 to 5.3.3 and summarized in Fig. 5.1. In Section 5.3.4, we prove that our formulation of IK is equivalent to the distance geometry problem (DGP) defined in Section 2.3 for a broad class of manipulators. Finally, in Section 5.3.5 we show how robots with planar and spherical joints constitute special cases for which our formulation can be trivially simplified.

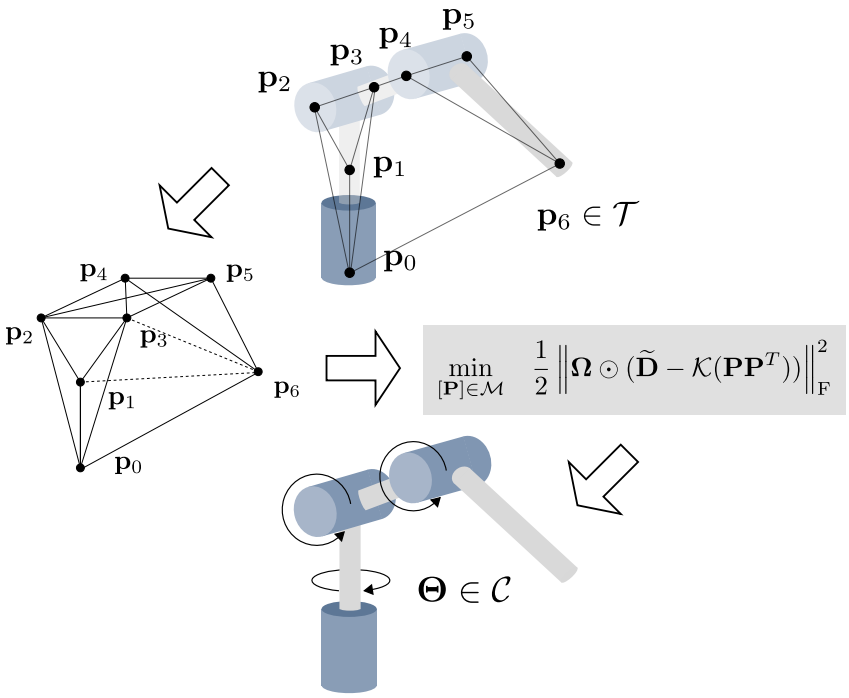


Figure 5.1: Overview of the proposed algorithm. A goal end-effector position $\mathbf{p}_6$ is defined for a 3-DOF robotic manipulator (top); the inverse kinematics problem is to find the corresponding joint angles $\mathbf{\Theta}$. Our method uses the matrix $\widetilde{\mathbf{D}}$ of distances between points $\mathbf{P}$ common to all feasible IK solutions to define an incomplete graph whose edges are weighted by known distances. Then, we apply Euclidean distance matrix completion with the known distance selection matrix $\mathbf{\Omega}$ to recover the weights corresponding to the unknown edges, solving the IK problem.

5.3.1 Kinematic Model

Articulated robots are comprised of a series of single-axis revolute joints oriented to provide a useful range of poses in their task spaces. Our goal in this section is to construct a graph representation of such mechanisms that is compatible with the DGP formulation in Problem 2. We achieve this by rigidly attaching a pair of points to the rotation axis of each joint in a manner similar to [Porta et al., 2005b]. As the example in Fig. 5.2 illustrates, the distances between points corresponding to neighbouring joints are invariant to changes in their angles during movement. These distances are key to describing the degrees of freedom of the robot.

Consider the points attached to the rotation axes of neighbouring joints, shown in Fig. 5.2 and labelled u and v. We denote the positions of these points and the orientations of their respective coordinate frames in $\underrightarrow{\mathcal{F}}_s$ as $\mathbf{p}_s^u$, $\mathbf{p}_s^v$, $\mathbf{R}_{su}$, and $\mathbf{R}_{sv}$, respectively. Further, $\mathbf{p}_u^v$ and $\mathbf{R}_{uv}$ denote the position and orientation of the fixed coordinate frame at v in the rotating coordinate frame at u. The matrix $\mathbf{R}_z(\theta_u)$ rotates $\mathbf{p}_u^v$ and the associated child joints about the $\hat{\mathbf{z}}$ axis by the joint angle θ_u. Given the joint angles $\mathbf{\Theta}$, these positions and orientations can be computed recursively as:

$$\begin{aligned} \mathbf{R}_{sv} &= \mathbf{R}_{su}\,\mathbf{R}_z(\theta_u)\,\mathbf{R}_{uv}\,, \\ \mathbf{p}_s^v &= \mathbf{p}_s^u + \mathbf{R}_{su}\,\mathbf{R}_z(\theta_u)\,\mathbf{p}_u^v \end{aligned} \quad \forall\,(u,v) : v \succ u, \tag{5.18}$$

where $\succ$ indicates that the joint indexed by u is the parent of the joint indexed by v in the directed graph describing the robot structure.[2] For neighbouring joints, $\mathbf{p}_u^v$ and $\mathbf{R}_{uv}$ are determined by the robot model parametriza-

[2]We assume that the graph is a tree whose root is the fixed robot base and whose leaves are the end-effectors; we leave extensions of

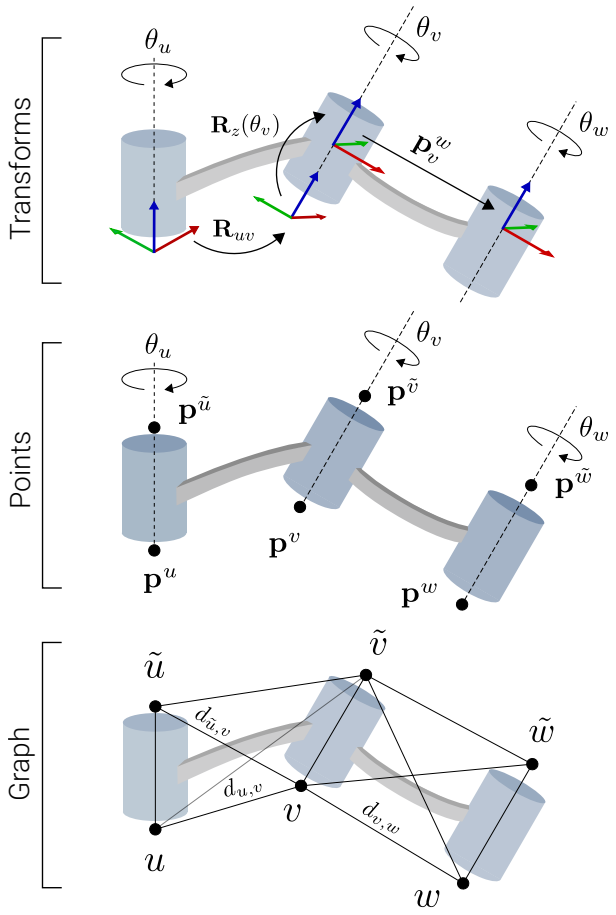


Figure 5.2: Visualization of the point placement used to describe a generic linkage of revolute joints and the corresponding graph representation. The top graphic shows the transformations used to obtain the poses of the joint coordinate frames. The middle graphic shows how pairs of points indexed by $(u, \tilde{u})$, $(v, \tilde{v})$, and $(w, \tilde{w})$ are placed along the rotation axis of their respective joints. The bottom graphic shows how the corresponding vertex representations form a graph whose edges are weighted by the known inter-point distances defined by link geometry in Eq. (5.20).

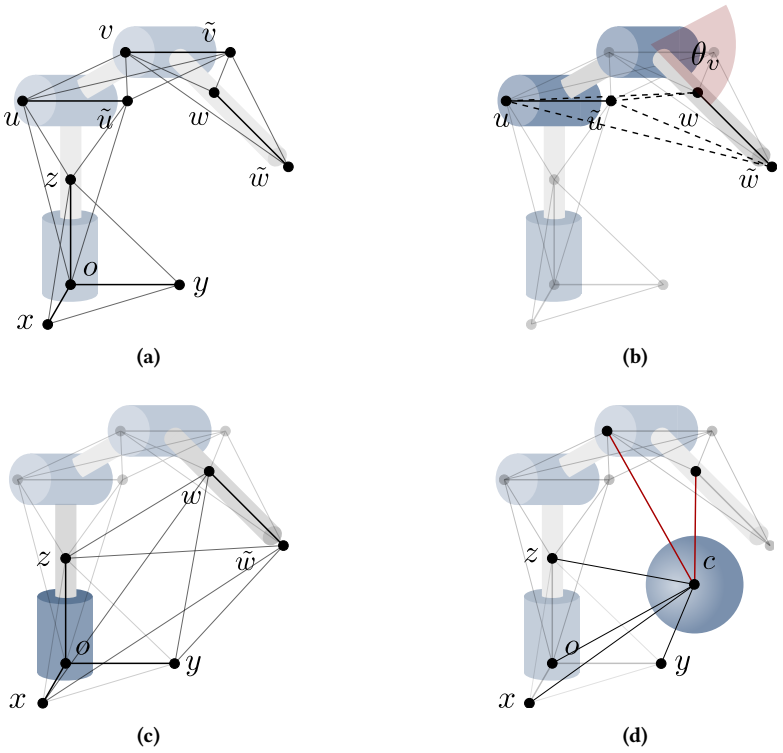


Figure 5.3: Visualization of the procedure in Section 5.3 for a 3-DOF revolute manipulator with joint limits. a) The solid black lines represent fixed distances between neighbouring joints and between base vertices. b) The dashed lines correspond to the distances constrained to some interval $[d^-,\ d^+]$ determined by symmetric limits on the joint angle θ_v. c) The solid black lines represent the distances fixed by setting a desired end-effector pose. d) Spherical obstacles are represented as vertices in the graph whose position is fixed by defining the distances to the base nodes. The lines drawn in red represent some distances whose lower bounds can be set in order to achieve obstacle avoidance.

tion (e.g., the DH convention [Hartenberg and Denavit, 1955] or Lie groups [Lynch and Park, 2017]). The second pair of points, labelled by $\tilde{u}$ and $\tilde{v}$, are obtained by translation along axis of rotation of each joint:

$$\begin{aligned} \mathbf{p}_s^{\tilde{u}} &= \mathbf{p}_s^u + \mathbf{R}_{su}\hat{\mathbf{z}}\,, \\ \mathbf{p}_s^{\tilde{v}} &= \mathbf{p}_s^v + \mathbf{R}_{sv}\hat{\mathbf{z}}\,. \end{aligned} \tag{5.19}$$

Together, these four points describe the relative position and orientation of the joints' rotation axes. For a given robot, we index the points obtained using Eq. (5.18) and Eq. (5.19) with the set of vertices $V_s \subset V$ of an incomplete graph $G = (V, E)$, where the set of edges E is weighted by interpoint distances that are known a priori. These distances describe the overall geometry and degrees of freedom of the robot, and they are invariant to the feasible motions of the robot (i.e., they remain constant in spite of changes to the configuration $\boldsymbol{\theta}$):

$$\begin{aligned} d_{u,\tilde{u}} &= d_{v,\tilde{v}} = 1, \\ d_{u,v} &= \|\mathbf{p}_u^v\|, \\ d_{u,\tilde{v}} &= \|\mathbf{p}_u^v + \mathbf{R}_{uv}\hat{\mathbf{z}}\|, \qquad \forall\,(u,v) \in V_s\,. \\ d_{\tilde{u},v} &= \|\mathbf{p}_u^v - \hat{\mathbf{z}}\|, \\ d_{\tilde{u},\tilde{v}} &= \|\mathbf{p}_u^v - \hat{\mathbf{z}} + \mathbf{R}_{uv}\hat{\mathbf{z}}\|\,. \end{aligned} \tag{5.20}$$

Depending on the specific link geometry, some identities in Eq. (5.20) may vanish, allowing us to merge identical points and reduce the overall size of the graph.

5.3.2 Constraints

In addition to describing the rigid structure of a robot using the distances in Eq. (5.20), we also need to introduce distance constraints that encode features of specific IK problem instances. To that end, this section describes how additional vertices and distances may be used to constrain the end-effector(s) of a robot, implement obstacle avoidance, and enforce joint limits.

Base Structure

In order to uniquely specify points with known positions (i.e., end-effectors) in terms of distances, we define the "base vertices" of a robot as $V_b = \{o, x, y, z\} \subset V$, where o is the root or base joint. The elements of V_b are used to form a coordinate frame with vertex o at the origin, as shown in Fig. 5.3a. We achieve this by defining the following edge weights:

$$\begin{aligned} d_{o,x} = d_{o,y} = d_{o,z} &= 1, \\ d_{x,y} = d_{x,z} = d_{y,z} &= \sqrt{2}. \end{aligned} \tag{5.21}$$

This base structure is used in the remainder of the section to specify, in terms of distance constraints, end-effector poses and joint limit constraints for links starting at the root. Note that the vertices x and y may be dropped in cases where the root joint angle is not limited to some interval, as this makes the solution set depend only on the distances from the goal points to the root o and to each other. Moreover, we may trivially reduce the graph size by using the points $\mathbf{p}^o$ and $\mathbf{p}^{\tilde{o}}$ attached to the base joint axis instead of the vertices o and z [Porta et al., 2005b].

our formulation to parallel manipulators, which contain loops, for future work.

End-Effector Pose

We consider two types of task space goals for an end-effector:

1. a 3-DOF position goal ($\mathcal{T}_p = \mathbb{R}^3$), and
2. a 5-DOF "direction goal" defined by the position of two distinct points ($\mathcal{T}_d = \mathcal{T}_p \times \mathcal{T}_p$).[3]

In both cases, $\mathbf{w} \in \mathcal{T}$ is encoded as a set of points fixed to the end-effector. These points are indexed by vertices $V_e \subset V$, and their positions are completely determined by the goal $\mathbf{w}$. Thus, an end-effector goal can be enforced by weighting the relevant edges with distances

$$d_{u,v} = \|\mathbf{p}^u - \mathbf{p}^v\|\,,\ u \in V_e\,, v \in V_b\,. \tag{5.22}$$

Joint Limits

Depending on the kinematic structure of the robot, symmetric joint limits can be represented by using distance intervals. In Fig. 5.3b, we see that a given joint angle can be represented (up to sign) using up to four distinct distances:

$$d^-_{u,w} = \sqrt{d^2_{u,v} + d^2_{v,w} - 2d_{u,v}d_{v,w}\cos(\theta^{\lim}_v)}\,,$$
$$d^+_{u,w} = d_{u,v} + d_{v,w}\,.$$

It follows that joint values can be restricted to symmetric intervals by constraining the distances between vertices assigned to the parent and child of a particular joint. Generally, limiting one of the aforementioned distances to an interval results in a distinct set of joint angle limits, depending on the particular distance chosen. However, it is important to note that the nature of the distance-based representation may make it difficult to implement arbitrary joint limits, as undesired symmetries may occur in certain ranges.

Obstacle Avoidance

We extend our model to incorporate spherical obstacles whose centers are indexed with the set of vertices $V_o \subset V$. The radius of each obstacle is given by the function $\rho : V_o \to \mathbb{R}_+$. Much like the elementary basis vectors in V_b, we can fix each center $\mathbf{p}_c$, $\forall\, c \in V_o$, in the global reference frame and augment G to include the constant interpoint distances for edges in $V_o \times V_o$ and $V_o \times V_b$:

$$\|\mathbf{p}^i - \mathbf{p}^j\| = d_{i,j}\ \forall\,(i,j) \in V_o \times V_o \cup V_o \times V_b. \tag{5.23}$$

Finally, points attached to the joints of a robot (i.e., those indexed by V_s) can be constrained to lie outside of each obstacle:

$$\|\mathbf{p}^u - \mathbf{p}^c\| \geq \rho(c)\ \forall (u,c) \in V_s \times V_o. \tag{5.24}$$

The radii given by ρ can be inflated to account for the shape and size of the robot's joints or as a conservative safety measure. For robots with long links, auxiliary points indexed by $V_{s'}$ can be easily added between points in V_s for higher precision collision avoidance.

[3] A full 6-DOF pose goal is not supported, as purely distance-geometric constraints cannot prevent reflections of the tool frame: this is equivalent to the assumption that the final joint has no angle limits when that axis is aligned with the final joint (e.g., for a common spherical wrist).

5.3.3 Solution Recovery

The remaining edge weights can be determined by completing the resulting partial EDM and a canonical realization $\mathbf{P}^*$ can be recovered. This result is achieved by identifying the points indexed by V_b with the origin and K elementary basis vectors in $\mathbb{R}^K$, which act as anchors in the solution of the Procrustes procedure [Dokmanic et al., 2015] discussed in Section 2.3.1. For $K = 3$, since the anchors form a right-handed frame that fully specifies a 6-DOF pose, $\mathbf{P}^*$ is unique. In Proposition 2, we prove that $\mathbf{P}^*$ will correspond to a unique feasible configuration $\boldsymbol{\theta} \in \mathcal{C}$ if the successive joint axes of our robot are *coplanar*. Once $\mathbf{P}^*$ is obtained, we can iteratively recover all the joint values by solving

$$\begin{aligned}\theta_u = \min_{\theta} &\|\mathbf{R}_{su}\mathbf{R}_z(\theta)\mathbf{p}_u^v - (\mathbf{p}_s^v - \mathbf{p}_s^u)\|^2 \\ &+\|\mathbf{R}_{su}\mathbf{R}_z(\theta)\mathbf{p}_u^v + \mathbf{R}_v\hat{\mathbf{z}} - (\mathbf{p}_s^{\tilde{v}} - \mathbf{p}_s^u)\|^2 .\end{aligned} \tag{5.25}$$

This problem can be reduced to finding the roots of a quartic polynomial and therefore admits a fast closed-form solution. Alternatively, $\boldsymbol{\theta}$ can be recovered from Eq. (5.18) and Eq. (5.19) with inverse trigonometric functions. Notably, the optimization formulation of Eq. (5.25) is robust to numerical errors in the calculation of $\mathbf{P}^*$ by Algorithm 2.

5.3.4 Equivalence to Distance Geometry

In this section, we prove that the robot models (i.e., the proposed graph descriptions) discussed thus far allow us to solve inverse kinematics (Definition 9) by means of the DGP (Problem 1). The main result is as follows:

Proposition 2 (IK $\equiv$ DGP)**.** *Suppose that the kinematic model of Section 5.3.1 describes a robot whose successive joint axes are* coplanar*. Then, the solutions to Problem 1 correspond one-to-one with the solutions to Definition 9. More precisely, if $\mathbf{p}^u, \mathbf{p}^{\tilde{u}}, \mathbf{p}^v$, and $\mathbf{p}^{\tilde{v}}$ are coplanar for all $v \succ u$, then for any end-effector target $\mathbf{w} \in \mathcal{T}$, we have a corresponding DGP encoded in G and there exists a bijection*

$$Q : \mathcal{C}_{\mathbf{w}} \to \mathcal{G}, \tag{5.26}$$

where $\mathcal{C}_{\mathbf{w}} \subset \mathcal{C}$ is the set of configurations achieving $\mathbf{w}$, and $\mathcal{G}$ is the space of all realizations (up to an arbitrary Euclidean transformation) of G.

Proof. We begin by recalling that the presence of base structure constraints (Eq. (5.21)) in our model allows us to identify "equivalent" realizations of a completion G with a canonical point assignment $\mathbf{P}^*$. We will assume $K = 3$ for the entire proof: $K = 2$ is a special case which can be simplified by noting that all joints share the same axis of rotation and by removing the "auxiliary" points defined by Eq. (5.19).

For a given robot and $\mathbf{w} \in \mathcal{T}$, the preceding sections described a set of DGP constraints which we write as the incomplete graph $G = (V, E)$. We will now proceed to prove that there is a bijection between $\mathcal{C}_{\mathbf{w}}$ and the equivalence classes representing distinct solutions to G (each equivalence class is represented by its canonical solution $\mathbf{P}^*$). It suffices to construct a map Q that is injective and surjective, where Q is simply the iterative procedure described by Eq. (5.18) and Eq. (5.19).

