Adaptive, Constraint-Aware Inverse Kinematics for Efficient Robotic Manipulator Control

Shibao Yang

PhD

University of York

Computer Science

September 2025

Abstract

Robotic arm motion planning and Inverse Kinematics (IK) represent critical components in enabling robots to perform sophisticated tasks in complex and dynamic environments. Traditional robotic control approaches typically rely on extensive manual programming and deterministic solvers, resulting in limited adaptability, high expert dependency, and challenges in real-time responsiveness and robustness.

This thesis addresses these limitations by proposing an integrated adaptive approach that enhances robotic motion planning by improving the interaction between motion planners and IK solvers. The research develops an advanced Dynamically Weighted Damped Least Squares Inverse Kinematics (DLS-IK) method combined with learned motion representations through Probabilistic Movement Primitives (ProMPs). The approach leverages probabilistic modelling and adaptive weighting strategies to dynamically adjust task constraints and improve motion feasibility, effectively handling singularities, joint limitations, and environmental uncertainties.

Three primary contributions are presented. First, the thesis investigates environmental factors influencing robotic motion planning, offering comprehensive analyses of workspace characteristics and comparative studies of prominent robotic arms. Second, it presents an enhanced DLS-IK method that integrates velocity damping control and joint constraint adherence, significantly improving trajectory accuracy and stability near singularities. Third, it introduces an Adaptive Weighted and Regularised Optimisation for Efficient Inverse Kinematics (AWARE-IK) integrated with DLS-IK, enabling robust, adaptive trajectory generation that efficiently manages conflicting kinematic objectives, such as end-effector accuracy, joint-limit avoidance, and singularity regularisation, within a unified multi-objective optimisation framework.

Evaluations across diverse and cluttered environments demonstrate that the proposed adaptive framework significantly outperforms existing methods regarding task generalisation, computational efficiency, and resilience to environmental complexity. By bridging learning-based methods with optimisation-based IK solutions, this thesis contributes substantially towards practical, reliable robotic manipulation for real-world applications.

Contents

List of Tables

List of Figures

Acknowledgements

First and foremost, I would like to express my deepest gratitude to everyone who has supported me during my four years of PhD research.

I am especially grateful to my supervisors, Dr. Pengcheng Liu and Professor Nick Pears. From the beginning of my PhD journey to its completion, the challenges of overcoming research bottlenecks, and finally the intense writing phase, they have always been there to support me. Their advice, guidance, and support were crucial to the progress of my research, and I am sincerely thankful.

Special thanks go to my friends: Tianda Sun, Sittikrai and Zhiyuan Zhang. Their support during the remote phase when I started my PhD was invaluable, helping me quickly acclimatise to the PhD journey and greatly easing the sense of isolation that often accompanies research. Our discussions on various research topics and creative ideas not only provided me with valuable inspiration but also expanded and deepened my understanding of other fields.

I am also grateful to the University of York for providing the resources and environment necessary for this research.

Lastly, I would like to thank my parents, Guorong Yang and Fengming Pan. Throughout my PhD studies, their unwavering support and care, both physically and emotionally, have been invaluable to me, and I am deeply grateful.

Declaration

'I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for a degree or other qualification at this University or elsewhere. All sources are acknowledged as references'.

Peer-reviewed papers:

- Shibao Yang, Pengcheng Liu and Nick Pears. Benchmarking of robot arm motion planning in cluttered environments. In *2023 28th International Conference on Automation and Computing (ICAC)* (pp. 1-6). IEEE. [1]
- Shibao Yang, Pengcheng Liu and Nick Pears. Enhancing Learning from Demonstration with DLS-IK and ProMPs. In *2024 29th International Conference on Automation and Computing (ICAC)* (pp. 1-6). IEEE. [2]
- Shibao Yang, Pengcheng Liu and Nick Pears. Reinforcement Learning-based Adaptive Probabilistic Movement Primitives in Hybrid Scenarios. In *2023 24th Annual Conference Towards Autonomous Robotic Systems (TAROS)* [3]
- Shibao Yang, Pengcheng Liu and Nick Pears. Jacobian-Guided Active Learning for Gaussian Process-Based Inverse Kinematics. Perception and Planning for Mobile Manipulation in Changing Environments (PM2CE) Workshop at IEEE/RSJ IROS 2025 — accepted workshop paper.

Portions of this thesis incorporate material from the aforementioned publications, which have been cited appropriately throughout the text. All sources are acknowledged as references.

Chapter 1

Introduction

1.1 Robotics Motion Planning

In our daily life, a human can easily and fluently grasp an object, push something or fold laundry - such tasks are not easily reproduced by a robot. Robot arm manipulators can realise such actions to either automate them fully or to assist humans. Thus a robot interacts with objects, which requires it to plan and control the hand and arm to execute some actions automatically. Robotic manipulation has been extensively applied to industrial tasks. The Unimate [4] was the very first industrial robot designed by Joseph Engelberger - the 'Father of Robotics', which revolutionised manufacturing worldwide. With robotic manipulation employed in industry, this reduced dependence on human labour and improved production efficiency. The PUMA [5] was a programmable industrial robot arm famous in both laboratory research and industrial assembly.

In traditional programming scenarios, human programmers have to write robot controllers that can respond to any situation the robot may get. The entire task may need to be broken down into tens or hundreds of smaller steps, and each of these steps should be tested for robustness before the robot leaves the factory. If a failure occurs in the field, highly skilled technicians must be dispatched to update the system for the new situation. Therefore, establishing a robotic arm operating system that does not rely too much on expert programming will help reduce industrial production costs. It can also help to apply complex robotic arms to people's daily lives and improve living standards. The overall motion planning pipeline addressed throughout this thesis, in which a high-level motion planner is coupled to an inverse kinematics solver, is summarised in the process flowchart of Fig. 1.1. When planning these motions the arm is essentially faced with two canonical scenarios, an obstacle-free scene, where it can follow a direct, kinematically optimal path to the goal (Fig. 1.2), and an obstacle-laden scene, where non-penetration constraints, clearance margins, and potential detours or speed changes must be

respected (Fig. 1.3). These settings respectively let the planner emphasise pure efficiency or compel it to integrate collision-avoidance logic into every step of the trajectory. Both scenes are simulated in RViz, in which the gripper goal can be repositioned interactively by dragging its marker or by scripting the pose programmatically, and various obstacles like squares, spheres, cones, or custom meshes, those can be added directly to the scene to test avoidance behaviour.

Motion planning [6] is the path planning in robot arm manipulation that breaks down the desired movement task into discrete motions. These satisfy movement constraints and possibly optimise some aspects of the movement. Popular motion planning approaches like sampling-based approaches [7] and optimisation-based approaches [8] have great success in motion planning problems. They mainly utilise tree searching, roadmap creation and cost functions to find the trajectory. However, such approaches have limitations in relying on tree construction and hand coding cost functions for motion feature learning, which is unsuitable for reducing reliance on expert programming. Learning from Demonstration (LfD) approaches, in turn, are better than those approaches. These directly learn from human demonstration and adapt to the new environment [9].

LfD [10–12] is a machine learning technique in robotics that enables robots to acquire skills by observing human demonstrations. The aim of LfD is to enable robots to perform tasks in a similar manner to how a human would perform them, by drawing on the knowledge and expertise of human operators. LfD has emerged as an important area of research in robotics due to its potential to simplify the task of programming robots and increase their ability to perform tasks in unstructured environments. The basic idea behind LfD is to use human demonstrations as input to a machine learning algorithm, which then generates a model of the task that can be used to control the robot. This approach has been found to be effective in incorporating a priori knowledge through demonstrations, relying on humans to show the robot possible solutions to a given task from which it can then generalise.

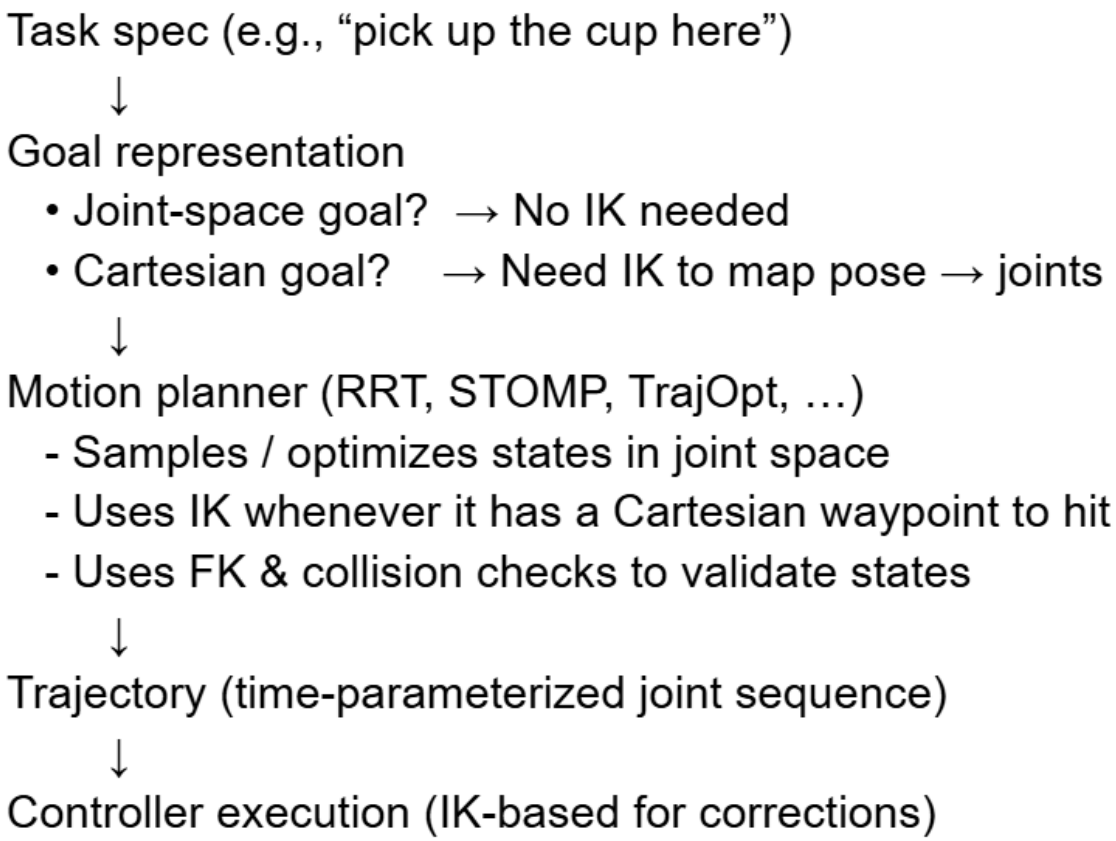
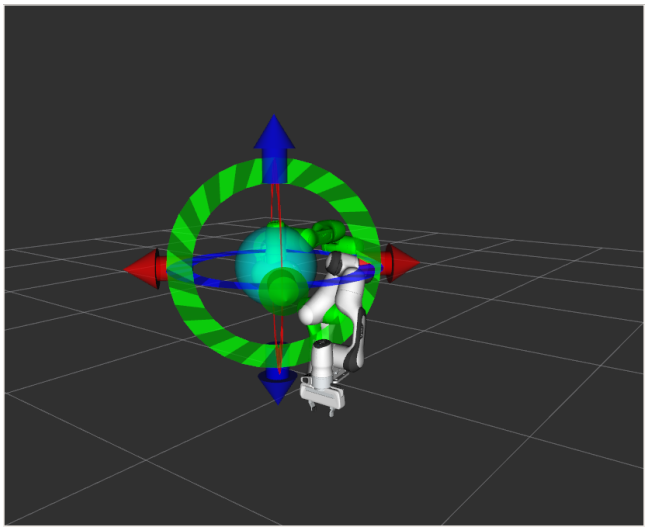
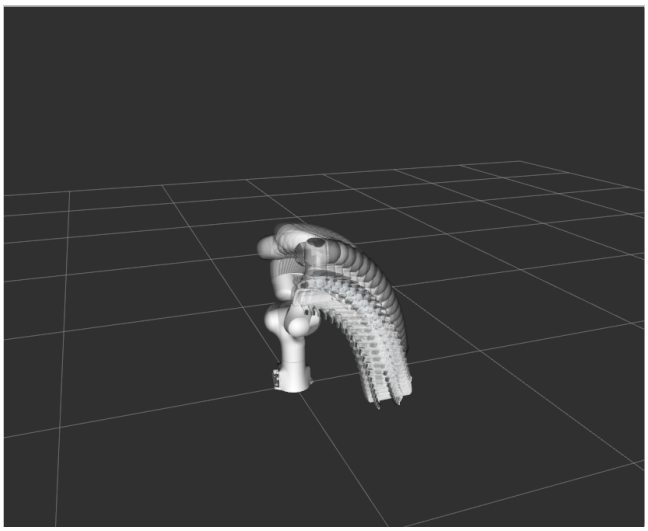


Figure 1.1: Motion Planning Process

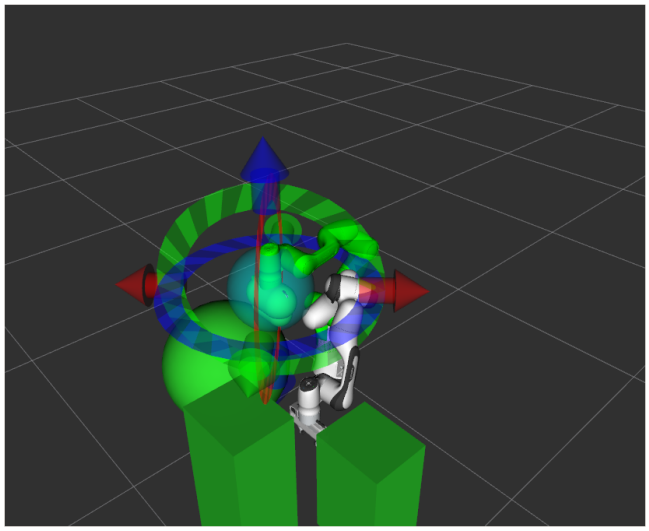


(a) General process

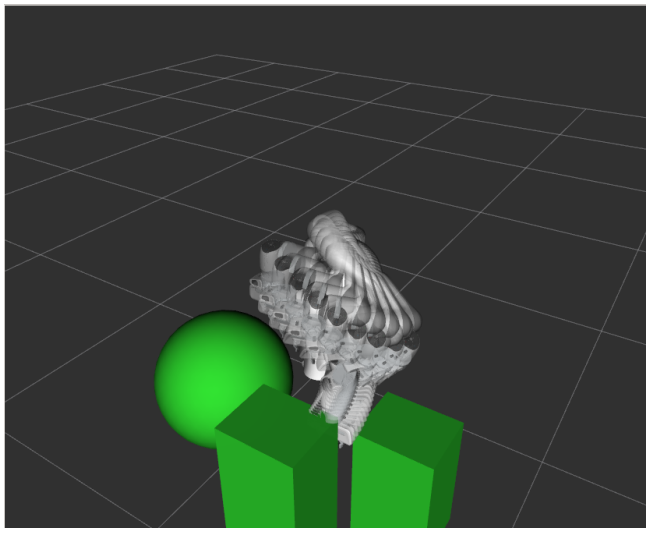


(b) Process with trail

Figure 1.2: Simulation in an obstacle-free environment: a shows the robot arm moving from the marker to the gripper position, while b visualises the full trail from start to end.



(a) General process



(b) Process with trail

Figure 1.3: Simulation in an obstacle environment: a shows the robot arm moving from the marker to the gripper position, while b shows the arm's full, trail-visualised path.

Inverse Kinematics (IK) computes the joint variables that place a robot's end-effector at a desired pose (position and orientation). It inverts Forward Kinematics (FK) and is often nonlinear, yielding multiple or no solutions depending on the manipulator's geometry and constraints. When the structure is simple (e.g., 6-DOF arms with a spherical wrist), closed-form analytic solutions can be derived. For higher-DOF or more complex robots, IK is typically solved numerically using the Jacobian and iterative schemes (e.g., pseudoinverse or damped least squares methods). As the flowchart in Fig. 1.1 shows, once a task is specified, a joint-space goal bypasses IK, whereas a Cartesian goal triggers IK to map pose to joints. Motion planners still work in joint space but call IK whenever they satisfy a Cartesian waypoint, then validate candidates with FK and collision checks. Throughout this thesis we deliberately distinguish a waypoint, an

Table 1.1: Summary of Traditional IK solvers and typical limitations.

Solver	**Core Function**	**Limitations**
Closed-form (analytic)	Exploit kinematic structure to derive explicit joint solutions (e.g., 6-DOF with spherical wrist).	Applies only to specific architectures; brittle to slight kinematic changes; multiple branches require careful selection and can cause discontinuities; poor handling of joint limits/secondary tasks; no inherent collision/constraint awareness.
Numerical (Jacobian pseudoinverse / transpose)	Iterative update using Jacobian to drive pose error to zero.	Convergence depends on initialisation and step sizes; susceptible to local minima and oscillations; ill-conditioned near singularities; no built-in constraints (joint limits, velocity/acceleration bounds, collision) or task prioritisation.
DLS-IK	Adds isotropic damping to stabilise near singularities.	Requires damping tuning (accuracy–stability trade-off); residual pose error near singularities; still unconstrained (no collision/joint-limit guarantees); slower convergence if over-damped.
SDLS-IK	Applies damping selectively along near-singular directions (via SVD).	Extra computation (SVD each iteration); parameter sensitivity; non-smooth behaviour across singular value thresholds; still lacks explicit constraints or global optimality.
TRAC-IK	Heuristic/optimisation-backed IK with bounds checking and random restarts to improve success rate and speed.	Seed and timeout dependent; no collision awareness by default; no global optimality guarantees; may violate secondary objectives; performance can degrade near singularities or tight tolerances.
Bio-IK	Multi-objective, evolutionary optimisation (pose, joint limits, collision proxies, etc.).	Stochastic and parameter-heavy; unpredictable runtimes; can yield small residual errors; scaling to high-DOF or tight real-time budgets is challenging.
Relaxed-IK	Optimisation-based IK that "relaxes" exact pose to balance smoothness, limits, collision, and orientation/position.	Intended residual error by design; careful weight tuning needed; may be slower than closed-form/Jacobian methods; no strict guarantees on global optimality.

intermediate end-effector pose that the planner's path is required to pass through as a hard geometric constraint, from a via-point, the term adopted in the movement primitive literature for an intermediate pose on which a learned trajectory distribution is probabilistically conditioned. The two coincide geometrically, in that both denote an intermediate pose the trajectory should attain, but they differ in how they are enforced: a waypoint acts as a hard path constraint imposed on the planner, whereas a via-point is imposed through Bayesian conditioning of the primitive distribution, so that the constraint is satisfied only in expectation and subject to the learned covariance. After a time-parameterised joint trajectory is produced, controllers often keep using Jacobian-based IK (pseudoinverse or damped least squares) to correct tracking errors online.

In robotic motion planning, the IK solver works with the motion planner. The IK solver maps a desired end-effector pose from task-space to one or more valid joint-space configurations. The motion planner then searches this joint-space for a collision-free and dynamically feasible path to

a goal configuration.

For any Cartesian goal, the planner relies on the IK solver to provide candidate joint states that seed its search. The planner evaluates these states and can, in turn, constrain the IK solver with contextual information like joint limits or redundancy preferences. This cycle ensures that local solutions from the IK solver align with the global trajectory being planned. Upon completion, the final trajectory is executed by a controller, which may also use differential IK for real-time adjustments. The IK solver thus performs the local pose-to-configuration mapping, the planner handles the global search, and together they translate abstract tasks into executable motion.

Programming robots is difficult. There are a considerable number of possible tasks, often there are unique environmental demands, often tasks are difficult to describe formally, and typically the use of an expert's engineering skill is impractical. Conventional robotic arm control pipelines exhibit a fundamental inflexibility when transitioning between manipulation tasks involving different objects. This limitation arises because traditional controllers are typically hard coding for specific object geometries, predefined grasp configurations, and fixed motion sequences, none of which generalise to novel objects without explicit reprogramming. Consequently, adapting a robot arm from one task to another requires an expert engineer to manually redefine waypoints, collision geometries, and force profiles, a process that is both time-consuming and labour-intensive. This dependence on task-specific expert intervention represents a significant barrier to deploying robotic manipulators in unstructured or variable environments, and directly motivates the adoption of Learning from Demonstration as a more adaptive and expressive programming paradigm. LfD provides a more natural and expressive way of programming than the previous methods. Most importantly, it does not require expert knowledge of domain dynamics, which depends heavily on the accuracy of the world model. Moreover, the robot can receive valuable human intuition from manipulating complex tasks. LfD still has some limitations that we should focus on.

A core advantage of LfD is that it is designed to be accessible to demonstrators across a wide technical backgrounds, ranging from robotics researchers to domain specialists with no robotics training at all, such as factory operators, clinicians, or domestic users. By shifting the skill requirement from explicit robot programming to task-level competence, LfD substantially broadens the pool of users capable of deploying and adapting robotic systems for their own applications. This accessibility is, however, accompanied by a practical limitation that the quality of the resulting policy is strongly coupled to the quality of the demonstrations, and non-expert demonstrators may inadvertently provide trajectories that violate joint limits, approach kinematic singularities, or are otherwise mechanically suboptimal for the target manipulator. End users, who are frequently novices, additionally expect the robot to acquire the desired behaviour from as few demonstrations as possible. These considerations motivate the constraint-aware LfD framework developed in Chapter 4, in which the integration of Damped Least Squares inverse kinematics with Probabilistic Movement Primitives explicitly compensates for mechanical infeasibilities that may arise from non-expert demonstration data, keeping the method accessible without loss of execution

quality.

While recent advances in adaptation technology have addressed certain challenges in robotic trajectory planning, significant limitations remain. The unified probabilistic framework proposed by [13,14] attempts to integrate obstacle avoidance, via-points, and mutual avoidance capabilities. The approach is theoretically sound, but it introduces computational overhead that can hinder real-time performance in complex scenarios [15, 16]. As [15] demonstrates, its effectiveness also drops in rapidly changing environments and when several constraints conflict at once.

The versatility and flexibility claimed for current adaptation techniques warrant closer scrutiny [17]. Methods that show promise in controlled settings [18] often degrade substantially in genuinely dynamic environments, where multiple uncertainties interact [19]. The promise of seamless trajectory modification frequently comes at the cost of computational efficiency or stability guarantees [20]. Approaches that incorporate advanced constraints such as via-points and mutual avoidance often rely on simplifying assumptions that may not hold in real-world applications [21, 22]. And although these techniques scale to high-dimensional systems in theory, practical implementations encounter numerical stability issues and computational bottlenecks that limit their use in complex robotic systems [23, 24].

A critical examination of the current literature on LfD and its integration with motion planning and inverse kinematics reveals three specific gaps that this thesis directly addresses. First, existing benchmarking studies evaluate motion planners in narrow experimental settings, typically focusing on a single robotic platform or a limited set of planners, and consequently offer little guidance on how planner performance scales with environmental complexity across different manipulator architectures. Second, standard LfD frameworks such as Probabilistic Movement Primitives (ProMPs) and Dynamic Movement Primitives (DMPs) learn motion distributions from demonstrations but do not inherently enforce kinematic constraints, with the result that demonstrated trajectories may violate joint limits or drive the manipulator toward singular configurations when reproduced on hardware. Third, existing inverse kinematics solvers typically treat singularity management, joint-limit avoidance, and task-space accuracy as separate concerns with statically weighted objectives, rather than as coupled requirements that must be balanced dynamically in response to the manipulator's instantaneous kinematic state.

These gaps are compounded by three structural weaknesses in the surrounding literature. First, existing surveys of LfD compartmentalise its methodological categories typically treating dynamical-system formulations such as DMPs, probabilistic formulations such as ProMPs and GMMs, and neural network formulations such as CNMPs in isolation [17, 25] without adequately analysing the interconnections, trade-offs, and conflicts that arise when these paradigms are combined with classical motion planners or constraint-aware inverse kinematics solvers. Second, the field's tendency to emphasise methodological novelty over deployment has produced algorithms whose validation relies heavily on simulation that typical LfD evaluation protocols use physics engines

such as Gazebo, MuJoCo, or PyBullet, or kinematic simulators coupled to MoveIt, to reproduce learned trajectories under idealised sensing, actuation, and contact models. Such simulations rarely capture the sensor noise, compliance, joint friction, and calibration errors that govern real hardware, and consequently reported successes often do not transfer to physical deployment without substantial additional engineering. Third, although the LfD literature is extensive when considered as a methodological whole, there is comparatively little work that provides an integrative overview of movement primitive adaptation, that is, the mechanisms by which learned primitives are modified online to accommodate new goals, via-points, obstacles, or kinematic constraints, despite the fact that adaptation is the operative mechanism that determines whether a learned skill generalises beyond its training conditions.

The persistence of these gaps can in turn be traced to a recurring pattern of methodological and validation shortcomings across the literature, which we summarise here as a diagnostic reference. Specifically, four interrelated symptoms of the broader theory–practice disconnect recur across recent LfD, motion planning, and IK studies:

- **Insufficient empirical breadth:** Few studies provide comprehensive evaluations of system reliability, computational efficiency, and robustness under varying environmental conditions and across multiple manipulator platforms [26, 27].
- **Idealised modelling assumptions:** Many theoretical frameworks assume accurate sensing, ideal actuation, and deterministic environments, assumptions that rarely hold in real-world deployments on physical hardware [28].
- **Theory–practice disconnect:** The mismatch between the environments in which algorithms are developed and evaluated and those in which they are ultimately deployed limits the credibility of reported empirical results [26–28].
- **Barriers to industrial adoption:** Taken together, these shortcomings create a significant barrier to the widespread industrial uptake of learning-based and optimisation-based manipulation technologies, even when individual components demonstrate strong performance in isolation [24].

These recurring symptoms motivate the methodological stance adopted throughout this thesis: each contribution is evaluated across multiple robot platforms, environmental complexities, or noise conditions, and the AWARE-IK solver is additionally validated on physical hardware, so that the claims advanced in the following chapters rest on evidence that addresses rather than reproduces the shortcomings identified above.

1.2 Motivation and Research Problem

Robotic arm motion planning in complex and cluttered environments remains a fundamental challenge in deploying robots for real-world applications. Efficient and feasible motion generation is critical for ensuring task success, especially in dynamic settings where robots must avoid obstacles, maintain kinematic feasibility, and adhere to task-specific constraints. A comprehensive benchmarking study [1] of widely used planning algorithms including RRT, PRM, STOMP, and TrajOpt highlights both the strengths and limitations of existing methods.

Key performance metrics, such as computation time, success rate, collision-avoidance capability, and trajectory smoothness, indicate that while modern planners perform well in simple environments, they frequently encounter difficulties along two distinct and largely independent axes of problem difficulty. The first is geometric complexity, arising from cluttered workspaces, narrow passages, and tight task-space tolerances, which increases the cost and frequency of collision queries and shrinks the volume of feasible configurations. The second is the dimensionality of the manipulator's configuration space, which for serial arms coincides with the number of degrees of freedom and ranges from six for standard industrial arms to seven or more for redundant manipulators such as the Franka and KUKA used in this thesis. Higher configuration-space dimensionality amplifies planning cost, increases the density of kinematic singularities and local minima, and expands the null-space that redundancy-aware solvers must exploit or manage. Throughout this thesis, unless otherwise stated, the term "high-dimensional" refers specifically to configuration-space dimensionality in this sense.

As illustrated in Fig. 1.1, the effectiveness of robotic arm motion planning depends on the interaction between the motion planner and the IK solver. The motion planner optimises states primarily in joint space, but Cartesian goals and waypoints necessitate IK solutions to convert target poses into joint configurations. Consequently, the quality and efficiency of the IK solver directly affect the motion planner's ability to validate feasible states through FK and collision checking. Improving how these two components work together is therefore important.

Moreover, a persistent challenge in robotic manipulation is handling kinematic singularities configurations where the robot loses degrees of freedom, complicating trajectory tracking and obstacle avoidance. Traditional IK solvers often struggle under dynamic conditions, particularly when confronted with joint limits, environmental uncertainty, and sensor noise. This research focuses on two interrelated aspects: improving the coordination between motion planners and IK solvers, and advancing the accuracy, robustness, and computational efficiency of IK methods. Progress on both improves the reliability and overall performance of robotic arm systems in practical scenarios.