Injective: We will use a proof by contradiction. Suppose $\exists\, \boldsymbol{\theta}_1 \neq \boldsymbol{\theta}_2 \in \mathcal{C}_{\mathbf{w}}$ such that $Q(\boldsymbol{\theta}_1) = \mathbf{P}^* = Q(\boldsymbol{\theta}_2)$. Let u be the vertex label for a joint whose corresponding angle $\boldsymbol{\theta}_1^u = \theta_{u_1} \neq \theta_{u_2} = \boldsymbol{\theta}_2^u$, but has $\theta_{s_1} = \theta_{s_2}$ for all ancestors s of u. At least one such u is guaranteed to exist because Eq. (5.18) and Eq. (5.19) tell us that the axis

points $\mathbf{p}^v$ and $\mathbf{p}^{\tilde{v}}$ of joint $v \succ u$ only depend on θ_u and the positions of all its ancestor joints' points. This gives us:

$$\begin{aligned} \mathbf{p}_s^u + \mathbf{R}_{su}\mathbf{R}_z(\theta_{u_1})\mathbf{p}_u^v &= \mathbf{p}_s^u + \mathbf{R}_{su}\mathbf{R}_z(\theta_{u_2})\mathbf{p}_u^v, \\ \implies \mathbf{R}_z(\theta_{u_1})^\mathsf{T}\mathbf{R}_z(\theta_{u_2}) &= \mathbf{I}, \\ \implies \theta_{u_1} &= \theta_{u_2}, \end{aligned} \tag{5.27}$$

where we have assumed without loss of generality that $\mathbf{p}_u^v \neq c\,\hat{\mathbf{z}}$ for some $c \in \mathbb{R}$.[4] This contradicts our premise that $\theta_{u_1} \neq \theta_{u_2}$ and proves that the mapping is injective.

Surjective: We will show that for each $\mathbf{P}^* \in \mathcal{G}$ there exists $\boldsymbol{\theta} \in \mathcal{C}_\mathbf{w}$ such that $Q(\boldsymbol{\theta}) = \mathbf{P}^* \in \mathcal{G}$. By the definition of $\mathcal{G}$, $\|\mathbf{P}_v^* - \mathbf{P}_u^*\| = \|\mathbf{p}_u^v\|$ and $\|\mathbf{P}_v^* - \mathbf{P}_{\tilde{u}}^*\| = \|\mathbf{p}_u^v - \hat{\mathbf{z}}\|$ for all $v \succ u$, therefore we can always find θ_u such that

$$\mathbf{P}_v^* = \mathbf{P}_u^* + \mathbf{R}_{su}\,\mathbf{R}_z(\theta_u)\,\mathbf{p}_u^v, \tag{5.28}$$

as required by Eq. (5.18). Since we have assumed that the points $\mathbf{p}_s^u$, $\mathbf{p}_s^{\tilde{u}}$, $\mathbf{p}_s^v$, and $\mathbf{p}_s^{\tilde{v}}$ are coplanar, the position of $\mathbf{P}_{\tilde{v}}^*$ is uniquely determined by the other three points and therefore *must* take the form specified by Q and given by θ_u in Eq. (5.19):

$$\mathbf{P}_{\tilde{v}}^* = \mathbf{P}_v^* + \mathbf{R}_v\hat{\mathbf{z}} = \mathbf{P}_u^* + \mathbf{R}_{su}\,\mathbf{R}_z(\theta_u)\,\mathbf{p}_u^v + \mathbf{R}_{sv}\hat{\mathbf{z}}. \tag{5.29}$$

If the points were *not* coplanar, the six distance constraints in Eq. (5.20) would only specify their relative positions up to a reflection ambiguity (i.e., there would be two feasible tetrahedra with opposite "chirality" or "handedness"). This situation is illustrated in Fig. 5.4.

Finally, we address the interval constraints in G which correspond to joint angle limits of $\mathcal{C}$ and obstacle avoidance in $\mathcal{T}$. We have established the desired bijection $Q : \mathcal{C}_\mathbf{w} \to \mathcal{G}$ for IK problems defined only by equality constraints. Including interval constraints simply limits the space of DGP solutions to $\mathcal{G}' \subset \mathcal{G}$. Since the inverse of a bijection is a bijection, and a subset of the domain of a bijection induces a bijection, we have the desired $Q' : \mathcal{C}_\mathbf{w} \to \mathcal{G}'$ and the proof is complete. □

Proposition 2 establishes that the IK problem for a large class of robots can be formulated as a DGP. Our experiments in Section 5.5 demonstrate that the requirement of coplanar neighbouring axes is satisfied by many popular commercial manipulators. Additionally, it is worth noting that planar and spherical manipulators satisfy this requirement by definition.

5.3.5 Special Cases

In many cases, the generic DGP formulation of the IK problem presented in this section will result in kinematically redundant points in V_s. One example is points $\mathbf{p}_u$, $\mathbf{p}_v$ that can be selected to coincide (i.e., $\mathbf{p}_u = \mathbf{p}_v$ for all values of $\boldsymbol{\theta}$) because they lie on joint axes $u \succ v$ that intersect. While finding a generic strategy for generating DGP representations with minimal graph sizes is beyond the scope of our research, we can identify two cases of practical importance where this reduction is both trivial and significant.

By reducing the point dimensionality to $K = 2$, we can represent planar mechanisms using a single point per joint, as shown in Fig. 5.5a. Without loss of generality, this significantly reduces both the number of points

[4] In the special case where $\mathbf{p}_u^v = c\,\hat{\mathbf{z}}$, injectivity can be proved with $\mathbf{p}_s^{\tilde{v}}$ instead of $\mathbf{p}_s^v$. If $\mathbf{p}_s^{\tilde{v}}$ is collinear with $\mathbf{p}_s^v$, $\mathbf{p}_s^u$, and $\mathbf{p}_s^{\tilde{u}}$, then joint v is a rotation around the same axis as joint u and they can be effectively combined into a single joint.

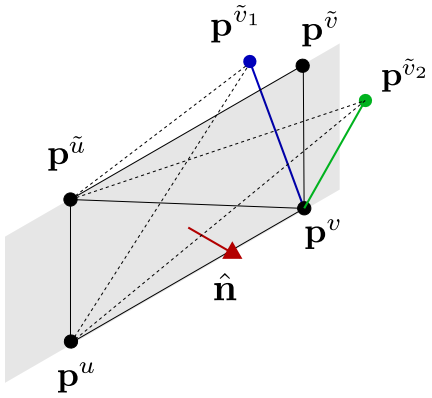


Figure 5.4: Illustration of the *chirality* or *handedness* issue that arises for non-coplanar pairs of joint axes $v \succ u$. If $\mathbf{p}^u, \mathbf{p}^{\tilde{u}}, \mathbf{p}^v$, and $\mathbf{p}^{\tilde{v}}$ are coplanar, our distance-geometric approach cannot introduce spurious solutions based on reflections of the true robot geometry. If $\mathbf{p}^{\tilde{v}}$ were replaced with $\mathbf{p}^{\tilde{v_1}}$ as shown, the spurious point $\mathbf{p}^{\tilde{v_2}}$ would also satisfy the distance constraints in Eq. (5.18) because it is a reflection of $\mathbf{p}^{\tilde{v_1}}$ across the plane containing $\mathbf{p}^u, \mathbf{p}^{\tilde{u}}$ and $\mathbf{p}^v$.

and distances used to describe the IK problem, resulting in lower overall computation times. For a detailed derivation of this simplified planar problem formulation, see [Marić et al., 2020].

A similar simplification can be obtained for a class of robots with joints allowing full or partial spherical motion, such as those of the human shoulder and hip. In addition to humanoids, these spherical joints can be found in highly redundant snake-like robots used in applications that include pipe inspection and surgery [Anantharayanan and Ordóñez, 2015]. As shown in Fig. 5.5b, the joints can be represented as two orthogonal revolute joints in the same position, which again allows us to drop the points used to define the rotation axes, while still allowing us to limit the elevation angle and thereby restricting link motions to a spherical cone. The remaining constraints can be trivially derived from those presented for the general case.

5.4 Algorithm

To summarize so far, in Section 5.3.1 we derived distance geometric IK formulations for a variety of different robot types. In Section 5.2 and Section 5.2.1 we showed how a Riemannian optimization approach to low-rank matrix completion can be used to efficiently solve instances of the DGP. In this section, we propose an extension to the optimization problem in Section 5.2.1, making it compatible with the distance-geometric IK formulation.

Modifying the matrix $\mathbf{\Omega}$ from Eq. (5.2) to select only those EDM elements corresponding to known invariant

(a) (b)

Figure 5.5: Visualization of planar and spherical mechanisms. These are two common examples of models for which our IK formulation uses a reduced number of variables.

distances in $\widetilde{\mathbf{D}}$, we introduce the matrices $\mathbf{\Psi}_-$ and $\mathbf{\Psi}_+$ that select interpoint distances with lower or upper bounds defined by joint limits or obstacle avoidance constraints (i.e., the known elements of $\widetilde{\mathbf{D}}_-$ and $\widetilde{\mathbf{D}}_+$). We can now formulate the DGP as the Riemannian optimization problem

$$\begin{aligned}\min_{[\mathbf{P}]\in\mathcal{M}} \phi([\mathbf{P}]) \triangleq \frac{1}{2}\left\|\mathbf{\Omega}\odot(\widetilde{\mathbf{D}}-\mathcal{K}\left(\mathbf{P}\mathbf{P}^T\right))\right\|_{\mathrm{F}}^2 \\ +\frac{1}{2}\left\|\max\left\{\mathbf{\Psi}_\pm\odot(\pm\widetilde{\mathbf{D}}_\pm\mp\mathcal{K}(\mathbf{P}\mathbf{P}^T)),0\right\}\right\|_{\mathrm{F}}^2 .\end{aligned} \tag{5.30}$$

The first term in this cost serves to enforce equality constraints on the distance matrix $\mathcal{K}(\mathbf{X})$, representing the constant set of distances defined by robot and task geometry. The second term enforces joint limits and obstacle avoidance constraints by setting either a lower or upper bound on a set of distances related to the joint rotation angle or distances from the obstacle center, with the element-wise $\max$ operator producing a nonzero value when these bounds are violated. Assuming that the joint limits are symmetric, the lower or upper bound on the distances will be implicitly enforced by the link length constraints, meaning that only one bound needs to be explicitly included in Eq. (5.30).

5.4.1 The Riemannian Trust-Region Algorithm

The quality of the configuration recovered by the reconstruction step outlined in Section 5.3.3 depends on highly accurate solutions to Eq. (5.30). As mentioned in Section 5.2, the superlinear convergence guarantee of second-order optimization methods helps to quickly obtain accurate solutions. To maintain this guarantee for the non-isolated minima of Eq. (5.30), we use the second-order Riemannian trust region (RTR) algorithm introduced in Absil et al. [2007]. Briefly, the trust region algorithm focuses on sequentially solving the problem

$$\begin{aligned}&\min_{\mathbf{Z}\in\mathcal{H}_\mathbf{P}\mathcal{M}} m_\mathbf{P}(\mathbf{Z}),\\ &\text{s.t.}\quad g_\mathbf{P}(\mathbf{Z},\mathbf{Z})\le\Delta^2,\end{aligned} \tag{5.31}$$

where

$$m_\mathbf{P}(\mathbf{Z}) \triangleq \phi(\mathbf{P}) + g_\mathbf{P}(\mathbf{Z},\nabla\phi) + \frac{1}{2}g_\mathbf{P}(\mathbf{Z},\mathbf{H}(\phi)[\mathbf{Z}]). \tag{5.32}$$

In other words, a quadratic approximation of the model constructed at point $\mathbf{P}$ informs a search for the optimal descent direction $\mathbf{Z}$ within a trust region of radius Δ. Accepting or rejecting a candidate descent direction and updating the trust region radius is based on the quotient

$$\rho = \frac{\phi(\mathbf{P})-\phi(R_\mathbf{P}(\mathbf{Z}))}{m_\mathbf{P}(0)-m_\mathbf{P}(\mathbf{Z})}. \tag{5.33}$$

A basic variant of this procedure is formalized in Algorithm 2, where the subproblem in Eq. (5.31) is approximately solved using the truncated conjugate gradient method introduced in [Absil et al., 2007]. The numerical cost per iteration of this approach was shown in [Mishra et al., 2011] to be $\mathcal{O}(dK+NK+NK^2+K^3)$, where d is the number of known entries in the EDM. Since $K\in[2,3]$ for IK problems, the complexity of Algorithm 2 is linear with respect to the number of points and known distances. Similarly to conventional trust region methods, the dominant computational bottleneck of RTR is the Hessian calculation, making the additional overhead added by the horizontal projection in Eq. (5.14) comparatively insignificant.

Algorithm 2: Riemannian Trust-Region (RTR)

Input: Initial point $\mathbf{P}_0$
Data: Cost function ϕ
Parameters : $\bar{\Delta} > 0$, $\Delta_0 \in [0, \bar{\Delta}]$, $\rho' \in [0, \frac{1}{4})$
Result: Solution $\mathbf{P}_N$
for $k = 0, 1, \ldots N$ **do**
  $\mathbf{Z}_k \leftarrow$ Eq. (5.31) and Eq. (5.32) ▷ Compute step
  $\rho \leftarrow$ Eq. (5.33)
  if $\rho < \frac{1}{4}$ **then** ▷ Update TR radius
    $\Delta_{k+1} \leftarrow \frac{1}{4}\Delta_k$
  else if $\rho > \frac{3}{4}$ *and* $\|\mathbf{Z}_k\| = \Delta_k$ **then**
    $\Delta_{k+1} \leftarrow \min(2\Delta_k, \bar{\Delta})$
  else
    $\Delta_{k+1} \leftarrow \Delta_k$
  end
  if $\rho > \rho'$ **then** ▷ Accept or reject step
    $\mathbf{P}_{k+1} \leftarrow R_{\mathbf{P}}(\mathbf{Z}_k)$ ▷ Retraction (Eq. (5.17))
  else
    $\mathbf{P}_{k+1} \leftarrow \mathbf{P}_k$
  end
end

5.4.2 Bound Smoothing

It is well established that the convergence of local optimization methods depends on the choice of starting point. The distance-based parametrization of the IK problem has the unique advantage of admitting informed initializations generated via a procedure known as *bound smoothing* [Havel, 2002]. First, we take the known set of distance bounds generated by the robot structure and problem constraints as described in Section 5.3 and form the graph G. Next, a bipartite graph is formed with two copies of G, with the vertices of the two graphs connected by edges weighted by their negative respective distance. Finally, the resulting all-pairs shortest path problem is solved using the Floyd-Warshall algorithm in $\mathcal{O}(|V|^3)$ time. For the IK problems analyzed in herein, the computation time of this search is on the order of 1 ms for a simple Python implementation on a laptop computer. The lower bounds on the distances between vertices can now be obtained by taking the shortest path between their representations in different subgraphs, with the upper bounds being the shortest path within one subgraph. An initial guess for the distance matrix, known as a pre-EDM, can then be generated by sampling individual elements within these bounds. Note that this procedure can be applied iteratively to produce even better approximations. Moreover, the computation time can be further reduced through the use of parallelization.

The full algorithm is described in Algorithm 3. We assume that an incomplete graph G describing the IK problem and an initializing conformation $\mathbf{P}_0$ are provided as input. Alternatively, we may also generate an initialization using the bound smoothing procedure described in the previous section. Next, we solve the local optimization problem using Algorithm 2 and transform the resulting configuration back to the canonical coordinate system. Finally, we recover the joint angle variables using Eq. (5.25).

Algorithm 3: Inverse Kinematics

Input: Incomplete graph G, Initial point $\mathbf{P}_0$
Result: Configuration $\boldsymbol{\Theta}$
Define ϕ from G via Eq. (5.30)
if *not* $\mathbf{P}_0$ **then**
  Get $\mathbf{P}_0$ using bound smoothing (Section 5.4.2)
end
$\mathbf{P}^* \leftarrow \text{RTR}(\mathbf{P}_0; \phi)$
$\mathbf{P} \leftarrow \text{OrthogonalProcrustes}(\mathbf{P}^*)$
Obtain $\boldsymbol{\Theta}$ from $\mathbf{P}$ using Eq. (5.25)

5.5 Experimental Results

In this section, we present an analysis of the performance of our method relative to multiple benchmark algorithms in a series of simulation studies. A variety of 2D and 3D kinematic models, including commercial manipulators and hyper-redundant kinematic chains and trees, were tested with and without joint angle limits and spherical obstacles. All Python code used in our experiments is freely available in our Git repository.[5]

5.5.1 Experimental Methodology

The results for each experiment were obtained with the following procedure:

1. generate a kinematic model (e.g., a planar robot with links arranged in a perfect binary tree with randomly generated symmetric joint angle limits);
2. randomly sample an angle configuration $\boldsymbol{\theta}_g \in \mathcal{C}$ for this model from a uniform distribution over the joint angle limits;
3. determine the target position(s) or pose(s) $\mathbf{w} \in \mathcal{T}$ of the end-effector(s) using $\boldsymbol{\theta}_g$ and the model's forward kinematics (i.e., $\mathbf{w} = F(\boldsymbol{\theta}_g)$);
4. run each IK algorithm on the problem instance defined by the kinematic model and the goal $\mathbf{w}$ using $\boldsymbol{\theta}_0 = \mathbf{0}$ as the initial configuration;
5. record statistics for each algorithm, including the number of iterations required until convergence, runtime, end-effector error(s), and joint angle limit violations;
6. repeat steps 2-5 above N times and summarize the statistics.

In all tables and figures, the success rate of an algorithm for a particular experiment was determined as the portion of runs where the solution satisfied all of the following criteria:

- joint angle limits were obeyed to within a tolerance of 1% of the bound magnitude;
- obstacle avoidance constraints were obeyed to within a tolerance of 0.01 m;
- the sum of the position errors of the end-effectors was less than 0.01 m; and
- the sum of the rotation errors of the end-effectors was less than 0.01 rad.

[5] https://github.com/utiasSTARS/GraphIK

The success rates of experiments reported in the tables and waterfall curves correspond to 95% Jeffreys confidence intervals [Tony Cai, 2005]. The statistics on solution error, runtime, and the number of iterations required for convergence that appear in various tables and figures are computed using the entire set of N runs (not just the successful portions). We denote joint angle-limited variants of experiments with a $+$ symbol (e.g., results labelled "6-DOF+" use the same robot as those labelled "6-DOF" but additionally enforce randomly generated joint angle limits).

In all relevant figures, the results from our algorithm are labelled RTR for "Riemannian Trust Region." When the bound smoothing procedure of Section 5.4.2 is used, `-B` is appended to the label (i.e., `RTR-B`). For each experiment, we compare our approach with a variety of benchmark algorithms from the optimization and IK literature. While we report runtime statistics for many of our experiments, we stress that our selection of baseline algorithms is designed to illustrate a variety of techniques and highlight the unique advantages of our novel distance-geometric formulation. Therefore, we did not choose particularly fast or industrially-proven implementations and opted for Python variants of all algorithms. Finally, all experiments were performed on a laptop computer with a 2.9 GHz Dual-Core Intel Core i5 processor.