In robotic motion planning, IK is the link between high-level task specifications and low-level joint configurations. While planners often reason about motion in Cartesian space, such as specifying

that the robot's end-effector must reach a certain position and orientation, they must ultimately convert these goals into joint-level commands [29]. IK provides this translation. Moreover, in redundant manipulators where multiple joint configurations can achieve the same end-effector pose, planners rely on IK not just once, but repeatedly, to evaluate and select solutions that maintain smoothness, avoid collisions, and respect joint limits [30]. IK also enables constraint projection, allowing planners to project sampled or optimised joint states back onto feasible Cartesian manifolds, such as ensuring the end-effector remains on a desired surface or plane. Without efficient and robust IK, motion planning would struggle to maintain task feasibility, particularly in high-dimensional, constrained environments. Thus, enhancing IK solvers is not merely an optimisation, but a necessity for enabling practical and reliable planning solutions in complex robotic applications.

One response to these challenges has been a growing interest in incorporating learning-based methods into motion planning frameworks. Rather than relying solely on deterministic planners or hand-tuned IK solvers, recent approaches explore data-driven strategies to improve generalisation and adaptability. A prominent class of such methods is built on movement primitives, which are parameterised, low-dimensional representations of elementary motion patterns that can be acquired from human demonstration and subsequently invoked to generate trajectories at execution time. Movement primitives are described as modular because individual primitives encode self-contained motion units that can be sequenced in time or superposed in parallel to construct more complex behaviours, and as reusable because a single learned primitive is parameterised in a way that permits generalisation to new start states, goal states, temporal scalings, and via-points without retraining. A formal treatment of the principal movement primitive formulations, including DMPs, Gaussian Mixture Models (GMMs), and ProMPs, is provided in Section 2.2.3. When movement primitives are combined with probabilistic learning, as in the ProMP framework, robots can learn distributions over motion patterns from past data or demonstrations and adapt them to new contexts through Bayesian conditioning on observed states or desired via-points.

In this thesis, we propose an integrated approach that enhances IK solvers through Dynamically Weighted Damped Least Squares Inverse Kinematics (DLS-IK) in combination with learned motion representations. The goal is to develop a framework that retains the reliability of classical IK methods while benefiting from the adaptability and generalisation afforded by learning-based motion planning. By dynamically adjusting weights based on task constraints and predicted motion feasibility, the proposed method enables the robot to operate robustly in environments with obstacles, tight clearances, and variable goals.

We focus on three core research objectives:

- Investigate the effect of environmental complexity on the adaptability and motion feasibility of robotic arms, particularly in cluttered settings where planning becomes nontrivial.

- Develop enhanced strategies for adaptive motion planning, by integrating learned motion priors with optimisation-based IK to improve real-time performance and constraint handling.
- Formulate robust mechanisms for generalisation and flexibility, ensuring the system performs consistently across diverse tasks, configurations, and workspace constraints.

The three contributions of this thesis, formally stated in Section 1.3 below, are organised as a coherent progression. The first contribution establishes a comprehensive benchmarking framework that systematically characterises the interaction among twelve motion planners, three robotic arms, and three levels of environmental clutter, thereby providing the empirical foundation on which subsequent methodological choices rest. Building on these insights, the second contribution integrates DLS-IK with ProMPs to produce kinematically feasible, mechanically compliant trajectories from human demonstrations, closing a known gap between demonstration-based learning and constraint satisfaction. The third contribution develops and validates AWARE-IK, an Adaptive Weighted and Regularised Optimisation framework for efficient inverse kinematics that dynamically balances task-space accuracy, joint-limit avoidance, and singularity management through configuration-aware weighting. Taken together, these three contributions, referenced hereafter as Contributions 1, 2, and 3 respectively, produce a framework that demonstrates improved task generalisation, computational efficiency, and resilience to workspace complexity, offering a practical solution for real-world robotic manipulation challenges.

1.3 Contributions of the Research

This thesis presents three principal contributions to the field of robotic motion planning and inverse kinematics, each corresponding to a specific chapter that addresses critical challenges in robotic manipulation.

Contribution 1: Comprehensive Benchmarking Framework for Motion Planning in Cluttered Environments (Chapter 3)

This contribution establishes a systematic evaluation methodology for motion planning algorithms across varying environmental complexities and robotic platforms. The specific advancements include:

- A systematic characterisation of how workspace dimensions and obstacle height govern planning time and success rate across three manipulator architectures, establishing how planner performance scales with environmental complexity rather than reporting performance only at a single fixed configuration.

- A three-axis evaluation framework spanning time efficiency, success rate, and range-parameter sensitivity that exposes performance differences concealed by conventional fixed-parameter benchmarks and thereby distinguishes planners that are robust to parameter variation from those that are not.
- A weighted cost function for scenario-aware planner selection that converts the raw benchmark data into concrete, task-specific planner arm recommendations, rather than reporting metrics in isolation.
- An empirical comparison of the Franka, UR5, and KUKA manipulators in cluttered environments that identifies the best-performing planner arm combinations for efficiency and robustness, yielding deployment guidance not available from single platform studies.

Contribution 2: Integration of Damped Least Squares Inverse Kinematics with Probabilistic Movement Primitives for Learning from Demonstration (Chapter 4)

This contribution addresses the fundamental challenge of transferring human demonstrations to mechanically compliant robot trajectories through a novel integration of DLS-IK with ProMPs. The key innovations comprise:

- Proposed an advanced DLS-IK method integrating velocity damping control and joint constraint adherence via the robotic arm's null space principle, significantly improving control accuracy and mechanical compliance.
- Integrated the refined DLS-IK approach within the ProMPs framework, substantially enhancing the robot's capability to learn and reproduce mechanically compliant and efficient trajectories while effectively managing singularity challenges and joint limitations.
- Conducted an in-depth analysis of the influence of the Jacobian condition number on DLS-IK stability, particularly near singular configurations, demonstrating enhanced stability and robustness through localised optimal parameter selection.

Contribution 3: Adaptive Weighted and Regularised Optimisation for Efficient Inverse Kinematics (AWARE-IK) (Chapter 5)

This contribution presents an adaptive framework that dynamically balances multiple objectives in inverse kinematics through configuration-aware optimisation. The methodology addresses limitations of existing approaches through the following technical developments:

- We propose a novel method that combines DLS-IK with adaptive QP weights to generate kinematically feasible trajectories that avoid singularities and respect joint limits, enhancing adaptability and precision.

- Through the development and validation of an adaptive weighting mechanism, the proposed method effectively balances accuracy and smoothness in multi-objective scenarios. By dynamically adjusting the relative priority realised as the per-objective weighting within a single cost function assigned to each competing kinematic objective, namely task-space pose accuracy, joint-limit avoidance, neutral posture regulation, and singularity damping, and by ensuring smooth transitions between them, the approach addresses limitations of traditional methods like DLS-IK and Selective Damped Least Squares IK (SDLS-IK), enabling more stable and precise performance in complex robotic tasks.
- We evaluate the proposed AWARE-IK using comprehensive metrics for kinematic accuracy, efficiency, and robustness to noise, demonstrating its capability to avoid singularities, adhere to constraints, and ensure smooth trajectory execution through adaptive weighting.

These contributions collectively advance the theoretical understanding and practical implementation of motion planning and IK for robotic manipulators, providing validated methodologies for deployment in complex environments requiring high precision, mechanical compliance, and computational efficiency.

1.4 Structure of the Thesis

The thesis is structured as follows:

1. **Chapter 1:** Introduction establishes the research context by identifying critical challenges in robotic motion planning and inverse kinematics. This chapter articulates the research motivation, defines the scope of investigation, and presents the primary objectives.
2. **Chapter 2:** Background and Related Work covers the theoretical foundations and existing methodologies in motion planning and inverse kinematics. This chapter reviews established approaches including sampling-based motion planners, numerical inverse kinematics methods, and learning from demonstration techniques.
3. **Chapter 3:** Benchmarking of Robot Arm Motion Planning presents a systematic evaluation framework for motion planning algorithms in cluttered environments. This chapter introduces a comprehensive benchmarking methodology using the MotionBenchMaker tool to assess twelve popular motion planners across three robotic platforms under varying environmental complexities.
4. **Chapter 4:** Enhancing LfD with DLS-IK and ProMPs addresses the challenge of transferring human demonstrations into mechanically compliant robot trajectories. This chapter

develops an integrated framework combining DLS-IK with ProMPs to ensure kinematic feasibility while preserving the intent of human demonstrations.

5. **Chapter 5:** AWARE-IK: Adaptive Weighted and Regularised Optimisation for Efficient Inverse Kinematics introduces an advanced inverse kinematics solver that dynamically adapts to varying task requirements and kinematic configurations. This chapter presents an optimisation framework that combines adaptive weighting mechanisms with hybrid solving strategies to balance multiple objectives including accuracy, joint limit avoidance, and singularity management.

6. **Chapter 6:** Conclusions and Future Work synthesises the research contributions and their implications for robotic manipulation systems. This chapter consolidates the key findings from the benchmarking studies, the enhanced learning from demonstration framework, and the adaptive inverse kinematics solver.

Chapter 2

Literature Review

2.1 Introduction

We present a systematic review of motion primitive adaptation methodologies within LfD for robotic manipulators, emphasising how these primitives can be modified and generalised across different scenarios. We cover the fundamental approaches, including DMPs, ProMPs, GMMs and Kernelised Movement Primitives (KMPs) with an analysis of both their theoretical foundations and their adaptive capabilities. Adaptation techniques are divided into distinct categories: trajectory modification through intermediate points, real-time obstacle avoidance, joint limit compliance, dynamic impedance control, and tactile skill adaptation. For each category, we evaluate the underlying mathematical frameworks, implementation challenges, and practical applications in industrial and service robotics. Through comparative analysis, we highlight the strengths and limitations of different adaptation strategies, particularly focusing on their robustness in uncertain environments, computational efficiency, and scalability. We also explore emerging applications in human-robot collaboration and multi-robot systems, where adaptive motion primitives are central to safe and efficient interaction.

The remainder of this chapter is organised to mirror the three contributions developed in this thesis. Section 2.2 surveys the motion planning literature, beginning with sampling-based planners (Section 2.2.1), proceeding to existing benchmarking studies and their methodological limitations (Section 2.2.3), and concluding with LfD as an alternative paradigm together with its principal movement-primitive formulations (Section 2.2.4); this section provides the methodological foundation for the benchmarking framework presented in Chapter 3 and motivates the integration of demonstration-based learning with constraint-aware control in Chapter 4. Section 2.3 then turns to IK, summarising FK conventions before reviewing the principal classes of IK solvers analytical, Jacobian-based, damped least squares, selective damping, optimisation-based (TRAC-IK, Bio-IK,

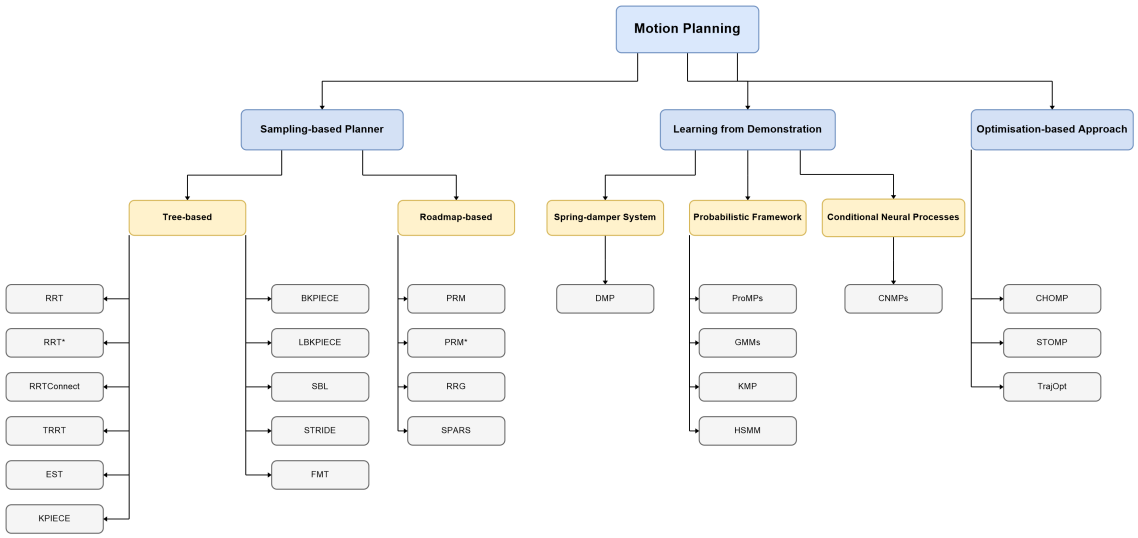


Figure 2.1: Hierarchical taxonomy of motion-planning methods, organised into three top-level paradigms and refined into their constituent method families and individual algorithms.

Relaxed-IK), and learning-based, and analysing their respective strengths and limitations; this review establishes the technical baselines against which the AWARE-IK framework of Chapter 5 is positioned. Section 2.4 synthesises the gaps identified across these studies and articulates how the three thesis contributions address them.

2.2 Motion Planning

Fig. 2.1 organises the motion planning methods into three top-level paradigms, with arrows tracing the refinement of each paradigm from the general planning category to the specific planners surveyed below. The Sampling-based Planner paradigm comprises tree-based planners (RRT, RRT*, RRTConnect, TRRT, EST, KPIECE, BKPIECE, LBKPIECE, SBL, STRIDE, and FMT) and roadmap-based planners (PRM, PRM*, RRG, and SPARS). The LfD paradigm is subdivided into spring damper systems (DMP), probabilistic frameworks (ProMPs, GMMs, KMP, and HSMM), and conditional neural processes (CNMPs). The Optimisation-based Approach paradigm comprises trajectory-optimisation methods (CHOMP, STOMP, and TrajOpt).

Sampling-based motion planners are conventionally divided into two categories according to how their search structure is reused. Single-query planners construct a search tree or graph dedicated to a specific start–goal pair and discard it once the query is answered, which makes them well suited to problems in which the environment or task changes between queries. Multi-query planners, by contrast, invest greater preprocessing effort to construct a reusable roadmap of the free configuration space, which can then be queried many times at low marginal cost; this trade-

off favours static environments in which planning requests are repeated. The planners surveyed below are predominantly single-query, with PRM-based methods (Section 2.2.1) representing the multi-query alternative.

The RRT [31] planner uses randomised sampling to grow a tree in configuration space from the start state, repeatedly extending the tree by a fixed step size toward newly drawn samples until a node within the goal region is reached. RRTConnect [31] extends this idea bidirectionally: two trees are grown simultaneously, one rooted at the start configuration and the other at the goal, and at each iteration the planner samples a new point, extends one tree toward it, and then attempts to "connect" the other tree to the newly added node by repeatedly stepping toward it until either the node is reached (in which case a feasible path is recovered by concatenating the two tree branches) or an obstacle is encountered (in which case the iteration terminates and the trees swap roles). This greedy connect heuristic is the source of RRTConnect's substantial speed advantage over plain RRT in most practical settings. RRTstar [31] further modifies the basic RRT by introducing cost-based rewiring: when a new node is added, neighbouring nodes are re-evaluated and reconnected through the new node whenever doing so reduces their accumulated path cost, yielding asymptotically optimal paths at the price of additional bookkeeping. TRRT [32] augments RRT with adaptive sampling that biases tree growth based on a temperature-controlled cost criterion, while EST [33] grows a tree by preferentially expanding from regions of low local sample density to maintain coverage of unexplored space.

SBL [34], KPIECE [35], BKPIECE [35], and LBKPIECE [35] are further single-query planners that exploit projections of the configuration space onto lower-dimensional grids or cells: SBL grows two trees and tests connection lazily, while the KPIECE family discretises a low-dimensional projection to bias exploration toward under-explored cells, with BKPIECE and LBKPIECE introducing bidirectional and lazy variants of the same idea. STRIDE [36] uses a tree-based search informed by a Generalised Binary Search Tree to handle high-dimensional and dynamic environments, and FMT (Fast Marching Tree) [37] is a single-query asymptotically optimal planner that processes a fixed batch of samples in cost-to-come order using a dynamic-programming recursion, providing the convergence guarantees of optimal sampling-based planners with reduced computational overhead per sample.

2.2.1 Sampling-based planner

Sampling-based motion planning constructs paths by probing the robot's continuous configuration space with random samples and attempting short, local connections between nearby collision-free states, thereby avoiding the need for a fixed discretisation that would otherwise scale exponentially with the dimension of the configuration space. As shown in Fig. 2.1, sampling-based planners are conventionally divided into two families. Tree-based planners (e.g., RRT, RRTstar, RRTConnect,

EST, KPIECE) grow a search tree from the start configuration and, in bidirectional variants, also from the goal, and are designed primarily for single-query problems. Roadmap-based planners (e.g., PRM, PRM*, RRG, SPARS) instead construct a reusable graph of the free configuration space, amortising substantial preprocessing cost across many subsequent queries in multi-query settings.

The principal theoretical attractions of sampling-based methods are two distinct convergence guarantees that apply to specific planner subsets rather than to the family as a whole. Probabilistic completeness, the property that the probability of finding a feasible path, if one exists, approaches one as the number of samples grows, is provided by the standard formulations of RRT, RRTConnect, EST, SBL, the KPIECE family, PRM, and FMT. Asymptotic optimality, the stronger property that the cost of the returned path additionally converges to the optimum as samples grow, is provided by the optimal variants RRT, PRM*, RRG, and FMT*, which differ from their non-optimal counterparts in that they perform either rewiring or batch processing of samples in cost-to-come order. In practice, the runtime of all sampling-based methods is dominated by collision checking and nearest neighbour queries, and the choice between tree-based and roadmap-based formulations turns on the trade-off between build cost and query speed in light of whether the application is single-query or multi-query in character. Table 2.1 consolidates the principal sampling-based planners surveyed here, listing for each its tree or roadmap-based category, its defining algorithmic feature, and its characteristic query complexity, so that the completeness and optimality properties discussed above can be read alongside the practical cost of constructing and querying each planner's search structure.

There are two sampling-based approaches, tree-based approaches and roadmap-based approaches. The Rapidly exploring Random Tree, known as RRT [38], is the primary Tree-based motion planner. The planner begins by rooting a tree at the starting configuration of the robot. RRTConnect [40] is a bi-directional version of RRT, where two trees are grown simultaneously from start and goal nodes until they meet each other. RRTstar [39] is an RRT planner with an optimising step, where a new sample node is connected to the tree so that the state space is more locally refined. Probabilistic Roadmap (PRM) [41] is a basic Probabilistic Road Map technique. It is similar to RRT and generates a limited number of random points within a given area.

In practice, the behaviour of sampling-based planners is dependent on the number and specific ordering of samples drawn, since the search structure they construct is a stochastic approximation to the connectivity of the configuration space [44, 45]. This stochasticity has two consequences that matter for applications. First, algorithm performance on any individual planning query becomes a random variable rather than a deterministic quantity, which necessitates statistical evaluation over multiple runs, a methodological point that directly motivates the benchmarking study of Chapter 3. Second, the formal guarantees introduced earlier in this section (probabilistic completeness for standard variants, asymptotic optimality for optimal variants) are all asymptotic in the number of samples, which means that they constrain the behaviour of a planner in the limit

Table 2.1: Sampling-based Approaches

Planner	**Category**	**Feature**	**Complexity**	**Ref**
RRT	Tree-based	Randomly sampling and incrementally growing trees to generate the trajectory of the motion planning.	Moderate build; single-query. Cost mainly from collision checks.	[38]
RRT*	Tree-based	Following the basic principle of RRT, but delivering the shortest possible path from the beginning to the goal.	Moderate High build (like RRT + rewiring); single-query.	[39]
EST	Tree-based	Tree expands towards the less explored space to generate an optimal path.	Moderate build; extra book-keeping for coverage.	[33]
RRTConnect	Tree-based	Two growing trees generate the trajectory of the motion planning.	Moderate build; typically faster in practice than RRT; single-query.	[7, 31, 40]
PRM	Roadmap-based	Multi-query; build a roadmap, then determine the optimal path with a search algorithm.	High to build (many connections); Fast to query (multi-query).	[41, 42]
PRM*	Roadmap-based	Improves PRM by generating an optimised roadmap without a fixed radius and removing the whole cluster system.	High to build (denser/optimised roadmap); Fast to query.	[39]
RRG	Roadmap-based	Incremental (vs. batch) algorithm to build a connected roadmap, possibly with cycles.	Moderate High to build (many near neighbour links); Fast to query.	[39]
SPARSE	Roadmap-based	Asymptotic near-optimality and a stopping criterion to build the roadmap.	Low Moderate to build (keeps graph small); Fast to query.	[43]

but do not determine its behaviour under the finite sample budgets that practical deployments impose. The central practical trade-off of sampling-based planning is therefore between planning speed and solution quality under finite time constraints that aggressive early termination yields fast but potentially suboptimal paths, whereas extended sample budgets approach the theoretical guarantees at the cost of computation time. This trade-off is a principal factor in real-world applicability and is examined quantitatively in Chapter 3.

2.2.2 Optimisation-based planners

A third family of motion planners, shown in Fig. 2.1, formulates planning as a continuous optimisation problem over trajectories rather than as a discrete search over sampled states. CHOMP [46] performs gradient descent on a functional that combines a smoothness term (penalising high trajectory curvature) with an obstacle-clearance term (computed from a signed distance field), producing locally optimal trajectories at the cost of sensitivity to initialisation. STOMP [47] replaces the exact gradient of CHOMP with a stochastic sampling of trajectory perturbations, which removes the requirement that the cost functional be differentiable and permits the incorporation of arbitrary cost terms at the price of noisier convergence. TrajOpt [48] formulates trajectory optimisation as a sequential quadratic programming problem subject to non-convex collision constraints, using convex–concave relaxations to handle the non-convexity and achieving competitive runtimes on high-dimensional manipulation problems. Optimisation-based planners typically produce smoother trajectories than sampling-based methods and support the direct encoding of task-specific cost terms, but they are generally local and depend on an initial seed trajectory, which motivates hybrid pipelines in which a sampling-based planner provides the seed and an optimisation-based planner refines it.

2.2.3 Evaluation Methodology and Limitations of Existing Benchmarks

Having reviewed the principal families of motion planners, sampling-based, optimisation-based, and LfD, we now turn to how these planners have been evaluated and compared in the existing literature, and we identify the methodological gaps that motivate the benchmarking study presented in Chapter 3 of this thesis.

Chamzas et al. [49] introduced MotionBenchMaker, a tool that standardises the generation of diverse manipulation scenarios across multiple robotic arms, enabling fair comparisons using common benchmarking metrics such as planning time and path cost. However, their initial experiments evaluated only three planners, which limits the ability to generalise conclusions about the behaviour of the sampling-based motion planners implemented in the Open Motion Planning Library (OMPL) [50,51], an open source library that provides reference implementations of RRT,

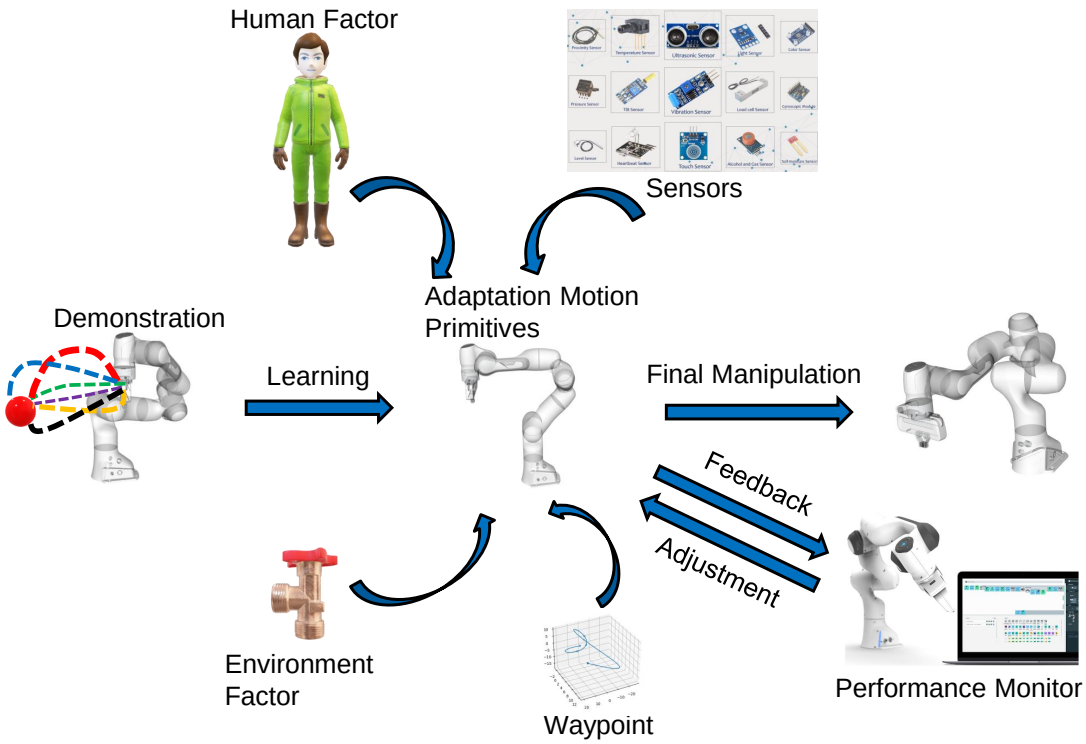


Figure 2.2: General scheme of movement primitive adaptation in LfD: demonstrations are encoded by learning algorithms into adaptive motion primitives, whose parameters are refined in a cycle through feedback from the Performance Monitor.

RRT*, RRTConnect, EST, KPIECE and related algorithms, and which has become the de facto standard for benchmarking sampling-based planners in the robotics community, across different clutter levels and manipulator architectures. In the benchmarking study presented in Chapter 3 of this thesis, we address this limitation by extending the MotionBenchMaker evaluation protocol to twelve widely used OMPL planners assessed across three representative robot arms (Franka, UR5, KUKA) and three levels of environmental clutter, thereby producing a substantially broader empirical characterisation of planner–arm interactions than the original study provided.

2.2.4 Learning from Demonstration

A movement primitives adaptation technique [13,54,59,61] is commonly used in LfD to generalise and adapt a demonstrated motion to new situations. This technique allows a robot or agent to learn from a human demonstrator's movements and then use that knowledge to perform similar tasks in new contexts, see Fig. 2.2 for the general scheme. In this scheme, motion demonstrations are processed through learning algorithms to generate adaptive motion primitives, which dynamically adjust based on human factors, environmental conditions, sensor inputs, and way-

Table 2.2: LfD planners

Planner	Structure	Feature	Complexity	Ref
DMP	Spring-damper system	Encodes a demonstrated trajectory via differential equations with a non-linear forcing term	Train: Low, simple least-squares over basis weights; Infer: Very low, constant-time per step integration on CPU	[52, 53]
ProMPs	Probabilistic framework	Learns trajectory distributions from stochastic movements with a stochastic feedback controller	Train: Low–Medium, fit per-demo weights then moments; Infer: Low–Medium, fast rollout; conditioning adds small matrix solve	[54, 55]
GMMs	Probabilistic framework	Models joint density as a mixture of Gaussians	Train: Medium–High, EM over all data; grows with #states and dimensionality; Infer: Low, fast GMR per query	[56]
CNMPs	Conditional Neural Processes	Movement primitives via conditional neural processes	Train: Medium–High, neural network training (epochs/architecture dependent, benefits from GPU); Infer: Low, single forward pass	[57, 58]
KMP	Probabilistic framework	Non-parametric imitation learning with kernels / basis functions	Train: Medium, one-off kernel matrix factorisation (cubic in support size); Infer: Low, linear to quadratic in support size	[59]
HSMM	Probabilistic framework	Switches across dynamical systems with explicit state durations	Train: Medium–High, EM with forward–backward over full sequences; Infer: Medium, Viterbi/forward–backward over horizon	[60]

point constraints. The Performance Monitor evaluates the executed manipulation against target criteria, providing real-time feedback that directly modifies the motion primitives through adjustment parameters. This architecture enables the system to iteratively refine robot movements by updating primitive parameters based on performance metrics, so the system adapts to changing task requirements and environments without extensive reprogramming. Movement primitives are low-level, reusable building blocks of movements that can be combined to perform complex tasks. By adapting these primitives, robots can adjust their movements to new situations and environments, which widens the range of tasks they can perform. However, one of the challenges of movement primitives adaptation is to ensure that the robot can complete the task while taking into account the disturbance in the environment. To address this challenge, researchers have developed various techniques, including inverse reinforcement learning, probabilistic models, and model-based reinforcement learning.