5.5.2 Benchmark Algorithms

As discussed in Section 3.3, generalized approaches to solving the IK problem most often resort to numerical methods that search for a joint configuration $\boldsymbol{\theta} \in \mathcal{C}$ satisfying the defined constraints. Among the large variety of such approaches, local nonlinear programming is perhaps the most common and versatile. Therefore, we primarily compare our algorithm to the formulation

$$\begin{aligned} \min_{\boldsymbol{\theta}\in\mathcal{C}} \quad & \left\|\mathbf{e}\left(F(\boldsymbol{\theta}), \mathbf{w}\right)\right\|^2 \\ \text{s.t.} \quad & \left\|\mathbf{p}_i(\boldsymbol{\theta}) - \mathbf{c}_j\right\|^2 \geq l_j^2 \;\; \forall\, i \in V_s, \;\; \forall\, j \in V_o, \\ & \boldsymbol{\theta}_{\min} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_{\max}, \end{aligned} \tag{5.34}$$

where $F(\boldsymbol{\theta})$ is the forward kinematic mapping and $\mathbf{w}$ is the goal, as defined in Definition 9. The vector-valued function $\mathbf{e}$ in the objective represents an appropriate error for the task space, while inequality constraints serve to enforce obstacle avoidance and joint limits. In cases where some exact tolerance $\boldsymbol{\epsilon}$ on $\mathbf{e}$ is defined, the error can also be encoded using an (in-)equality constraint such as

$$\begin{aligned} \min_{\boldsymbol{\theta}\in\mathcal{C}} \quad & \left\|\boldsymbol{\theta} - \boldsymbol{\theta}_0\right\|^2 \\ \text{s.t.} \quad & \mathbf{e}\left(F(\boldsymbol{\theta}), \mathbf{w}\right) \leq \boldsymbol{\epsilon} \\ & \left\|\mathbf{p}_i(\boldsymbol{\theta}) - \mathbf{c}_j\right\|^2 \geq l_j^2 \;\; \forall\, i \in V_s, \;\; \forall\, j \in V_o, \\ & \boldsymbol{\theta}_{\min} \leq \boldsymbol{\theta} \leq \boldsymbol{\theta}_{\max}. \end{aligned}$$

However, we have empirically found that incorporating the error in an objective function (Eq. (5.34)) results in higher success rates—this phenomenon is also reported in [Beeson and Ames, 2015].

5.5.3 Hyper-Redundant and Tree-Like Robots

We begin by analyzing the performance of our algorithm for hyper-redundant and tree-like planar robots. This approach helps to avoid introducing confounding factors in the analysis, as the choice of any particular revo-

Table 5.1: Results for planar chain manipulators over 1,000 random experiments with pose goals. The + indicates joint angle limits. Bolded values indicate significantly higher sucess rates.

Method	`trust-exact/constr`		FABRIK		RTR			RTR-B		
	Success (%)	Iter. μ (σ)	Success (%)	Iter. μ (σ)	Success (%)	Iter. μ (σ)	Runtime [ms] μ (σ)	Success (%)	Iter. μ (σ)	Runtime [ms] μ (σ)
6-DOF	100.0 ± 0.1	26 (5)	100.0 ± 0.1	10 (19)	100.0 ± 0.1	16 (2)	2.28 (0.39)	100.0 ± 0.1	9 (2)	1.54 (0.37)
6-DOF+	72.0 ± 2.8	35 (14)	36.0 ± 3.0	1333 (916)	91.0 ± 1.7	32 (22)	5.24 (2.83)	$\mathbf{98.0 \pm 0.9}$	24 (17)	5.02 (3.01)
10-DOF	100.0 ± 0.1	28 (2)	100.0 ± 0.1	9 (13)	100.0 ± 0.1	20 (2)	3.27 (0.53)	100.0 ± 0.1	10 (2)	2.25 (0.43)
10-DOF+	97.0 ± 1.0	43 (54)	59.0 ± 3.0	856 (971)	88.0 ± 2.0	74 (101)	14.6 (1.15)	$\mathbf{98.0 \pm 0.8}$	23 (46)	5.47 (3.85)

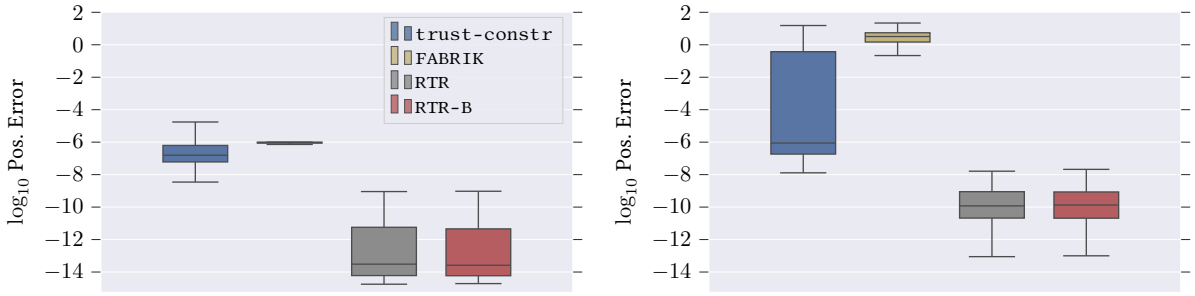
Table 5.2: Results for planar tree manipulators over 1,000 random experiments with pose goals. The + indicates joint angle limits.

Method	`trust-exact/constr`		FABRIK		RTR			RTR-B		
	Success (%)	Iter. μ (σ)	Success (%)	Iter. μ (σ)	Success (%)	Iter. μ (σ)	Runtime [ms] μ (σ)	Success (%)	Iter. μ (σ)	Runtime [ms] μ (σ)
14-DOF	54.0 ± 3.1	14 (4)	56.0 ± 3.1	949 (964)	64.0 ± 3.0	20 (4)	4.39 (1.00)	$\mathbf{67.0 \pm 2.9}$	9 (3)	4.12 (1.11)
14-DOF+	90.0 ± 1.9	189 (423)	85.0 ± 2.2	463 (716)	$\mathbf{96.0 \pm 1.2}$	18 (4)	4.98 (1.40)	96.0 ± 1.2	8 (2)	4.79 (1.76)
30-DOF	16.0 ± 2.3	23 (7)	12.0 ± 2.0	1848 (479)	$\mathbf{20.0 \pm 2.5}$	34 (10)	20.75 (8.52)	18.0 ± 2.4	18 (10)	45.25 (32.38)
30-DOF+	49.0 ± 3.1	818 (830)	0.0 ± 0.3	2000 (0)	$\mathbf{85.0 \pm 2.2}$	36 (19)	31.64 (17.95)	85.0 ± 2.2	20 (15)	43.98 (32.62)

lute manipulator opaquely affects the difficulty of IK problems. More importantly, these mechanisms allow us to systematically scale the size and number of constraints of the IK problem by adding joints and introducing multiple end-effectors, while minimizing the number of redundant points and fixed distances as noted in Section 5.3.5. Since the full pose of a planar end-effector is determined by its position and the position of its parent joint, the error $\mathbf{e}$ in the cost function of the joint angle-based approach in Eq. (5.34) can simply be defined as the difference between the end-effectors' and their parent joints' positions and their goal positions in $\mathbf{w}$. We solve Eq. (5.34) using second-order trust region methods with similar convergence guarantees to those provided by our algorithm, referring to the unconstrained method as `trust-exact` and the joint angle-constrained method as `trust-constr`. Our open source code provides an interface for testing this problem formulation with other solvers provided by `scipy.optimize`. In addition to the formulation in Eq. (5.34), we implemented the FABRIK [Aristidou and Lasenby, 2011] heuristic IK solver in Python as an additional benchmark for our experiments. In contrast to generic nonlinear solvers, this is an IK-specific heuristic method tailored to planar and spherical robots.

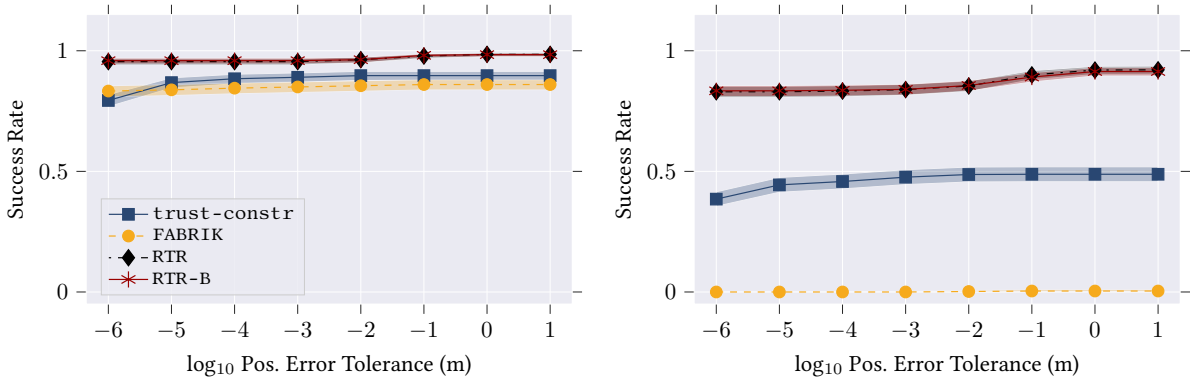
For the experiments in this section, all algorithms were allowed a total of 2,000 iterations and used a numerical tolerance of 10^{-9} for all stopping criteria. Where applicable, the magnitude of the gradient of the cost function or Lagrangian was used as the stopping criterion. Otherwise, the magnitude of the cost function or norm of the variable change in one iteration was used. All other parameters were assigned their default values as provided by the `pymanopt` [Townsend et al., 2016] and `scipy.optimize` libraries [Virtanen et al., 2020b]. Since the implementations of these benchmark algorithms were not extensively tuned for performance, we place greater emphasis on the number of iterations taken by each algorithm as a more meaningful statistic than runtime in the results to follow. FABRIK was allowed a maximum of 2,000 full forward and backward iterations per problem instance.

Table 5.1 summarizes our results for 6- and 10-DOF planar manipulators of the type shown in Fig. 5.5a, with and without joint angle limits. Unsurprisingly, all four algorithms achieve a 100% success rate when no joint angle limits are present. When joint angle limits are introduced, FABRIK performs far worse, while the RTR algorithm performs similarly to `trust-constr` in terms of success rate and iterations required. Initializing the problem using bound smoothing reduces the number of iterations needed while increasing the success rate of our RTR-B algorithm compared to a naive initialization. Curiously, both `trust-constr` and FABRIK's performance improve as the number of DOF increases. We suspect that the additional DOF give the heuristic



(a) 14-DOF+ **(b)** 30-DOF+

Figure 5.6: Box-and-whiskers plots summarizing end-effector position error over 1,000 experiments with planar binary tree robots with joint angle limits.



(a) 14-DOF+ **(b)** 30-DOF+

Figure 5.7: Waterfall curves of success rate versus position error tolerance for 1,000 experiments with planar tree robots with joint angle limits. The shaded regions are 95% Jeffreys confidence intervals centered on the solid lines. The rotation error tolerance is fixed at 0.01 rad.

approach of FABRIK more time to "steer" away from a difficult initial configuration induced by pose goals. The `trust-constr` algorithm may benefit from more DOF that can be used to "escape" from local minima or extremely flat regions of the cost function landscape. While we only report on experiments involving *pose* goals, we found that when joint angle limits were introduced, FABRIK was much better suited to *position* goals (i.e., without a specified orientation) for the robot end-effector(s).

Table 5.2 contains results for experiments involving binary tree-like robots, such as the 6-DOF example with a height of two shown in Fig. 5.8. The 14- and 30-DOF results correspond to robots with a perfect binary tree kinematic structure of height three and four, respectively. While not as practical as chain-like robots or the revolute manipulators of Section 5.5.4, these experiments serve to showcase the performance of IK methods on highly kinematically-constrained mechanisms, while also scaling naturally to a higher number of DOF. In all cases, the RTR and RTR-B algorithms outperform the three benchmark approaches in terms of success rate. When no joint limits are present, the overall difference in success rates is relatively small, with RTR and `trust-exact` having a similar number of outer iterations. When joint limits are introduced, the experimental procedure in Section 5.5.1 generates problems within a tighter range around a naive initialization, resulting in higher success rates for all approaches. Owing to a high number of constraints, the number of outer iterations for `trust-constr` increases more than ten-fold, significantly degrading performance. In contrast, this effect does not occur for RTR and RTR-B, where the number of iterations remains considerably lower while a similar increase in success rate is observed. The box-and-whiskers plots in Fig. 5.6 summarize the position error statistics for each algorithm over *all* runs in the constrained case, including those runs that did *not* qualify as a success. The waterfall curves in Fig. 5.7 display the success rate as a function of an increasing position error tolerance, demonstrating that the higher success rate of our algorithm is maintained for different accuracy requirements.

Mean runtimes of both RTR and RTR-B across all planar experiments remain below 0.1 s, with the 30-DOF tree-like robot unsurprisingly resulting in the most computationally-intensive problems. The runtime for FABRIK across all planar experiments was 3.4 s, while the local solvers had a mean runtime of 10.2 s. Both of these sets of runtime statistics are influenced by the large proportion of unsuccessful runs that required all 2,000 allowed iterations before terminating. However, since we cannot guarantee that the local algorithms were provided with equally well-tuned implementations of core subroutines (e.g., Hessian computations for the second-order trust region solvers), we urge our readers to treat the statistics on iterations reported in Table 5.1 and Table 5.2 as more qualitative indicators of performance.

To illustrate the optimization procedure and help elucidate the relatively superior performance of our method on branching tree-like robots, we conducted a simple empirical analysis on one of the many problem instances where RTR outperformed the benchmark algorithms. Fig. 5.8 shows the convergence of four solvers with the same initial condition on a sample low-dimensional IK problem involving a 6-DOF binary planar tree robot with symmetric joint angle limits and end-effector position goals. Only our algorithm, RTR, is able to find the global minimum. The algorithms all perform similarly for the first eight iterations, but the three competitors are unable to escape from the same local minimum. The difference in behaviour is explained by Fig. 5.9, which compares contour maps of different cost function terms used by the algorithms, overlaid with the progress of each algorithm across iterations. Fig. 5.9a illustrates the Euclidean distance of the end-effector controlled by θ_1 and θ_3 and its goal position, which is used in the cost function of `L-BFGS-B` and `trust-constr`. In contrast, the contour map in Fig. 5.9b is the logarithm of the quartic function containing the terms in the distance-geometric cost function of Eq. (5.3) involving the position of the joint actuated by θ_1. The angular cost function is ill-conditioned, leading the algorithms that minimize it to converge to the local minimum of Fig. 5.8, whereas RTR minimizes the distance-geometric cost of Fig. 5.9b and quickly converges to the global minimum, which

has a large and well-conditioned basin of convergence. In spite of its simplicity, this toy problem illustrates the behavioural differences of the algorithms in a state space with low enough dimension to visualize clearly.

5.5.4 Revolute Manipulators

Next, Table 5.3 compares the performance of our algorithm and the trust region benchmark algorithms on 3D robots with revolute joints and within an unconstrained workspace. These problems are of greater practical interest than the planar results in Section 5.5.3 and showcase the expressiveness of our problem formulation. All experiments are conducted for the Universal Robots UR10, KUKA IIWA, Schunk LWA4D, and Schunk LWA4P manipulators shown in Fig. 5.10. For these robots, the distance-geometric IK formulation derived using the procedure described in Section 5.3.1 results in points that always overlap (i.e., have a fixed distance of zero). While these points could be "merged" to reduce the graph size—thereby improving overall performance—we used the generic models for transparency.

In terms of success rate, our algorithm outperforms `trust-exact` and `trust-constr` on the UR10 and KUKA IIWA manipulators with and without joint angle limits, as well as on the Schunk LWA4P with joint limits. The bound smoothing procedure used to initialize RTR-B reduces the overall number of iterations in all cases, but has a variable effect on the success rate. Both RTR and RTR-B require a significantly larger number of iterations to converge compared to the planar case, while `trust-*` remains in a similar range to that observed in Table 5.1. We can partially attribute this to the iteration complexity discussed in Section 5.4, which is increased by the unfavourably high ratio of points to DOF in the kinematic models of these mechanisms. Again, this effect could be mitigated in future work by removing overlapping points from the kinematic models, reducing the overall number of variables.

The box-and-whiskers plots in Fig. 5.11 summarize the position error statistics for each algorithm over *all* runs with the UR10 manipulator, with and without joint limits. In both cases the `trust-*` algorithms converge to significantly lower cost function values. This suggests that the highly variable magnitudes of known distances may cause numerical issues in the gradient for our algorithms, causing early termination. We suspect this issue can be avoided by using a weighting matrix to regularize elements of the cost function, as shown in [Nguyen et al., 2019a]. The waterfall curves in Fig. 5.12 corroborate these findings, showing that the success rate of our algorithm drops as the position error tolerance decreases, and suggesting that decreasing the gradient termination criteria may increase accuracy at the cost of increased computation time.

The mean runtime of both RTR and RTR-B remains below 1.0 s for all robot models, reaching the lowest mean runtimes of less than 0.2 s on the UR10. In the same instance, the `trust-*` algorithms have a slightly higher mean runtime of 0.3 s. We observe the highest computation times for our algorithms with the 7-DOF Schunk LWA4D, with mean runtimes slightly below 1.0 s. While the `trust-*` methods exhibit similarly worse performance with a mean runtime of 0.5 s, the overall increase in runtime is smaller due to the number of variables only increasing from six to seven.

5.5.5 Obstacle Avoidance

Finally, we analyze the performance of our algorithm on revolute manipulators in environments with a varying number of spherical obstacles, as shown in Fig. 5.13. For each environment and robot pair, 3,000 IK problems were randomly generated, some of which may not be solvable (i.e., no collision-free solutions exist). For experiments performed in this section we chose $\mathbf{e} = \log(\mathbf{T}(\boldsymbol{\theta})^{-1}\mathbf{T}_{goal})$ as the error in Eq. (5.34), which is the vector of exponential coordinates describing the screw motion between the current pose and goal pose of the

Table 5.3: Results for revolute chain manipulators over 2,000 random experiments with pose goals. The + indicates joint angle limits.

Method	`trust-exact/constr`		RTR			RTR-B		
	Success (%)	Iter. μ (σ)	Success (%)	Iter. μ (σ)	Runtime [ms] μ (σ)	Success (%)	Iter. μ (σ)	Runtime [ms] μ (σ)
UR10	90.0 ± 1.3	12 (7)	87 ± 1.0	364 (523)	205.8 (264.1)	$\mathbf{95.0 \pm 1.0}$	**319** (**493**)	198.4 (239.3)
UR10+	63.0 ± 2.1	36 (19)	$\mathbf{77.0 \pm 1.8}$	**364** (**523**)	206.6 (264.2)	66.0 ± 2.0	251 (457)	138.2 (193.0)
KUKA	100.0 ± 0.1	20 (6)	100.0 ± 0.2	317 (373)	242.8 (211.1)	100.0 ± 0.1	315 (386)	268.3 (220.0)
KUKA+	87.0 ± 1.5	48 (18)	82.0 ± 1.7	734 (771)	432.4 (395.1)	$\mathbf{89 \pm 1.3}$	**506** (**660**)	386.8 (374.8)
LWA4P	$\mathbf{100.0 \pm 0.2}$	**20** (**17**)	89.0 ± 1.4	513 (624)	416.6 (542.4)	90.0 ± 1.3	503 (614)	290.0 (330.1)
LWA4P+	77.0 ± 1.8	46 (25)	$\mathbf{87.0 \pm 1.5}$	**435** (**569**)	246.7 (282.2)	81 ± 1.7	324 (482)	201.1 (219.7)
LWA4D	$\mathbf{100.0 \pm 0.1}$	**24** (**17**)	99.0 ± 0.5	868 (747)	969.9 (878.0)	97.0 ± 0.7	713 (699)	895.6 (897.1)
LWA4D+	$\mathbf{96.0 \pm 0.8}$	**47** (**17**)	91.0 ± 1.3	867 (769)	900.2 (874.1)	90.0 ± 1.3	825 (758)	991.2 (943.8)

end-effector. Further, the FK and Jacobian computations are carried out using the popular *product of exponentials* approach [Lynch and Park, 2017], avoiding trigonometric identities inherent to the DH parametrization. We solve the problem in Eq. (5.34) using sequential quadratic programming, namely the SLSQP solver implemented in the `scipy` library. Due to its speed and accuracy, this method is a popular choice for nonlinear programming solutions to IK [Beeson and Ames, 2015]. We empirically determined that a maximum of 200 iterations and an objective function value of 10^{-7} were effective stopping criteria for our experiments.