Much of the existing LfD adaptation literature consists of methods specialised to a single class of adaptation requirement rather than to the integrated handling of multiple requirements. Representative examples include the obstacle-avoidance DMP variants of Park et al. [62] and Hoffmann et al. [63], which augment a base primitive with a repulsive forcing term tuned to scene geometry; via-point conditioned ProMP variants such as those of Zhou et al. [21] and Paraschos et al. [64], which enforce intermediate-pose constraints through Bayesian conditioning of the trajectory distribution; and impedance or compliance-shaping approaches reviewed in Calinon [65], which adapt the stiffness profile of the executed motion. Each of these methods is effective within its target adaptation regime, but industrial deployments, where productivity gains depend on broad generality, typically require several adaptation regimes to be active simultaneously, for example obstacle avoidance combined with via-point enforcement and joint-limit compliance during a single manipulation. The literature offers comparatively little integrative analysis of how such adaptation regimes interact, conflict, or compose, and this thesis treats the integration of constraint-aware control with demonstration-based learning (Chapter 4) and with adaptive multi-objective inverse kinematics (Chapter 5) as principal contributions to closing this gap.

This section examines the implementation of various movement primitive adaptation techniques within LfD, with the aim of identifying methods that yield more robust and adaptive robotic systems capable of coping with the complexity of real-world environments. We classify these adaptation techniques into the following five categories: midpoint adaptation, obstacle avoidance, joint-constraint compliance, impedance control, and tactile-skill adaptation. For each category, we analyse strengths and drawbacks to help researchers generalise their robotic manipulation tasks to different scenarios based on the LfD approach, and so improve real-world performance.

The taxonomy summarised in Table 2.2, together with the category-level analysis above, exposes a recurring pattern across movement primitive frameworks that each formulation prioritises a specific subset of adaptation regimes, temporal stability and goal generalisation in DMPs, distribution-level conditioning in ProMPs, multi-modal demonstration handling in GMMs and

HSMMs, and kernelised generalisation in KMPs, at the cost of weaker support for the regimes prioritised by the others. Critically, none of these frameworks inherently enforces hard kinematic constraints such as joint limits, singularity avoidance, or collision-free execution in environments that differ from those encountered during demonstration. This observation motivates the technical contribution of Chapter 4, which we introduce next.

ProMPs, as the closing paragraph above noted, learn robot motion trajectories as distributions over weighted basis functions inferred from human demonstrations. The framework is, however, dependent on the quality of those demonstrations [10, 66] and does not inherently address the singularity problem, with the consequence that the generated motions can be suboptimal or kinematically infeasible whenever the demonstrations fail to account for singularity avoidance or joint-limit compliance. To address this gap, we integrate inverse kinematics principles, specifically the DLS method, directly into the ProMPs framework, so that singularity regularisation and joint-limit handling are enforced during both the learning and the generation phases. This integration is developed in detail in Chapter 4 of this thesis, where we show that the resulting framework produces motions that retain the adaptability of demonstration-based learning while satisfying the kinematic constraints that standard ProMPs do not enforce.

Table 2.3: Overview of representative IK solvers (Part I): computational principle, singularity handling, and hyperparameters.

Solver	Core computational principle	Singularity-robust mechanism	Typical hyperparameters	Ref.
DLS-IK	Damped pseudo-inverse update, $\dot{q} = J^{\mathsf{T}}(JJ^{\mathsf{T}} + \lambda^2 I)^{-1}\dot{x}$.	Global scalar damping factor $\lambda > 0$ regularises the inverse.	Damping gain λ; velocity limits.	[67]
SDLS-IK	Direction-dependent damping: each singular vector of J is damped separately, with stronger damping on large-error directions.	Per-singular-value damping gains.	Error-clamp threshold; per-axis gains.	[68]
G-DLS-IK	Gaussian-profile damping, $\lambda_i = \lambda_{\max} e^{-(\sigma_i/\varepsilon)^2}$, applied per singular vector of J.	Smooth continuous damping eliminates velocity discontinuities; GA tunes $\lambda_{\max}$ and ε offline.	$\lambda_{\max}$; ε; GA population, crossover, mutation.	[69]
TRAC-IK	Damped pseudo-inverse run in parallel with SQP; returns the first feasible (or best) solution.	Combines Jacobian damping with constraint-aware SQP.	Timeout; joint-limit weight; tolerance radius.	[70]
Bio-IK	Memetic evolution (GA + PSO + L-BFGS-B) over genotype-encoded joint configurations with recombination, mutation, and elitism.	Direct joint-space sampling avoids singular configurations; niching tracks multiple solutions.	Population size; elite count; timeout; error thresholds.	[71]
Relaxed-IK	Weighted-sum nonlinear optimisation over normalised Gaussian-polynomial objectives (pose, smoothness, collision, singularity).	SVD-based condition-number monitoring with hard cut-off below $\mu_c - 2\sigma_c$; NN approximates collision state.	Per-objective weights; loss params (n, s, c, r); condition-number threshold.	[72]

Table 2.4: Overview of representative IK solvers (Part II): strengths and limitations.

Solver	Strengths	Limitations
DLS-IK	Simple to implement; substantially improves robustness near singularities.	Choice of λ trades accuracy against stability; convergence can be slow.
SDLS-IK	Preserves accuracy along well-conditioned directions; fewer oscillations on unreachable targets.	Marginally higher per-iteration cost than DLS; still local and gradient-based.
G-DLS-IK	Minimal end-effector tracking error versus constant or piecewise damping; no unnecessary damping on well-conditioned directions; smooth joint-velocity profiles through singularities.	Requires an offline, problem-specific GA tuning step; heavier than plain DLS during tuning.
TRAC-IK	Fast average solve time; higher success rate than KDL across many robots; open-source ROS library.	Non-deterministic runtime due to thread race; tuning needed for tight tolerances.
Bio-IK	Handles multiple objectives and chains simultaneously; high success rate; scales to high DOF; supports real-time collision avoidance.	Objective-weight tuning is difficult; stochastic nature can cause solution jumps; cost grows with number of objectives.
Relaxed-IK	Achieves pose matching with motion feasibility; real-time collision avoidance; smooth minimum-jerk motions; works on under-actuated robots.	No convergence guarantees; slower than standard IK ($\sim$17 ms); requires 20–30 min of offline preprocessing; no dynamics yet.

2.3 Inverse Kinematics

Frame	a (m)	d (m)	α (rad)	θ (rad)
Joint 1	0	0.333	0	q_1
Joint 2	0	0	-π/2	q_2
Joint 3	0	0.316	π/2	q_3
Joint 4	0.0825	0	π/2	q_4
Joint 5	-0.0825	0.384	-π/2	q_5
Joint 6	0	0	π/2	q_6
Joint 7	0.088	0	π/2	q_7
Flange (F)	0	0.107	0	0
End effector (EE)	0	0.1034	0	π/4

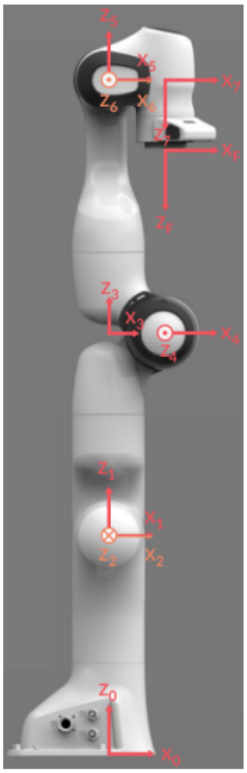


Figure 2.3: Franka Emika kinematic chain and Denavit–Hartenberg parameters [73]

IK forms a fundamental component of robotic manipulation, translating desired end-effector positions and orientations into corresponding joint configurations. In robotic arm applications, this computational method is essential for executing precise and adaptable movements, especially within complex and obstacle-filled environments [74, 75].

FK [76, 77] describes how a robot's internal joint states determine the position and orientation of its end-effector in the workspace. Starting from a vector of joint variables, angles for revolute joints or linear displacements for prismatic ones, FK chains together the rigid-body transformations of each link to produce a unique pose of the tool frame with respect to the base. Two parameterisations dominate practice:

- **Denavit–Hartenberg (DH) convention (Fig. 2.3)** [78, 79] A compact, four-parameter description that has become the de-facto standard in industrial manipulators thanks to its tabular clarity and long-standing software support.
- **Product of Exponentials (PoE) formulation** [80, 81] A coordinate-free, twist-based representation that lends itself to symbolic differentiation, modern automatic differentiation

libraries, and direct extension to spatial mechanics.

Because FK is algebraically straightforward and runs in microseconds on modern hardware, it sits at the core of real-time control cycles, simulation engines, and digital twins. The forward map feeds the analytical Jacobian, which links joint velocities to end-effector motion and underpins inverse kinematics, operational-space control, and dynamic modelling. Accurate FK is therefore essential not only for motion planning but also for calibration, state estimation, and safety envelopes.

Emerging robot classes, such as redundant, parallel, continuum, or soft mechanisms, have rekindled research interest in generalising FK to richer configuration spaces and more expressive geometric representations. Contemporary open source libraries now offer plug-and-play FK routines that support both DH and PoE, expose automatic differentiation, and integrate with machine learning toolchains, reflecting the field's shift toward data-driven and model-based hybrid workflows.

Traditional IK methods, including the Jacobian Transpose, Jacobian Pseudoinverse, and Analytical IK, remain widely employed due to their straightforward implementation and computational efficiency [82, 83]. However, these conventional techniques suffer from well-documented limitations that constrain their practical applicability. Near kinematic singularities, where the Jacobian loses rank, these methods exhibit numerical instability that can produce erratic joint velocities or complete failure to converge. Additionally, their local optimisation nature makes them susceptible to local minima, particularly in redundant manipulators where multiple solutions exist for a given end-effector pose.

The DLS (Table 2.3) method was an important advance, introducing damping parameters that improve numerical stability near singularities [84]. While DLS successfully prevents the numerical explosion of joint velocities, the selection of appropriate damping factors remains problematic. A fixed damping factor inevitably trades accuracy for stability, resulting in steady-state errors even when feasible solutions exist. This fundamental limitation motivated the development of SDLS (Table 2.3), which applies direction dependent damping by scaling each singular vector of the Jacobian separately [68]. The Gaussian-DLS variant further refines this approach through continuous damping profiles tuned via genetic algorithms, achieving smoother joint-velocity profiles through singularities at the cost of offline optimisation requirements [69].

The limitations of gradient-based methods have driven the development of sophisticated optimisation frameworks that reformulate IK as constrained optimisation problems. TRAC-IK (Table 2.3) is an open source IK solver that extends the classical Newton–pseudoinverse approach of Orocos KDL [85] with two key elements: (i) a local-minimum detector that triggers stochastic restarts to avoid joint-limit singularities, and (ii) a parallel Sequential Quadratic Programming (SQP) optimiser whose Quasi-Newton updates cope with the resulting non-smooth search land-

scape. The two sub-solvers are coordinated by a lightweight thread manager that returns the first converged result and terminates the others, avoiding wasted computation once a feasible pose is found [70]. This race and terminate coordination is an instance of the more general parallel-portfolio paradigm well established in optimisation and combinatorial search [70], and is in principle applicable to any IK formulation that admits multiple complementary sub-solvers, for example, a damped pseudoinverse solver running alongside a constrained-optimisation solver, as in TRAC-IK itself, or analogous combinations of analytical and numerical solvers. However, in the IK literature, the specific first convergence thread management mode is still most closely related to TRAC-IK; related but different ideas appear in Bio-IK's multi-startup population search and some multi-restart heuristics used by optimisation-based solvers, but the application of the complete parallel portfolio competition architecture outside of TRAC-IK remains limited. These design choices make TRAC-IK a robust, real-time capable baseline against which newer optimisation and learning-based IK methods are routinely benchmarked.

Bio-IK (Table 2.3) takes a fundamentally different approach by employing memetic evolution that combines genetic algorithms, particle swarm optimisation, and gradient-based refinement [71]. By encoding joint configurations as genotypes and evolving populations through recombination and mutation, Bio-IK inherently avoids singularities through its joint-space sampling strategy. The solver's multi-modal optimisation capabilities, enhanced by niching mechanisms, enable simultaneous tracking of multiple solutions, making it particularly valuable for redundant manipulators and multi-chain systems. However, this probabilistic nature introduces challenges in deterministic applications that require consistent solutions, and the computational cost grows with the number of competing objectives.

Relaxed-IK (Table 2.3) presents a unified framework that encodes multiple objectives including pose matching, smoothness, collision avoidance, and singularity avoidance as normalised Gaussian-polynomial terms within a weighted-sum optimisation [72]. The method's innovation lies in its real-time collision state approximation through neural networks and its SVD-based condition number monitoring that actively avoids poorly conditioned configurations. While Relaxed-IK achieves high motion quality with minimum-jerk characteristics, it cannot provide convergence guarantees and requires substantial preprocessing time (20-30 minutes) to train the collision approximation networks.

No current IK solver is universal and the methods form a spectrum with complementary strengths. Gradient-based methods excel in computational efficiency and predictability but fail catastrophically near singularities. Optimisation-based approaches offer superior robustness and flexibility but at the cost of increased computational complexity and potential non-determinism. The choice of solver depends on the specific application requirements that TRAC-IK suits general purpose manipulation with its balance of speed and reliability, Bio-IK excels in highly constrained multi-objective scenarios despite its computational overhead, while Relaxed-IK provides superior motion quality for applications prioritising smoothness and collision avoidance.

The scalability of IK solvers to high-DOF manipulators reveals fundamental trade-offs between computational complexity and solution quality. Classical Jacobian-based methods exhibit linear per-iteration complexity in the number of joints for the forward-kinematics and Jacobian evaluations themselves, but their convergence behaviour deteriorates dramatically as dimensionality increases, owing to the proliferation of kinematic singularities and local minima in the configuration space. The dominant computational cost lies in the linear-algebra step that maps the task-space residual into a joint-space update. In Jacobian-based formulations such as DLS (Eq. 4.3), each iteration requires forming the matrix product $J^TJ + \lambda^2 I$ or equivalently $JJ^T + \lambda^2 I$, depending on the formulation, and either inverting it or solving the associated linear system. For direct dense factorisations such as LU decomposition, Cholesky factorisation (applicable here because $J^TJ + \lambda^2 I$ is symmetric positive definite for any $\lambda > 0$), or singular value decomposition, the cost of factorising an $n \times n$ dense matrix is $O(n^3)$, and this cubic scaling dominates the per-iteration runtime in moderate to high DOFs manipulators. For iterative Krylov-subspace methods such as conjugate gradient, the dominant per-iteration cost is a single matrix–vector product at $O(n^2)$, and the total cost reduces to $O(n^2)$ when the number of iterations required for convergence is bounded independently of n, which holds in well-conditioned regimes but degrades near kinematic singularities where the system becomes ill-conditioned and iteration counts grow. The practical consequence is that direct factorisation remains the default for the moderate DOFs manipulators ($n \leq 10$) considered in this thesis, where the cubic cost is acceptable, while iterative methods become attractive primarily for higher DOFs systems such as humanoids and continuum manipulators, provided that conditioning can be maintained through regularisation or preconditioning.

Population-based methods like Bio-IK face exponential growth in the solution space, requiring larger populations and longer evolution times to maintain coverage. The population size typically needs to scale superlinearly with DOF to maintain solution quality, leading to computational requirements that can exceed real-time constraints for manipulators with more than 10-12 joints. Quadratic Programming approaches encounter similar challenges, as the number of inequality constraints grows with both the DOF and the complexity of the environment, resulting in solve times that can vary significantly based on the active constraint set.

Deep learning methods for inverse kinematics present a fundamental paradox in their application to robotics. Li et al. [86] combine invertible neural networks with error decoupling mechanisms specifically to handle occlusions, while Rice et al. [87] relegate their neural network to a supporting role, merely seeding an evolutionary algorithm rather than solving IK directly. These hybrid architectures suggest that deep learning alone cannot reliably handle the precision requirements of robotic manipulation. These models exhibit poor generalisation across different robot configurations, a neural network trained for one kinematic chain cannot transfer to another without complete retraining, unlike analytical methods that can be systematically derived for any configuration.

The third contribution of this research addresses the identified challenges by integrating adaptive Quadratic Programming weighting with an enhanced Damped Least Squares framework. This novel combination draws on the stability benefits of selective damping while retaining the flexible task prioritisation of QP. Unlike previous approaches that treated singularity avoidance and multi-objective optimisation as separate problems, our method unifies these concerns within a single computational framework. By dynamically adjusting both damping factors and objective weights based on the robot's configuration and task requirements, we achieve superior stability near singularities while efficiently managing multiple objectives. This integration overcomes the computational overhead of traditional SDLS methods and the inflexibility of standard QP approaches, providing a robust and versatile solution for complex robotic manipulation tasks.

2.4 Conclusion

This chapter has reviewed motion planning, LfD, and IK for manipulators, finding strong progress but fragmented integration and persistent gaps that impede real-world deployment. Sampling-based planners have good trade-offs, yet cross-planner comparisons remain narrow; LfD has advanced from primitives to deep models but still struggles with constraints and singularities (e.g., ProMPs can yield infeasible motions); and IK methods split between fast but brittle Jacobian schemes and slower, more robust optimisers. These challenges are amplified by the sim-to-real gap, siloed component design, scarce standardised benchmarks, and poor scalability to high-DOF systems, leaving practitioners without reliable guidance.

Motivated by these gaps, our work broadens empirical comparison across twelve planners and varied clutter regimes to inform algorithm selection, integrates IK principles into ProMPs to achieve constraint-aware, singularity-robust adaptation, and unifies IK with adaptive damping and dynamic objective weighting to enable real-time, multi-objective control. We thus anchor the contributions that follow, shifting from isolated algorithm tweaks toward integrated, rigorously validated systems capable of robust, efficient manipulation in noisy, uncertain environments.

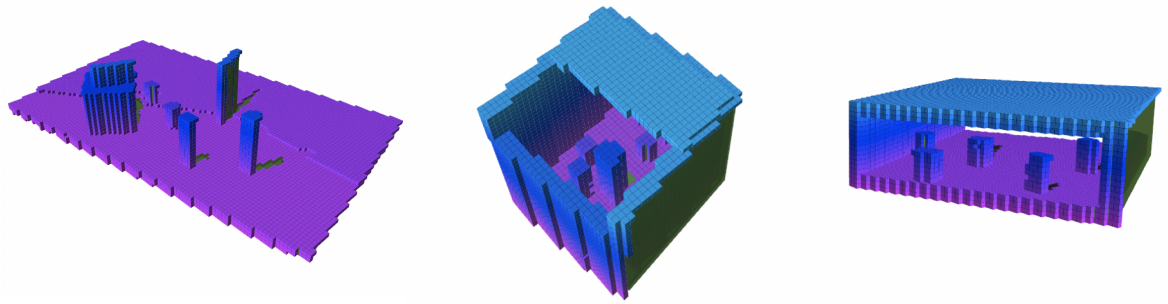
Chapter 3

Benchmarking of Robot Arm Motion Planning

Motion planning is essential for robotic automation across various industries. However, generalising research outcomes has been challenging due to the narrow focus of previous work on a specific robot arm system. Here, we take a broader approach by exploring the combinations of three popular robot arm systems, three levels of clutter in the environment, and twelve popular motion planners. To conduct the necessary performance analysis, we employ the MotionBenchMaker tool [49] and introduce a sensitivity metric. Our approach is structured and accessible, enabling the identification of the best-performing planner-robotic arm combinations. We find that the LBKPIECE, RRTConnect, and BKPIECE planners with Franka and UR5 offer the best balance of effectiveness and robustness. More generally, our results help researchers and practitioners make informed decisions when selecting robotic arms and motion planners, for use in environments with different degrees of clutter.

3.1 Introduction

Motion planning is essential for industrial robots to ensure precise movements and avoid obstacles [88]. However, two interrelated challenges persist in the existing benchmarking literature. The first is limited generalisability across robotic-arm architectures, since most studies evaluate a single platform and offer little guidance on how planner performance transfers to other manipulators. The second is the absence of a systematic empirical characterisation of how planner performance varies with controlled changes in the geometric properties of the workspace [89–91]. Throughout this chapter, the term "cluttered environment" is used in the specific sense of a workspace whose geometric properties, workspace volume, obstacle height, and the spatial con-



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Figure 3.1: The three benchmarking scenarios: (a) a flat surface with obstacles of different shapes; (b) a semi-closed box; (c) a drawer-like enclosure.

finement imposed by surrounding structure, restrict the volume of feasible configurations available to the planner. The three scenarios depicted in Fig. 3.1 instantiate three increasing levels of spatial confinement: an open plane with scattered obstacles of different shapes, the simplest scene without space constraints; a semi-closed box, which imposes object access restrictions; and a drawer-like enclosure, in which the working space is the most severely limited of the three. By varying these geometric properties in a controlled manner across the three scenarios, the benchmarking study presented in this chapter provides the systematic empirical characterisation that the existing literature lacks. Several benchmarking studies have been conducted to evaluate and compare the performance of motion planning algorithms and robotic arm systems. For instance, a study [92] focused on the optimisation and evaluation of motion planning algorithms in various scenarios, proposing a motion planning pipeline connecting the Open Motion Planning Library (OMPL) with optimised CHOMP or STOMP algorithms. Also [93] performed benchmarking tests on a 7-DOF robotic arm with various controllers to evaluate their accuracy, control efficiency, jitter, and robustness. While these studies provide valuable insights into motion planning algorithm performance, there are still gaps that need to be addressed. [49] introduced the MotionBenchMaker tool to generate and benchmark motion planning datasets. Another study [50] presented an extensible infrastructure for the analysis and visualisation of motion planning algorithms. While these studies provide valuable insights into motion planning algorithm performance, various gaps still need to be addressed.

One of the major gaps in existing studies is the limited generalisability of the results to other robotic arm systems, as many studies focused on one robotic arm. Moreover, there is a lack of comprehensive investigation into the impacts of the working environment on motion planning, especially in the context of cluttered environments. The influence of the environment's properties, such as the size of the working space and the dimensions of obstacles, on motion planning performance, remains unexplored. Furthermore, there is a need for a standardised framework that enables the systematic comparison and evaluation of motion planners and robotic arm systems in various environments.

To address this gap, we conduct benchmarking studies that compare the performance of twelve OMPL [51] motion planners used with three different robotic arms: Franka [78], UR5 [94], and KUKA [95], see (Table 3.1), in three environments of different levels of clutter. The MotionBenchMaker tool [49] is utilised to facilitate the benchmarking process, providing a unified platform for performing the evaluation of different motion planners and robotic arms. Our experiments investigate the performance of three robotic arms to determine their suitability for motion planning tasks in cluttered environments. The motion planners are tested in three distinct cluttered environments with varying levels of complexity: simple, moderate, and difficult, based on the benchmarks proposed by [96]. These environments will present unique features and require different planning strategies. The performance of the motion planners and robotic arms will be evaluated using the following metrics suggested by [97], time efficiency, success rate, and sensitivity to the *range* parameter.

In OMPL planners, range is the maximum length of a single motion (edge) that the local planner is permitted to add when expanding the tree or graph, so any attempt to extend toward a target is truncated to this bound. The choice of range governs a fundamental coverage resolution trade-off. Larger range values allow each extension to cover a greater volume of the configuration space, which accelerates global coverage and reduces the total number of extension attempts required to reach the goal region. However, they also increase the per-edge collision checking cost (because each edge contains more discrete check points) and, more importantly, raise the probability that an extension will overshoot a narrow passage in the free configuration space, since the planner cannot represent feasible motions at a finer scale than its step size. Smaller range values, conversely, refine the search at higher resolution and are better able to traverse narrow passages and tight corridors, but they require more extension attempts to cover a given volume and therefore pay a larger total bookkeeping and nearest neighbour cost. The impact on planning time is therefore non monotone in range and depends on scene geometry that larger values tend to be advantageous in open scenes with few narrow features, while smaller values are typically necessary in cluttered or constrained scenes such as those examined in this chapter.

Contributions are: (1) exploration and analysis of the influence of the working environment properties on motion planning for robotic arms, with a focus on the size of the working space and the height of obstacles, (2) introduction of a three-axis evaluation framework (time efficiency, success rate, and parameter sensitivity) analysing the OMPL range parameter, which reveals performance variations between planners that prior fixed-parameter benchmarks have concealed, (3) developing a cost function that can score motion planners across different scenarios, and recommend the appropriate planner based on the specific task requirements, and (4) a comparison of the performance of three robotic arms (Franka, UR5, and KUKA) in various cluttered environments, providing insights into the most efficient and robust planner-arm combinations. Our results enable researchers and practitioners to make informed decisions when selecting robotic arms and motion planners for their specific applications, ultimately improving the efficiency and robustness

Table 3.1: Features of the robotic arm

Robotic Arm	**Feature**	**Application**
Franka	7 DOFs, real-time motion planning, compliance control, advanced sensing capabilities, scalability	Assembly complex mechanical parts in manufacturing, inspections and measurements in research, surgical procedures in healthcare
UR5	6 DOFs, user-friendly interface, safe operation, repeatability, flexibility integration	Picking and placing goods, testing and evaluating new robotic algorithms, assisting to patients
KUKA	6 DOFs, precision and accuracy, high speed and performance, safe operation, customisation, integration	It can be used in industrial production to automate the process of placing goods or products onto pallets

of robotic systems in complex environments.

In the following section, we review related work on motion planning in benchmarking. Section 3.3 presents the variations in scenarios and queries, as well as the metrics used for benchmarking. In Section 3.4, we describe the experimental setup and methodology for the benchmarking. A final section draws conclusions and discusses the implications of the findings.

3.2 Software

We employ the Robot Operating System (ROS) [98] framework, the MoveIt [99] library for planning and executing robotic arm movements, and the MotionBenchMaker [49] repository for benchmarking motion planning algorithms. ROS is an open source platform for building robot applications, while MoveIt provides a unified interface for performing complex tasks, such as grasping objects and navigating in three-dimensional space. The MotionBenchMaker repository offers a standardised and modular interface for benchmarking motion planning algorithms, supporting a wide range of robotic platforms and planning algorithms for performance evaluation and comparison.

3.3 Problem formulation

3.3.1 Variation definitions

MotionBenchMaker (MBM) generates benchmark instances in two stages. In the first stage, it perturbs a nominal scene at two levels: (i) a global world frame transform applied uniformly to all scene elements, and (ii) each object SE(3) noise applied independently to each object pose. In the second stage, it samples end-effector (EE) query offsets relative to the perturbed scene to produce concrete start–goal pairs. Together, these two stages yield diverse yet parameterised benchmarks suitable for systematic comparison across planners and robot platforms. For object i with nominal pose $T_i \in SE(3)$, the sampled pose is

$$T_i' \;=\; T_{\text{world}}(\delta\mathbf{t}_w, \delta\boldsymbol{\theta}_w)\; T_i\; T_{\text{local},i}(\delta\mathbf{t}_i, \delta\boldsymbol{\theta}_i), \tag{3.1}$$

where translations $\delta\mathbf{t}_{\{\cdot\}} \in \mathbb{R}^3$ and roll–pitch–yaw angles $\delta\boldsymbol{\theta}_{\{\cdot\}} \in \mathbb{R}^3$ are drawn independently per axis either from Gaussian noise, $\delta x \sim \mathcal{N}(0, \sigma_x^2)$ (similarly for $y, z,$ roll, pitch, yaw), or from symmetric uniform bounds, $\delta x \sim \mathcal{U}[-b_x, b_x]$; rotations are composed as $R(\delta\boldsymbol{\theta}) = R_z(\delta\psi)R_y(\delta\theta)R_x(\delta\phi)$. In the experiments reported in this chapter, translation perturbations are drawn from $U[-0.05, 0.05]$

m on each Cartesian axis and rotation perturbations from $U[-0.1, 0.1]$ rad on each rotational axis. These bounds were chosen to be large enough to produce meaningful instance diversity, empirically yielding distinct query geometries across runs while preserving the qualitative character of each scenario, but small enough that perturbed scenes remain reachable by all three manipulators (Franka, UR5, KUKA) considered in this chapter and remain within the difficulty range tractable by the planners under evaluation. Larger bounds were trialled in pilot runs and were observed to push a non-trivial fraction of instances into the infeasible regime, where success rate measurements would conflate planner failure with instance infeasibility. For articulated geometry with joint vector q, MBM can sample $q' = \Pi_{\mathcal{J}}(q + \delta q)$ with $\delta q \sim \mathcal{N}(0, \Sigma_q)$ and projection $\Pi_{\mathcal{J}}$ enforcing joint limits (yielding self-collision-free kinematics). EE start/goal queries are generated by applying offsets in a chosen frame F: ${}^{F}T'_{\mathsf{EE}} = {}^{F}T_{\mathsf{EE}}\, T(\delta\mathbf{t}_q, \delta\boldsymbol{\theta}_q)$; valid robot states are then obtained via IK with collision checking. Since SE(3) perturbation cannot guarantee that independent objects do not overlap, the actual dataset uses constraint range and rejection sampling, that is, any perturbation scenario containing scene-scene collision samples will be discarded and resampled, thus cleaning up the output. Setting random number generator (RNG) seeds ensures repeatability; reporting (σ or b) per axis, the rejection policy, and the seeds used makes the results fully reproducible. The experiments reported in Section 3.4 fix the RNG seed at 42 across all planner, arm and scenario combinations to ensure that comparisons isolate the algorithmic differences between planners rather than reflecting sampling variability.