The results of our evaluation are shown in Fig. 5.14. Each column represents a different fixed set of obstacles in the manipulators' environment. The first column contains results for obstacle-free problems, while the remaining columns use spherical obstacles placed on the vertices of an octahedron, cube, and icosahedron, respectively. The top row compares the success rate of RTR-B and SLSQP, with dashed lines indicating the portion of problems that are known to be feasible (i.e., the configuration used to generate the problem is not in collision). The box-and-whiskers plots in the middle two rows describe the distribution of position and rotation errors, with the thresholds for success (0.01 m and 0.01 rad) drawn as dashed lines. Finally, the bottom row compares the distribution of solution times.

In terms of success rate, our algorithm outperforms SLSQP in all experiments. The position and rotation error distributions displayed in the box-and-whiskers plots reveal that this is due to RTR-B providing solutions with lower position and rotation error on average. The performance gap is particularly large for the UR10, which both algorithms struggled most with in all environments. Finally, the runtime for our Riemannian solver is expectedly higher than for SLSQP in the obstacle-free case, as seen in previous experiments. However, the addition of obstacles leads to significantly faster relative performance for RTR-B on all the manipulators tested. More importantly, we note that the runtime of SLSQP exhibits a more significant relative increase than RTR-B when obstacles are introduced. We suspect that this is due to the nonlinear mapping between joint angles and obstacle locations that makes the problem significantly more difficult to solve in configuration space. In contrast, the effect is avoided by our distance-based approach because collision avoidance constraints are treated in the same way as structural and joint limit constrains.

5.6 Summary and Conclusions

This chapter describes a novel and elegant procedure for formulating many inverse kinematics problems in the language of distance geometry. The distance-geometric perspective on IK allows us to leverage powerful low-rank matrix completion methods, resulting in an algorithm (RTR-*) that can efficiently compute IK solutions for a variety of robots using Riemannian optimization. Our experiments show that RTR-* outperforms competing algorithms in terms of success rate and number of iterations on IK problems for robots with multiple

end-effectors, both with and without the inclusion of joint limits. We have also demonstrated the feasibility of this approach to solve IK for commercial manipulators, achieving success rates competitive with both comparable [Erleben and Andrews, 2019] and conventional [Lynch and Park, 2017] angle-based methods. Notably, we observe that our algorithm performs significantly better than a conventional method, both in terms of success rate and runtime, when the IK problem requires finding configurations that are not in collision with obstacles (modelled as spheres). Overall, these experimental results indicate that a distance-geometric approach is particularly advantageous when a large number of workspace constraints are present. We believe our algorithm provides a valuable benchmark for IK solvers, as well as a good starting point for future research into distance-geometric formulations of this problem.

We have identified several exciting research directions for the distance-geometric IK formulation presented herein. Our algorithm utilizes a Riemannian optimization-based solution because it is fast, can easily be extended (e.g., to include obstacles or other cost terms), and avoids problems associated with redundant degrees of freedom. However, we believe a thorough exploration of the vast body of literature on distance geometry may yield other effective approaches or insights into our particular problem structure. This will permit us to compare the runtime of our algorithm against a greater variety of IK solvers, including complex algorithms like TRAC-IK that utilize multiple methods in parallel [Beeson and Ames, 2015]. Additionally, we are eager to develop an optimized version of our algorithm in a fast compiled language such as C. We believe that the distance-geometric IK formulation described herein provides a strong mathematical basis for future research in kinematics, motion planning, and control.

5.6.1 Limitations

Avoiding reflections in the solution set of certain problems remains a core challenge when using a purely distance-based IK approach. As noted in Proposition 2, this spells out an important limitation of our formulation: its inability to handle revolute manipulators with non-coplanar consecutive axes of rotation. This also restricts the capacity of our formulation to represent joint angle limits to those that are symmetrical about $\boldsymbol{\theta}_0 = \mathbf{0}$ (i.e., of the form $[-\theta_{\text{lim}},\ \theta_{\text{lim}}]$). At the cost of some of the theoretical foundations laid out in Section 5.3.4, it is possible to address these ambiguities by extending our formulation to include cross products (allowing "handedness" to be expressed). For example, solutions that include reflections for non-coplanar joints u and v in Fig. 5.2 could be removed by taking $\mathbf{c} = (\mathbf{p}^{\tilde{u}} - \mathbf{p}^{u}) \times (\mathbf{p}^{\tilde{v}} - \mathbf{p}^{v})$ and constraining the sign of the dot product $\mathbf{c} \cdot (\mathbf{p}^{v} - \mathbf{p}^{u})$, thereby only allowing one of the two possible relative orientations satisfying distance constraints. Similarly, a nonsymmetric joint limit on v can be obtained by taking $\mathbf{a} = (\mathbf{p}^{v} - \mathbf{p}^{u}) \times (\mathbf{p}^{\tilde{v}} - \mathbf{p}^{v})$, $\mathbf{b} = (\mathbf{p}^{w} - \mathbf{p}^{u}) \times (\mathbf{p}^{\tilde{v}} - \mathbf{p}^{v})$ and constraining the sign of $\mathbf{a} \cdot \mathbf{b}$. Since adding these constraints would require a lengthy and detailed deviation from the purely distance-geometric view of IK developed herein, we leave their characterization as future work.

5.6.2 Associated Publications

This section features a method for solving the inverse kinematics problem through a distance-based characterization that enables the use of novel optimization methods for finding solutions in constrained workspace. This contribution was published alongside the particular method for solving IK using local optimization over a Riemannian quotient manifold described herein.

1. Marić, F., Giamou, M., Hall, A.W., Khoubyarian, S., Petrović, I. and Kelly, J., 2022. Riemannian Optimization for Distance-Geometric Inverse Kinematics. In IEEE Transactions on Robotics, vol. 38, no. 3, (pp. 1703-1722), doi: 10.1109/TRO.2021.3123841.

It is important to mention that this work is also associated with our previously published IK approach based on sum-of-squares optimization, which will not be described in this thesis.

2. Marić, F., Giamou, M., Khoubyarian, S., Petrović, I. and Kelly, J., 2020, May. Inverse kinematics for serial kinematic chains via sum of squares optimization. In 2020 IEEE International Conference on Robotics and Automation (ICRA) (pp. 7101-7107).

Finally, this work was followed by an important result describing a convex relaxation of the IK problem.

3. Giamou, M., Maric, F., Rosen, D., Peretroukhin, V., Roy, N., Petrovic, I. and Kelly, J., 2022. Convex Iteration for Distance-Geometric Inverse Kinematics. IEEE Robotics and Automation Letters, vol. 7, no. 2, (pp. 1952-1959), doi: 10.1109/LRA.2022.3141763.

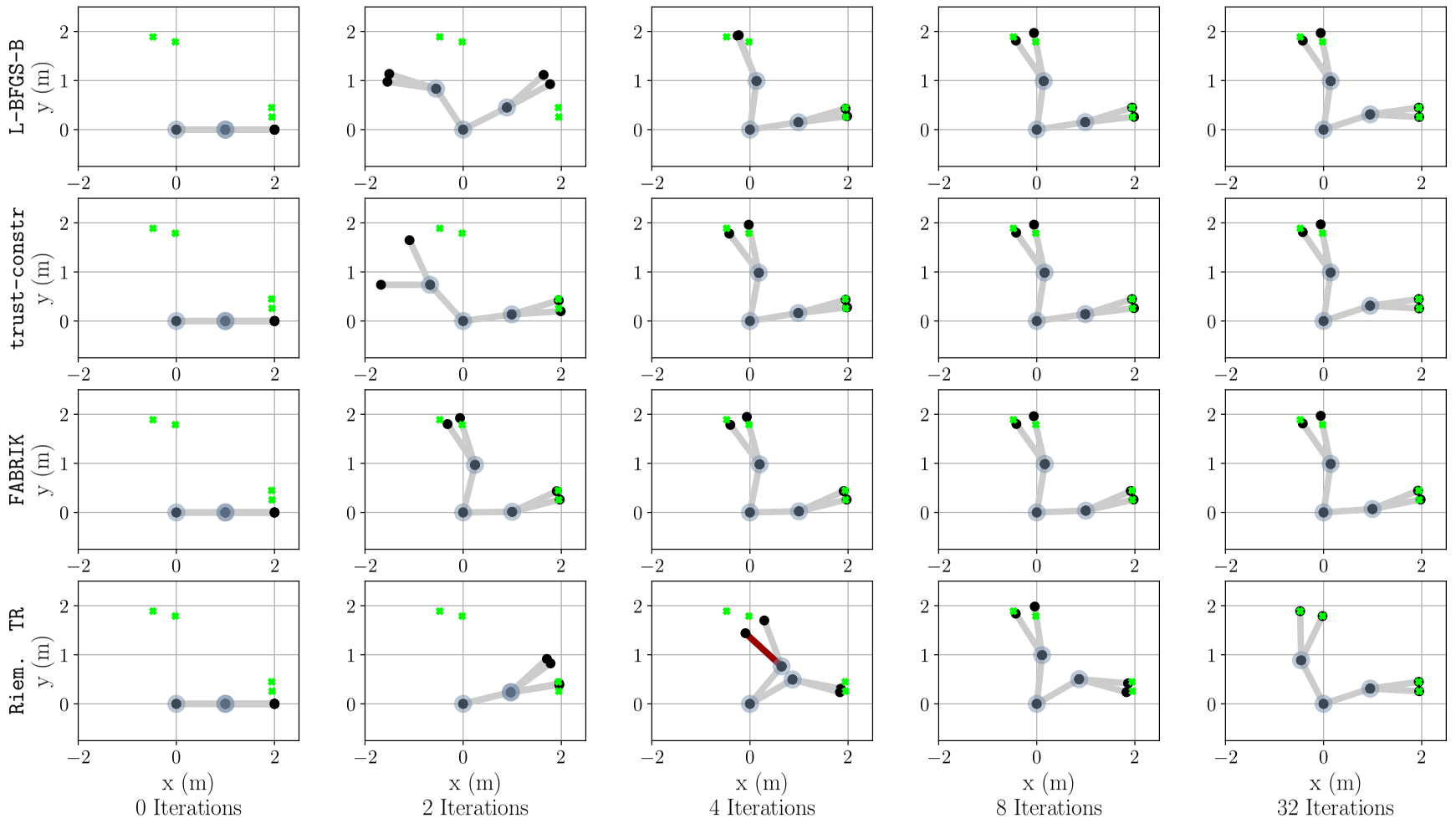
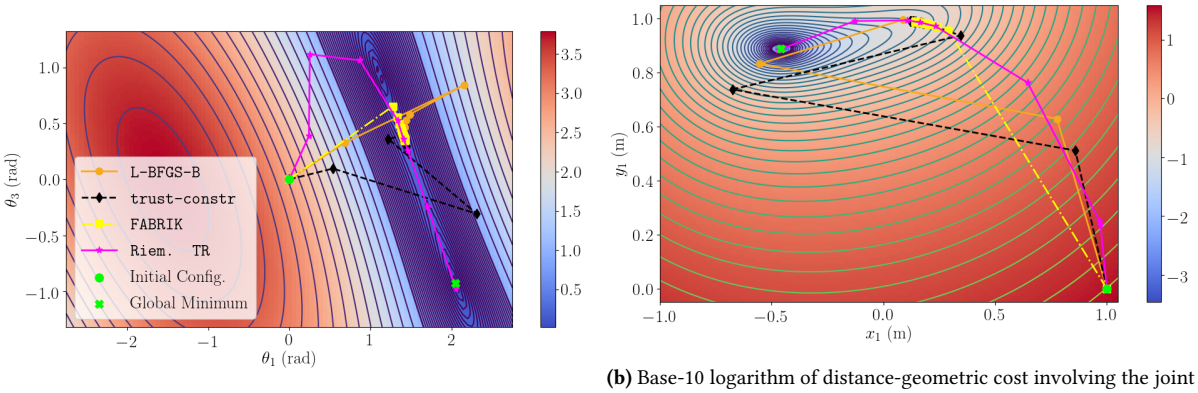


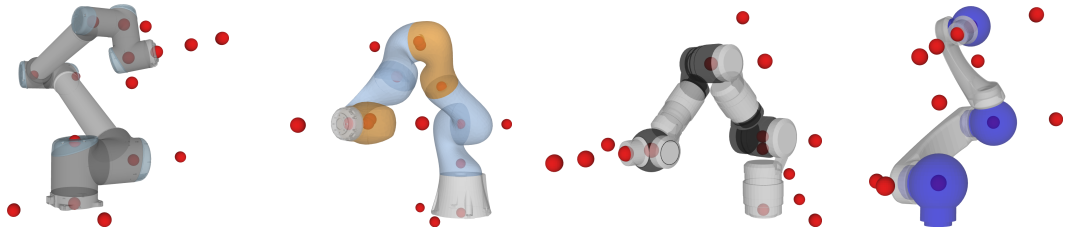
Figure 5.8: Convergence of various algorithms on a 6-DOF binary tree robots with joint angle limits. FABRIK and the angle-parametrized `L-BFGS-B` and `trust-constr` all converge to a local minimum, whereas `Riem. TR` is able to converge to the the global minimum from the same initial condition ($\boldsymbol{\theta} = \mathbf{0}$). Note that FABRIK quickly converges but is unable to accurately reach any of the four end-effector targets. The red link in the RTR solution after four iterations indicates that the link's parent joint is violating its joint angle limits.



(a) L_2 error for the end-effector actuated by θ_1 and θ_3.

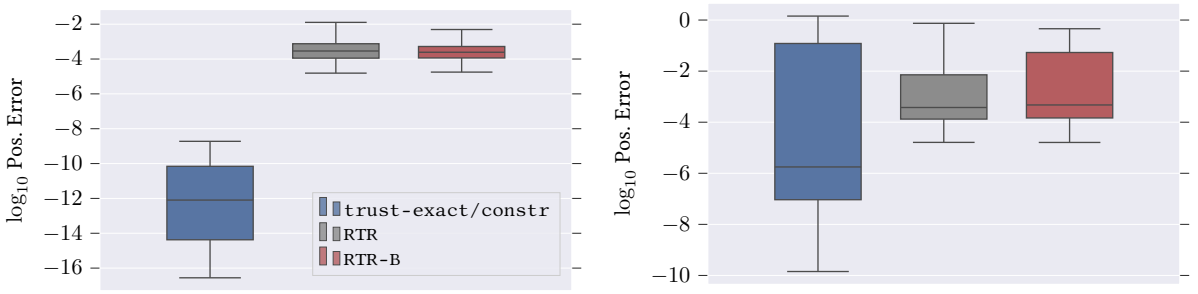
(b) Base-10 logarithm of distance-geometric cost involving the joint position which is actuated by θ_1.

Figure 5.9: Contour maps of different cost function components overlaid with solver trajectories for the IK problem instance depicted in Fig. 5.8. The angle θ_1 is the angle of the joint connected to the root and pointing towards the top left of corner of the plot of `Riem. TR` after 32 iterations in Fig. 5.8. Angles θ_3 and θ_4 are the angles of the two child links of the link actuated by θ_1. In Fig. 5.9b, $x_1 = \cos\theta_1$ and $y_1 = \sin\theta_1$ are the coordinates of the point actuated by θ_1, and the cost function is the quartic terms of Eq. (5.3) involving x_1 and y_1. This distance-geometric cost function is very well posed, and the Riemannian solver follows its contours to the global minimum. The other three methods attempt to minimize the ill-conditioned cost function in Fig. 5.9a and return a suboptimal solution.



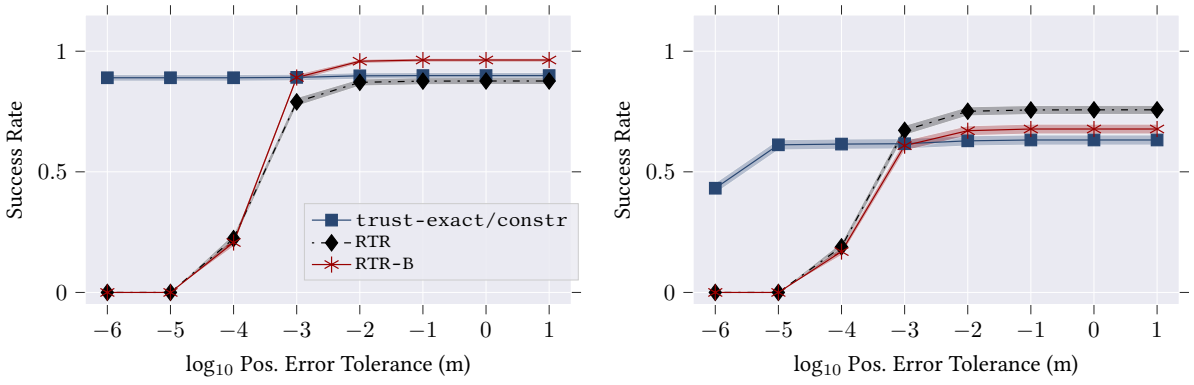
(a) UR10 **(b)** KUKA-IIWA **(c)** Schunk-LWA4D **(d)** Schunk-LWA4P

Figure 5.10: Points in the distance-based models of the commercial manipulators used in our experiments. Note that the distances between pairs of points on individual rotation axes have been reduced for clarity.



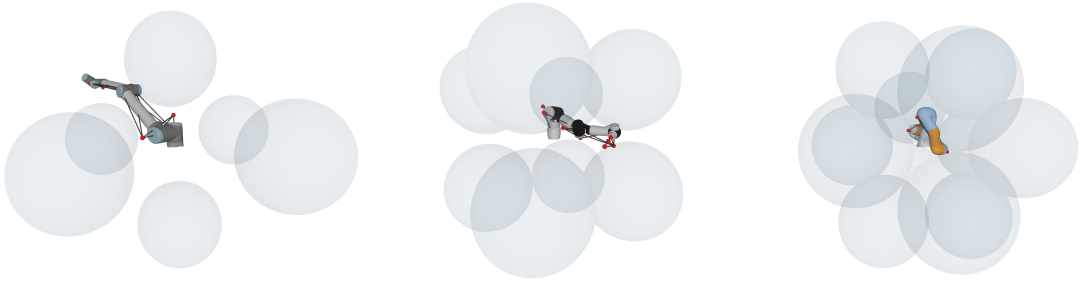
(a) UR10 **(b)** UR10+

Figure 5.11: Box-and-whiskers plots summarizing end-effector position error over 2,000 experiments with the UR10 manipulator, (a) without and (b) with joint angle limits.



(a) UR10 **(b)** UR10+

Figure 5.12: Waterfall curves of success rate versus position error tolerance for 2,000 experiments with the UR10 manipulator, without (a) and with (b) joint angle limits. The shaded regions are 95% Jeffreys confidence intervals centered on the solid lines. The rotation error tolerance is fixed at 0.01 rad.



(a) Universal Robotics UR10 (*octahedron*) **(b)** Schunk LWA4D (*cube*) **(c)** KUKA-IIWA (*icosahedron*)

Figure 5.13: A selection of robot and environment combinations used to generate IK problems in Section 5.5.5.