3.3.2 Scenarios

The three benchmarking scenarios used throughout this chapter are designed to span the spatial confinement axis introduced in Section 3.1, with each scenario instantiating a distinct point on the spectrum from minimal to severe geometric restriction of the manipulator's reachable workspace. The scenarios are not generated by MBM itself but are instead manually designed base scenes that serve as nominal templates; each base scene comprises a fixed set of collision objects with defined poses, and constitutes the core component encapsulating the benchmark tests for its scenario. MBM, as described in Section 3.3.1, is then applied to each base scene to generate stochastic instance variations through the perturbation pipeline. This two-tier structure separates the deterministic design choices that distinguish the three scenarios from the stochastic instance variations within each scenario, allowing us to attribute observed performance differences either to scenario-level geometry (across scenarios) or to instance variability (within scenarios).

Across the experiments reported in Section 3.4, the following elements are held fixed within each scenario, the manipulator model and its kinematic parameters; the topological structure of the scenario (i.e., whether the workspace is a flat surface, a semi-closed box, or a drawer-like enclosure); the dimensions of any enclosing structure (box walls or drawer walls); and the broad spatial layout that distinguishes one scenario from another. The following elements are

randomised across runs by the MBM pipeline, the precise SE(3) pose of each internal obstacle, drawn from the bounded uniform perturbation specified in Section 3.3.1; the start and goal EE poses, generated by collision-aware inverse kinematics; and the RNG seed governing the sampling, as detailed in Section 3.3.1. Each (planner, robot, scenario) combination is then evaluated over 100 such randomised instances within its scenario.

We acknowledge that restricting attention to motion planning and excluding object grasping is a scoping limitation of this study, since in full manipulation pipelines the planner and the grasping subsystem are coupled that pre-grasp approach geometry, EE orientation constraints, and post grasp retraction all interact with the planner's behaviour. The exclusion is, however, a deliberate methodological choice consistent with the standard practice of motion planning benchmarks. Introducing grasping at this stage would conflate planner performance with gripper kinematics, EE design, and object specific stability constraints, all of which vary substantially across deployments and would obscure the cross planner and arm comparisons that this chapter is designed to support.

The first scenario (Fig. 3.1.a) involves cluttered environments without spatial constraints, where obstacles of different shapes are placed on a single plane. A scene generation module is utilised to add random noise to the pose of collision objects relative to the global frame in a standard scene, as described in [100]. This study solely focuses on motion planning and does not take into account object grasping. The task of the robotic arm is to enter the workspace and reach one of the objects.

The second scenario (Fig. 3.1.b) introduces spatial constraints by enclosing the manipulator within a semi-closed box, restricting its reachable volume relative to Scenario 1. The robotic arm must plan its movements within this restricted space, and the task is to enter the box and reach one of the objects placed inside. Compared to Scenario 1, this configuration imposes lateral and vertical bounds that constrain the joint motion required to approach internal targets. A representative Scenario 2 motion planning instance is illustrated in Fig. 3.2, in which the Franka end-effector moves between objects inside the box without collision, while the resulting planned trajectory is traced as a blue line in Fig. 3.3.

The third scenario (Fig. 3.1.c) resembles a living room drawer, with an even more restricted workspace than the previous two, and the obstacles within the drawer cannot be bypassed directly from the open space above. The task of the robotic arm is entering inside the drawer, simulating motion planning in an even more restricted space compared to the previous two, and reaching one of the objects.

Table 3.2 summarises the geometric properties that distinguish the three scenarios, namely the type of enclosing structure, its internal dimensions and access opening size, and the resulting fraction of the Scenario 1 free workspace volume that remains reachable. These quantities are

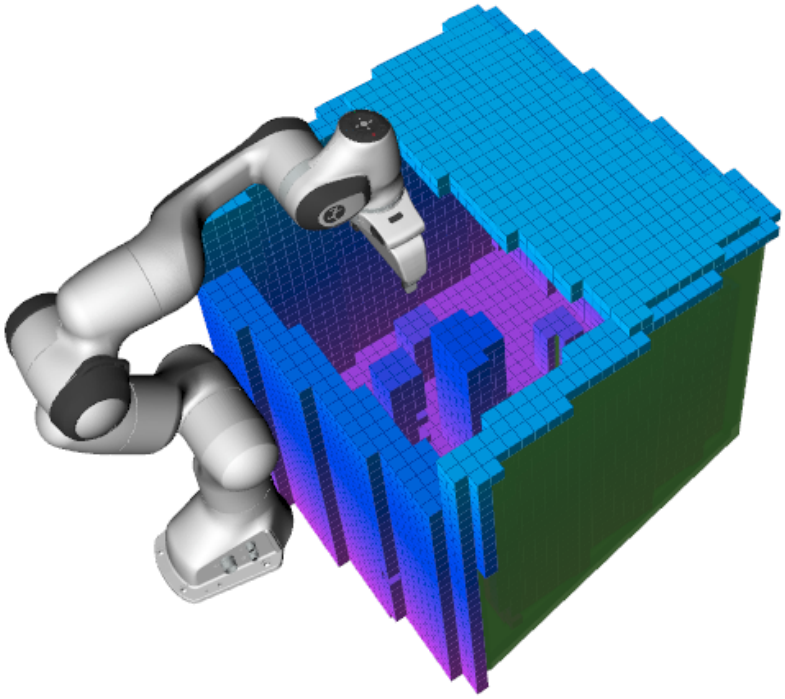


Figure 3.2: An illustration of motion planning for Scenario 2. The end-effector of the Franka robot arm is inside the box, moving from one of the objects to the top of the other object without obstacle collisions in the process.

design parameters of the manually constructed base scenes rather than stochastic outputs of the MBM perturbation pipeline, and they make the otherwise qualitative notion of "moderately cluttered" precise that Scenario 1 is the unconstrained reference; Scenario 2 reduces the reachable volume to approximately 45%; and Scenario 3 reduces it further to approximately 25% by means of a drawer enclosure whose closed top restricts access to a single frontal opening.

3.3.3 Query formulation

The query generation process is configured through YAML files distributed with the MBM scenario definitions and the experimental code accompanying this thesis, available at 'repository URL'. For each scenario described in Section 3.3.2, the corresponding query specifications reside in 'queries/scenario1.yaml', where each entry defines a single start–target pair. A representative excerpt from 'queries/scenario2.yaml' is shown below:

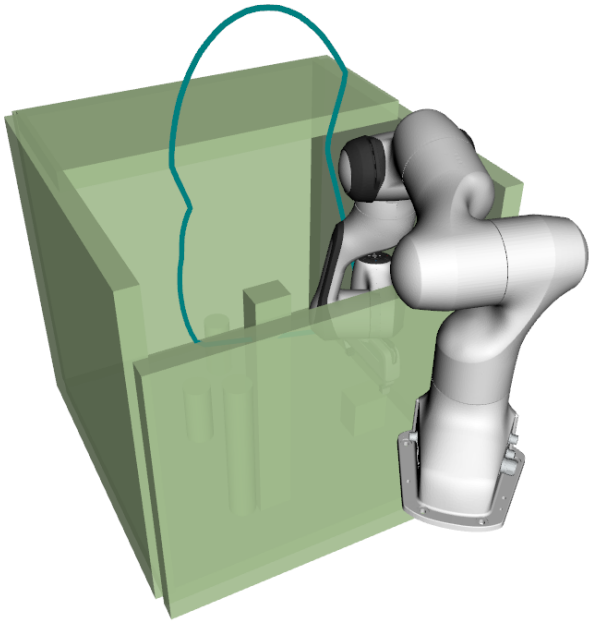


Figure 3.3: In Scenario 2, the Franka robot arm's motion planning resulted in a trajectory depicted by a blue line.

Table 3.2: Geometric properties of the three benchmarking scenarios. The free workspace volume fraction is expressed relative to the unconstrained Scenario 1 workspace, which serves as the reference. The listed quantities are design parameters of the manually constructed base scenes (Section 3.3.2), not stochastic outputs of the MBM perturbation pipeline.

Scenario	**Enclosure type**	**Enclosure / access geometry**	**Free vol. (%)**
1 (open plane)	None (planar surface)	Unbounded above and laterally; obstacles 0.3–0.4 m tall	100
2 (semi-closed box)	Five-sided box	Internal $0.6 \times 0.6 \times 0.5$ m; single open face 0.6×0.5 m	45
3 (drawer enclosure)	Drawer (closed top)	Internal $0.6 \times 0.6 \times 0.3$ m; frontal opening 0.6×0.3 m	25

```yaml
query_0001:
  start:
    joint_values: [0.0, -0.785, 0.0, -2.356, 0.0, 1.571, 0.785]
  goal:
    position: [0.45, 0.10, 0.55]
    orientation: [0.0, 1.0, 0.0, 0.0]   # quaternion (w, x, y, z)
    tolerance:
      position: 0.001
```

```
9       orientation: 0.01
```

Each query specifies the start configuration in joint space and the goal EE pose in Cartesian space, together with the position and orientation tolerances within which the planner is required to converge. The full set of queries used for the experiments reported in Section 3.4 is included in the repository, together with the RNG seeds (Section 3.3.1) needed to reproduce the perturbed scene instances and the corresponding query results.

3.3.4 Evaluation Metrics for Planner Comparison

We evaluate each (planner, robot, scenario) combination along three metrics that together characterise both its raw performance and its robustness to parameter choice.

Time efficiency is defined as the average actual time (in seconds) required for the planner to compute a feasible path for a given query, which is the average of 100 random instances drawn from the corresponding scenario (Section 3.3.2). Lower values indicate faster planning. Time efficiency is reported only over successful runs, since the time recorded for a failed run reflects the timeout rather than meaningful planner behaviour.

Success rate is defined as the percentage of randomised instances for which the planner returns a feasible, collision-free path within the planning timeout (set to 30 s in the experiments of Section 3.4). This metric quantifies the planner's reliability under instance variability and is the principal indicator of robustness to scene perturbation.

Parameter sensitivity is defined as the variability of time efficiency and success rate as the OMPL range parameter is swept over a controlled interval. The range parameter and its role in the coverage resolution trade-off were introduced and analysed in detail in Section 3.1 and here we use the parameter sensitivity of each planner as a third evaluation dimension that exposes performance differences which prior benchmarks, conducted at fixed default range values, would have concealed. The specific sweep used in the experiments is described in Section 3.4.2.

These three metrics are used throughout the experiments of Section 3.4 and are aggregated through the weighted scoring scheme described in Section 3.3.5 to produce planner rankings within and across scenarios.

3.3.5 Planners' score

For each planner, scenario and arm combination, we first calculate the average time efficiency (T_{avg}), which represents the mean computation time required for the planner to generate a feasible

path across all test runs within that specific configuration. To enable meaningful comparisons across planners with vastly different time scales, we apply min-max normalisation to transform these raw time measurements into a standardised score:

$$N_t = 1 - \frac{T_{avg} - \min(T_{avg})}{\max(T_{avg}) - \min(T_{avg})} \tag{3.2}$$

where N_t is the normalised time efficiency score (range: 0 to 1) and T_{avg} is the average computation time for a specific planner in a given scenario. The normalisation is inverted (1 minus the standard normalisation) to ensure that lower computation times yield higher scores, with 1.0 representing optimal performance and 0.0 representing the poorest performance.

Given that our three test scenarios present varying levels of complexity and computational challenge, we employ a weighted averaging scheme to compute each planner's overall performance score. This approach acknowledges that success in more challenging environments should contribute more substantially to the final evaluation:

$$W_{\text{avg}} = \sum_{i=1}^{3} w_i \, N_{t,i} \tag{3.3}$$

$$W_{avg} = w_1 N_{t_1} + w_2 N_{t_2} + w_3 N_{t_3} \tag{3.4}$$

where W_{avg} is the weighted average score for a planner across all scenarios, $N_{t,i}$ is the normalised time efficiency score for scenario i and w_i is the weight assigned to scenario i ($w_1 = 1/6, w_2 = 1/3, w_3 = 1/2$).

The weight distribution reflects the increasing difficulty: Scenario 1 (open environment) receives the lowest weight, Scenario 2 (semi-enclosed box) receives moderate weight, and Scenario 3 (drawer-like environment) receives the highest weight, as it presents the most significant motion planning challenges.

3.4 Experiments

We conducted experiments using the MBM tool to create scenarios for benchmarking motion planners in conjunction with robotic arms. The C++ and Python scripts referenced throughout this chapter constitute the control layer through which we configure MBM, specify base scene parameters, set the perturbation bounds and RNG seeds, and invoke the planners under evaluation. For environmental configuration benchmarking, we used single-query planners from OMPL,

Table 3.3: Scenario 1, deterministic obstacle height 0.3 m. Mean planning time and success rate for each (planner, robot) combination, evaluated over 100 randomised instances generated by MBM SE(3) perturbations around the Scenario 1 base scene (Section 3.3.2) under the noise specification of Section 3.3.1. The lowest mean time per robot is shown in bold.

	Franka		**UR5**		**KUKA**	
Planner	Time (s)	Succ. (%)	Time (s)	Succ. (%)	Time (s)	Succ. (%)
RRT	1.504	89	4.793	59	1.622	65
RRTConnect	**0.014**	100	**0.069**	100	**0.070**	100
RRTstar	30.003	75	30.040	47	30.030	60
TRRT	1.288	66	5.150	37	1.216	70
EST	0.073	100	1.670	100	0.696	100
SBL	0.042	100	0.354	100	0.397	100
KPIECE	0.060	100	1.771	100	0.617	100
BKPIECE	0.063	100	0.793	100	1.542	100
LBKPIECE	0.066	100	1.507	100	1.579	100
PDST	0.059	99	3.351	97	0.685	100
STRIDE	0.095	100	1.762	100	0.794	100
FMT	0.615	100	5.643	100	22.058	99

selecting the most popular motion planners for comparison during parameter sensitivity benchmarking. Each test in Section 3.4.1 consisted of 100 runs with a randomised environment and a planned timeout of 30 seconds. Planners' scores depend on assigned weights (1/6, 1/3, 1/2) for Scenarios 1, 2, and 3. For Section 3.4.2, each parameter *range* trial included 100 runs with a 30-second scheduled timeout, where a 0 mean time indicated all experiments failed. The mean time units in the experiments were seconds, and the success rate was expressed as a percentage. We conducted the experiment on a computer equipped with an Intel i9-11900 processor and an NVIDIA 3050. To preserve legibility while presenting the complete Franka, UR5, and KUKA comparison within a single portrait page, the per-scenario results tables (Tables 3.3–3.6) are necessarily typeset in a compact font; the narrower aggregated planner-score table (Table 3.7), which contains only two columns, is set at the normal document size for maximum readability.

3.4.1 Environment configuration benchmarking

The performance metrics derived from Tables 3.3, 3.5, and 3.6 reveal a striking distinction among the assessed planners in their ability to operate under diverse scenarios. Four planners RRTConnect, SBL, BKPIECE, and LBKPIECE exhibit impressive success rates across all three scenarios. The metric employed for evaluation is the normalised time efficiency score, calculated as a weighted average for each scenario. The resulting scores place RRTConnect (0.88) at the forefront, followed by SBL (0.82), LBKPIECE (0.75), and BKPIECE (0.04). Notwithstanding BKPIECE's high success rate, its planning time of approximately 10 seconds is noticeably

Table 3.4: Scenario 1, deterministic obstacle height 0.4 m. Mean planning time and success rate for each (planner, robot) combination, evaluated over 100 randomised instances under the same protocol as Table 3.3. The lowest mean time per robot is shown in bold.

	Franka		UR5		KUKA	
Planner	Time (s)	Succ. (%)	Time (s)	Succ. (%)	Time (s)	Succ. (%)
RRT	0.681	90	3.629	65	3.656	65
RRTConnect	**0.017**	100	**0.089**	100	**0.087**	100
RRTstar	30.002	88	30.071	46	30.045	62
TRRT	1.590	68	5.116	34	2.010	65
EST	0.125	100	2.074	98	1.054	100
SBL	0.052	100	0.387	100	0.506	100
KPIECE	0.155	100	1.852	98	1.075	100
BKPIECE	0.074	100	0.823	100	1.619	100
LBKPIECE	0.077	100	1.555	100	1.720	100
PDST	0.478	100	2.523	91	1.215	99
STRIDE	0.148	100	2.709	98	1.111	100
FMT	0.643	100	5.836	100	21.298	100

Table 3.5: Scenario 2 (semi-enclosed box). Mean planning time and success rate for each (planner, robot) combination, evaluated over 100 randomised instances under the same protocol as Table 3.3. The lowest mean time per robot among solvers achieving $\geq$95% success is shown in bold.

	Franka		UR5		KUKA	
Planner	Time (s)	Succ. (%)	Time (s)	Succ. (%)	Time (s)	Succ. (%)
RRT	2.196	9	7.935	43	0.048	47
RRTConnect	0.408	100	**1.047**	100	3.616	99
RRTstar	30.001	4	30.058	31	30.057	52
TRRT	0.592	30	2.343	26	0.069	48
EST	2.239	98	5.228	97	2.086	53
SBL	0.529	100	1.505	99	**5.762**	100
KPIECE	2.521	94	6.349	92	4.157	57
BKPIECE	0.975	100	3.883	98	9.398	85
LBKPIECE	**0.235**	100	2.351	99	4.916	100
PDST	1.154	100	6.587	76	2.119	53
STRIDE	1.967	97	6.256	87	2.727	60
FMT	1.989	98	6.240	100	8.476	66

longer than its counterparts, most apparent in clutter-dense scenarios such as 2 and 3. RRTConnect emerges as the most time efficient planner across these scenarios, trailed closely by SBL. LBKPIECE ranks third, while BKPIECE falls behind due to its longer planning time.

According to Tables 3.3, 3.5, and 3.6, planners such as PDST, STRIDE, EST, KPIECE, and FMT

Table 3.6: Scenario 3 (drawer-like enclosure). Mean planning time and success rate for each (planner, robot) combination, evaluated over 100 randomised instances under the same protocol as Table 3.3. The lowest mean time per robot among solvers achieving $\geq$95% success is shown in bold. Following the convention of Section 3.4, a mean time of 0 s indicates that all 100 trials failed.

	Franka		**UR5**		**KUKA**	
Planner	Time (s)	Succ. (%)	Time (s)	Succ. (%)	Time (s)	Succ. (%)
RRT	2.219	63	2.063	65	5.462	2
RRTConnect	0.092	100	**0.330**	100	7.351	79
RRTstar	30.002	50	30.023	55	30.005	1
TRRT	3.987	23	5.089	40	0.000	0
EST	2.419	95	4.255	88	15.719	5
SBL	0.156	100	0.488	100	6.100	91
KPIECE	1.084	84	4.039	85	15.166	6
BKPIECE	0.571	100	1.543	100	10.484	62
LBKPIECE	**0.086**	100	1.486	100	**4.650**	100
PDST	1.648	85	5.822	61	8.602	3
STRIDE	2.183	93	4.414	81	10.424	6
FMT	1.789	90	5.791	100	22.036	17

Table 3.7: Planner scores aggregated across all scenario, arm combinations, computed with the weighted scoring scheme of Section 3.3.5 ($w_1 = 1/6$, $w_2 = 1/3$, $w_3 = 1/2$). Higher is better; the top three planners are shown in bold.

Planner	**Score**
LBKPIECE	**0.99**
SBL	**0.98**
RRTConnect	**0.97**
BKPIECE	0.90
FMT	0.77
EST	0.76
STRIDE	0.74
KPIECE	0.73
PDST	0.68
RRT	0.44
TRRT	0.30
RRTstar	0.07

demonstrate moderate success rates across all scenarios. The performance ranking for this group of planners is PDST (0.98), STRIDE (0.90), EST (0.77), KPIECE (0.68), and FMT (0.00). PDST and STRIDE, in particular, display time efficiency almost matching that of the high-performing RRTConnect, SBL, BKPIECE, and LBKPIECE, and are even marginally quicker by 2

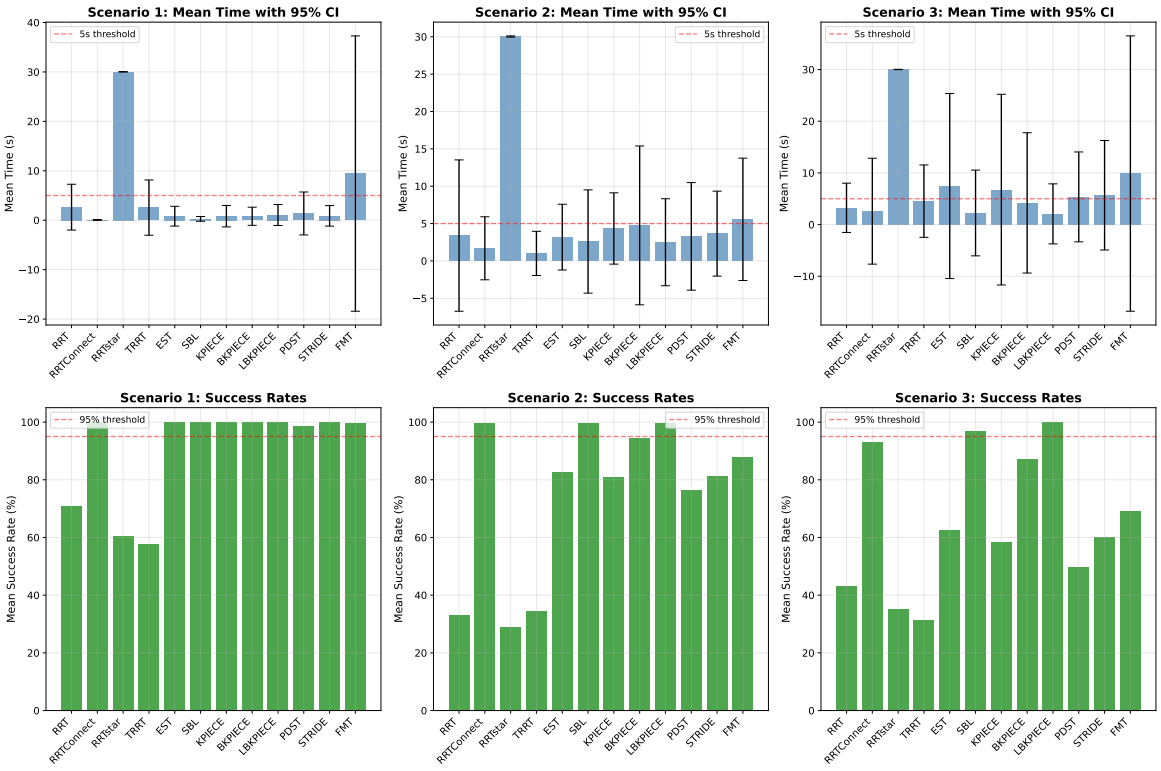


Figure 3.4: Comparative statistical analysis of the 12 motion planners across the three scenarios, mean planning times with 95% confidence intervals (top) and mean success rates across the three robot arms (bottom).

seconds in Scenario 2. This indicates that PDST and STRIDE remain competitive at intermediate spatial confinement levels defined here, following Section 3.1, as scenarios in which the reachable workspace is restricted by surrounding structure but not by an enclosure with a small access opening relative to the manipulator's link lengths. In our experiments, Scenario 2 instantiates this intermediate level with an enclosing box of internal dimensions $0.6\,\mathrm{m} \times 0.6\,\mathrm{m} \times 0.5\,\mathrm{m}$ and a single open face of $0.6\,\mathrm{m} \times 0.5\,\mathrm{m}$, which together restrict the manipulator to approximately 45% of the free workspace volume available in Scenario 1 while preserving full access from above and from one lateral direction (see Table 3.2 for a complete summary of the geometric properties of each scenario). While EST and KPIECE perform satisfactorily, securing third and fourth ranks respectively, FMT disappointingly underperforms, suggesting it may be less suited for these scenarios.

The Tables 3.3, 3.5, and 3.6 also highlight RRT, TRRT, and RRTstar, which demonstrate lower success rates in all scenarios. In this category, RRT (1.00) outperforms with the highest score, signifying its competency in handling low success rate scenarios. TRRT, while ranking second, does not quite match RRT's efficiency. RRTstar struggles with the lowest score, indicating it may be inadequate for such scenarios.

Tables 3.3 and 3.4 report results for two experimental conditions of Scenario 1 that differ only in the deterministic uniform height of the cylindrical obstacles placed on the surface: 0.3 m in the configuration of Table 3.3 and 0.4 m in the configuration of Table 3.4. The obstacle height is held fixed at the specified value for every obstacle in every randomised instance within a given configuration, while stochastic variation across runs is introduced solely by the MBM SE(3) pose perturbations described in Section 3.3.1. The two height values were chosen to span a regime in which the obstacles are tall enough to require non-trivial collision avoidance by all three manipulators considered (Franka, UR5, KUKA) but short enough that the workspace remains predominantly free, isolating the effect of obstacle scale on planner behaviour from the effect of overall workspace confinement examined in Scenarios 2 and 3. Across both height conditions, LBKPIECE, RRTConnect, and SBL maintained high success rates and time efficiency, while RRT, RRTstar, and TRRT showed substantial fluctuations in success rates and computation times. This pattern emphasises the importance of considering both the scenario geometry and the robotic arm when selecting a planner.

Considering planners' scores across all scenario-arm combinations (Table 3.7), LBKPIECE, SBL, RRTConnect, and BKPIECE consistently demonstrated the best efficiency and robustness across all scenarios. RRT, RRTstar and TRRT performed the worst, with the lowest efficiency and success rates. EST, KPIECE, PDST, STRIDE, and FMT showed varying success rates depending on the robot arm, with most planners exhibiting lower success rates for the KUKA robot arm.

The statistical analysis (Fig. 3.4) provides quantitative validation of the observed performance distinctions through rigorous confidence interval and variance metrics. In both panels of the figure, environmental complexity increases from Scenario 1 (minimal obstacles) to Scenario 3 (cluttered environment); the red dashed lines mark the 5-second planning-time threshold for real-time applications (top panel) and the 95% reliability threshold (bottom panel), and the error bars represent 95% confidence intervals computed using Student's t-distribution (n=3). Following standard practice in robotics benchmarking literature, we computed 95% confidence intervals to balance statistical rigour with practical interval widths given our sample size of three robot platforms per scenario. This confidence level, universally accepted in engineering research, indicates that we can be 95% certain the true population mean falls within the reported intervals, with only a 5% probability of Type I error. The results reveal critical insights about planner reliability across different configurations. RRTConnect demonstrates exceptional stability with a mean planning time of 0.051 seconds (95% CI: [0.014, 0.088]) in Scenario 1, maintaining consistently narrow confidence intervals across all scenarios, which indicates the highly predictable performance essential for industrial deployment. Similarly, SBL and LBKPIECE exhibit tight confidence intervals with mean times consistently below 2 seconds, confirming their operational reliability under varying environmental conditions. In contrast, RRTstar shows consistently poor performance with mean times approaching 30 seconds (95% CI: [29.9, 30.1]) across all scenarios, where the narrow confidence interval paradoxically confirms predictable inefficiency rather than

variable performance.