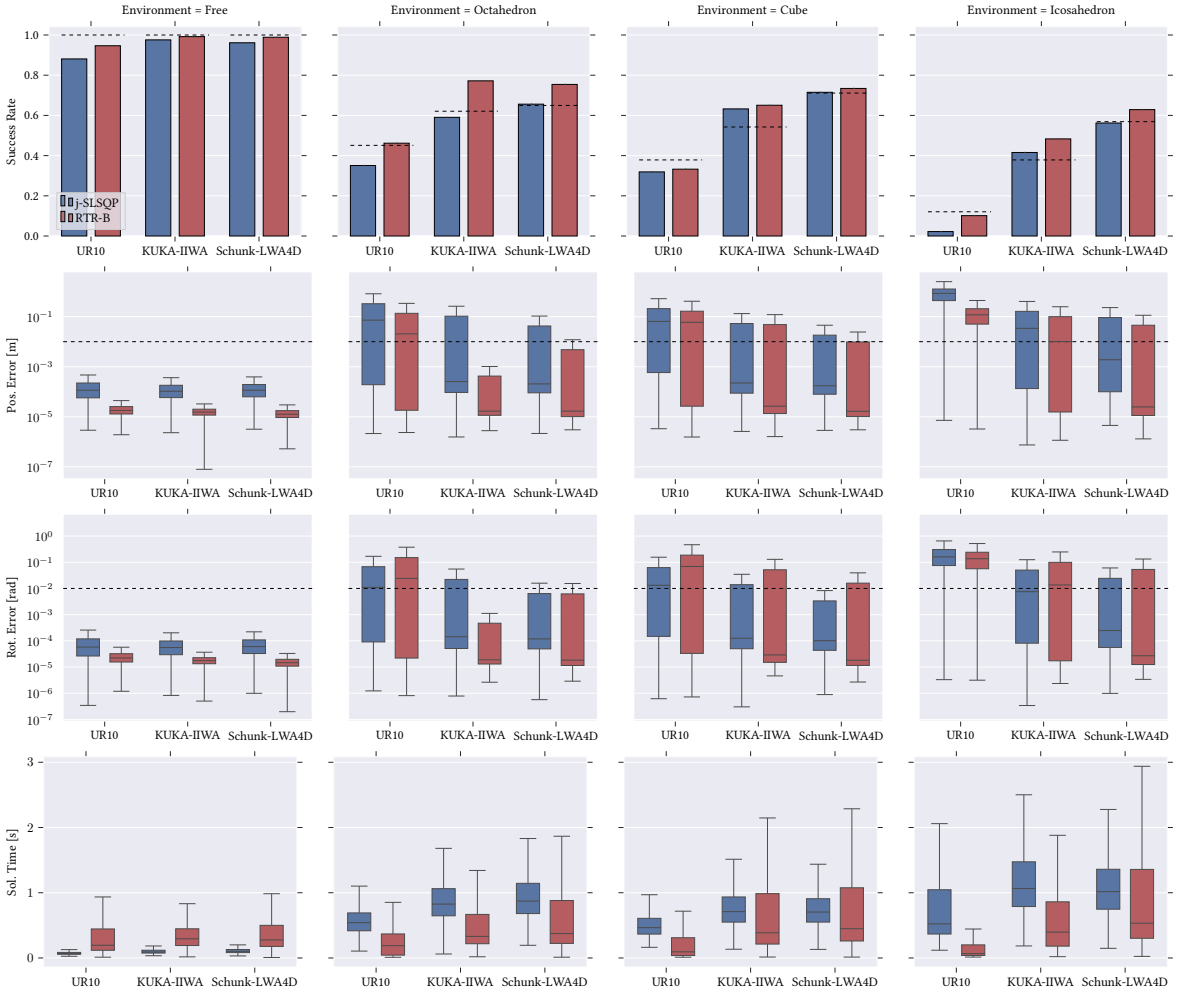


Figure 5.14: Experimental results comparing our method with local optimization. Each column contains the success rates, position errors, rotation errors, and solution times for 3,000 randomly generated problems in the titular environment. The success rate in the top row is measured relative to the total number of generated problems, many of which are infeasible. The upper bound defining success in each box-and-whiskers plot is shown as a dashed line. The dashed lines in the top row indicate the lower bound on the number of feasible problems for each robot-environment pair.

Chapter 6

Generative Graphical Inverse Kinematics

> There is no royal road to geometry.
>
> —Euclid

Quickly and reliably finding accurate inverse kinematics (IK) solutions remains a challenging problem. IK is even more difficult when hard constraints, such as obstacle avoidance, or soft constraints, such as "natural-looking" poses and motions, must be considered. Existing numerical solvers are broadly applicable, but rely on local search techniques to manage highly nonconvex objective functions. Recently, learning-based approaches have shown promise as a means to generate fast and accurate IK results; learned solvers can easily be integrated with other learning algorithms in end-to-end systems. However, learning-based methods have an Achilles' heel: each robot of interest requires a specialized model which must be trained from scratch. To address this key shortcoming, in this chapter we investigate a novel distance-geometric robot representation coupled with a graph structure that allows us to leverage the flexibility of graph neural networks (GNNs). We use this approach to train the first learned generative graphical inverse kinematics (GGIK) solver that is, crucially, "robot-agnostic"—a single model is able to provide IK solutions for a variety of different robots. In addition, the generative nature of GGIK allows the solver to produce a large number of diverse solutions in parallel with minimal additional computation time, making it appropriate for applications such as sampling-based motion planning. Finally, GGIK can complement local IK solvers by providing reliable initializations. These advantages, as well as the ability to use task-relevant priors and to continuously improve with new data, suggest that GGIK has the potential to be a key component of flexible, learning-based robotic manipulation systems.

6.1 Motivation and Related Work

In terms of success rate, learned models that output individual solutions to IK are able to compete with the best numerical IK solvers when high accuracy is not required [von Oehsen et al., 2020]. Data-driven methods have proven useful for integrating abstract criteria such as "human-like" poses or motions [Aristidou et al., 2018]. Generative approaches [Ren and Ben-Tzvi, 2020, Ho and King, 2022] have demonstrated the ability to rapidly produce a large number of approximate IK solutions and even model the entire solution set [Ames et al., 2021] for specific robots. Access to large number of configurations fitting desired constraints has proven beneficial in

motion planning applications [Lembono et al., 2021]. Unfortunately, the configurations and end-effector poses used as input-output vector pairs of deep neural networks (DNNs) associated with these architectures make it impossible to transfer learned solutions to robots that vary in link geometry and DOF. This ultimately limits the utility of learning for IK over well-established numerical methods [Beeson and Ames, 2015] that are easier to implement and generalize.

In contrast with existing DNN-based approaches [Ren and Ben-Tzvi, 2020, Lembono et al., 2021, von Oehsen et al., 2020, Ho and King, 2022, Ames et al., 2021], we explore a new pathway towards learning a generalized IK mapping by adopting a *graphical* model of robot kinematics [Porta et al., 2005a, Maric et al., 2021]. This description allows us to make use of *graph neural networks* (GNNs) described in Section 2.4.3, which we imbue with relational inductive biases in the form of specific architectural assumptions [Battaglia et al., 2018a] that capture varying robot geometries and different configuration space dimensionalities. GNNs have been successfully employed to learn object manipulation policies that generalize to tasks with a greater number of objects than seen during training [Li et al., 2019] and to varying agent structures [Wang et al., 2018, Whitman et al., 2021]. Motivated by the performance of generative models, our architecture leverages the proven capacity of Conditional Variational Autoencoders (CVAEs) to represent diverse solution sets [Kruse et al., 2021]. We describe our novel method, a generative graphical inverse kinematics (GGIK) model, and explain its capacity for representing general (i.e., not tied to a single robot model or environment) IK mappings. Our results show that, even when trained on a sparse and completely randomized dataset, our model is able to provide reasonable approximations of IK solutions for a multitude of robot manipulators.

6.1.1 Learning Inverse Kinematics

Jordan and Rumelhart [Jordan and Rumelhart, 1992] showed that the nonuniqueness of IK solutions presents a major difficulty for learning algorithms, which learn an average of the nonconvex solution set. D'Souza et al. [D'Souza et al., 2001] address this problem for differential IK by observing that the set of solution angle changes is locally convex around particular configurations. Bocsi et al. [Bócsi et al., 2011] use an *support vector machine* to model a quadratic program cost, whose solutions match those of position-only IK in particular workspace regions. In computer graphics, Villegas et al. [Villegas et al., 2018] use an RNN to solve a highly constrained IK instance of transferring motion between identical skeletons with different bone lengths. We show that a GNN-based model allows a higher degree of generalization that captures not only different link lengths, but also different numbers of DOF.

Recently, generative models have shown potential in capturing the full set of IK solutions. A number of invertible architectures [Ardizzone et al., 2018, Kruse et al., 2021] have been able to successfully capture the IK solution set for a multi-jointed 2D kinematic chain. *Generative Adversarial Networks* (GANs) have been used to learn inverse kinematics and dynamics of an 8-DOF robot [Ren and Ben-Tzvi, 2020] and improve motion planning performance by sampling configurations constrained by link positions and (partial) orientations [Lembono et al., 2021]. Recently, Ho et al. [Ho and King, 2022] proposed a model that retrives configurations reaching a target position by decoding *posture indices* for the closest position in the database. Finally, Ames et al. [Ames et al., 2021] use a model based on normalizing flows to generate a distribution of IK solutions for a desired end-effector pose. Our architecture differs from previous work by allowing the learned distribution to be generalized to a large class of robots, removing the requirement of training a new model for specific robots.

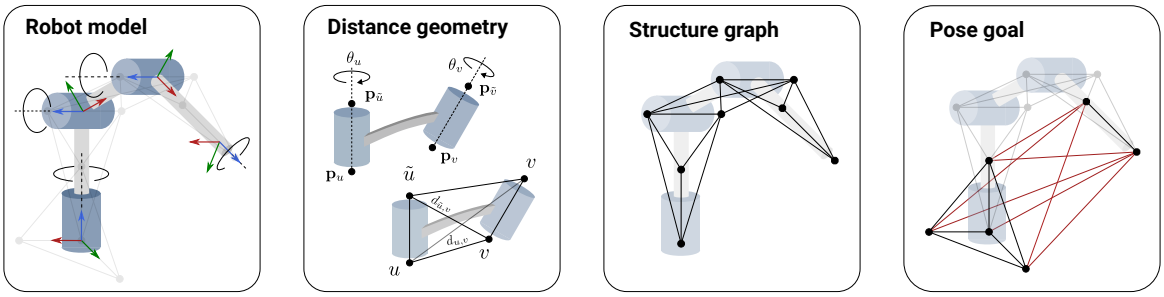


Figure 6.1: The process of defining an IK problem as an incomplete or *partial* graph $\widetilde{G}$ of inter-point distances.

6.1.2 Learning for Motion Planning

A significant amount of recent work has investigated the use of learning for classical motion planning problems. Prior applications of learning include warm-starting optimization-based methods [Ichnowski et al., 2020], learning sampling distributions for sample-based methods [Khan et al., 2020, Ichter et al., 2018], and directly learning a motion planner [Qureshi et al., 2020]. GNNs have been leveraged for their ability to efficiently encode the topology of planning problems in a permutation invariant manner [Khan et al., 2020]. The authors of [Ichter et al., 2018] and [Khan et al., 2020] explored the use of generative models to learn distributions of motion planning solutions. While the learning methodology of our work is partially similar to the aforementioned papers, we are interested in the inverse kinematics problem and not the motion planning problem. The inverse kinematics problem introduces additional interesting challenges from the perspective of learning: capturing multiple solutions and having to handle multiple manipulator classes.

6.2 Distance-Geometric Graph Representation of Robots

We eschew the common angle-based representation of the configuration space in favour of a *distance-geometric* model of robotic manipulators comprised of revolute joints based on [Porta et al., 2005a] and further developed in [Maric et al., 2021] as part of the contribution previously described in Chapter 5. This approach allows us to represent configurations $\boldsymbol{\theta}$ of robotic manipulators as complete graphs $G = (V, E)$. The graph edges E are weighted by distances d between a collection of N points with position coordinates $\mathbf{p} = \{\mathbf{p}_i\}_{i=1}^{N} \in \mathbb{R}^{N \times D}$ indexed by vertices V, where $D \in \{2, 3\}$ is the workspace dimension. The coordinates of points associated with a particular set of distances are recovered by solving the distance geometry problem (DGP) defined in Section 2.3.

Our research [Maric et al., 2021] presented in the previous chapter has shown that any solution $\mathbf{p} \in DGP(G)$ may be mapped to a unique[1] corresponding configuration $\boldsymbol{\theta}$. Moreover, we have shown that the set of configurations $IK(\mathbf{T})$ reaching a particular end-effector pose $\mathbf{T}$ may be represented by a partial graph $\widetilde{G} = (V, \widetilde{E})$. The partial graph $\widetilde{G}$ can be constructed by defining weights for edges $\widetilde{E} \subset E$ corresponding to distances defined by the end-effector pose and the robot's structure (i.e., those shared by $IK(\mathbf{T})$) where all $\mathbf{p} \in DGP(\widetilde{G})$ correspond to particular IK solutions $\boldsymbol{\theta} \in IK(\mathbf{T})$.

[1] Up to any Euclidean transformation of $\mathbf{p}$, since distances remain unchanged.

6.2.1 Partial Graph Construction

The generic procedure for constructing $\widetilde{G}$ as presented in [Maric et al., 2021] and Chapter 5 is demonstrated for a simple manipulator in Fig. 6.1. First, the *structure graph* $G_s = (V_s, E_s)$ shown in Fig. 6.1c is built by attaching two pairs of points labeled by vertices $u, \tilde{u}$ and $v, \tilde{v}$ to the rotation axes of neighbouring joints at a unit distance, as shown in Fig. 6.1b. The edges associated with every combination of points are then weighted by their respective distance, which is determined solely by the link geometry, and the process is repeated for every pair of neighbouring joints. The resulting set of vertices V_s and edges E_s, shown in Fig. 6.1c, describe the overall geometry and DOF of the robot and are invariant to feasible motions of the robot (i.e., they remain constant in spite of changes to the configuration $\boldsymbol{\theta}$).

In order to uniquely specify points with known positions (i.e., end-effectors) in terms of distances, we define the "base vertices" $V_b = \{o, x, y, z\}$, where o, z are the vertices in V_s associated with the base joint. Setting the distances weighting the edges E_b such that they form a coordinate frame with o as the origin, we specify edges E_p weighted by distances between vertices in $V_p \subset V_s$ associated with the end-effector and the base vertices V_b. The resulting subgraph $G_e(V_b \cup V_p, E_b \cup E_p)$, shown in Fig. 6.1d, uniquely specifies an end-effector pose under the assumption of unconstrained rotation of the final joint, while $\widetilde{G} = G_s \cup G_e$ is the partial graph that uniquely specifies the associated IK problem.

6.2.2 Dataset

To train our GGIK model, we require a dataset of partial graphs $\widetilde{G}$ representing IK problems associated with configurations $\boldsymbol{\theta}$ represented by complete graphs G. Conveniently, acquiring training data is fast and simple as any valid configuration can be used for training. This enables us to generate arbitrarily large datasets by sampling joint angles of various manipulators and using forward kinematics to obtain the end-effector pose. From the known robot geometry, end-effector pose, and joint angles, we are able to produce undirected partial and complete graphs $\widetilde{G} = (V, \widetilde{E})$ and $G = (V, E)$ using the procedure outlined in Section 6.2.1.

For a complete graph G, we define node features as a combination of point positions $\mathbf{p} = \{\mathbf{p}_i\}_{i=1}^{N} \in \mathbb{R}^{N \times D}$ and general features $\mathbf{h} = \{\mathbf{h}_i\}_{i=1}^{N}$, where each $\mathbf{h}_i$ is a feature vector containing extra information about each node. In GGIK, we use a three-dimensional one-hot-encoding, $\mathbf{h}_i \in \{0, 1\}^3$ and $\sum_{j=1}^{3} h_{i,j} = 1$, that indicates whether the node defines the base coordinate system, a general joint or link, or the end-effector. Similarly, we can define the M known positions of points corresponding to the end-effector and base nodes in the partial graph $\widetilde{G}$ as $\tilde{\mathbf{p}} = \{\tilde{\mathbf{p}}_i\}_{i=1}^{M} \in \mathbb{R}^{M \times D}$, and set the remaining unknown $N - M$ node positions to zero. The partial graph shares the same general features $\mathbf{h} = \{\mathbf{h}_i\}_{i=1}^{N}$ as the complete graph, given that we know which part of the robot each node belongs to in advance. However, the parital graph contains only a subset of edges defined in the complete graph, due to only certain inter-point distances being defined *a priori* by the structure and problem definition.

6.3 Learning to Generate Inverse Kinematics Solutions

We consider the problem of modelling complete graphs corresponding to IK solutions given partial graphs that define the problem instance (i.e., the robot's geometric information and the task space goal pose). Intuitively, we would like our network to map or "complete" partial graphs $\widetilde{G}$ into full graphs G. Since multiple or infinite joint configuration solutions may exist for a single task space goal $\mathbf{T}$ and robot architecture, a single partial graph may be associated with multiple or even infinite valid complete graphs corresponding to the entire solution

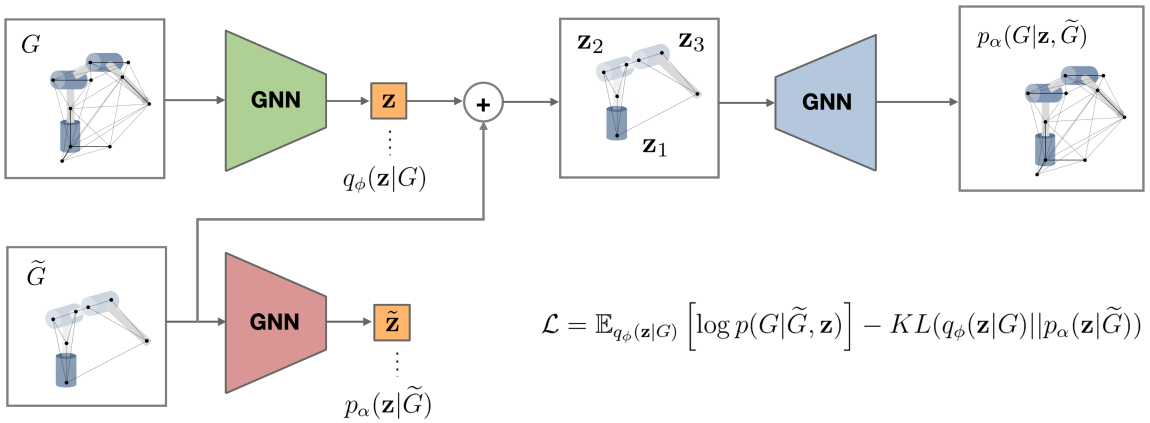


Figure 6.2: Our GGIK solver is based on the CVAE framework. The encoder GNN_{enc} (green) encodes a complete graph representation of a manipulator into a latent graph representation and GNN_{dec} (blue) "reconstructs" it. The prior network, GNN_{prior} (red), encodes the partial graph into a latent embedding that is optimized to remain near the embedding of the full graph using a KL divergence term in the loss. At test time, we decode the latent embedding of a partial graph into a complete graph to generate a solution.

set $IK(\mathbf{T})$. Having said this, we interpret the learning problem through the lens of generative modelling and treat the solution space as a multimodal distribution conditioned on a single problem instance. By sampling this distribution, we can generate diverse solutions to a given IK problem.

At its core, GGIK is a CVAE model [Sohn et al., 2015], a type of deep generative model that parameterizes a distribution conditioned by given inputs (see Section A.3). Crucially, GGIK parameterizes the conditional distribution $p(G \mid \widetilde{G})$ using GNNs using stochastic latent node embeddings $\mathbf{z}$, resulting in the generative distribution

$$p_\alpha(G \mid \widetilde{G}) = \int p_\alpha(G \mid \widetilde{G}, \mathbf{z})\, p_\alpha(\mathbf{z} \mid \widetilde{G})\, d\mathbf{z}, \tag{6.1}$$

where $p_\alpha(G \mid \widetilde{G}, \mathbf{z})$ is the likelihood of the full graph, $p_\alpha(\mathbf{z} \mid \widetilde{G})$ is the prior, and α are the learnable generative parameters. The likelihood of the full graph is given by

$$p_\alpha(G \mid \widetilde{G}, \mathbf{z}) = \prod_{i=1}^{N} p_\alpha(\mathbf{p}_i \mid \widetilde{G}, \mathbf{z}_i), \;\; \text{with} \;\; p_\alpha(\mathbf{p}_i \mid \widetilde{G}, \mathbf{z}_i) = \mathcal{N}(\mathbf{p}_i \mid \boldsymbol{\mu}_i, \mathbf{I}), \tag{6.2}$$

where $\mathbf{p} = \{\mathbf{p}_i\}_{i=i}^{N}$ are the positions of all N nodes, $\mathbf{z} = \{\mathbf{z}_i\}_{i=i}^{N}$ are the latent embeddings of each node, and $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}_{i=i}^{N}$ are the predicted means of the distribution of node positions. We parametrize the likelihood distribution with a GNN decoder, in other words, $\boldsymbol{\mu}$ is the output of $\text{GNN}_{dec}(\widetilde{G}, \mathbf{z})$. The GNN decoder propagates messages and updates the nodes at each intermediate layer and outputs the predicted means of all node positions at the final layer. In practice, for the input of $\text{GNN}_{dec}(\cdot)$, we concatenate each latent node with the respective position node features $\tilde{\mathbf{p}}$ of the original partial graph $\widetilde{G}$ when avialable. For positions of nodes that are unknown *a priori*, we concatenate the latent node embeddings with the initialized point positions set to zero. We follow the common practice of only learning the mean of the decoded Gaussian likelihood distribution and use a fixed diagonal covariance matrix $\mathbf{I}$ [Doersch, 2016]. The prior distribution is given by

$$p_\alpha(\mathbf{z} \mid \widetilde{G}) = \prod_{i=1}^{N} p_\alpha(\mathbf{z}_i \mid \widetilde{G}), \;\; \text{with} \;\; p_\alpha(\mathbf{z}_i \mid \widetilde{G}) = \sum_{k=1}^{K} \pi_{k,i}\, \mathcal{N}\left(\mathbf{z}_i \mid \boldsymbol{\mu}_{k,i}, \text{diag}(\boldsymbol{\sigma}_{k,i}^2)\right). \tag{6.3}$$

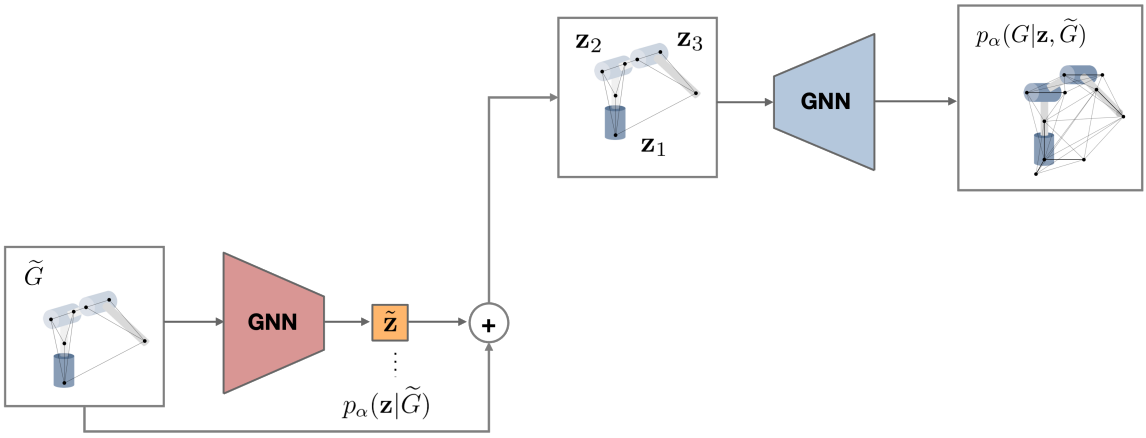


Figure 6.3: At test time, our GGIK solver uses the prior network GNN_{prior} (blue) to encode the partial graph into a multi-model latent distribution shown in Eq. (6.3). This distribution is then sampled an passed to the decoder GNN_{dec} that outputs a Gaussian distribution over node positions that approximates an IK solution.

Here, we parameterize the prior as a Gaussian mixture model with K components. Each Gaussian is in turn parameterized by a mean $\boldsymbol{\mu}_k = \{\boldsymbol{\mu}_{k,i}\}_{i=1}^N$, diagonal covariance $\boldsymbol{\sigma}_k = \{\boldsymbol{\sigma}_{k,i}\}_{i=1}^N$, and a mixing coefficient $\boldsymbol{\pi}_k = \{\pi_{k,i}\}_{i=1}^N$, where $\sum_{k=1}^K \pi_{k,i} = 1,\ \forall i = 1, ..., N$. We chose a mixture model as an expressive prior, capable of capturing the latent distribution representing multiple IK solutions. We parameterize the prior distribution with a multi-headed GNN encoder $\text{GNN}_{prior}(\widetilde{G})$ that outputs parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\sigma}_k, \boldsymbol{\pi}_k\}_{k=1}^K$.

The goal of learning is to maximize the marginal likelihood or evidence of the data as shown in Eq. (A.9). As commonly done in the variational inference literature [Kingma and Welling, 2014], we instead maximize a tractable evidence lower bound (ELBO):

$$\mathcal{L} = \mathbb{E}_{q_\phi(\mathbf{z}\,|\,G)}[\log p_\alpha(G\,|\,\widetilde{G}, \mathbf{z})] - KL(q_\phi(\mathbf{z}\,|\,G)||p_\alpha(\mathbf{z}\,|\,\widetilde{G})). \tag{6.4}$$

Finally, the inference model $q_\phi(\mathbf{z}\,|\,G)$ with learnable parameters ϕ is defined as:

$$q_\phi(\mathbf{z}\,|\,G) = \prod_{i=1}^N q_\phi(\mathbf{z}_i\,|\,G), \quad \text{with} \quad q_\phi(\mathbf{z}_i\,|\,G) = \mathcal{N}(\mathbf{z}_i\,|\,\boldsymbol{\mu}_i, \text{diag}(\boldsymbol{\sigma}_i^2)). \tag{6.5}$$

As with the prior distribution, we parameterize the inference distribution with a multi-headed GNN encoder, $\text{GNN}_{enc}(G)$, that outputs parameters $\boldsymbol{\mu} = \{\boldsymbol{\mu}_i\}_{i=1}^N$ and $\boldsymbol{\sigma} = \{\boldsymbol{\sigma}_i\}_{i=1}^N$. The inference model is an approximation of the intractable true posterior $p(\mathbf{z}\,|\,G)$. We note that the resulting ELBO objective in Eq. (6.4) is based on an expectation with respect to the inference distribution $q_\phi(\mathbf{z}\,|\,G)$, which itself is based on the parameters ϕ. Since we restrict $q_\phi(\mathbf{z}\,|\,G)$ to be a Gaussian variational approximation, we can use stochastic gradient descent (i.e., Monte Carlo gradient estimates) via the reparameterization trick [Kingma and Welling, 2014] to optimize the lower bound with respect to parameters α and ϕ.

At test time, given a goal pose and the manipulator's geometric information encapsulated in a partial graph $\widetilde{G}$, we can use the prior network, GNN_{prior}, to encode the partial graph into a latent distribution $p_\alpha(\mathbf{z}\,|\,\widetilde{G})$. During training, the distribution $p_\alpha(\mathbf{z}\,|\,\widetilde{G})$ is optimized to be simultaneously near multiple encodings of valid solutions contained in the inference distribution $q_\phi(\mathbf{z}\,|\,G)$ by way of the KL divergence term in Eq. (6.4). We can sample this multimodal distribution $\mathbf{z} \sim p_\alpha(\mathbf{z}\,|\,\widetilde{G})$ as many times as needed, and subsequently decode all of the samples with the decoder network GNN_{dec} to generate IK solutions represented as complete graphs. This

procedure is demonstrated in Fig. 6.3, and can be done quickly and in parallel on the GPU.

6.4 $E(n)$ Equivariant Network Architecture

In this section, we discuss the choice of architecture for networks GNN_{dec}, GNN_{enc}, and GNN_{prior}. Recall that we are interested in mapping partial graphs $\widetilde{G}$ into full graphs G. Once trained, our model maps partial point sets to full point sets $f : \mathbb{R}^{M\times D} \to \mathbb{R}^{N\times D}$, where f is a combination of networks GNN_{prior} and GNN_{dec} applied sequentially. The point positions (i.e., $\mathbf{p}$ and $\tilde{\mathbf{p}}$) assigned to each node and the distances weighting the associated edges in the distance geometry problem contain underlying geometric relationships that we would like to preserve in our choice of architecture. Crucially, the point sets are *equivariant* to the Euclidean group $E(n)$ of rotations, translations, and reflections. Let $S : \mathbb{R}^{M\times D} \to \mathbb{R}^{M\times D}$ be a transformation consisting of some combination of rotations, translations and reflections on the initial partial point set $\tilde{\mathbf{p}}$. Then, there exists an equivalent transformation $T : \mathbb{R}^{N\times D} \to \mathbb{R}^{N\times D}$ on the complete point set $\mathbf{p}$ such that:

$$f(S(\tilde{\mathbf{p}})) = T(f(\tilde{\mathbf{p}})). \tag{6.6}$$

To leverage this structure or geometric prior in the data, we use $E(n)$-equivariant graph neural networks (EGNNs) [Satorras et al., 2021] for GNN_{dec}, GNN_{enc}, and GNN_{prior}.

The EGNN layer splits up the node features into an equivariant coordinate or position-based part and a nonequivariant part. We treat the positions $\mathbf{p}$ and $\tilde{\mathbf{p}}$ as the equivariant portion and the general features $\mathbf{h}$ as nonequivariant. Using the graph network formalism for describing generic GNNs defined in Section 2.4.3, we can state the EGNN update equations used in GNN_{enc}, GNN_{dec} and GNN_{prior}. First, the MLP NN^e generates a generalized edge feature embedding

$$\mathbf{e}'_{ij} = \text{NN}^e\left(\mathbf{h}^l_i, \mathbf{h}^l_j, \|\mathbf{p}^l_i - \mathbf{p}^l_j\|^2, d_{i,j}\right) \tag{6.7}$$

for every node pair, where $d_{i,j}$ is their relative distance, and $\mathbf{p}$ and $\mathbf{h}$ are the equivariant and general features of each node, respectively. Next, each node's equivariant feature $\mathbf{p}$ is updated by

$$\mathbf{p}'_i = \mathbf{p}_i + \text{NN}^p\left(\sum_{j\neq i}(\mathbf{p}^l_i - \mathbf{p}^l_j)\mathbf{e}'_{ij}\right), \tag{6.8}$$

computing a forward pass of the MLP NN^p. Finally, each node's general (nonequivariant) feature $\mathbf{h}$ is updated by

$$\mathbf{h}'_i = \text{NN}^h\left(\mathbf{h}_i, \sum_{j\in N(i)} \mathbf{e}'_{ij}\right), \tag{6.9}$$

where NN^h is the MLP dedicated to this particular set of features.

Together, the EGNN identities in Eq. (6.7), Eq. (6.8) and Eq. (6.9) form a forward pass of a single GNN layer that updates a graph G as

$$G'(\mathbf{p}', \mathbf{h}') = \text{GNN}(G(\mathbf{p}, \mathbf{h}))\,. \tag{6.10}$$

Note that GNN_{enc}, GNN_{dec} and GNN_{prior} are comprised of multiple (generally two or three) EGNN layers, each parameterized by its own set of MLPs. Alternatively, the forward pass of one layer can be repeated multiple times, but we have empirically found this to produce worse results overall. For more details about the model

Table 6.1: Performance of GGIK on 2,000 randomly generated IK problems for five different robotic manipulators. Taking 50 samples from the learned distribution, the error statistics are presented as the mean and mean minimum error per problem, as well as the percentage of "successes," defined as solutions with a position error lower than 1 cm and rotation error lower than 1 degree. The mean MMD score measures how different GGIK's samples are from a uniform distribution approximated by rejection sampling with error tolerances of 8 cm and 8 degrees. Note that all solutions were produced by a *single* model.

Robot	Err. Pos. [mm]					Err. Rot. [deg]					Success [%]	MMD
	mean	min	25%	50%	75%	mean	min	25%	50%	75%		
kuka	15.1	2.7	7.1	10.2	15.4	2.0	0.5	1.1	1.5	2.1	99.0	0.06
lwa4d	13.9	2.3	6.5	9.7	14.6	2.3	0.7	1.4	1.8	2.5	99.0	0.06
lwa4p	6.6	2.3	4.6	6.3	8.3	1.8	1.0	1.4	1.7	2.1	98.2	0.21
panda	36.5	3.8	13.7	24.4	43.7	3.6	0.4	1.4	2.4	4.3	97.5	0.08
ur10	17.0	5.1	9.9	14.3	20.7	1.7	0.8	1.2	1.5	1.9	90.0	0.21

and a proof of the equivariance property, we refer readers to [Satorras et al., 2021]. We present ablation studies on the use of the EGNN network architecture in Section 6.5.

6.5 Experimental Results

In this section, we (i) evaluate GGIK's capability to learn accurate solutions and generalize across common manipulators, (ii) determine whether GGIK can be used effectively to initialize local numerical IK solvers, and (iii) investigate the importance of our choice of learning architecture.

6.5.1 Accuracy and Generalization of GGIK Across Manipulators

In our first experiment, we evaluate the accuracy and generalization capacity of GGIK by training a single model on a dataset comprised of a variety of manipulators featuring different structures and numbers of joints. All experiments were performed on a laptop computer with a six-core 2.20 GHz Intel i7-8750H CPU and an NVIDIA GeForce GTX 1050 Ti Mobile GPU.

We evaluate a model trained on a total of 256,000 data points uniformly distributed over five different commercial manipulators. The success rates in Table 6.1, obtained by selecting the lowest-error sample from the learned distribution, suggest that this approach is feasible for directly generating solutions in a variety of practical applications. Moreover, the position and orientation error percentiles indicate that the majority of sampled configurations correspond to end-effector poses in close proximity to the goal pose. Consequently, samples from the learned distributions such as those shown in Fig. 6.4 can be refined to arbitrary accuracy with only a few iterations of a local optimizer. The maximum mean discrepancy (MMD) score [Gretton et al., 2012] between GGIK's samples and a uniform distribution computed with rejection sampling is reported for each robot as the mean over 50 goal poses with 50 sample solutions for each goal pose. The MMD attains a minimum value of zero for identical distributions; a low MMD indicates that GGIK has learned a distribution that is close to uniform over the solution set for each goal pose [Ames et al., 2021]. Interestingly, while Fig. 6.4 indicates that GGIK successfully captures both the continuous solution sets of the redundant robots and the discrete solution sets characteristic of the 6-DOF Schunk LWA4P and UR10, the MMD scores for the two 6-DOF manipulators are significantly greater.

Table 6.2: Comparison of random samples and GGIK samples as initializations for local optimization on the same problems as in Table 6.1. GGIK significantly reduces the number of iterations needed for convergence.

Initialization	Err. Pos. [mm]					Err. Rot. [deg]					Success [%]	Num. Iter.
	mean	min	25%	50%	75%	mean	min	25%	50%	75%		
Random	3.8	0.0	0.2	1.4	6.3	0.04	0.0	0.0	0.0	0.04	99.9	28.7
GGIK	1.0	0.0	0.3	0.7	1.4	0.0	0.0	0.0	0.0	0.0	99.9	11.1

Table 6.3: Comparison of different network architectures. EGNN outperforms existing architectures that are not equivariant in terms of overall accuracy and test ELBO.

Model Name	Err. Pos. [mm]					Err. Rot. [deg]					Test ELBO	Success [%]
	mean	min	25%	50%	75%	mean	min	25%	50%	75%		
EGNN	18.4	6.4	14.1	16.5	17.1	1.7	0.8	1.1	1.6	1.7	-3.8	96.3
MPNN	143.2	48.9	114.6	174.8	175.9	17.7	15.3	15.4	17.5	19.4	-8.3	13.1
GAT	-	-	-	-	-	-	-	-	-	-	-12.41	0.0
GCN	-	-	-	-	-	-	-	-	-	-	-12.42	0.0
GRAPHsage	-	-	-	-	-	-	-	-	-	-	-10.5	0.0

6.5.2 Initializing a Local Numerical IK Solver with GGIK

GGIK learns a sampling distribution capable of producing multiple approximate solutions in parallel, which may be used as initializations for optimization-based methods. Table 6.2 shows the results of repeating the experiment in Section 6.5.1 using an IK solver based on the SLSQP algortihm implemented in the `scipy.optimize` package [Virtanen et al., 2020a], setting a maximum of 100 iterations and keeping other termination criteria at their default values. We compare the solver's accuracy and number of iterations required before convergence when initialized with 32 random configurations per problem and 32 samples from our learned distribution. We average the results over all problems and all robots. While both approaches achieve a high degree of success and accuracy on these unconstrained problems, the results clearly show that using our model to initialize the optimization produces better overall performance. Specifically, the mean number of iterations required compared to random initializations is almost three times lower, and the accuracy of the solutions is notably higher as well. Our model could be used as a fallback approach for difficult or highly constrained instances of IK, or as a general initializer ensuring accurate final results.

6.5.3 Ablation Study on the Equivariant Network Architecture

We conduct an ablation experiment to evaluate the importance of capturing the underlying $E(n)$ equivariance of the distance geometry problem (Problem 1) in our learning architecture. We compare the use of the EGNN network [Satorras et al., 2021] to four common and popular GNN layers that are not $E(n)$ equivariant: GRAPHsage [Hamilton et al., 2017], GAT [Velickovic et al., 2018], GCN [Kipf and Welling, 2017] and MPNN [Gilmer et al., 2017]. We match the number of parameters for each GNN architecture as closely as possible and keep all other experimental parameters fixed. Our dataset is the same one used in Section 6.5.1, however the results are averaged over all manipulators as shown in Table 6.3. Out of the five different architectures that we compare, only the EGNN and MPNN output point sets that can be successfully mapped to valid joint configurations. Point sets that are too far from those representing a valid joint configuration result in the configuration reconstruction procedure diverging. The equivariant EGNN model outperforms all other models in terms of the ELBO value attained on a held-out test set from our original training data. Our ablation results emphasize the importance

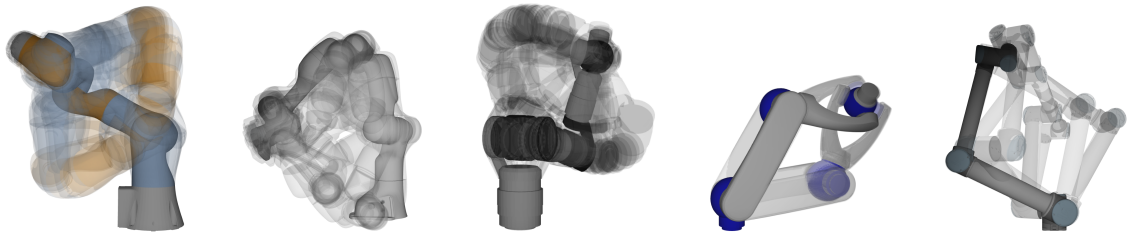


Figure 6.4: Sampled conditional distributions for various robotic manipulators. From left to right: KUKA `IIWA`, `Franka Emika Panda`, `Schunk LWA4D`, `Schunk LWA4P`, and `Universal Robots UR10`.

of choosing a learning architecture that properly captures the representation utilized in the distance-geometric IK problem formulation.

6.6 Summary and Conclusions

We have presented GGIK, a generative graphical IK solver that is able to produce multiple diverse and accurate solutions in parallel across many different manipulator types. This capability is achieved through a distance-geometric representation of the IK problem in concert with GNNs. The accuracy of the generated solutions points to the potential of GGIK both as an approximate standalone solver and as an initialization method for local approaches. To our knowledge, this is the first approach that allows the generation of multiple solutions for multiple different robots using only a single model. More important, because GGIK is fully differentiable, it can be incorporated as a flexible IK solver as part of an end-to-end learning-based robotic manipulation framework.