The variance analysis reveals another crucial dimension of planner behaviour that directly impacts real-world deployment decisions. Top-tier planners demonstrate remarkably low variance, with RRTConnect maintaining $\sigma^2 < 1.0$ across all scenarios, while middle-tier planners such as BKPIECE exhibit high variance ($\sigma = 4.23$ in Scenario 2) despite achieving comparable success rates. This variance differential becomes particularly pronounced in cluttered environments, where Scenario 3 results show dramatic performance degradation for most planners except the consistently performing top tier. The statistical significance of these performance differences was confirmed through one-way ANOVA tests ($p < 0.001$), establishing that the observed variations represent genuine algorithmic differences rather than random fluctuations. Furthermore, the Kruskal-Wallis non-parametric tests corroborate these findings, providing additional robustness to our statistical conclusions regardless of distribution assumptions. These comprehensive statistical measures transform our empirical observations into quantifiable reliability metrics, definitively establishing that only RRTConnect, SBL, and LBKPIECE consistently achieve the critical combination of sub-2-second planning times with 95% confidence and great success rates necessary for deployment in time-critical robotic applications. The statistical rigour applied here ensures that our recommendations for planner selection are grounded in robust quantitative evidence suitable for industrial decision-making.

Among all, RRTConnect, SBL, and LBKPIECE consistently illustrate superior robustness and efficiency, exhibiting scalability across varied clutter levels. They uphold high success rates and relatively low mean times, which positions them as suitable candidates for a wide range of environments. Despite this, BKPIECE, PDST, and STRIDE show potential and could benefit from further research. Conversely, RRT, RRTstar, and TRRT present less promising results, underlining the need for further optimisation of these planners.

3.4.2 Parameter sensitivity benchmarking

To complement the cross-planner comparison of Section 3.4.1, we now quantify how strongly each planner's performance depends on the choice of its principal tuning parameter. For all sampling-based planners considered in this thesis, this parameter is the OMPL *range* parameter, which limits the maximum extension step the local planner is permitted to add during tree or graph expansion in configuration space. The role of the *range* parameter and its mediation of the coverage resolution trade-off were introduced in detail in Section 3.1. We are concerned specifically with how planner performance varies as this parameter is swept across a controlled interval, and with how robust each planner is to changes in this single tuning choice. To make this concrete and to provide tuning guidance rather than defaults, we systematically sweep *range* from 0 to 3 in increments of 0.25 and evaluate seven widely used planners (EST, TRRT, SBL,

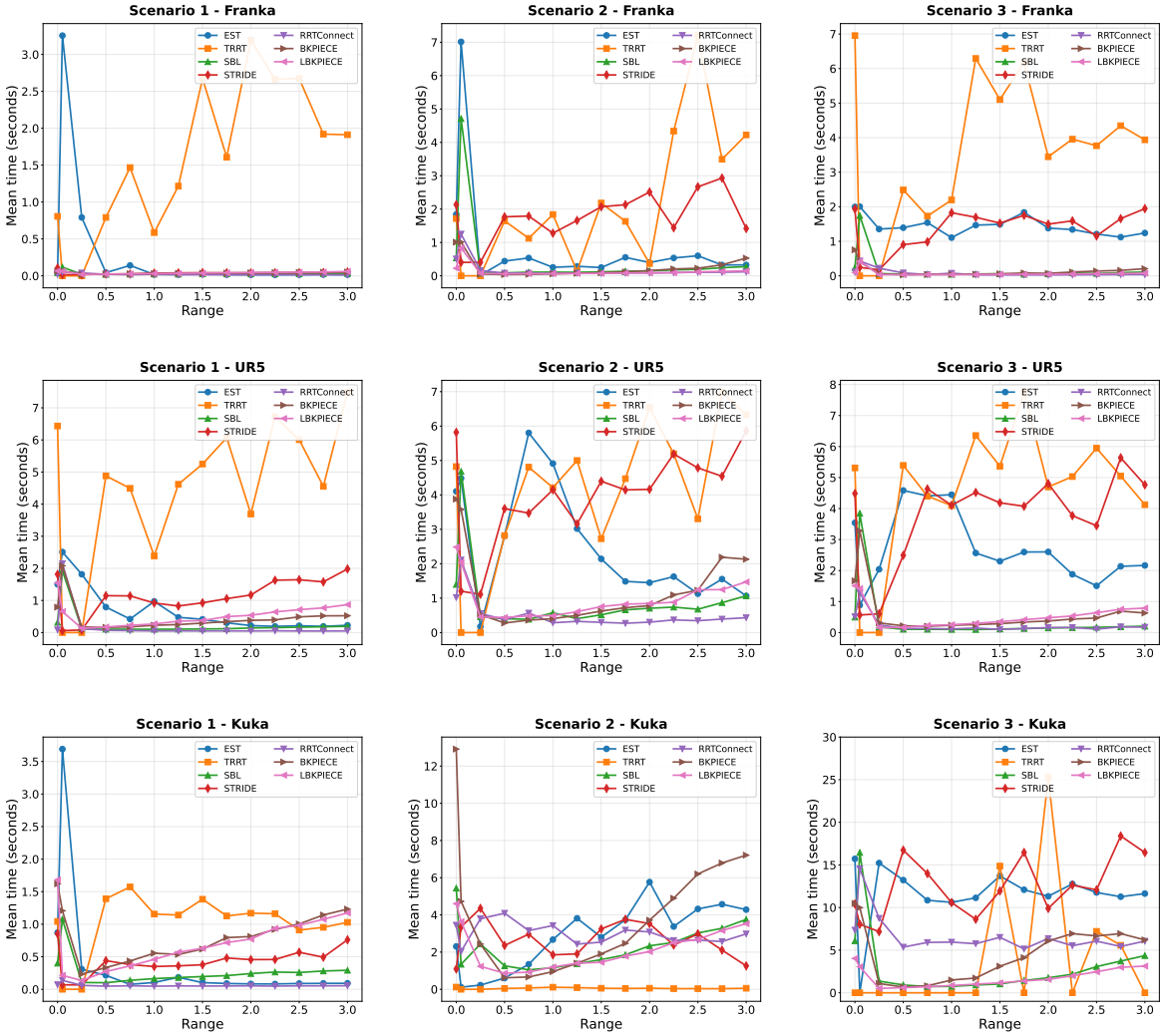


Figure 3.5: Mean planning time, in seconds, as a function of the *range* parameter for seven motion planners (EST, TRRT, SBL, STRIDE, RRTConnect, BKPIECE, and LBKPIECE) on the three robotic arms.

STRIDE, RRTConnect, BKPIECE, LBKPIECE) on three robotic arms (Franka, UR5, KUKA). For each (*range*, planner, robot) setting we run 100 trials with a 30 s timeout and record the mean planning time; a mean of 0 s indicates that all runs failed at that setting (i.e., 0% success).

In the benchmarking analysis illustrated in Fig. 3.5, the relative ranking of the top performing planners is arm-dependent. On the Franka and UR5, LBKPIECE, RRTConnect, and BKPIECE consistently deliver superior time efficiency and high success rates across all three scenarios, confirming the cross-planner ranking established in the single-shot evaluation of Section 3.4.1. On the KUKA, however, only LBKPIECE retains this robustness across all scenarios, achieving a 100% success rate even in the highly confined Scenario 3 (Table 3.6, 4.650 s mean time). RRTConnect and BKPIECE on the KUKA both exhibit markedly degraded performance in the more confined scenarios. RRTConnect drops to 79% success in Scenario 3 (7.351 s), and BKPIECE to 62% success (10.484 s), and several other planners that succeed on the Franka and UR5 collapse to single-digit success rates on the KUKA in Scenario 3 (Table 3.6). This asymmetric pattern is consistent with the kinematic differences between the three platforms. The KUKA arm used in this study is a 6-DOF manipulator, lacking the kinematic redundancy of the 7-DOF Franka, and its joint-limit configuration combined with its reach envelope leaves comparatively little manoeuvring volume in the most confined scenario; the result is that planners must thread relatively narrow corridors of feasible configurations, a regime in which lazy bidirectional projection-based methods such as LBKPIECE retain their advantage but in which simpler tree-based methods (RRTConnect) and non-lazy KPIECE variants (BKPIECE) lose their reliability. The 6-DOF UR5 does not exhibit the same degradation, suggesting that the issue is specific to the KUKA's reach envelope rather than to its degree of redundancy alone. SBL exhibits strong performance in the less confined scenarios and shows potential for further improvement with appropriately tuned *range* parameters, as discussed below. EST, TRRT, and STRIDE generally had higher mean times and lower success rates across all robotic arms. This result points to these planners' comparative inefficiency.

An additional observation was that the majority of the planners displayed lower mean times and higher success rates within the parameter *range* interval of 0.25 to 0.5. Most of the response curves appeared to stabilise regionally once the parameter value exceeded 1.5. This trend was even noticed for the typically unstable TRRT, which showed a decline in amplitude frequency beyond this point.

In summary, LBKPIECE, RRTConnect, and BKPIECE showcased robust and reliable performance across all robotic arms and scenarios, indicating their versatility. Although SBL required optimal *range* adjustments, it showed promise in specific conditions. In contrast, EST, TRRT, and STRIDE were generally less efficient. It is also important to note that performance varied across robotic arms, Franka, UR5, and KUKA, based on the planner used and the specific *range* parameters.

3.4.3 Discussion

LBKPIECE, RRTConnect, and BKPIECE consistently demonstrate time-efficient performance for all three robot arms in the tested scenarios, featuring low mean times, high success rates, and robust performance across various *range* parameter values. EST, and SBL could also be viable options, depending on the desired performance, but their effectiveness may vary depending on specific *range* values and environmental complexity. The influence of the *range* parameter on LBKPIECE, RRTConnect, and BKPIECE's performance is less pronounced in simpler cluttered environments, where these planners can often find feasible paths quickly, regardless of the *range* value. Conversely, for EST and SBL, a smaller *range* value may slow the planning process due to increased samples and connections, while a larger value could accelerate the convergence of trees.

In Scenario 1's open space, planners benefit from multiple path options, resulting in high timeliness and success rates, although obstacle configurations can still impact performance. Scenarios 2 and 3 illustrate that workspace size significantly affects robot arm motion planning. Restricted spaces constrain arm joint mobility, reducing both arm and planner performance in cluttered environments, where obstacles also impede computational efficiency. Nevertheless, fine-tuning the *range* parameter in complex environments can enhance planning performance and enable planners to thrive in otherwise challenging settings. Empirical studies suggest adjusting the *range* parameter to optimise performance across different environments, with a *range* of 0.25 to 0.5 typically yielding good performance. However, in constrained and cluttered environments like Scenario 3, increasing the parameter to 1.5 or higher has shown improved robustness and success rates. This adjustment should account for the specific characteristics and constraints of the environment. Further research is needed to investigate the impact of *range* parameter adjustments on performance in diverse environments.

The empirical findings of this chapter support the following conclusions. The 7-DOF Franka consistently achieves high success rates and competitive planning times across all three scenarios when paired with LBKPIECE, RRTConnect, or BKPIECE, making it the most reliable choice in our experiments for tasks that require navigation through both open and confined workspaces. The 6-DOF UR5 also achieves high success rates with the same three planners across all three scenarios, but its mean planning times are typically two to five times longer than those of the Franka under the same conditions (Tables 3.3–3.6), suggesting that its lack of kinematic redundancy increases the search effort required by sampling-based planners even when feasible solutions exist. The 6-DOF KUKA exhibits a markedly different pattern that performs comparably to the other arms in Scenario 1 (open workspace), but its success rates degrade substantially in Scenarios 2 and 3 with all planners other than LBKPIECE, as documented in detail in Section 3.4.2. The practical recommendation that follows from these measurements is that LBKPIECE is the most defensible default planner across all three platforms, while RRTConnect and BKPIECE are com-

petitive alternatives on the Franka and UR5 in less confined scenarios but become unreliable on the KUKA as workspace confinement increases. These conclusions concern the planning time and success rate behaviour measured in this study and then properties such as interface ergonomics, safety certification, and industrial integration, which often weigh in deployment decisions, are outside the scope of the present empirical evaluation.

3.5 Conclusion

This chapter has presented a comprehensive benchmarking study evaluating the performance of various motion planners across different robotic platforms in environments of varying complexity. Through systematic experimentation using the MotionBenchMaker tool, we have identified critical performance patterns that inform practical deployment decisions for robotic manipulation systems.

Our experimental results demonstrate that LBKPIECE, RRTConnect, and BKPIECE consistently achieve superior time efficiency and robustness across all tested robotic arms, with LBKPIECE achieving the highest overall score when considering weighted performance across all scenarios. The study reveals that certain planners consistently achieve superior time efficiency and robustness across all tested robotic arms, while others exhibit significant performance degradation as environmental complexity increases. The parameter sensitivity analysis establishes that planner performance depends on careful parameter tuning, with optimal values varying significantly based on environmental constraints and robot kinematics. These findings show that both the specific robotic platform and the operational environment need to be considered when selecting motion planning algorithms.

The benchmarking framework has provided valuable insights into the relationship between robot kinematics and planner performance. Robots with additional degrees of freedom demonstrated clear advantages when paired with appropriate planners, particularly in highly constrained scenarios. The analysis also reveals that while some platforms excel in standard industrial applications, they may exhibit limitations when deployed in complex manipulation tasks requiring navigation through cluttered or confined spaces. This observation has important implications for system design and robot selection in practical applications.

However, this comprehensive evaluation also reveals fundamental limitations inherent in current motion planning approaches. While sampling-based planners demonstrate effectiveness in specific scenarios, their performance remains heavily dependent on manual parameter tuning and expert knowledge for optimal configuration. The substantial variation in success rates across different environmental complexities indicates that no single planner provides a universally robust solution. Furthermore, the computational burden of exhaustive parameter optimisation for each

new scenario presents a significant barrier to practical deployment in dynamic industrial settings.

These limitations point to the need for alternative approaches that can draw on human expertise while maintaining mechanical feasibility. Rather than relying solely on algorithmic optimisation and manual parameter tuning, incorporating human demonstrations could provide valuable guidance for navigating complex manipulation tasks. This observation motivates our exploration in the following chapter of LfD techniques, specifically examining how human insights can be effectively transferred to robotic systems while ensuring kinematic compliance and mechanical constraints are respected. By combining the systematic understanding gained from our benchmarking study with the adaptability offered by demonstration-based learning, we aim to address the generalisation challenges that purely algorithmic approaches have yet to overcome.

Chapter 4

Enhancing LfD with DLS-IK and ProMPs

LfD techniques are invaluable for capturing complex human behaviours for robotic arm manipulations, yet they frequently encounter challenges such as avoiding singularities and respecting joint limits when directly applied to robotic systems. These challenges often lead to mechanical non-compliance, inaccuracies, and increased computational demands during control method adjustments. To address these issues, this study introduces a robust approach that integrates DLS-IK with ProMPs. By using DLS-IK to generate kinematically feasible trajectories, and embedding these within the ProMPs framework, our method not only ensures mechanical compliance but also benefits from the probabilistic modelling capabilities of ProMPs. This synergy addresses a significant gap in traditional LfD applications, aligning human demonstrations with the mechanical constraints of robotics, independent of the demonstrator's expertise. Our integrated approach refines the LfD process, enabling the generation of precise, reliable, and mechanically compliant movements in robotic arms, thereby reducing the typical inaccuracies and computational burdens associated with conventional LfD methods.

4.1 Introduction

The challenge of kinematic singularities is a prominent concern within the field of robotic manipulators [101, 102]. At such points, the robotic arm encounters configurations where it loses one or more degrees of freedom, complicating control over its movements. Traditionally, these issues are addressed using IK solutions [103–105]. However, the implementation of LfD techniques [1, 10, 88], where robots learn actions from human demonstrations, intensifies the difficulty of managing singularities. Human demonstrators often overlook the robotic arm's susceptibility

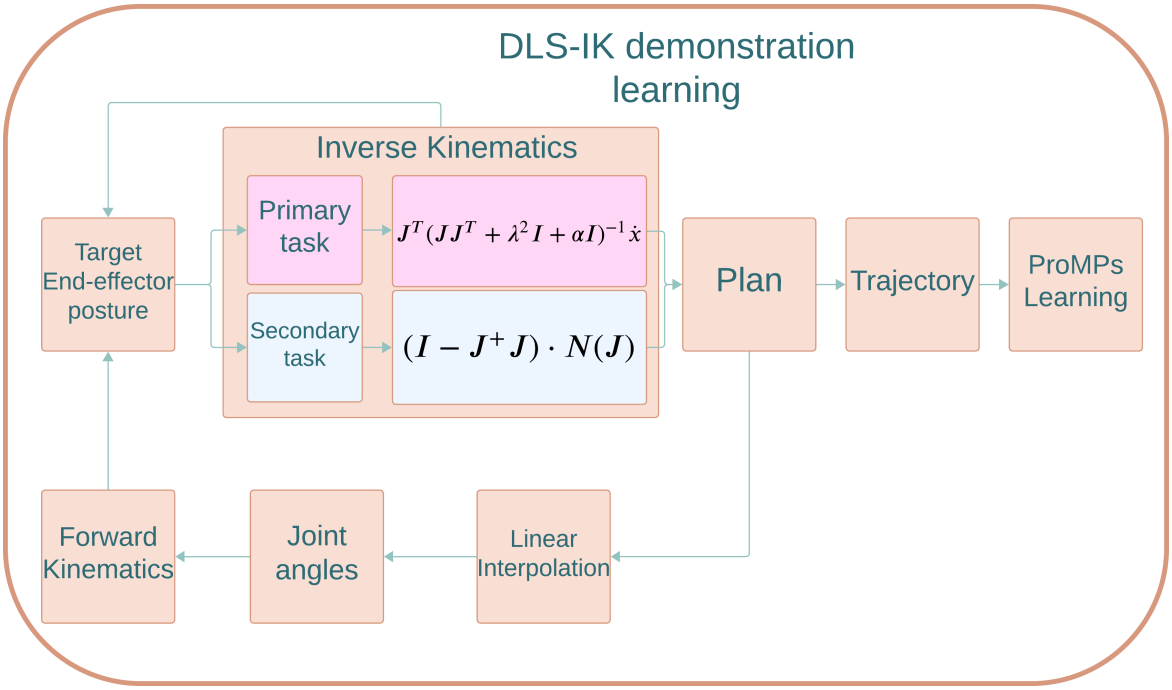


Figure 4.1: Overview of the proposed DLS-IK with ProMPs demonstration learning pipeline: an iterative IK cycle (FK, primary and secondary tasks, joint update) produces a kinematically feasible joint trajectory, which the Plan block resamples and passes to ProMPs learning. The pipeline stages are described in detail in Section 4.2.5.

to singularities and joint limits, which can lead to suboptimal guidance during demonstrations.

This study seeks to enhance LfD methods by addressing such inherent challenges when employing human demonstrations for training the robotic arm. Current LfD approaches may fail to consider the full mechanical constraints of robotics, resulting in increased computational complexity and suboptimal robotic performance. We introduce a novel integration of the DLS-IK (Fig. 4.1) [106] with ProMPs [13, 55], aimed at overcoming these limitations by ensuring both kinematically-feasible and mechanically-compliant trajectory learning.

The gap in traditional LfD methods lies in the alignment of human demonstrations with the mechanical and operational principles of robotic arms [107]. This misalignment often necessitates various improvised adjustments and control methods, which compromise the accuracy of the outcomes and increase the system's computational burden. Our approach uses the robust trajectory generation capabilities of DLS-IK to handle singularities and joint constraints effectively. Integrating these trajectories within the ProMPs framework utilises the probabilistic modelling strengths of ProMPs, while ensuring that the learning process adheres to mechanical feasibility.

It is necessary to visualise the problems on the physical platform used in this chapter, and to build upon this foundation for discussion. Figures 4.2, 4.3, and 4.4 illustrate three distinct singular or near-singular configurations of the 7-DoF Franka Emika Panda arm encountered during

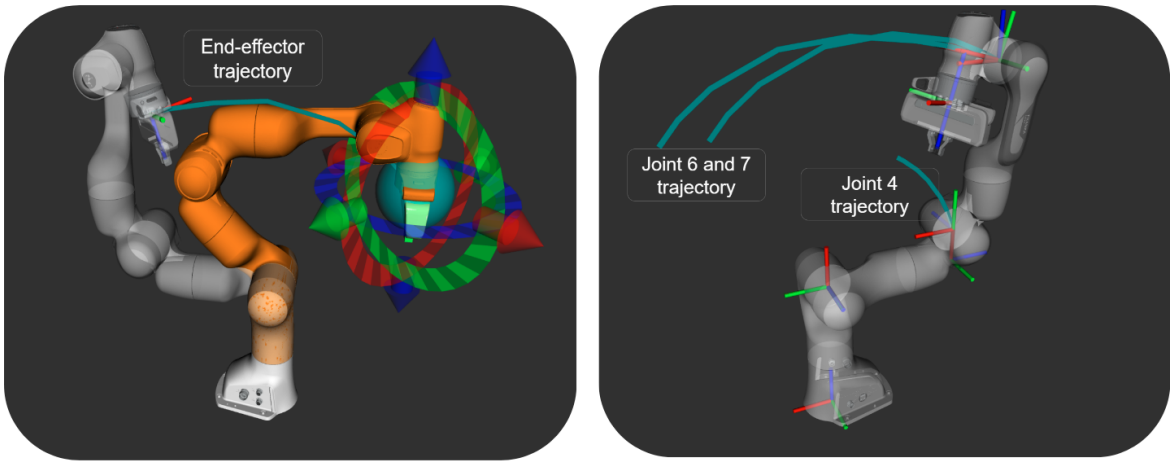


Figure 4.2: A representative singular configuration of the 7-DoF Franka Emika Panda arm encountered during a kinesthetic demonstration: (a) full-arm view, with the end-effector path traced in green; (b) close-up of joints 4, 6, and 7, whose individual trajectories are drawn separately.

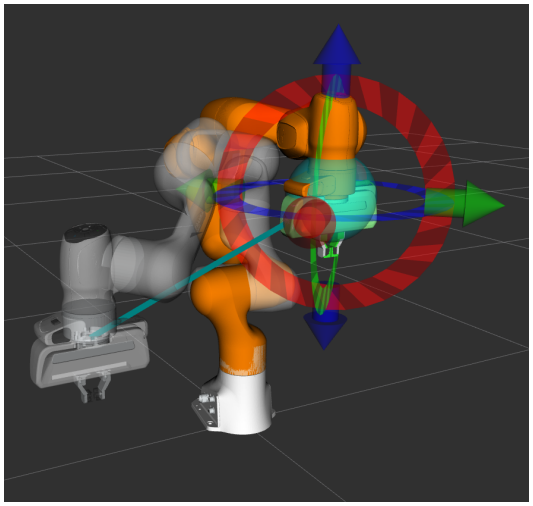


Figure 4.3: Wrist-alignment singularity: the axes of the final three joints become coplanar, so rotations about the wrist no longer span a full three-dimensional orientation space.

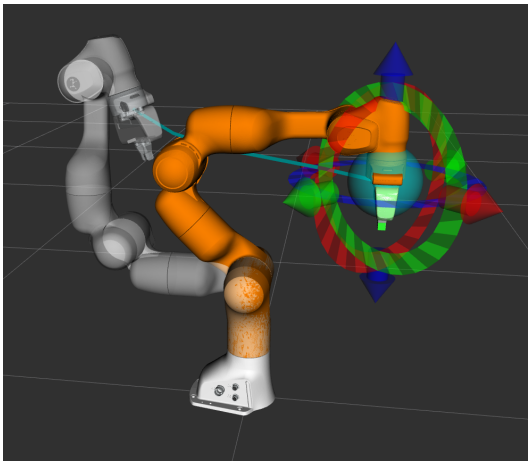


Figure 4.4: Elbow-extension singularity: the arm is fully extended, so the shoulder, elbow, and wrist lie almost in a straight line at the boundary of the reachable workspace.

preliminary kinesthetic demonstrations. Figure 4.2 shows a generic case in which the arm has reached a configuration where two consecutive revolute axes have become near-collinear, so motion of the affected joints produces no independent end-effector displacement that the end-effector trajectory remains well-defined whilst the underlying joint trajectories of joints 4, 6, and 7 lose their kinematic independence, collapsing onto coincident or near-parallel paths that signal rank deficiency of the manipulator Jacobian at this pose. This is the situation that gives rise to

the unstable joint velocities analysed quantitatively in section 4.3.1. Figures 4.3 and 4.4 depict two structurally distinct singularity classes that recur throughout the demonstration corpus: a wrist-alignment singularity, in which the axes of the final three revolute joints become coplanar (Fig. 4.3), and an elbow-extension singularity, in which the manipulator approaches full extension and the redundant degree of freedom collapses (Fig. 4.4). In the latter case the end-effector pose appears unremarkable from a task-space perspective, yet the Jacobian becomes severely ill-conditioned and any further commanded motion in the direction of extension cannot be realised. Both classes are routinely produced by human demonstrators who, as noted earlier, guide the end-effector towards task-relevant goals without regard for the manipulator's internal kinematic state. Each class causes classical pseudoinverse-based IK to fail in a characteristic way: wrist-alignment singularities produce unbounded joint velocities about the wrist axes, whereas elbow-extension singularities produce ill-conditioned solutions accompanied by erratic torque signatures. The DLS-IK formulation developed in section 4.2 is designed to remain stable in both regimes, and the experiments of section 4.3.1 quantify this stability on instances drawn from each class.

In summary, our contributions are: (1) An advanced DLS-IK method that not only controls velocity damping but also adheres to joint constraints through the null space principle of the robotic arm. (2) Incorporation of this refined DLS-IK approach into the ProMPs framework, which significantly enhances the robot's ability to learn and generate motion trajectories that are both mechanically compliant and efficient, effectively addressing singularity issues and joint limitations. (3) An investigation into the impact of the Jacobian condition number in the DLS-IK process on the stability and robustness of robotic arm movements, particularly near singularities. Our findings demonstrate improved stability and velocity control via local optimal parameter selection.

4.2 Methods

4.2.1 Inverse Kinematics

IK [108] is a crucial concept in robotics, used for determining the necessary joint parameters that achieve a desired position of the robot's end-effector. This process is the inverse of FK, where the position and orientation of the end-effector are computed given the joint parameters. The complexity of the IK equations can vary significantly depending on the robot's architecture (e.g., the number of joints, the type of joints, the geometry of the linkages).

Analytical methods involve solving the equations derived from the robot's geometry algebraically. These methods can provide exact solutions but may not be feasible for complex robots due to the mathematical difficulty of solving the equations.

For robots with multiple degrees of freedom, a common approach is to use the Jacobian matrix, which relates the velocities of the robot's joints to the velocity of the end-effector. The IK problem can be solved by calculating the inverse (or pseudo-inverse) of the Jacobian matrix:

$$\dot{x} = \mathbf{J}\dot{\theta} \tag{4.1}$$

where $\dot{\theta}$ denotes the vector of joint velocities, $\mathbf{J}$ is the Jacobian matrix, and $\dot{x}$ represents the Cartesian velocity of the end effector. This iterative method helps approximate joint angles for complex robotic arms.

4.2.2 Damped Least Squares

The DLS method modifies the basic least squares solution by adding a damping term to improve stability and handle singular configurations [109]. The Jacobian matrix ($\mathbf{J}$) of the robot, which relates changes in joint angles to changes in end effector position and orientation, plays a central role in this method.

Given a desired change in end effector position (Δx), the change in joint angles ($\Delta\theta$) can be approximated using the pseudo-inverse of the Jacobian:

$$\Delta\theta = \mathbf{J}^{\dagger}\Delta x \tag{4.2}$$

However, directly using the pseudo-inverse can lead to large joint velocities in the vicinity of singularities. To mitigate this, the DLS method introduces a damping factor (λ), modifying the formula to:

$$\Delta\theta = (\mathbf{J}^T\mathbf{J} + \lambda^2\mathbf{I})^{-1}\mathbf{J}^T\Delta x \tag{4.3}$$

where $\mathbf{J}^T$ is the transpose of the Jacobian, $\mathbf{I}$ is the identity matrix, and λ is the damping factor, a small positive constant. This addition ensures that the solution is less sensitive to singularities and results in smoother motion.

Additionally, the relationship between the damping factor λ and the condition number of the Jacobian $\kappa(\mathbf{J})$ in the context of the DLS is critical for maintaining the stability of the robotic arm's movements, especially near singularities.

$$\kappa(\mathbf{J}) = \frac{\sigma_{\mathsf{max}}}{\sigma_{\mathsf{min}}} \tag{4.4}$$

where σ_{max} and σ_{min} are the maximum and minimum singular values of the Jacobian matrix $\mathbf{J}$, respectively. High condition number of Jacobian ($\kappa(\mathbf{J})$ large) indicates that the Jacobian matrix is near singular or ill-conditioned. To reduce the influence of these instabilities, the larger λ provides more damping, which effectively regularises the pseudo-inverse by reducing the contribution of directions associated with small singular values. This helps to prevent excessively large joint velocities and torques.