In future work, we would like to learn constrained distributions of robot configurations that account for obstacles in the task space; obstacles can be easily incorporated in the distance-geometric representation of IK [Maric et al., 2021, Giamou et al., 2022]. Learning an obstacle-aware distribution would yield a solver that implements obstacle avoidance by way of message passing between manipulator and obstacle nodes. Having access to such distributions would be highly useful in motion planning applications, where configurations could be sampled within regions that satisfy some set of task space constraints, likely speeding up many existing motion planning ealgorithms.

6.6.1 Limitations

While our proposed architecture demonstrates a high capacity for generalization, generated solutions may require post-processing by local optimization methods in certain applications in order to adhere to constraints more tightly. Moreover, the underlying distance-geometric model is subject to ambiguities in cases where the rotation axes of consecutive joints are not co-planar, as explained in Section 5.3.5. The learned architecture has the capacity to resolve this ambiguity by only observing feasible configurations in the training set, however the model does generally require more training samples of such robots to achieve baseline reasults.

6.6.2 Associated Publications

This research extends the work presented in Chapter 5, making it the final contribution of this thesis both semantically and chronologically. While these results have not yet been published, the initial idea was presented in an invited talk at the 2021 Robotics: Science and Systems conference, titled “Of Ellipsoids and Distances: Reconsidering Robot Kinematics”. We have also recently compiled this work in a shared-authorship preprint

- Limoyo O., Marić F., Giamou M., Alexson P., Petrović, I., Kelly, J. (2022). One Model, Many Robots: Generative Graphical Inverse Kinematics

submitted to the 2022 Conference on Robot Learning (CoRL).

Chapter 7

Conclusion

> If you know the way broadly you will see it in everything.
>
> Miyamoto Musashi

In this thesis, we explored a variety of novel geometric formulations of problems involving the identification and optimization of feasible robotic manipulator configurations. The primary motivation behind this endeavour was to examine the idea that the alternative, geometric perspective reveals important, common 'structural' properties within these problems. These properties can then be exploited to improve the performance of existing algorithms, improving their robustness by circumventing failure modes, for example, or by increasing their overall success rate.

7.1 Summary of Contributions

In our first contribution, we have demonstrated that the important problem of singularity avoidance admits a geometrically-interpretable singularity index that helps detect and avoid the common failure modes of existing indices used for singularity avoidance. Crucially, we have found that this index can be differentiated with respect to joint angles using a result from computational matrix analysis. We have shown that optimizing the geometry-aware singularity index improves singularity avoidance performance of task space control (as well as differential and pose IK) algorithms with little to no additional computational cost.

Next, we developed a detailed and algorithmic approach for constructing distance-based models of manipulator kinematics, unifying the configuration and task spaces. This systematic approach has enabled us to draw a formal equivalence between the distance geometry problem and inverse kinematics, thereby linking inverse kinematics to a plethora of diverse and previously unused solution methods. We have shown that taking on this geometric perspective enables the development of robust solution formulations that are able to handle problems with a high number of spatial constraints. The particular Riemannian formulation presented in this thesis has shown superior performance in terms of success rate compared to traditional angle-based solvers.

Finally, we explored the potential of using our distance-geometric model to structure and transform robot configuration data into a distance geometric graph, revealing an intuitive and robot-agnostic inverse kinematics generative model architecture. Remarkably, we have found that our generative model surpasses the capabilities of many existing learning approaches to inverse kinematics problem, not only by providing highly accurate

solutions, but by generalizing to different robots. Moreover, we have also shown that this generative architecture may serve as an efficient initializer for optimization-based solution methods, increasing the success rate and reducing the number of iterations required to reach a solution.

Here we include a summary of the contributions in this thesis:

- a measure of proximity to kinematic singularities inherent to robotic manipulators, derived from Riemannian geometry;
- a closed-form expression for computing the geometry-aware singularity index Jacobian;
- a fully algorithmic approach to modeling inverse kinematics problems as an instance of the distance geometry problem;
- a formal proof of equivalence between inverse kinematics and the distance gometry problem;
- a method for solving IK of the distance-based kinematic model using Riemannian geometry and a distance matrix completion method;
- a way to exploit the inherent graph structure of the distance-based kinematic model to formulate inverse kinematics as a learning problem for graph neural networks (GNNs); and
- a learned model of an inverse kinematics solver capable of generalizing to a variety of different robots using GNNs and graph-structured data

Overall, we hope that the contributions in this thesis insipire other authors by proving that there is untapped potential in exploiting the known geometry of various elements in robotic manipulation for developing new and more efficient planning and control algortihms.

7.2 Future Work

The research presented in this thesis has been focused on unlocking the potential for applying novel tools to existing problems, opening many potential avenues for future work. The geometry-aware singularity avoidance index presented in Chapter 4 may be integrated in many existing motion planning and control pipelines for manipulation to provide more dexterous and stable configurations than baseline approaches. More importantly, we expect that developing methods for choosing appropriate reference elliposids for particular tasks may provide an even more significant improvement.

The distance-distance geometric model in Chapter 5 holds promise for applications in motion planning due to its unification of the task and configuration spaces. For example, motion planning in cluttered environments would be improved by using this more robust approach to identify key collision-free configurations that are to be interpolated using a trajectory optimization algortihm. Further, the distance-based model opens up possibilities for using solution methods from other related fields, such as the one explored in [Giamou et al., 2022]. Working in a single space of inter-point distances (as opposed to the twin configuration and task spaces), these methods hold promise for providing higher success rates in difficult IK problem instances contatining a large number of spatial constraints. Overall we see this approach as a basis for a new generation of "heavy-duty" IK solvers.

Finally, the contribution in Chapter 6 has demonstrated the potential of learning approaches for problems in kinematics where access to a large variety of solutions is beneficial. In future work, we hope to further develop this approach to find the first truly general learned IK solver that accounts for constraints such as obstacle avoidance. Sampling-based motion planners would benefit from access to spatially-localized samples that repsect elements of workspaces geometry. We believe that the overall GNN-based generative architecture may be extended to other geometrc representations, such as the more common matrix Lie groups.

Appendices

Appendix A

Learning

This appendix to the thesis brifly introduces two types of learning tasks and a particular learning architecture relevant to our contributions. We begin by introducing the concepts of unsupervised learning and supervised learning, which gives more insight into the "placement" of the contribution in Chapter 6 in the overall body of research. We follow this up by introducing deep generative models, that lie at the intersection of the two learning types and serve as a bases for our final contribution.

A.1 Unsupervised Learning

In *unsupervised learning* involves capturing patterns in unlabeled training data $\mathcal{D} \triangleq \{(\mathbf{x}_i)\}_1^N$ with the goal of finding lower-dimensional representations or models that are able to produce new data similar to the training set. Examples of unsupervised learning include tasks such as synthesizing writing or images, as well as generating motion plans for robots. Here, the learned model then defines a likelihood $p_{\boldsymbol{\alpha}}(\mathbf{x}) = p(\mathbf{x}; \boldsymbol{\alpha})$, which can be maximized by minimizing the negative log-likelihood $-E_{\mathbf{x}} \log p_{\boldsymbol{\alpha}}(\mathbf{x})$. This identity can be used to find an estimated parameter value $\boldsymbol{\alpha}$ using maximum likelihood (ML) or *maximum a posteriori* (MAP) estimation, representing a frequentist or Bayesian approach.

We can apply the principle of maximum likelihood to find $\boldsymbol{\alpha}$ by directly solving the optimization problem

$$\boldsymbol{\alpha}_{ML} = \max_{\boldsymbol{\alpha}} E_{\mathbf{x}} p_{\boldsymbol{\alpha}}(\mathbf{x}) = \min_{\boldsymbol{\alpha}} -E_{\mathbf{x}} \log p_{\boldsymbol{\alpha}}(\mathbf{x}) \,, \tag{A.1}$$

For example, assuming that $p_{\boldsymbol{\alpha}}(\mathbf{x}) = \mathcal{N}\left(\mathrm{NN}(\mathbf{x}; \boldsymbol{\alpha}), \mathbf{I}\right)$, we arrive at

$$\mathcal{L}_{ML}(\mathbf{x}, \boldsymbol{\alpha}) = -E_{\mathbf{x}} \log p_{\boldsymbol{\alpha}}(\mathbf{x}) = \frac{1}{2} E_{\mathbf{x}} \left\| \mathbf{x} - \mathrm{NN}(\mathbf{x}; \boldsymbol{\alpha}) \right\|^2 + \mathrm{const} \,, \tag{A.2}$$

which is simply the mean-squared error (MSE) loss.

Unlike the ML approach, which assumes the true set of parameters $\boldsymbol{\alpha}$ is unknown but fixed, MAP estimation assigns them a prior distribution $p(\boldsymbol{\alpha})$. From the Bayes' rule we have $p(\boldsymbol{\alpha}|\mathbf{x}) \propto p(\mathbf{x}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha})$, where $p(\mathbf{x}|\boldsymbol{\alpha}) = p_{\boldsymbol{\alpha}}(\mathbf{x})$. This allows us to estimate the parameters $\boldsymbol{\alpha}$ by solving the optimization problem

$$\boldsymbol{\alpha}_{MAP} = \max_{\boldsymbol{\alpha}} E_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) = \min_{\boldsymbol{\alpha}} -E_{\mathbf{x}} \left(\log p_{\boldsymbol{\alpha}}(\mathbf{x}) + \log p(\boldsymbol{\alpha}) \right) . \tag{A.3}$$

Assuming that $p_{\boldsymbol{\alpha}}(\mathbf{x}) = \mathcal{N}\left(\mathrm{NN}(\mathbf{x};\boldsymbol{\alpha}), \mathbf{I}\right)$ and $p(\boldsymbol{\alpha}) = \mathcal{N}(0, \lambda^{-2}\mathbf{I})$ results in

$$\mathcal{L}_{MAP}(\mathbf{x},\boldsymbol{\alpha}) = \frac{1}{2}E_{\mathbf{x}}\left\|\mathbf{x} - \mathrm{NN}(\mathbf{x};\boldsymbol{\alpha})\right\|^2 + \lambda\boldsymbol{\alpha}\,\boldsymbol{\alpha}^\intercal + \mathrm{const}\,. \tag{A.4}$$

Which is equivalent to an MSE with an added regularizing weight decay penalty.

A.2 Supervised learning

In *supervised learning*, the goal is to learn a function based on N input-output pairs found in the training data $\mathcal{D} \triangleq \{(\mathbf{x}_i, \mathbf{y}_i)\}_1^N$. Examples of supervised learning tasks include time-series prediction, handwriting recognition and spam detection. Here, the learned model then defines a distribution $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) = p(\mathbf{y}|\mathbf{x};\boldsymbol{\alpha})$. Again, this identity can be used to find an estimated parameter value $\boldsymbol{\alpha}$ using maximum likelihood (ML) or *maximum a posteriori* (MAP) estimation.

We can apply the principle of maximum likelihood to find an appropriate $\boldsymbol{\alpha}$ by directly solving the optimization problem

$$\boldsymbol{\alpha}_{ML} = \max_{\boldsymbol{\alpha}} E_{\mathbf{x},\mathbf{y}} p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) = \min_{\boldsymbol{\alpha}} -E_{\mathbf{x},\mathbf{y}} \log p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x})\,. \tag{A.5}$$

Assuming that $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathrm{NN}(\mathbf{x};\boldsymbol{\alpha}), \mathbf{I}\right)$, we arrive at

$$\mathcal{L}_{ML}(\mathbf{y},\mathbf{x},\boldsymbol{\alpha}) = -E_{\mathbf{x},\mathbf{y}} \log p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) = \frac{1}{2}E_{\mathbf{x},\mathbf{y}}\left\|\mathbf{y} - \mathrm{NN}(\mathbf{x};\boldsymbol{\alpha})\right\|^2 + \mathrm{const}\,, \tag{A.6}$$

which is simply the mean-squared error (MSE) loss. Note that $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x})$ may be defined differently for tasks such as binary classification, where the output labels have only two discrete categories (e.g., hot dog or not hot dog).

For the MAP estimator, the Bayes' rule gives $p(\boldsymbol{\alpha}|\mathbf{x},\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x},\boldsymbol{\alpha})p(\boldsymbol{\alpha})$, where $p(\mathbf{y}|\mathbf{x},\boldsymbol{\alpha}) = p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x})$. This allows us to estimate the parameters $\boldsymbol{\alpha}$ by solving the optimization problem

$$\boldsymbol{\alpha}_{MAP} = \min_{\boldsymbol{\alpha}} -E_{\mathbf{x},\mathbf{y}}(\log p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) + \log p(\boldsymbol{\alpha}))\,. \tag{A.7}$$

Assuming that $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) = \mathcal{N}\left(\mathrm{NN}(\mathbf{x};\boldsymbol{\alpha}), \mathbf{I}\right)$ and $p(\boldsymbol{\alpha}) = \mathcal{N}(0, \lambda^{-2}\mathbf{I})$ results in

$$\mathcal{L}_{MAP}(\mathbf{x},\mathbf{y},\boldsymbol{\alpha}) = \frac{1}{2}E_{\mathbf{x},\mathbf{y}}\left\|\mathbf{y} - \mathrm{NN}(\mathbf{x};\boldsymbol{\alpha})\right\|^2 + \lambda\boldsymbol{\alpha}\,\boldsymbol{\alpha}^\intercal + \mathrm{const}\,. \tag{A.8}$$

This again equates to an MSE with a weight decay penalty.

A.3 Deep Generative Models

Unlike conventional feedforward neural networks, generative models are designed to output a probability distribution that is used for generating novel examples resembling those in the training set. Depending on the task, these models may be used for either supervised or unsupervised learning.

A.3.1 Variational Autoencoder

The variational autoencoder (VAE) [Rezende et al., 2014, Kingma and Welling, 2013] uses learned approximate inference and can be trained using standard gradient-based methods. The goal of this network is to generate a sample from a distribution $p(\mathbf{x})$ of unlabeled data $\mathcal{D} \triangleq \{(\mathbf{x}_i)\}_1^N$ by sampling a latent variable $\mathbf{z}$ from a prior

distribution $p(\mathbf{z})$ and passing it through a decoder networks associated with the conditional distribution $p_{\boldsymbol{\alpha}}(\mathbf{x}|\mathbf{z})$. However, at training time $\mathbf{z}$ is generated from the distribution $q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})$ associated with the encoder network.

Crucially, by introducing a stochastic latent variable $\mathbf{z}$, the data distribution may be modeled as

$$p_{\boldsymbol{\alpha}}(\mathbf{x}) = \int p_{\boldsymbol{\alpha}}(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}, \tag{A.9}$$

where $p_{\boldsymbol{\alpha}}(\mathbf{x}|\mathbf{z})$ is the likelihood of $\mathbf{x}$, $p_{\boldsymbol{\alpha}}(\mathbf{z})$ is the true prior distribution of $\mathbf{z}$, and $\boldsymbol{\alpha}$ are the learnable encoder and decoder parameters. The goal of learning is to maximize the maximum likelihood of the data $p_{\boldsymbol{\alpha}}(\mathbf{x})$. However, instead of maximizing $p_{\boldsymbol{\alpha}}(\mathbf{x})$, it is possible to maximize a tractable evidence lower bound (ELBO):

$$\mathcal{L} = E_{\mathbf{x}}\left(E_{q_{\boldsymbol{\alpha}}(\mathbf{z})}[\log p_{\boldsymbol{\alpha}}(\mathbf{x}|\mathbf{z})] - KL(q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))\right) \leq E_{\mathbf{x}}\left(\log p_{\boldsymbol{\alpha}}(\mathbf{x})\right), \tag{A.10}$$

where $q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})$ is the encoder model

$$q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)). \tag{A.11}$$

generally defined as a Gaussian or some other tractable distribution. Since $q_{\boldsymbol{\alpha}}(\mathbf{z})$ is restricted to be a Gaussian variational approximation, we can use stochastic gradient descent (i.e., using Monte Carlo estimates of the gradient) via the reparameterization trick to optimize the lower bound in Eq. (A.10) w.r.t parameters $\boldsymbol{\alpha}$.

A.3.2 Conditional Variational Autoencoder

The conditional variational autoencoder (CVAE) [Sohn et al., 2015] learns the conditional distribution $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x})$, given a training set of correlated input and output data $\mathcal{D} \triangleq \{(\mathbf{x}_i, \mathbf{y}_i)\}_1^N$. At test time, the prior encoder encodes the input information $\mathbf{x}$ into a distribution $p_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})$ in the latent space. During training, the distribution $p_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})$ was optimized to be simultaneously near multiple encodings of valid outputs $\mathbf{y}$ by way of the KL divergence term in Eq. (A.14).

By introducing a stochastic latent variable $\mathbf{z}$, the conditional distribution may be modeled as

$$p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}) = \int p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}, \mathbf{z})p_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})d\mathbf{x}\mathbf{z}, \tag{A.12}$$

where $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}, \mathbf{z})$ is the likelihood of $\mathbf{y}$, $p_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x})$ is the prior distribution of $\mathbf{z}$ given $\mathbf{x}$, and $\boldsymbol{\alpha}$ are the learnable encoder and decoder parameters. The prior distribution of $\mathbf{z}$ given $\mathbf{x}$ is often chosen to be Gaussian

$$p_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)), \tag{A.13}$$

with the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\sigma}$ being outputs of an encoder with learnable parameters $\boldsymbol{\alpha}$. This allows the sampling of inferred outputs $\mathbf{y}$ from the generative distribution $p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x}, \mathbf{z})$ modeled by a decoder that takes $\mathbf{x}$ and a sampled $\mathbf{z}$ as inputs.

The goal of learning is to maximize the marginal likelihood or evidence of the data defined in Eq. (A.12). However, instead of maximizing $p(\mathbf{y}|\mathbf{x})$, it is possible to maximize a tractable evidence lower bound:

$$\mathcal{L} = E_{\mathbf{x},\mathbf{y}}\left(\mathbb{E}_{q_{\boldsymbol{\alpha}}(\mathbf{z}|G)}[\log p(\mathbf{y}|\mathbf{x}, \mathbf{z})] - KL(q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{y})||p_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{x}))\right) \leq E_{\mathbf{x},\mathbf{y}}\left(\log p_{\boldsymbol{\alpha}}(\mathbf{y}|\mathbf{x})\right), \tag{A.14}$$

where $q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{y})$ is the inference model, with learnable parameters $\boldsymbol{\alpha}$, defined as:

$$q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{y}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}^2)). \tag{A.15}$$

The inference model is an approximation of the intractable true posterior $p(\mathbf{z}|\mathbf{y})$. Similarly to the prior distribution, the inference model consists of an encoder network that outputs inference distribution parameters $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. The resulting objective, as denoted by Eq. (A.14), is based on an expectation with respect to the inference distribution $q_{\boldsymbol{\alpha}}(\mathbf{z}|\mathbf{y})$, which itself is based on the parameters $\boldsymbol{\alpha}$. The latent variables $\mathbf{z}$ allow for modeling multiple modes in the conditional distribution of output variables $\mathbf{y}$, making CVAEs suitable for representing one-to-many mappings such as inverse kinematics.

Bibliography

P.-A. Absil, C. G. Baker, and K. A. Gallivan. Trust-region methods on Riemannian manifolds. *Found. Comput. Math.*, 7(3):303–330, 2007. doi: 10.1007/s10208-005-0179-9.

P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

Abdo Y Alfakih, Amir Khandani, and Henry Wolkowicz. Solving euclidean distance matrix completion problems via semidefinite programming. *Comput. Optim. Appl.*, 12(1-3):13–30, 1999.

Barrett Ames, Jeremy Morgan, and George Konidaris. Ikflow: Generating diverse inverse kinematics solutions. *arXiv preprint arXiv:2111.08933*, 2021.

Hariharan Ananthanarayanan and Raúl Ordóñez. Real-time inverse kinematics of (2n+1) DOF hyper-redundant manipulator arm via a combined numerical and analytical approach. *Mech. Mach. Theory*, 91:209–226, September 2015.