Low condition number of Jacobian ($\kappa(\mathbf{J})$ small) indicates that the Jacobian matrix is well-conditioned. A smaller λ allows the control to more precisely follow the desired end effector path or force application.

The damping factor λ, together with the additional regulariser α introduced in the null-space formulation of Section 4.2.4, is held fixed throughout the experiments of this chapter at $\lambda = 0.01$ and $\alpha = 0.01$. Because the primary-task solve takes the form $(\mathbf{J}\mathbf{J}^\top + \lambda^2\mathbf{I} + \alpha\mathbf{I})^{-1}$ (Eq. 4.8), the two parameters combine into an effective isotropic damping of $\sqrt{\lambda^2 + \alpha} = \sqrt{10^{-4} + 10^{-2}} \approx 0.10$, which sits at the upper end of the conventional $0.01 \leq \lambda \leq 0.1$ range; note that $\alpha = 0.01$ dominates $\lambda^2 = 10^{-4}$, so the additive regulariser in fact supplies most of the damping.

This operating point was selected, rather than merely assumed, through a sensitivity analysis of the accuracy–stability trade-off along the elbow-singularity approach of Section 4.3.1, where the Jacobian condition number rises steeply. Table 4.1 and Fig. 4.5 report, as the effective damping is swept, the peak joint velocity reached during the approach (the stability indicator) against the residual error in task-space velocity tracking (the accuracy cost). An effective damping of ≈ 0.10 is the smallest value that holds the peak joint velocity below the Franka joint-velocity limit of $2.175\ \text{rad/s}$ (reaching $1.79\ \text{rad/s}$); weaker damping ($\lesssim 0.06$) leaves the joint velocity above the limit and therefore mechanically infeasible near the singularity, whereas stronger damping trades tracking accuracy for stabilisation that is no longer required. The fixed value is thus a deliberate, conservative operating point justified by the manipulability characteristic, while the condition number adaptive damping developed in Chapter 5 is its principled successor.

It is worth addressing here why the standard DLS formulation, rather than the more refined SDLS variant introduced in section 2.3, was selected as the IK backbone for the integration with ProMPs. Three considerations motivated this choice. First, the integration target a probabilistic movement primitive framework whose weights $\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$ already encode trajectory variability is most cleanly coupled with a deterministic IK module whose regularisation is governed by a single, interpretable scalar parameter; a per-direction damping schedule introduces a second source of stochasticity-like behaviour through threshold transitions and complicates the analysis of how IK regularisation propagates into the learned weight distribution. Second, the demonstration learning cycle generates a large number of IK queries per training episode (5,600 joint-angle samples for the configuration reported in section 4.3.2), and the SVD cost incurred per iteration by SDLS is poorly amortised in this offline, batch usage pattern, where DLS's closed-form

Table 4.1: Accuracy stability trade-off of the damped least squares solver along the elbow-singularity approach of Section 4.3.1, as a function of the effective isotropic damping $\lambda_{\text{eff}} = \sqrt{\lambda^2 + \alpha}$. The peak joint velocity is the stability indicator (it must remain below the Franka limit of 2.175 rad/s) and the task-velocity residual $\|\dot{\mathbf{x}} - \mathbf{J}\dot{\mathbf{q}}\|$ is the accuracy cost. The value used throughout this chapter ($\lambda_{\text{eff}} \approx 0.10$, from $\lambda = \alpha = 0.01$) is the smallest damping that keeps the peak joint velocity within the limit.

λ_{eff}	peak joint velocity (rad/s)	task-velocity residual
0 (undamped)	72.44	0.000
0.03	6.62	0.110
0.06	3.16	0.177
0.10 (used)	**1.79**	**0.236**
0.20	0.91	0.321
0.50	0.33	0.416

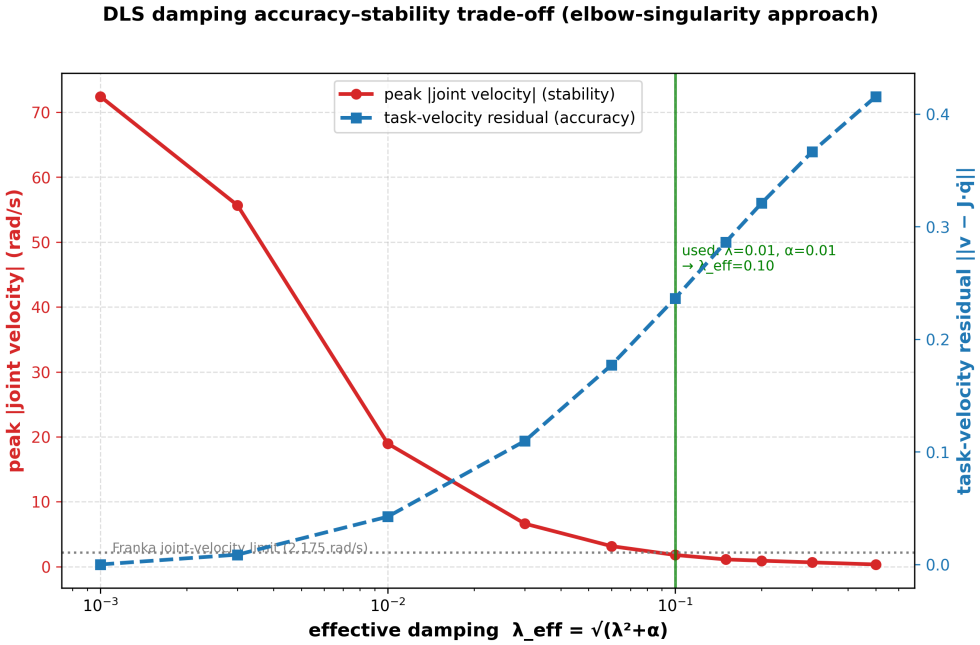


Figure 4.5: Damping accuracy–stability trade-off along the elbow-singularity approach: peak joint velocity (red, left axis) and task-velocity tracking residual (blue, right axis) as functions of the effective damping λ_{eff} (logarithmic axis). The dotted line marks the Franka joint-velocity limit and the vertical line the operating point used in this chapter.

update is preferable. Third, and most importantly from a thesis-architectural standpoint, the contribution of Chapter 4 is the integration of an IK-based feasibility layer with a learning-based motion-primitive framework, not the refinement of the IK solver itself; introducing SDLS at this stage would have entangled two distinct contributions and obscured which performance gains stem from kinematic regularisation versus from the LfD integration. The natural place for directional damping and adaptive regularisation is therefore Chapter 5, where SDLS-IK serves as one of the principal benchmark comparators against which AWARE-IK is evaluated, and where the

interaction between regularisation strategy and multi-objective optimisation is examined directly.

4.2.3 Selectively Damped Least Squares

While the DLS method introduced in the previous section successfully prevents numerical divergence near singularities, it does so at a measurable cost: the scalar damping factor λ regularises all directions of motion uniformly, including those that are well-conditioned and require no stabilisation. The consequence is a systematic loss of tracking accuracy along directions where the manipulator retains full mobility, manifesting as steady-state Cartesian error even when feasible solutions exist. This isotropic treatment of an inherently anisotropic problem motivated the development of the SDLS, which redistributes damping in a direction-dependent manner through explicit spectral decomposition of the Jacobian. The foundation of SDLS lies in the Singular Value Decomposition (SVD) of the manipulator Jacobian:

$$\mathbf{J} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^\top \tag{4.5}$$

where $\mathbf{U} \in \mathbb{R}^{m\times m}$ and $\mathbf{V} \in \mathbb{R}^{n\times n}$ are orthogonal matrices whose columns $\mathbf{u}_i$ and $\mathbf{v}_i$ represent the task-space and joint-space singular vectors, respectively, and $\mathbf{\Sigma}$ contains the singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ ordered by magnitude. Each singular value σ_i quantifies the gain along a corresponding pair of singular directions that small singular values identify near-singular directions where small Cartesian commands induce disproportionately large joint velocities, while large singular values correspond to directions of well-conditioned mobility.

Whereas DLS computes the joint update through the uniformly regularised inverse

$$\dot{\mathbf{q}}_{\text{DLS}} = \mathbf{J}^\top(\mathbf{J}\mathbf{J}^\top + \lambda^2\mathbf{I})^{-1}\dot{\mathbf{x}} \tag{4.6}$$

SDLS reformulates the update as a sum of per-direction contributions, applying damping to each spectral component independently:

$$\dot{\mathbf{q}}_{\text{SDLS}} = \sum_{i=1}^{r} \gamma_i \frac{\sigma_i}{\sigma_i^2 + \lambda_i^2} \mathbf{v}_i(\mathbf{u}_i^\top \dot{\mathbf{x}}) \tag{4.7}$$

where λ_i is a per-direction damping factor and $\gamma_i \in [0, 1]$ is a clamping coefficient that limits the magnitude of each component when the requested motion exceeds achievable bounds. The damping factor λ_i is typically scheduled as a function of σ_i, for instance, $\lambda_i = \lambda_{\max}(1-\sigma_i/\sigma_{\text{thresh}})$ for $\sigma_i < \sigma_{\text{thresh}}$ and $\lambda_i = 0$ otherwise, so that well-conditioned directions are inverted essentially

without regularisation while ill-conditioned directions receive aggressive damping. The clamping term γ_i further prevents any single component from dominating the joint velocity vector, which is particularly valuable when the desired end-effector velocity has components that exceed the manipulator's instantaneous reachability.

The practical effect of this selective regularisation is twofold. First, tracking accuracy along well-conditioned directions is preserved at near-undamped levels, eliminating the systematic Cartesian residual that uniform damping introduces. Second, the joint velocity through singular configurations becomes smoother because damping is engaged gradually as σ_i decreases, rather than being applied as a constant offset throughout the motion.

These advantages do not come without a cost. SDLS requires a full SVD at every control step, raising the per-iteration computational complexity from $\mathcal{O}(n^3)$ for DLS's matrix inversion to approximately $\mathcal{O}(\min(m,n)^2 \max(m,n))$ for the decomposition. While this remains tractable for typical 6 and 7-DOF manipulators, it is a non-trivial overhead for high-frequency control cycles or for trajectory generators that solve thousands of IK problems per planning episode. A second concern is parameter sensitivity that SDLS replaces DLS's single scalar λ with a damping schedule $\{\lambda_i\}$, a singularity threshold σ_{thresh}, and clamping bounds, each of which must be tuned to the manipulator and the task. Empirically, this expanded parameter space can produce non-smooth behaviour across the threshold value of σ_{thresh}, where a singular value crossing the threshold causes a discontinuous change in the damping applied to its direction.

4.2.4 Controlling in the null space

The following general terms are used throughout this section. Let n denote the number of actuated joints of the manipulator and m the dimension of the task space, with $m = 6$ when the task is full pose tracking (three positional and three rotational components). For the 7-DoF Franka used in all experiments of this chapter, $n = 7$ and $m = 6$. The joint configuration is $q \in \mathbb{R}^n$ with corresponding joint velocity $\dot{q} \in \mathbb{R}^n$; the end-effector pose lives on the manifold $SE(3)$, and we write $\dot{x} \in \mathbb{R}^m$ for its instantaneous twist. The manipulator Jacobian $\mathbf{J} \in \mathbb{R}^{m \times n}$ is the linear map between these two velocity spaces, $\dot{x} = \mathbf{J}\dot{q}$. We adopt the convention $\mathbf{I}_k \in \mathbb{R}^{k \times k}$ for the identity matrix of size k, so that $\mathbf{I}_m$ and $\mathbf{I}_m$ refer unambiguously to identity matrices acting in task space and joint space respectively.

Controlling in the null space refers to a technique used to achieve a specific task (the primary task), while simultaneously satisfying additional constraints or optimising secondary objectives to achieve some secondary task [110]. The null space refers to the space of motions that do not move the end effector and hence do not affect the primary task.

In this approach, we first calculate the joint velocity required for the primary task (e.g., reaching

a target end effector position and orientation) using the DLS method. This method modifies the pseudo-inverse calculation of the Jacobian to include damping factors that help prevent numerical instabilities and large joint velocities near singularities. The formula for the primary task joint velocity $\dot{q}_{primary} \in \mathbb{R}^n$ using DLS is:

$$\dot{\mathbf{q}}_{\text{primary}} = \mathbf{J}^\top(\mathbf{J}\mathbf{J}^\top + \lambda^2\mathbf{I}_m + \alpha\mathbf{I}_m)^{-1}\dot{\mathbf{x}} \tag{4.8}$$

$\dot{\mathbf{q}}_{\text{primary}} \in \mathbb{R}^n$, joint-velocity command for the primary task; $\mathbf{J} \in \mathbb{R}^{m\times n}$, manipulator Jacobian, with $\mathbf{J}^T \in \mathbb{R}^{n\times m}$ its transpose; $\dot{\mathbf{x}} \in \mathbb{R}^m$, desired end-effector twist in Cartesian (task) space; $\mathbf{I}_m \in \mathbb{R}^{m\times m}$, identity matrix acting in task space, this is distinct from the joint-space identity $\mathbf{I}_n$ that will appear in the secondary-task formulation in Eq. 4.9; $\lambda \in \mathbb{R}_{>0}$, Damping factor (typically $0.01 \leq \lambda \leq 0.1$) that prevents numerical instabilities near singularities by regularising the inversion of $\mathbf{J}\mathbf{J}^T$; $\alpha \in \mathbb{R}_{\geq 0}$, additional regularisation parameter penalising joint-velocity magnitude.

Dimensionally, the product $\mathbf{J}\mathbf{J}^T \in \mathbb{R}^{m\times m}$, so the bracketed quantity to be inverted is $m \times m$ and its inverse acts on the task-space residual $\dot{\mathbf{x}} \in \mathbb{R}^m$; left-multiplication by $\mathbf{J}^T \in \mathbb{R}^{n\times m}$ then projects the result back into joint space, yielding $\dot{\mathbf{q}}_{\text{primary}} \in \mathbb{R}^n$ as required. The damping factor λ specifically addresses the singularity problem by ensuring that $(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}_m)$ remains invertible even as $\mathbf{J}\mathbf{J}^T$ approaches rank deficiency. A larger λ provides more robust numerical behaviour at the cost of tracking accuracy. The supplementary parameter α affords further control over joint-velocity magnitude: larger values produce smaller commanded velocities and lower energy expenditure, at the cost of slower convergence.

The damping factor λ specifically addresses the singularity problem by ensuring the matrix $(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}_m)$ remains invertible even when $\mathbf{J}\mathbf{J}^T$ becomes singular. A larger λ value provides more robust numerical behaviour but may reduce tracking accuracy. The additional parameter α provides further control over joint velocity magnitudes, where larger values result in smaller joint velocities and potentially lower energy consumption, though this may slow convergence of the IK solution.

The secondary task in our implementation minimises the deviation of each joint's angle from the midpoint of its motion range. To prevent this secondary task from interfering with the primary task, the secondary joint velocity $\dot{q}_{secondary} \in \mathbb{R}^n$ is projected onto the null space of the Jacobian. The null space comprises all joint velocities that do not influence the movement of the end effector [102]. The projection of $\dot{q}_{secondary}$ onto the null space is given by:

$$\dot{q}_{null} = (\mathbf{I}_n - \mathbf{J}^+\mathbf{J}) \cdot \dot{q}_{secondary} \tag{4.9}$$

where: $\dot{\mathbf{q}}_{\text{null}} \in \mathbb{R}^n$, null-space projected component of the secondary-task velocity; $\mathbf{I}_n \in \mathbb{R}^{n\times n}$, identity matrix acting in joint space (distinct from $\mathbf{I}_m$); $\mathbf{J}^+ = \mathbf{J}^T(\mathbf{J}\mathbf{J}^T+\lambda^2\mathbf{I}_m+\alpha\mathbf{I}_m)^{-1} \in \mathbb{R}^{n\times m}$, damped pseudoinverse, consistent with Eq. 4.8; $\dot{\mathbf{q}}_{\text{secondary}} \in \mathbb{R}^n$, joint velocity computed for the secondary task before projection.

The product $\mathbf{J}^+\mathbf{J} \in \mathbb{R}^{n\times n}$ projects any joint-space vector onto the row space of $\mathbf{J}$ (i.e. the directions that do affect the end-effector); consequently $(\mathbf{I}_n - \mathbf{J}^+\mathbf{J}) \in \mathbb{R}^{n\times n}$ projects onto the null space of $\mathbf{J}$, guaranteeing by construction that $\mathbf{J}\dot{\mathbf{q}}_{\text{null}} = \mathbf{0}$ and so the secondary task cannot perturb the end-effector motion delivered by the primary task.

The matrix product $\mathbf{J}^+\mathbf{J}$ projects any vector onto the column space of $\mathbf{J}$, and thus $(\mathbf{I}_n - \mathbf{J}^+\mathbf{J})$ projects onto the null space, ensuring that $\dot{q}_{null}$ does not affect the end-effector motion.

The final joint velocities combine contributions from both tasks, prioritising the primary task while incorporating the secondary task within the available degrees of freedom:

$$\dot{q}_{final} = \dot{q}_{primary} + \dot{q}_{null} \tag{4.10}$$

where $\dot{q}_{final} \in \mathbb{R}^n$ represents the final joint velocity command sent to the robot. This formulation ensures that the primary task objectives are met exactly, while the secondary task objectives are achieved to the extent possible without interference.

4.2.5 DLS-IK

The DLS-IK demonstration learning framework summarised in Fig. 4.1 takes two inputs, a desired target end-effector pose $\mathbf{x}_{\text{target}} \in SE(3)$ and the manipulator's initial joint configuration $\mathbf{q}_0 \in \mathbb{R}^n$, and produces, as output, a probabilistic trajectory model $p(y(t))$ encoded by ProMPs. Between these two endpoints, the framework operates in two distinct phases that share a common joint-space representation but run on different time scales. The first phase is the iterative IK solver of Fig. 4.1, in which FK, the primary and secondary tasks, and the joint update repeatedly close a feedback cycle until the Cartesian residual against $\mathbf{x}_{\text{target}}$ falls below tolerance; this phase produces a discrete sequence of joint configurations $\{\mathbf{q}_k\}_{k=0}^{T}$ that is kinematically feasible by construction. The second phase is a one-off that the Plan block resamples this sequence onto a uniform time grid via linear interpolation in joint space, and the resulting time-parameterised trajectory is encoded by ProMPs as a distribution over basis-function weights, yielding a generative model from which novel mechanically compliant trajectories can subsequently be drawn. It begins with the target end effector posture, which serves as the desired outcome of the demonstration learning. The FK module provides the necessary context by predicting the end-effector's position from current joint angles.

Before tracing the operations step by step, we make explicit the design rationale that motivates the primary and secondary decomposition, since the motivation for splitting the task clarifies the mechanical description that follows. The choice is not stylistic but a deliberate exploitation of the manipulator's redundancy that the Franka robot possesses seven actuated joints whilst pose tracking imposes only six task space constraints, leaving a one-dimensional self-motion manifold along which the arm can reconfigure without disturbing the end-effector. A naïve damped pseudoinverse leaves this redundancy unused, and over long trajectories the solver tends to drift towards joint limits or towards configurations of poor manipulability. Splitting the task allows the primary component to deliver Cartesian accuracy through the damped pseudoinverse, while the secondary component spends the redundant DoF on a chosen ergonomic objective that in our case, regulation towards the centre of each joint's range projected onto the Jacobian's null space so that it cannot, by construction, interfere with task-space tracking. The role of the Plan block is then purely to render the discrete IK output suitable for downstream learning, linear interpolation in joint space resamples the sequence onto a uniform time grid, producing the kinematically feasible, time-parameterised trajectory that ProMPs encodes. The forward arrow from Plan to ProMPs Learning in Figure 4.1 is therefore a one-shot transfer; the apparent feedback path through FK that a casual reading of the schematic might suggest exists only inside the iterative phase and is closed before the Plan block is reached.

Concretely, the iterative phase proceeds as follows. The framework is initialised with the target end-effector pose $\mathbf{x}_{\text{target}}$ and the starting joint configuration $\mathbf{q}_0$. At each iteration k, FK is evaluated at the current joint state $\mathbf{q}_k$ to obtain the present end-effector pose; differencing this against $\mathbf{x}_{\text{target}}$ produces the Cartesian residual that drives the joint-velocity update. The update itself is decomposed into two contributions. The primary task uses the damped least squares formulation of Eq. 4.8 to compute the joint velocity that drives the end-effector towards the target whilst remaining stable near singularities. The secondary task, computed independently, biases the joints towards the centre of their motion ranges; it is then projected onto the null space of the Jacobian via Eq. 4.9 so that, by construction, it cannot perturb the primary task. The two contributions are summed in Eq. 4.10 to yield the final joint-velocity command $\dot{\mathbf{q}}_{\text{final}}$, which is integrated to obtain $\mathbf{q}_{k+1}$ and then enters back to FK closing the cycle. Iteration continues until the Cartesian residual falls below tolerance, at which point the accumulated joint sequence $\{\mathbf{q}_k\}_{k=0}^{T}$ is handed to the downstream encoding phase, linear interpolation in joint space resamples it onto a uniform time grid, and ProMPs encodes the resulting time-parameterised trajectory as a distribution over basis function weights, supporting smooth reproduction and conditioning on novel task contexts.

Throughout this iterative process, the robot's commanded motion is continuously refined by the interplay of primary-task tracking, null-space regulation, and the damping mechanism of Eq. 4.8. Once the framework converges and the resulting trajectory is encoded by ProMPs, the learned model can be queried to generate new trajectories that respect the same kinematic

constraints as the demonstration whilst adapting to perturbed via-points or revised goal poses. The mathematical machinery underlying the ProMPs encoding step is set out in section 4.2.6.

4.2.6 Probabilistic Movement Primitives

In the robotic motion planning and control, ProMPs [54] are a key methodology for encapsulating complex motor skills in a probabilistic framework. This approach significantly diverges from deterministic trajectory representations, opting instead to model movements as distributions over trajectories. Such a probabilistic representation not only accommodates for inherent execution variability but also adeptly navigates uncertainties, thereby ensuring robust adaptability to dynamic environmental interactions and alterations in task objectives. ProMPs learn from multiple task demonstrations to form a cohesive probabilistic model that encodes the movement's fundamentals and permissible variations.

Movement trajectory $\mathbf{y}(t)$ at any given time t is expressed as:

$$\mathbf{y}(t) = \boldsymbol{\phi}(t)^T \mathbf{w} \tag{4.11}$$

where $\boldsymbol{\phi}(t)$ denotes the chosen basis functions, such as Gaussian basis or polynomials, and $\mathbf{w}$ represents the weights that encode specific movement patterns.

ProMPs envisage the weights $\mathbf{w}$ as random variables adhering to a Gaussian distribution for simplicity:

$$\mathbf{w} \sim \mathcal{N}(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w) \tag{4.12}$$

This distribution encapsulates movement variability, with $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ denoting the mean and covariance matrix of the weights, respectively.

The objective is to ascertain the parameters $\boldsymbol{\mu}_w$ and $\boldsymbol{\Sigma}_w$ that most aptly characterise the demonstrated movements. Techniques such as maximum likelihood estimation or Bayesian inference are employed, integrating prior knowledge regarding the weights.

ProMPs excel in conditioning on observed trajectory segments or desired outcomes, facilitating movement adaptation. This is accomplished through Bayesian conditioning, which updates the weight distribution based on new evidence:

$$p(\mathbf{w}|\mathbf{y}_{\text{obs}}) \propto p(\mathbf{y}_{\text{obs}}|\mathbf{w})p(\mathbf{w}) \tag{4.13}$$

The revised distribution is instrumental in predicting likely movement completions or adaptations to novel circumstances. New movements are synthesised by sampling from the $\mathbf{w}$ distribution, followed by trajectory computation using the basis functions.

The variability that this Gaussian distribution over weights is designed to capture is of three principal kinds, all of which arise routinely in kinesthetic teaching of a 7-DoF manipulator. The first is trajectory variability, across repeated demonstrations of the same task, the operator typically traces end-effector paths that differ in shape, for instance, reaching towards a target along a slightly higher or lower arc, even when the task constraints (start pose, end pose, obstacle avoidance) are identical. The second is temporal variability, the same operator, or different operators, will execute the same gesture at different paces, with non-uniform local speed, leading to phase variation between otherwise similar trajectories. The third is execution variability, small perturbations within a single demonstration arising from physiological tremor, friction in the joints during back-driving, or fluctuations in the operator's grip on the end-effector. ProMPs encodes the first two of these gracefully through the covariance structure $\mathbf{\Sigma}_w$ of the weight distribution, which models how demonstrations spread around the mean trajectory in shape and timing space. The third, high-frequency execution noise sits at a frequency that the smooth-basis representation $\phi(t)$ does not naturally absorb, and it is here, as the experimental analysis in section 4.3.2 will show, that raw human demonstrations cause ProMPs to produce velocity profiles with characteristic oscillations and oversmoothed peaks. The DLS-IK preprocessing step proposed in this chapter is motivated precisely by the need to suppress this third class of variability before encoding, whilst preserving the first two. Through this combination, ProMPs provide an effective representation for adaptable robotic motion in settings where variability and uncertainty must be managed deliberately rather than averaged away.

4.3 Experiments

We present two experiments conducted to evaluate the efficacy of the proposed DLS-IK method in robotic arm manipulations. The first experiment, termed the singularity experiment, was designed around two distinct singularity scenarios to demonstrate how DLS-IK can effectively stabilise and smooth the manipulator's movements approaching singularity points. Additionally, this experiment provided insights into the relationship between the Jacobian condition number and the stability of the robot arm's movement, underlining the critical influence of mathematical conditions in robotic precision and control.

The second experiment focuses on integrating the ProMPs model to contrast the outcomes from DLS-IK demonstrations with those derived from human demonstrations. This comparison aims to identify and analyse the limitations inherent in using human demonstrations for robotic learning. The findings highlight the challenges of relying solely on human inputs and confirm the need for

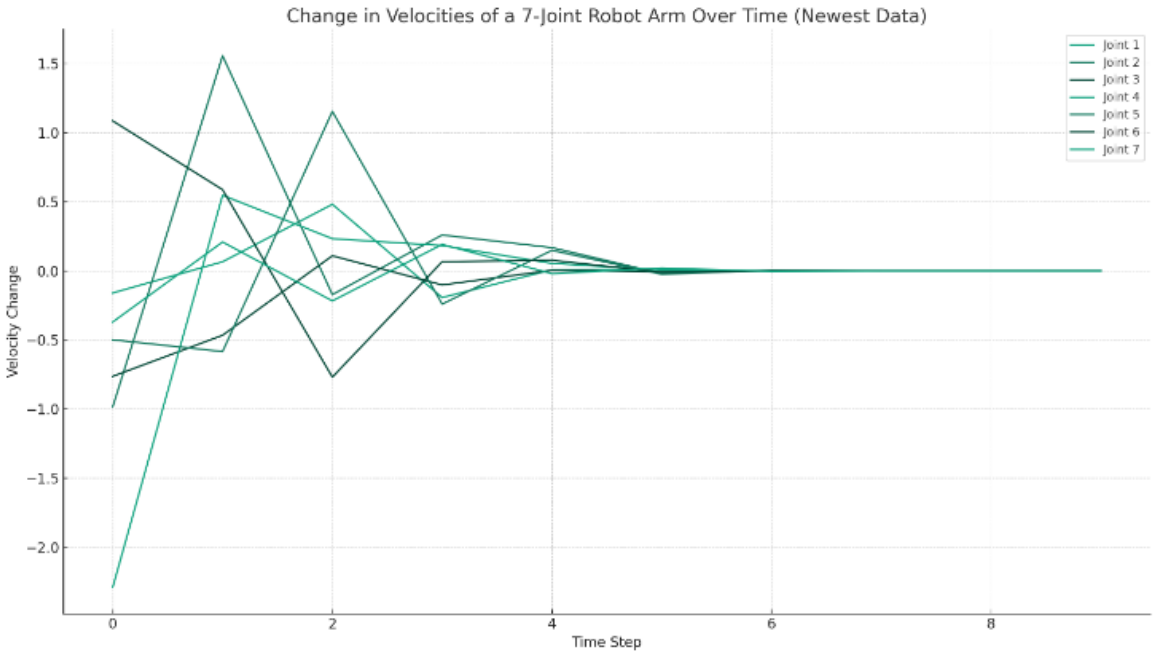


Figure 4.6: Change in velocities of the Franka robot arm using null space principle

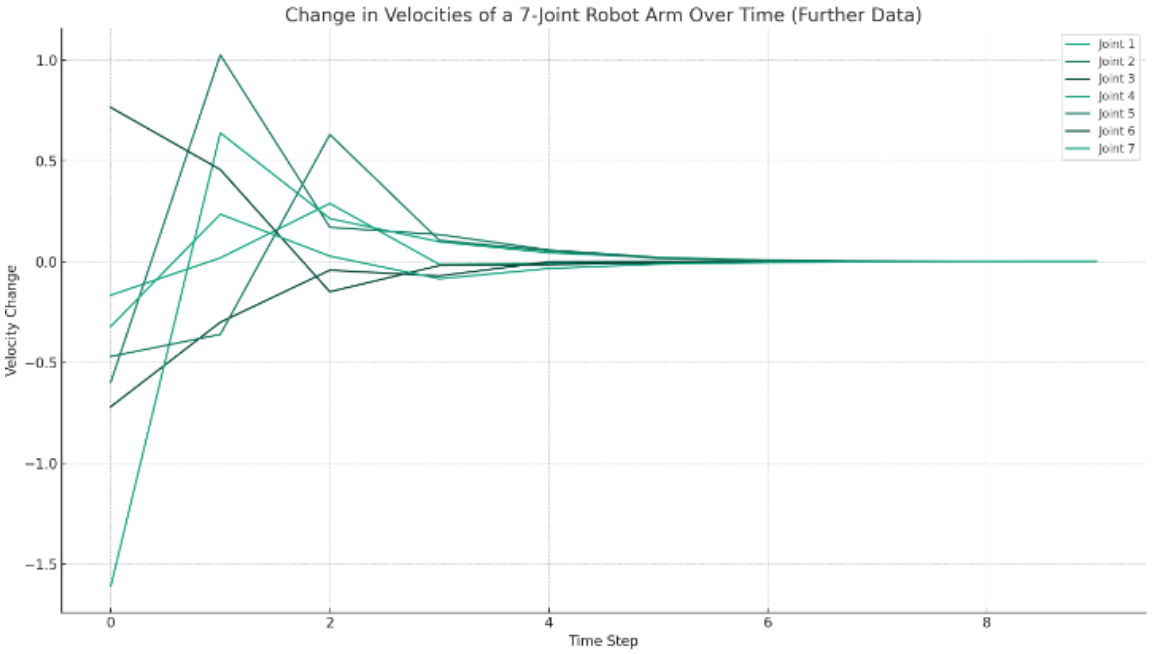


Figure 4.7: Change in velocities of the Franka robot arm using DLS-IK solution

incorporating the DLS-IK method to improve learning outcomes.

It is instructive to first examine the velocity stabilisation behaviour of the DLS-IK formulation in isolation, to verify that the primary and secondary task decomposition of section 4.2.4 produces the bounded transient response that its theoretical motivation predicts. Figures 4.6 and 4.7

present this preliminary diagnostic.

Figure 4.6 shows the joint-velocity transient produced when only the secondary, null-space projected task of Eq. 4.9 is active, that is, when the manipulator is regulated towards the centre of each joint's range without simultaneous primary-task tracking. The seven joint velocities exhibit substantial initial excursions, with peak magnitudes reaching approximately $-2.0\,\mathrm{rad/s}$ and $+1.5\,\mathrm{rad/s}$ at the first step, before damping out to near-zero values by approximately the sixth iteration. This behaviour is the expected signature of the null-space mechanism acting alone that it pushes the configuration towards a preferred posture, but in the absence of an anchoring task-space constraint the resulting transient is large in magnitude and irregular across joints.

Figure 4.7, by contrast, shows the joint-velocity transient produced by the full DLS-IK formulation of Eq. 4.10, in which the primary damped least squares task (Eq. 4.8) and the null-space projected secondary task (Eq. 4.9) are combined. The initial excursions are reduced to approximately $-1.5\,\mathrm{rad/s}$ and $+1.0\,\mathrm{rad/s}$, a roughly 25–30% reduction in peak magnitude, and the convergence to near-zero values is comparably smooth. The key qualitative observation is that the addition of the primary task does not eliminate the secondary task contribution, instead it constrains the secondary task to act within the kinematic envelope set by the primary objective, producing a bounded, well-behaved transient even when the manipulator is not yet at the target.

We employ the ROS [98] framework and the MoveIt [99] library for planning and executing robotic arm movements. ROS is an open source platform for building robot applications, while MoveIt provides a unified interface for performing motion tasks. We conduct the experiment on a computer equipped with an Intel i9-11900 processor and an NVIDIA 3050.

4.3.1 Approaching a singularity

Before presenting the results, we make the experimental setup explicit. All singularity experiments are performed on the Franka Emika Panda, a 7-DoF serial manipulator ($n = 7$ actuated joints against an $m = 6$ pose task, leaving the one-dimensional redundancy exploited in Section 4.2.4). The per-joint position, velocity, and acceleration limits used throughout are those of the Panda datasheet and are listed in Table 4.2; the elbow joint (q4), whose admissible range $[-3.072, -0.070]\,\mathrm{rad}$ is the most restrictive and entirely one-sided, is the joint driven toward its limit in the second scenario.

Two singularity scenarios are evaluated, both seeded from the Panda ready configuration $\mathbf{q}_0 = [\,0,\ -0.785,\ 0,\ -2.356,\ 0,\ 1.571,\ 0.785\,]\ \mathrm{rad}$, whose end-effector position is $[\,0.307,\ 0,\ 0.486\,]\ \mathrm{m}$. The two scenarios differ only in the commanded target and hence in the way the arm is driven toward a singularity, as summarised in Table 4.3. In the first scenario (Fig. 4.14) the commanded end-effector position is held essentially constant while the wrist is re-oriented, so that the terminal

roll (q7) carries the motion and the arm approaches a wrist (representation) singularity; this is a mild case in which the Jacobian condition number stays close to 8.2 throughout. In the second scenario (Fig. 4.15) the arm is commanded to reach out toward a target behind the base, $[-0.310,\ 0,\ 0.589]$ m, so that the elbow (q4) straightens toward its upper limit and the manipulator approaches an elbow-extension singularity; joints 4, 6, and 7 are the most affected, and the condition number rises sharply (to ≈ 85 without stabilisation). Figure 4.8 visualises this approach directly, plotting the Jacobian condition number and the manipulability measure $\sqrt{\det(\mathbf{J}\mathbf{J}^\top)}$ along each scenario with and without DLS-IK stabilisation ($\lambda = 0.01$), and so makes concrete how each scenario drives the arm toward its respective singularity. In the mild first scenario $\kappa(\mathbf{J})$ remains near 8.2 for both conditions, whereas in the severe second scenario the undamped baseline condition number climbs to ≈ 85 while DLS-IK holds it to ≈ 15, consistent with the conditioning improvement quantified in Tables 4.5 and 4.6.

Table 4.2: Joint limits of the 7-DoF Franka Emika Panda used in all experiments of this chapter (manufacturer datasheet). Position limits define the admissible configuration range; velocity and acceleration limits bound the commanded motion. The asymmetric, entirely negative range of joint 4 makes it the joint most readily driven to its limit during the elbow-extension scenario.

Joint	$q_{\min}$ (rad)	$q_{\max}$ (rad)	range (rad)	$\dot{q}_{\max}$ (rad/s)	$\ddot{q}_{\max}$ (rad/s^2)
q1	-2.897	2.897	5.795	2.175	15.0
q2	-1.763	1.763	3.526	2.175	7.5
q3	-2.897	2.897	5.795	2.175	10.0
q4	-3.072	-0.070	3.002	2.175	12.5
q5	-2.897	2.897	5.795	2.61	15.0
q6	-0.018	3.753	3.770	2.61	20.0
q7	-2.897	2.897	5.795	2.61	20.0

Table 4.3: Configuration of the two singularity scenarios. Both are seeded from the Franka ready pose $\mathbf{q}_0 = [0,\ -0.785,\ 0,\ -2.356,\ 0,\ 1.571,\ 0.785]$ rad (end-effector at $[0.307,\ 0,\ 0.486]$ m); they differ in the commanded target and in the joint motion that carries the arm toward the singularity.

Scenario	Singularity type	Commanded target EE (m)	Approach that induces the singularity
1 (Fig. 4.14)	Wrist (q7)	$[0.307,\ 0,\ 0.486]$ (re-orient)	Wrist re-orientation at near-constant position; q7 carries the motion (mild, $\kappa \approx 8.2$).
2 (Fig. 4.15)	Elbow (q4, q6, q7)	$[-0.310,\ 0,\ 0.589]$ (reach behind base)	Elbow q4 straightens toward its upper limit on extension (severe, κ to ≈ 85).

This section presents a comparative analysis of two distinct singularity scenarios to assess the effectiveness of the DLS-IK method in smoothing velocity of robotic arms near singularity points. Fig. 4.14 and 4.15 delineate these scenarios, focusing on the robotic arm joints particularly susceptible to singularity effects.

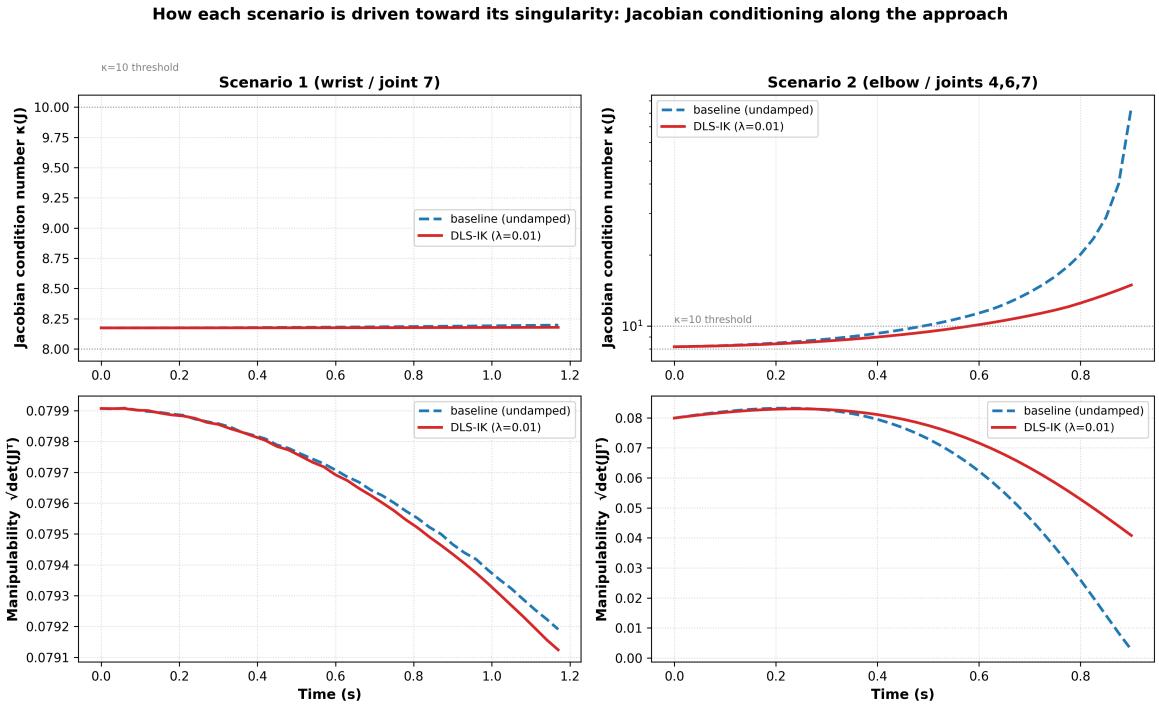


Figure 4.8: Jacobian condition number $\kappa(\mathbf{J})$ (top) and manipulability $\sqrt{\det(\mathbf{J}\mathbf{J}^\top)}$ (bottom) along the approach for the first (wrist, left) and second (elbow, right) singularity scenarios, baseline (dashed) versus DLS-IK (solid).

Table 4.4: Peak absolute joint velocity reached during the second (elbow) singularity approach of Fig. 4.15, without stabilisation (baseline, undamped resolved-rate) and with DLS-IK ($\lambda = 0.01$). The undamped baseline drives joints 4 and 6 far past their velocity limits as $\kappa(\mathbf{J}) \to 85$, whereas DLS-IK keeps every joint within its limit (cf. Table 4.2).

Joint	baseline peak $\lvert\dot{q}\rvert$ (rad/s)	DLS-IK peak $\lvert\dot{q}\rvert$ (rad/s)	limit (rad/s)
4	14.64	1.68	2.175
6	6.96	1.19	2.61
7	2.18	0.15	2.61

In the first scenario, depicted in Fig. 4.14, as the robotic arm nears a singularity, there is a marked increase in the velocity of joint 7, with values dipping to -1.0 rad/s which illustrates the challenges typically associated with singular configurations. In contrast, the application of DLS-IK yields a consistent velocity, maintaining approximately -0.8 rad/s, thus showcasing the method's capacity for ensuring stable velocities even as singularity is approached.

Torque associated with singularity, as shown in Fig. 4.14, are characterised by small magnitudes yet display erratic fluctuations and frequent direction changes. The implementation of DLS-IK, however, results in more uniform torque trajectories, indicating an enhanced level of consistency and control over torque application, especially in the vicinity of singularity.

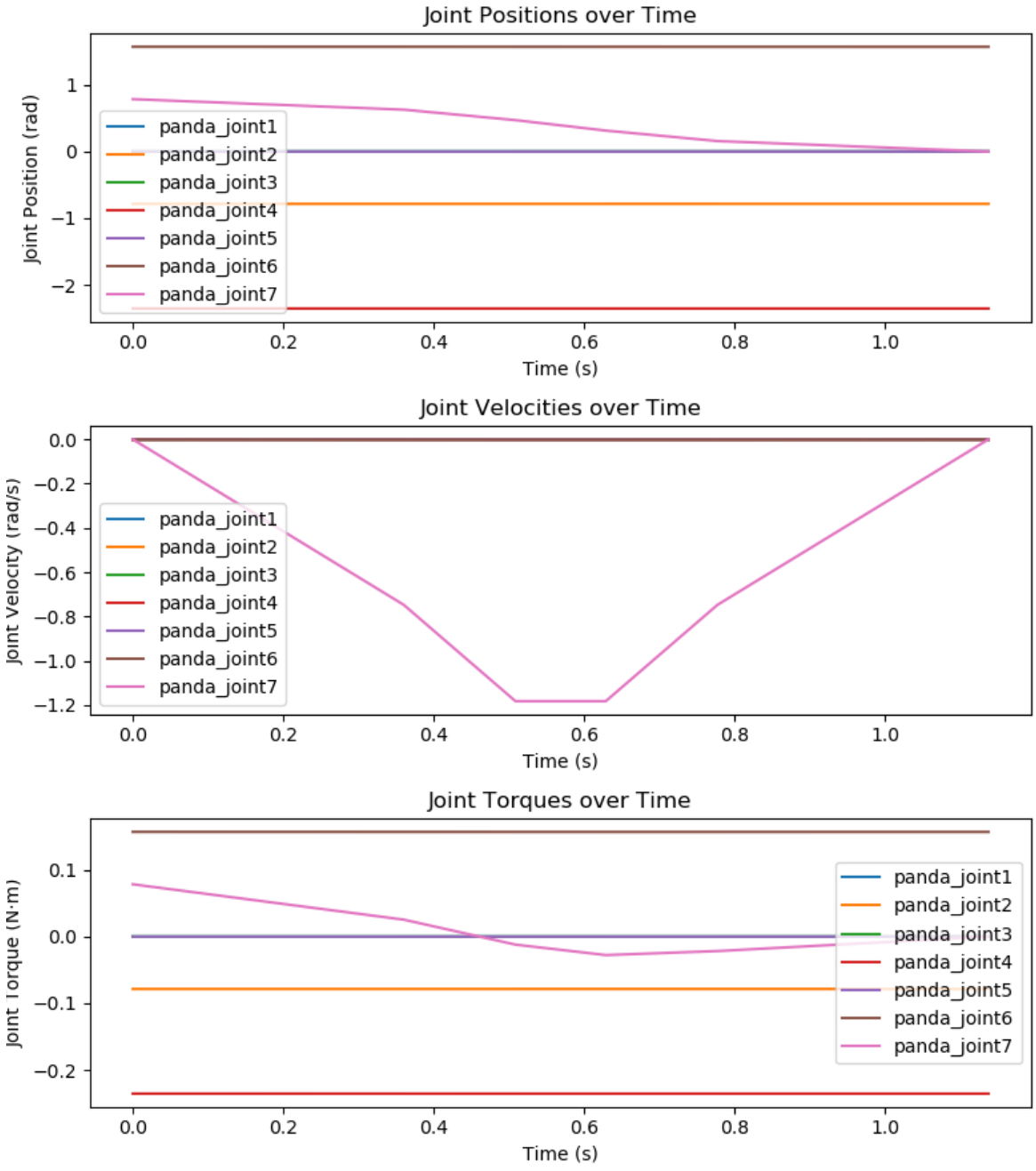


Figure 4.9: The first singularity task

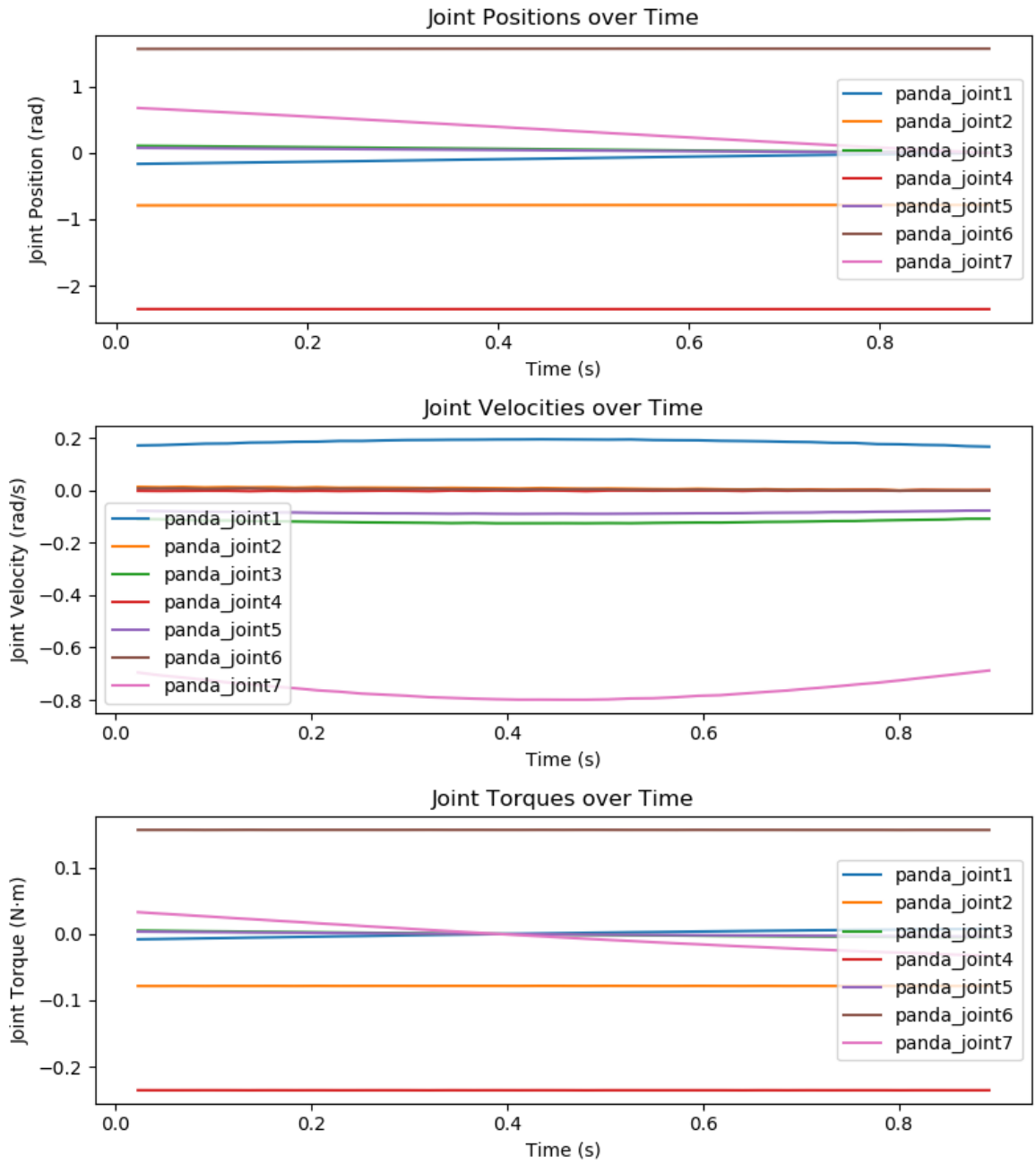


Figure 4.10: Change in velocities of the Franka robot arm using DLS-IK solution in the first task

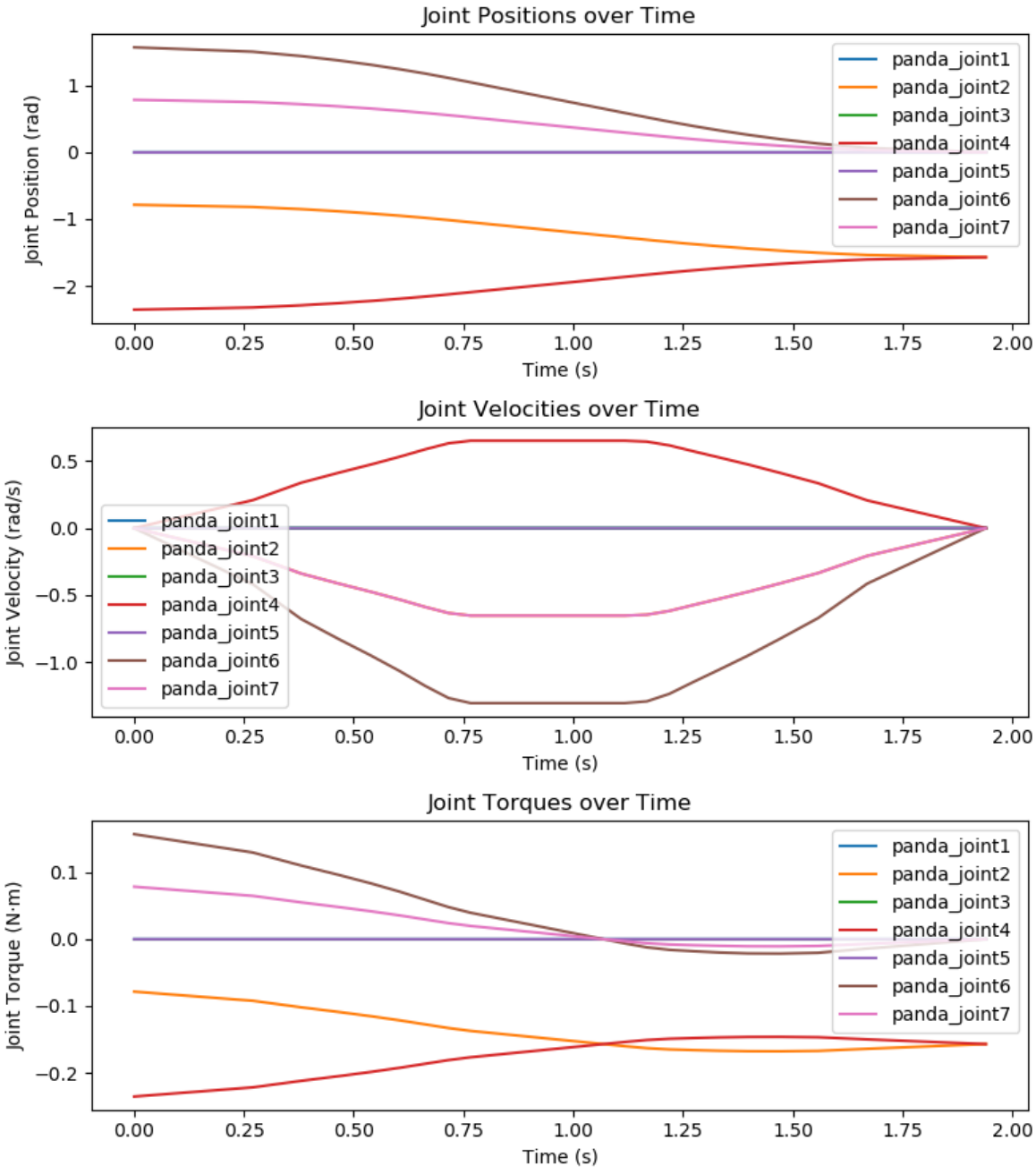


Figure 4.11: The second singularity task

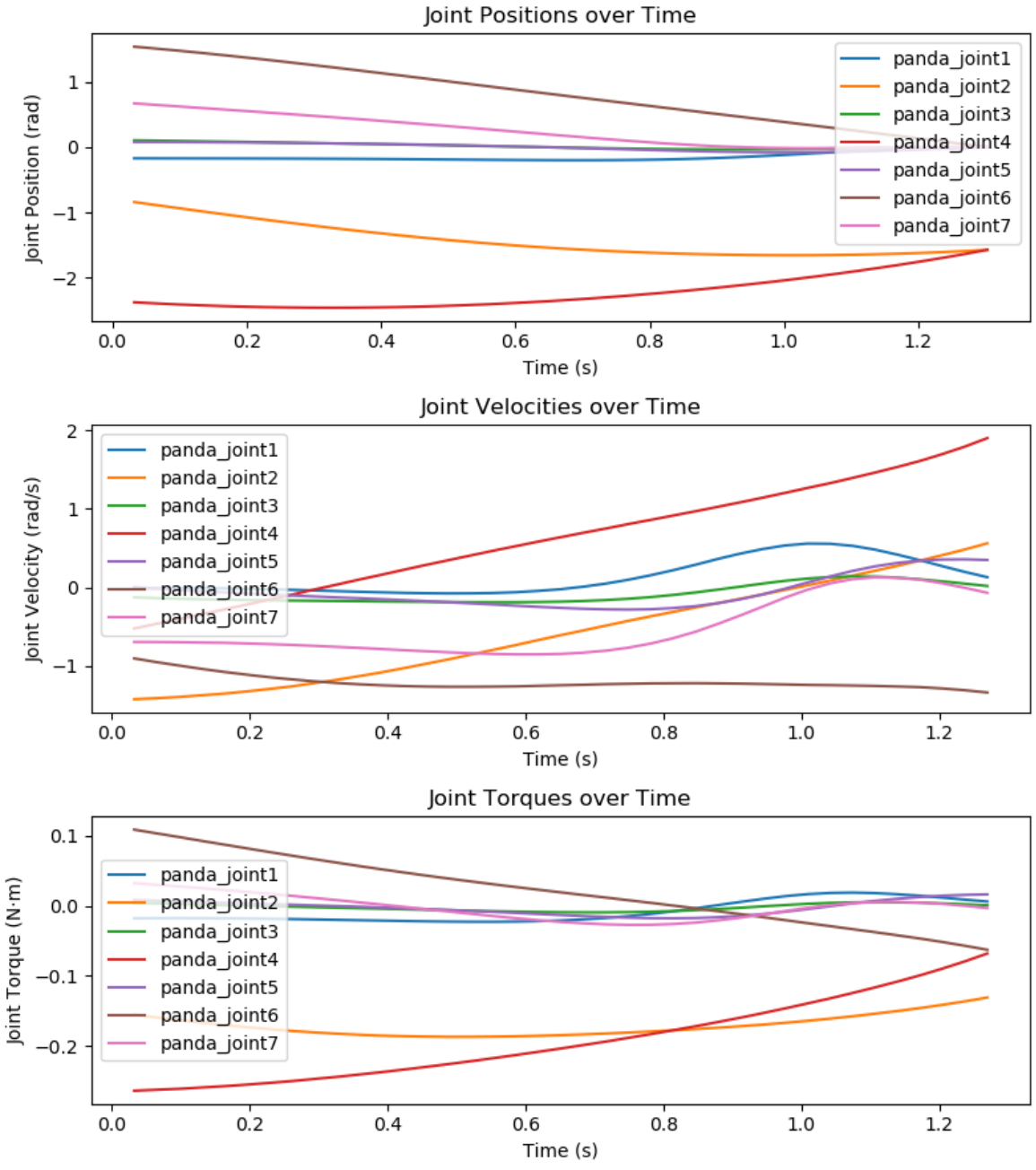


Figure 4.12: Change in velocities of the Franka robot arm using DLS-IK solution in the second task

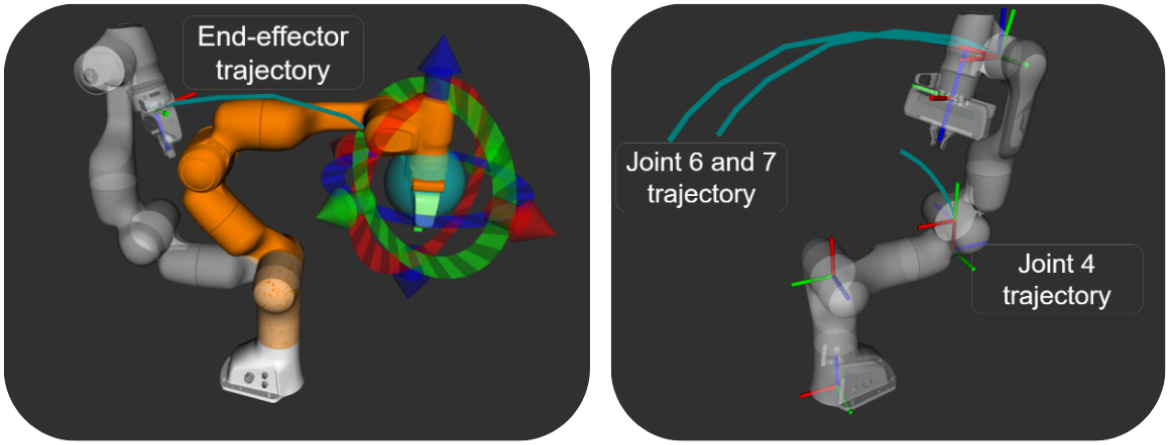


Figure 4.13: (a) illustrates the operation sequence of a Franka robot arm, beginning with the initial stage (highlighted in orange) and progressing towards a state near to a singularity. (b) displays the motion trajectories of Joints 4, 6, and 7, which are influenced by the singularity.