Jorge Angeles, Günter Hommel, and Peter Kovács. *Computational Kinematics*, volume 28. Springer Science & Business Media, 2013.

Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2018.

A. Aristidou, J. Lasenby, Y. Chrysanthou, and A. Shamir. Inverse Kinematics Techniques in Computer Graphics: A Survey. *Comput. Graph. Forum.*, 37(6):35–58, September 2018. ISSN 01677055. doi: 10.1111/cgf.13310.

Andreas Aristidou and Joan Lasenby. FABRIK: A fast, iterative solver for the inverse kinematics problem. *Graph. Models*, 73(5):243–260, September 2011. ISSN 1524-0703.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *ArXiv180601261 Cs Stat*, October 2018a.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018b.

Patrick Beeson and Barrett Ames. Trac-ik: An open-source library for improved solving of generic inverse kinematics. In *IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 928–935, 2015.

Leon Beiner. Singularity avoidance for Scara robots. *Robotics and autonomous systems*, 10(1):63–69, 1992.

Leon Beiner. Singularity avoidance for articulated robots. *Robotics and autonomous systems*, 20(1):39–47, 1997.

Pratik Biswas, Tzu-Chen Lian, Ta-Chung Wang, and Yinyu Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Trans. Sensor Networks (TOSN)*, 2(2):188–220, 2006.

Franco Blanchini, Gianfranco Fenu, Giulia Giordano, and Felice Andrea Pellegrino. Inverse kinematics by means of convex programming: Some developments. In *IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, pages 515–520, August 2015.

Franco Blanchini, Gianfranco Fenu, Giulia Giordano, and Felice Andrea Pellegrino. A convex programming approach to the inverse kinematics problem for manipulators under constraints. *Eur. J. Control*, 33:11–23, January 2017.

Botond Bócsi, Duy Nguyen-Tuong, Lehel Csató, Bernhard Schoelkopf, and Jan Peters. Learning inverse kinematics with structured prediction. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 698–703. IEEE, 2011.

N. Boumal and P.-A. Absil. Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. *Linear Algebra Appl.*, 475:200–239, 2015. doi: 10.1016/j.laa.2015.02.027.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Samuel Burer and Renato D.C. Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical Programming*, 103(3):427–444, December 2004. ISSN 1436-4646.

Samuel R Buss. Introduction to inverse kinematics with Jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19):16, 2004.

Samuel R Buss and Jin-Su Kim. Selectively damped least squares for inverse kinematics. *J. Graphics Tools*, 10: 37–49, 2005.

Karel Čapek. *RUR (Rossum's universal robots): a fantastic melodrama*. Garden City, NY: Doubleday, Page, 1923.

Philippe Cardou, Samuel Bouchard, and Clément Gosselin. Kinematic-sensitivity indices for dimensionally nonhomogeneous Jacobian matrices. *IEEE Transactions on Robotics*, 26(1):166–173, 2010.

Ching-An Cheng, Mustafa Mukadam, Jan Issac, Stan Birchfield, Dieter Fox, Byron Boots, and Nathan Ratliff. Rmpflow: A computational graph for automatic motion policy generation. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 441–457. Springer, 2018.

F-T Cheng, T-H Chen, Y-S Wang, and Y-Y Sun. Obstacle avoidance for redundant manipulators using the compact QP method. In *IEEE International Conference on Robotics and Automation*, pages 262–269, 1993.

Stefano Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997.

David I Chu, Hunter C Brown, and Moody T Chu. On Least Squares Euclidean Distance Matrix Approximation and Completion. Technical report, Department of Mathematics, North Carolina State University, 2003.

Michael AA Cox and Trevor F Cox. Multidimensional scaling. In *Handbook of Data Visualization*, pages 315–347. Springer, 2008.

Hongkai Dai, Gregory Izatt, and Russ Tedrake. Global inverse kinematics via mixed-integer convex optimization. *Int. J. Rob. Res.*, 38(12-13):1420–1441, October 2019.

Jon Dattorro. Convex optimization & Euclidean distance geometry, 2005.

Javier García de Jalón. Twenty-five years of natural coordinates. *Multibody Sys. Dyn.*, 18(1):15–33, Aug. 2007. ISSN 1573-272X. doi: 10.1007/s11044-007-9068-0.

Yichuan Ding, Nathan Krislock, Jiawei Qian, and Henry Wolkowicz. Sensor network localization, Euclidean distance matrix completions, and graph realization. *Optim. Eng.*, 11(1):45–66, February 2010. ISSN 1389-4420, 1573-2924. doi: 10.1007/s11081-008-9072-0.

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. Euclidean Distance Matrices: Essential Theory, Algorithms and Applications. *IEEE Signal Process. Mag.*, 32(6):12–30, November 2015.

Aaron D'Souza, Sethu Vijayakumar, and Stefan Schaal. Learning inverse kinematics. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, volume 1, pages 298–303. IEEE, 2001.

Joseph Duffy. *Analysis of mechanisms and robot manipulators*. Edward Arnold London, 1980.

Kévin Dufour and Wael Suleiman. On integrating manipulability index into inverse kinematics solver. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6967–6972, 2017.

Kenny Erleben and Sheldon Andrews. Solving inverse kinematics using exact Hessian matrices. *Comput. Graph.*, 78:1–11, February 2019.

Hawren Fang and Dianne P. O'Leary. Euclidean distance matrix completion problems. *Optim. Methods Softw.*, 27(4-5):695–717, October 2012. ISSN 1055-6788, 1029-4937. doi: 10.1080/10556788.2011.643888.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Matthew Giamou, Filip Maric, David M. Rosen, Valentin Peretroukhin, Nicholas Roy, Ivan Petrovic, and Jonathan Kelly. Convex iteration for distance-geometric inverse kinematics. *IEEE Robotics and Automation Letters*, 2022. doi: 10.1109/LRA.2022.3141763. URL https://arxiv.org/abs/2109.03374. To Appear.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. pages 1263–1272. PMLR, 2017.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. 30, 2017.

Li Han and Lee Rudolph. Inverse kinematics for a serial chain with joints under distance constraints. 2006.

Yuval Noah Harari. *Sapiens: A brief history of humankind*. Random House, 2014.

Richard S Hartenberg and Jacques Denavit. A kinematic notation for lower pair mechanisms based on matrices. *J. Appl. Mech.*, 77(2):215–221, 1955.

Timothy F. Havel. Distance Geometry: Theory, Algorithms, and Chemical Applications. In *Encyclopedia of Computational Chemistry*, pages 723–742. John Wiley & Sons, Ltd, April 2002. ISBN 978-0-470-84501-1. doi: 10.1002/0470845015.cda018.

Nicholas J Higham. *Functions of matrices: theory and computation*, volume 104. Siam, 2008.

Chi-Kai Ho and Chung-Ta King. Selective inverse kinematics: A novel approach to finding multiple solutions fast for high-dof robotic. *arXiv preprint arXiv:2202.07869*, 2022.

Jeffrey Ichnowski, Yahav Avigal, Vishal Satish, and Ken Goldberg. Deep learning can accelerate grasp-optimized motion planning. *Science Robotics*, 5(48):eabd7710, 2020. doi: 10.1126/scirobotics.abd7710. URL https://www.science.org/doi/abs/10.1126/scirobotics.abd7710.

Brian Ichter, James Harrison, and Marco Pavone. Learning sampling distributions for robot motion planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7087–7094. IEEE, 2018.

Noémie Jaquier, Leonel Rozo, Darwin G Caldwell, and Sylvain Calinon. Geometry-aware manipulability learning, tracking, and transfer. *The International Journal of Robotics Research*, 2020. doi: 10.1177/0278364920946815.

Long Jin, Shuai Li, Hung Manh La, and Xin Luo. Manipulability optimization of redundant manipulators using dynamic neural networks. *IEEE Transactions on Industrial Electronics*, 64(6):4710–4720, 2017.

Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive science*, 16(3):307–354, 1992.

M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. 20(5):2327–2351, January 2010. ISSN 1095-7189.

Arbaaz Khan, Alejandro Ribeiro, Vijay Kumar, and Anthony G Francis. Graph neural networks for motion planning. *arXiv preprint arXiv:2006.06248*, 2020.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. 2014. URL http://arxiv.org/abs/1312.6114.

Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. 2017.

Jakob Kruse, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Benchmarking invertible architectures on inverse problems. *arXiv preprint arXiv:2101.10763*, 2021.

Jean B. Lasserre. Global Optimization with Polynomials and the Problem of Moments. 11(3):796–817, January 2001.

Thibaut Le Naour, Nicolas Courty, and Sylvie Gibet. Kinematics in the metric space. *Comput. Graph.*, 84:13–23, November 2019a. ISSN 0097-8493.

Thibaut Le Naour, Nicolas Courty, and Sylvie Gibet. Kinematics in the metric space. *Comput. Graph.*, 84:13–23, 2019b.

John M Lee. *Introduction to Riemannian manifolds*. Graduate texts in mathematics. Springer, 2018. doi: 10.1007/978-3-319-91755-9.

Teguh Santoso Lembono, Emmanuel Pignat, Julius Jankowski, and Sylvain Calinon. Learning constrained distributions of robot configurations with generative adversarial network. *IEEE Robotics and Automation Letters*, 6(2):4233–4240, 2021.

Ngai-Hang Z Leung and Kim-Chuan Toh. An SDP-based divide-and-conquer algorithm for large-scale noisy anchor-free graph realization. 31(6):4351–4372, 2010.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards Practical Multi-Object Manipulation using Relational Reinforcement Learning. *arXiv:1912.11032 [cs]*, December 2019.

Leo Liberti, Carlile Lavor, and Nelson Maculan. A branch-and-prune algorithm for the molecular distance geometry problem. *Int. Trans. Oper. Res.*, 2008.

Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. Euclidean Distance Geometry and Applications. *SIAM Rev.*, 56(1):3–69, January 2014. ISSN 0036-1445, 1095-7200. doi: 10.1137/120875909.

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

Anthony A Maciejewski and Charles A Klein. The singular value decomposition: Computation and applications to robotics. *The International journal of robotics research*, 8(6):63–79, 1989.

Edwin Mandfield. The diffusion of industrial robots in japan and the united states. *Research Policy*, 18(4):183–192, 1989.

Giacomo Marani, Jinhyun Kim, Junku Yuh, and Wan Kyun Chung. A real-time approach for singularity avoidance in resolved motion rate control of robotic manipulators. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1973–1978, 2002.

Filip Marić, Ivan Jurin, Ivan Marković, Zoran Kalafatić, and Ivan Petrović. Robot arm teleoperation via RGBD sensor palm tracking. In *39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1093–1098. IEEE, 2016.

Filip Marić, Oliver Limoyo, Luka Petrović, Trevor Ablett, Ivan Petrović, and Jonathan Kelly. Fast manipulability maximization using continuous-time trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8258–8264. IEEE, 2019.

Filip Marić, Matthew Giamou, Soroush Khoubyarian, Ivan Petrović, and Jonathan Kelly. Inverse kinematics for serial kinematic chains via sum of squares optimization. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 7101–7107, August 2020.

Filip Maric, Matthew Giamou, Adam W. Hall, Soroush Khoubyarian, Ivan Petrovic, and Jonathan Kelly. Riemannian optimization for distance-geometric inverse kinematics. *IEEE Transactions on Robotics*, 2021. doi: 10.1109/TRO.2021.3123841. URL https://arxiv.org/abs/2108.13720.

Matthew T Mason. Toward robotic manipulation. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1), 2018.

D. Miller. *2015 NASA technology roadmaps: TA4: robotics and autonomous systems*. 2015.

B. Mishra, G. Meyer, and R. Sepulchre. Low-rank optimization for distance matrix completion. In *Proc. IEEE Conf. Decision and Control and Eur. Control Conf.*, pages 4455–4460, Dec 2011. doi: 10.1109/CDC.2011.6160810.

A. Müller. An O(n)-Algorithm for the Higher-Order Kinematics and Inverse Dynamics of Serial Manipulators Using Spatial Representation of Twists. *IEEE Robot. Autom. Lett.*, 6(2):397–404, April 2021. ISSN 2377-3766. doi: 10.1109/LRA.2020.3044028.

Richard M Murray, Zexiang Li, and S Shankar Sastry. *A mathematical introduction to robotic manipulation*. CRC press, 1994.

Yoshihiko Nakamura, Hideo Hanafusa, and Tsuneo Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal of Robotics Research*, 6(2):3–15, 1987.

Luong Trung Nguyen, Junhan Kim, Sangtae Kim, and Byonghyo Shim. Localization of IoT Networks via Low-Rank Matrix Completion. *IEEE Trans. Commun.*, 67(8):5833–5847, August 2019a. ISSN 0090-6778, 1558-0857. doi: 10.1109/TCOMM.2019.2915226.

Luong Trung Nguyen, Junhan Kim, and Byonghyo Shim. Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237, 2019b.

Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Math. Program.*, 96(2): 293–320, May 2003.

Sarosh Patel and Tarek Sobh. Manipulator performance measures-a comprehensive literature survey. *Journal of Intelligent & Robotic Systems*, 77(3-4):547–570, 2015.

Barak A. Pearlmutter. Fast exact multiplication by the Hessian. *Neural Comput.*, 6(1):147–160, January 1994. ISSN 1530-888X.

Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A Riemannian framework for tensor computing. *International Journal of computer vision*, 66(1):41–66, 2006.

Luka Petrović, Juraj Peršić, Marija Seder, and Ivan Marković. Cross-entropy based stochastic optimization of robot trajectories using heteroscedastic continuous-time Gaussian processes. *Robotics and Autonomous Systems*, 133:103618, 2020.

Luka Petrović, Filip Marić, Ivan Marković, Jonathan Kelly, and Ivan Petrović. Trajectory optimization with geometry-aware singularity avoidance for robot motion planning. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pages 1760–1765. IEEE, 2021.

Hoang-Lan Pham, Véronique Perdereau, Bruno Vilhena Adorno, and Philippe Fraisse. Position and orientation control of robot manipulators using dual quaternion feedback. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 658–663, 2010.

J.M. Porta, L. Ros, F. Thomas, and C. Torras. A branch-and-prune solver for distance constraints. *IEEE Trans. Robot.*, 21:176–187, April 2005a.

Josep M Porta, Lluís Ros, and Federico Thomas. Inverse kinematics by distance matrix completion. In *Proc. 12th Int. Workshop Computational Kinematics*. Elsevier, 2005b.

Josep M. Porta, Nicolás Rojas, and Federico Thomas. Distance geometry in active structures. In *Mechatronics for Cultural Heritage and Civil Engineering*, volume 92, pages 115–136. 2018. ISBN 978-3-319-68645-5 978-3-319-68646-2. doi: 10.1007/978-3-319-68646-2_5.

Ahmed Hussain Qureshi, Yinglong Miao, Anthony Simeonov, and Michael C Yip. Motion planning networks: Bridging the gap between learning-based and classical motion planners. *IEEE Transactions on Robotics*, 37(1): 48–66, 2020.

Hailin Ren and Pinhas Ben-Tzvi. Learning inverse kinematics and dynamics of a robotic manipulator using generative adversarial networks. *Robotics and Autonomous Systems*, 124:103386, 2020.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

Leonel Rozo, Noémie Jaquier, Sylvain Calinon, and Darwin G Caldwell. Learning manipulability ellipsoids for task compatibility in robot manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3183–3189, 2017.

J Kenneth Salisbury and John J Craig. Articulated hands: Force control and kinematic issues. *The International journal of Robotics research*, 1(1):4–17, 1982.

Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. In *International Conference on Machine Learning*, pages 9323–9332. PMLR, 2021.

John Schulman, Jonathan Ho, Alex X Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: Science and Systems*, volume 9, pages 1–10, 2013.

John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *Int. J. Rob. Res.*, 2014.

Lorenzo Sciavicco and Bruno Siciliano. Coordinate transformation: A solution algorithm for one class of robots. 16(4):550–559, 1986.

Lorenzo Sciavicco and Bruno Siciliano. *Modelling and Control of Robot Manipulators*. Advanced Textbooks in Control and Signal Processing. Springer, 2012.

Jon M Selig. *Geometric fundamentals of robotics*, volume 128. Springer, 2005.

Bruno Siciliano and Jean-Jacques Slotine. A general framework for managing multiple tasks in highly redundant robotic systems. *Advanced Robotics*, pages 1211–1216, 1991.

Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: Modelling, Planning and Control*. Springer Science & Business Media, 2010.

M. J. Sippl and H. A. Scheraga. Cayley-menger coordinates. *Proc. Natl. Acad. Sci.*, 83(8):2283–2287, April 1986. ISSN 1091-6490.

Anthony Man-Cho So and Yinyu Ye. Theory of semidefinite programming for Sensor Network Localization. *Math. Program.*, 109(2-3):367–384, January 2007.

Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NeurIPS'15, pages 3483–3491, Cambridge, MA, USA, 2015. MIT Press.

Joan Solà, Jeremie Deray, and Dinesh Atchuthan. A micro Lie theory for state estimation in robotics. *ArXiv181201537 Cs*, November 2020.

Mark W Spong, Seth Hutchinson, and Mathukumalli Vidyasagar. *Robot Modeling and Control*. John Wiley & Sons, 2005.

T. Tony Cai. One-sided confidence intervals in discrete distributions. *J. Stat. Plan. Inference*, 131(1):63–88, April 2005.

Vassilios D Tourassis and Marcelo H Ang Jr. Identification and analysis of robot manipulator singularities. *The International Journal of Robotics Research*, 11(3):248–259, 1992.

James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A Python toolbox for optimization on manifolds using automatic differentiation. *J. Mach. Learn. Res.*, 17(1):4755–4759, 2016.

Bart Vandereycken. Low-rank matrix completion by Riemannian optimization—extended version. 23(2):1214–1236, September 2012.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. 2018.

Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural Kinematic Networks for Unsupervised Motion Retargetting. *ArXiv180405653 Cs*, April 2018.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020a. doi: 10.1038/s41592-019-0686-2.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in Python. *Nat. Methods*, 17(3):261–272, 2020b.

Tim von Oehsen, Alexander Fabisch, Shivesh Kumar, and Frank Kirchner. Comparison of Distal Teacher Learning with Numerical and Analytical Methods to Solve Inverse Kinematics for Rigid-Body Mechanisms. *arXiv:2003.00225 [cs]*, February 2020.

Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. Nervenet: Learning structured policy with graph neural networks. In *International Conference on Learning Representations*, 2018.

Ke Wei, Jian-Feng Cai, Tony F. Chan, and Shingyu Leung. Guarantees of Riemannian Optimization for Low Rank Matrix Recovery. *SIAM J. Matrix Anal. Appl.*, 37(3), April 2016.

Julian Whitman, Matthew Travers, and Howie Choset. Learning modular robot control policies. *arXiv preprint arXiv:2105.10049*, 2021.

Bin Xian, Marcio S de Queiroz, D Dawson, and I Walker. Task-space tracking control of robot manipulators via quaternion feedback. *IEEE Transactions on Robotics and Automation*, 20(1):160–167, 2004.

Tarun Yenamandra, Florian Bernard, Jiayi Wang, Franziska Mueller, and Christian Theobalt. Convex Optimisation for Inverse Kinematics. *Proc. Int. Conf. 3D Vision (3DV)*, pages 318–327, September 2019.

Tsuneo Yoshikawa. Manipulability of robotic mechanisms. *The International Journal of Robotics Research*, 4(2): 3–9, 1985.

Yunong Zhang, Dongsheng Guo, Kene Li, and Jun Li. Manipulability-maximizing self-motion planning and control of redundant manipulators with experimental validation. In *IEEE International Conference on Mechatronics and Automation*, pages 1829–1834, 2012.

Yunong Zhang, Xiaogang Yan, Dechao Chen, Dongsheng Guo, and Weibing Li. QP-based refined manipulability-maximizing scheme for coordinated motion planning and control of physically constrained wheeled mobile redundant manipulators. *Nonlinear Dynamics*, 85(1):245–261, 2016.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997.