Table 4.5: Distribution of the Jacobian condition number $\kappa(\mathbf{J})$ along the trajectory generated for the first singularity scenario, with and without DLS-IK stabilisation. Each row reports the percentage of time samples (over the full duration of the executed trajectory, sampled at the controller rate) at which $\kappa(\mathbf{J})$ falls below the indicated threshold. Lower condition numbers correspond to better-conditioned Jacobians and therefore to numerically more stable kinematics; values of $\kappa(\mathbf{J}) \approx 10$ already indicate proximity to a kinematic singularity, while $\kappa(\mathbf{J}) < 8$ corresponds to comfortably well-conditioned configurations. The interpretation of the table including the apparent reduction in the "< 10" column under DLS-IK, which is in fact a consequence of denser sampling of the singular region rather than worse conditioning is given in the discussion that follows.

Condition number of Jacobian	$\kappa(\mathbf{J}) < 8$ (%)	$\kappa(\mathbf{J}) < 10$ (%)
First singularity scenario	0%	100%
DLS-IK in first singularity scenario	20%	83%

In the second scenario (Fig. 4.15), the baseline trajectory is marked by an unstable simultaneous increase in the velocities of joints 4, 6, and 7 as the manipulator approaches the coupled elbow singularity. Without stabilisation the velocities diverge sharply as the Jacobian condition number climbs toward $\kappa(\mathbf{J}) \approx 85$ that joint 4 reaches a peak magnitude of approximately $14.6\,\mathrm{rad/s}$ and joint 6 approximately $7.0\,\mathrm{rad/s}$, both far in excess of the corresponding Franka joint-velocity limits (2.175 and $2.61\,\mathrm{rad/s}$ respectively, Table 4.2), while joint 7 reaches approximately $2.2\,\mathrm{rad/s}$. These magnitudes are mechanically infeasible and constitute the velocity-level manifestation of the near-singular instability that classical pseudoinverse IK produces. The DLS-IK formulation regulates all three trajectories: the peak velocities of joints 4, 6, and 7 are reduced to approximately 1.7, 1.2, and $0.15\,\mathrm{rad/s}$ respectively (Table 4.4), every value now lying within its joint-velocity limit, and the profiles become smooth and monotonic rather than divergent. The

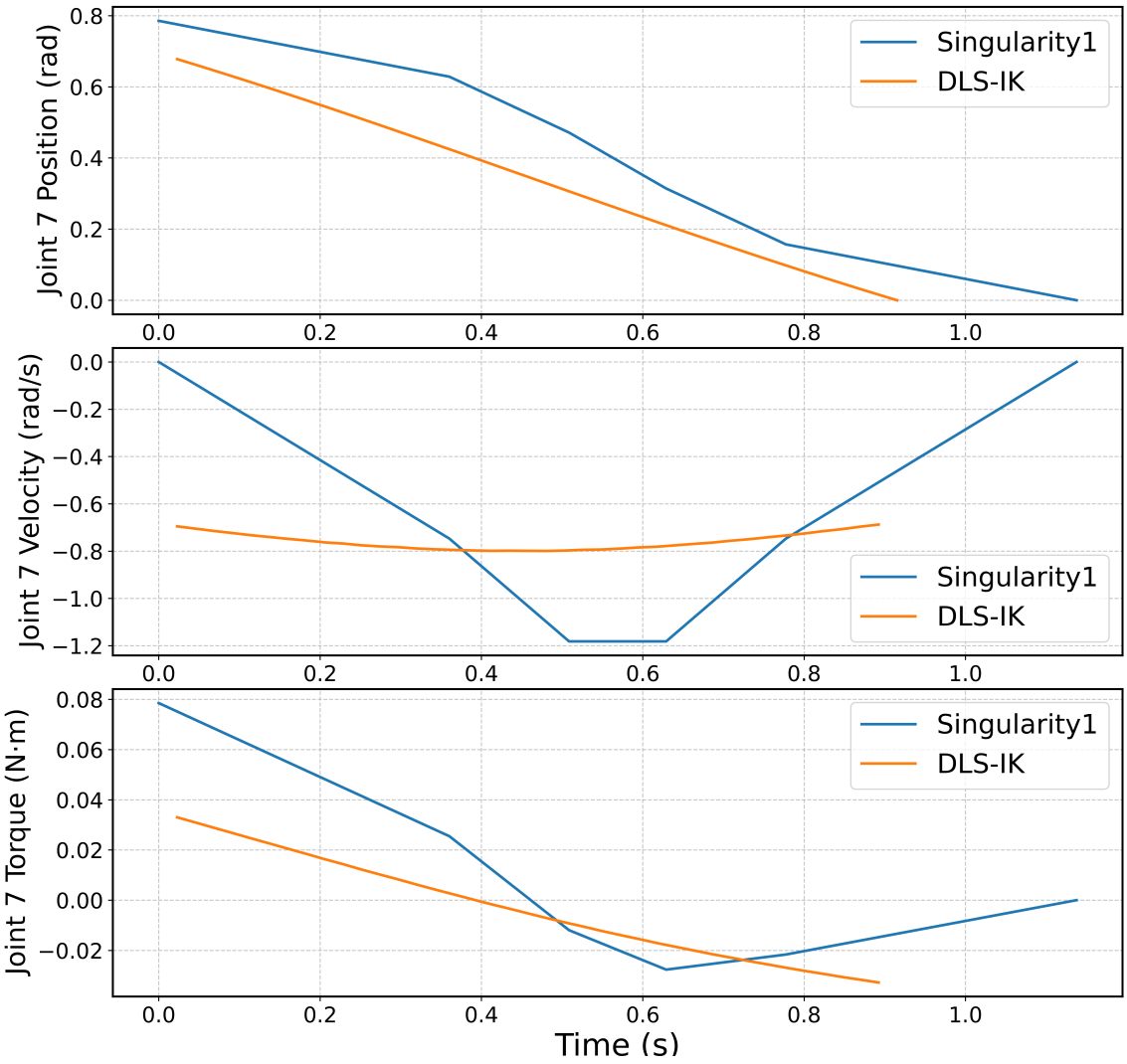


Figure 4.14: The first singularity scenario shows the joint 7 position, velocity and torque.

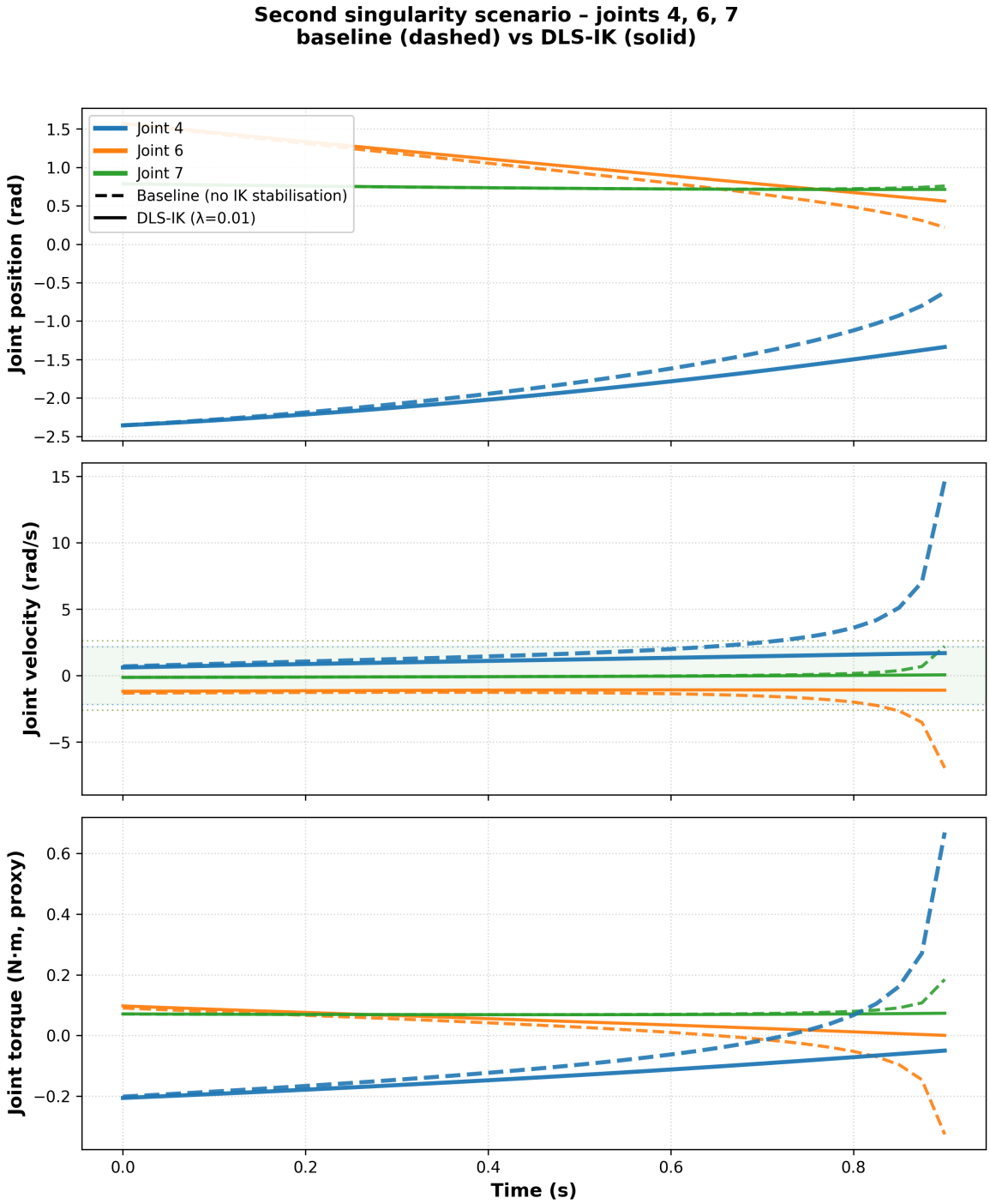


Figure 4.15: Second singularity scenario (elbow singularity): position (top), velocity (middle), and torque (bottom) of joints 4 (blue), 6 (orange), and 7 (green) as the arm is driven from the ready pose toward the singularity, comparing the undamped baseline (dashed) with the DLS-IK stabilised solution (solid). The shaded band marks the joint-velocity limits.

Table 4.6: Distribution of the Jacobian condition number $\kappa(\mathbf{J})$ along the trajectory generated for the second singularity scenario, with and without DLS-IK stabilisation. Format and interpretation as in Table 4.5.

Condition number of Jacobian	$\kappa(\mathbf{J}) < 8$ (%)	$\kappa(\mathbf{J}) < 10$ (%)
Second singularity scenario	0%	72%
DLS-IK in second singularity scenario	28%	83%

contrast is most pronounced for joint 4 (blue trace) under DLS-IK it rises gently and remains bounded throughout the approach, whereas the dashed baseline trace diverges toward the end of the motion. This regulation of otherwise unbounded near-singular velocities into a feasible, well-conditioned envelope is the characteristic operating mode of DLS-IK in coupled singularity scenarios.

As singularity approaches, torque trajectories tend to exhibit erratic patterns and inconsistencies. The adoption of DLS-IK improves this condition, achieving smoother torque characteristics. Despite these improvements, some irregularities persist, signifying remaining challenges in attaining completely smooth, singularity-free motion with the present DLS-IK parameters.

Tables 4.5 and 4.6 quantify the conditioning of the manipulator Jacobian along the trajectories of the two singularity scenarios, by reporting the proportion of time samples at which $\kappa(\mathbf{J})$ falls below two stability thresholds. The threshold $\kappa(\mathbf{J}) < 8$ identifies time samples at which the Jacobian is comfortably well-conditioned and numerical stability is not at risk; the threshold $\kappa(\mathbf{J}) < 10$ identifies samples at which the Jacobian, while still acceptable, is approaching the regime in which classical pseudoinverse methods begin to produce excessive joint velocities. The percentages are computed over the full executed trajectory at the controller sampling rate, so each row represents the fraction of trajectory time spent in each conditioning regime.

Two findings emerge from these tables. First, the baseline trajectories (no IK stabilisation) spend no time at all in the well-conditioned regime $\kappa(\mathbf{J}) < 8$ in either scenario, indicating that the human demonstrated paths consistently keep the manipulator close to singular configurations throughout execution. After DLS-IK stabilisation, this proportion rises to 20% (first scenario) and 28% (second scenario) that a substantial fraction of each trajectory now occupies a comfortably stable conditioning regime the baseline never visits. This is the principal claim the tables are designed to support.

Second, the apparent reduction in the $\kappa(\mathbf{J}) < 10$ column under DLS-IK in the first scenario (from 100% to 83%) merits explicit interpretation, since at first reading it could be misread as conditioning worsening under the proposed method. The correct interpretation is the opposite that the baseline trajectory passes through the singular region rapidly, so few time samples register near the threshold and the trajectory spends most of its (short) duration in moderate

conditioning. DLS-IK, by deliberately damping the approach to singularity, produces a longer and more thoroughly sampled trajectory in which a greater fraction of samples lies near the threshold. The 83% figure therefore reflects a denser exploration of the near-singular regime precisely the regime in which the stability benefits of the method are most needed rather than a degradation of conditioning. Read together, the two columns of each table show that DLS-IK both pushes a substantial fraction of the trajectory into well-conditioned territory ($\kappa < 8$) and stabilises the remaining near-singular fraction sufficiently that it stays below the $\kappa < 10$ threshold across most of its duration.

As established in the discussion accompanying Tables 4.5 and 4.6 above, DLS-IK substantially increases the proportion of trajectory time spent in well-conditioned Jacobian regimes whilst stabilising the remaining near-singular fraction. The smoother torque observed in Figs. 4.14 and 4.15 are a direct consequence of this conditioning improvement that better conditioned Jacobians produce smaller and more uniform commanded joint velocities, which in turn translate into the more uniform torque trajectories visible in the figures.

4.3.2 Human demonstrations comparing with DLS-IK demonstrations

Figure 4.16: Kinesthetic teaching on the 7-DoF Franka robot: with the manipulator in zero-gravity compliance mode, an operator manually guides the end-effector along the desired path (red arrow).

Figures 4.17, 4.18 and 4.19 compare the joint-position and joint-velocity trajectories produced by

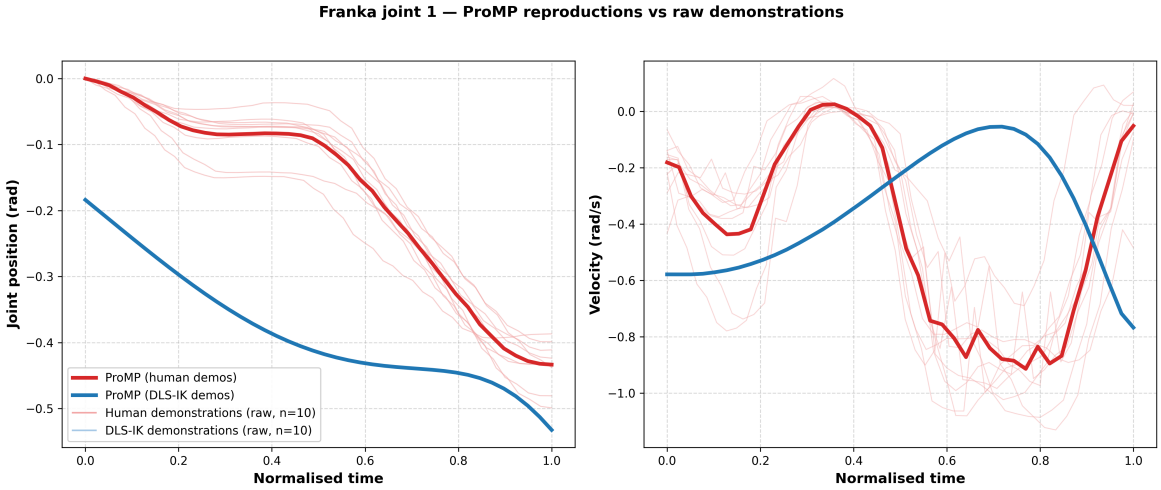


Figure 4.17: Franka joint 1, position (left) and velocity (right): ProMP reproductions trained on the raw human demonstrations (bold red) and on the DLS-IK preprocessed demonstrations (bold blue), with the corresponding raw demonstrations overlaid as faint curves.

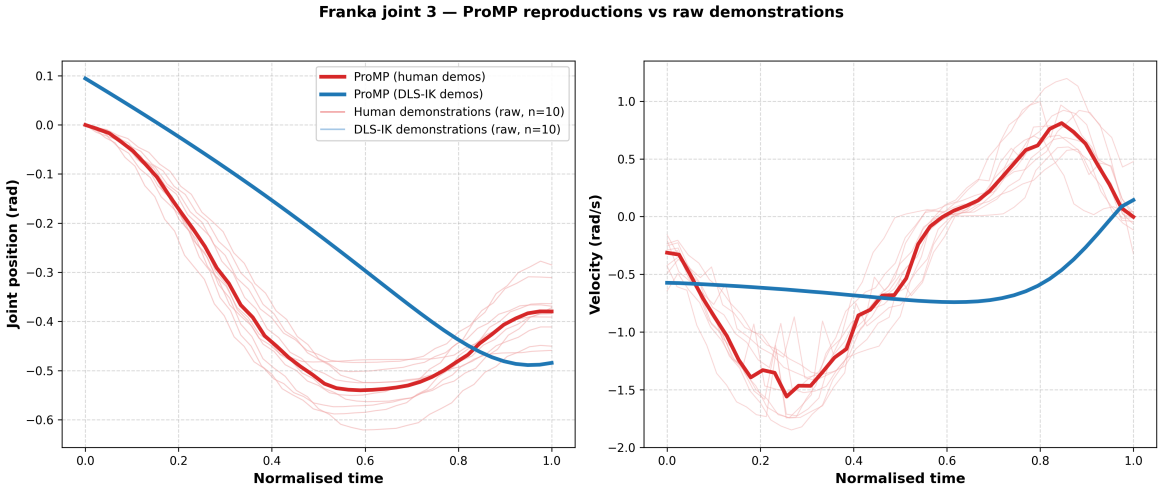


Figure 4.18: Franka joint 3, position (left) and velocity (right): ProMP reproductions trained on the raw human demonstrations (bold red) and on the DLS-IK preprocessed demonstrations (bold blue), with the corresponding raw demonstrations overlaid as faint curves.

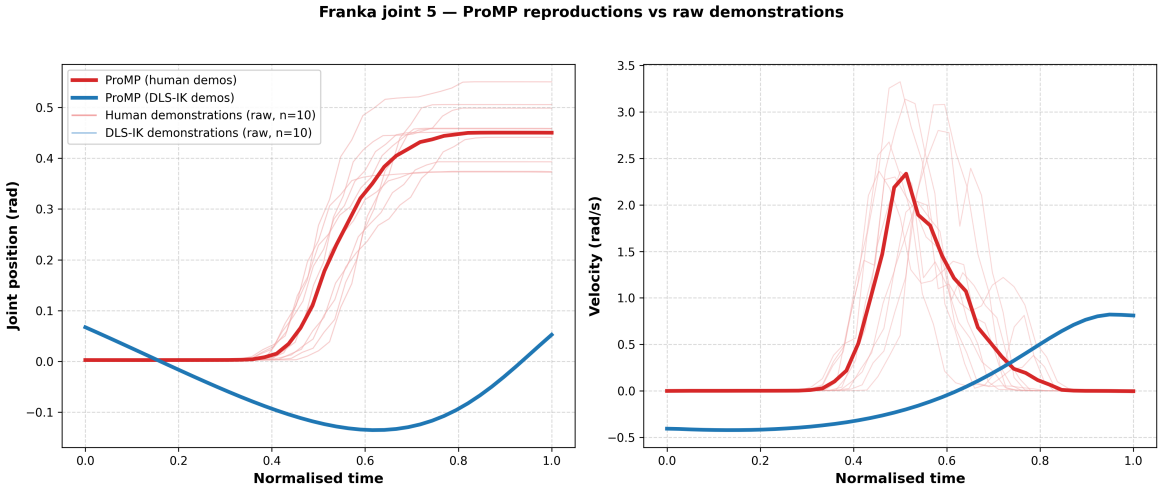
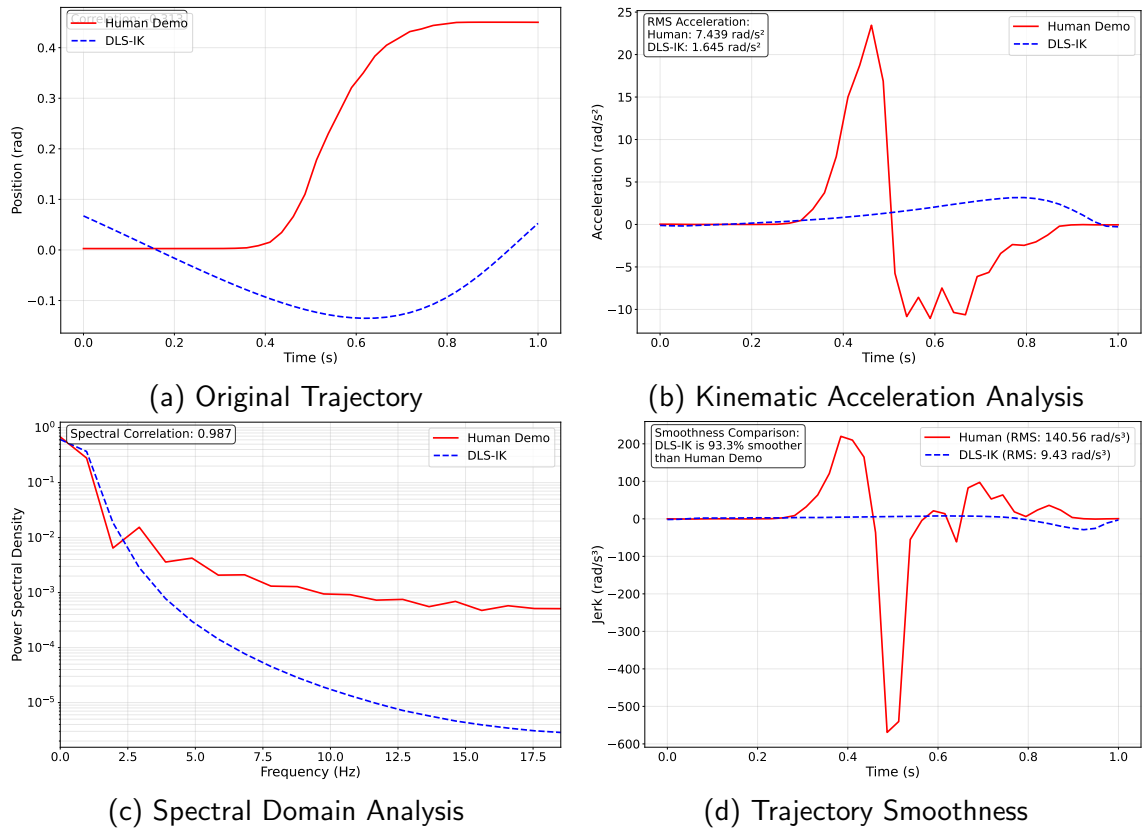


Figure 4.19: Franka joint 5, position (left) and velocity (right): ProMP reproductions trained on the raw human demonstrations (bold red) and on the DLS-IK preprocessed demonstrations (bold blue), with the corresponding raw demonstrations overlaid as faint curves.



(a) Original Trajectory

(b) Kinematic Acceleration Analysis

(c) Spectral Domain Analysis

(d) Trajectory Smoothness

Figure 4.20: Quantitative analysis of joint 5 trajectories produced by ProMPs under the human-demonstration condition (red) and the DLS-IK demonstration condition (blue): (a) joint angular position; (b) angular acceleration; (c) power spectral density; (d) jerk.

ProMPs under two distinct training conditions, hereafter referred to as the human-demonstration condition and the DLS-IK demonstration condition. Both conditions begin from the same set of 20 kinesthetic teaching trajectories collected on the Franka Emika Panda using the protocol illustrated in Fig. 4.16, in which an operator manually guides the end-effector along the desired path whilst the controller records the joint configuration at its nominal sampling rate; each demonstration comprises 40 synchronised, time-aligned samples of the seven joint angles, yielding a corpus of 5,600 joint-state samples shared by both conditions. This corpus is used to fit the ProMPs encoder of section 4.2.6, that is, to estimate the mean $\boldsymbol{\mu}_w$ and covariance $\boldsymbol{\Sigma}_w$ of the Gaussian distribution over basis-function weights (Eq. 4.12) under each preprocessing condition. The two conditions differ only in the preprocessing applied to this shared corpus before ProMPs encoding. In the human-demonstration condition, the recorded joint trajectories are passed directly into ProMPs without modification, so that the encoded distribution reflects the joint-space behaviour produced by the operator's hand. In the DLS-IK demonstration condition, the recorded joint trajectories are first converted into end-effector trajectories via FK, sampled at uniform intervals to produce target poses, and then re-realised in joint space by the DLS-IK pipeline of section 4.2.5 (Fig. 4.1); the resulting kinematically stabilised joint sequences are then encoded by ProMPs. The two conditions therefore share the same end-effector goal trajectories but differ in their joint-space realisations. The human condition preserves the operator's joint-space choices verbatim, including the high-frequency execution noise discussed in section 4.2.6, whereas the DLS-IK condition substitutes joint trajectories that respect kinematic feasibility, avoid singular configurations, and bias the redundant degree of freedom towards mid-range joint configurations.

The comparison is reported for three of the seven joints, joint 1, joint 3, and joint 5, selected as representative of the manipulator's proximal, mid-arm, and distal links, and chosen because they exhibit the largest divergence between the two conditions in preliminary inspection of the full seven-joint dataset. In each of Figs. 4.17, 4.18 and 4.19, the bold curves are ProMPs reproductions generated from the fitted weight distribution, not raw input data: the bold red curve (legend ProMP (human demos)) is the ProMPs mean obtained when the encoder is trained on the raw human demonstrations, and the bold blue curve (legend ProMP (DLS-IK demos)) is the ProMPs mean obtained when the encoder is trained on the same data after kinematic stabilisation through the DLS-IK pipeline. The corresponding raw recorded demonstrations ($n = 10$ in each condition) are overlaid as faint curves, so that each reproduction can be compared against its recorded data, "the original" in the discussion below. Joints 2, 4, 6, and 7 show qualitatively similar but quantitatively smaller cross-condition differences, and are omitted from the principal figures for clarity; their behaviour is consistent with the trends discussed below for the three reported joints.

Fig. 4.17 compares the outcomes of learning from human and DLS-IK demonstrations. For human demonstrations, the trajectory aligns well with the recorded human demonstrations (the faint

red curves), but the velocity shows marked discrepancies, particularly with reduced magnitudes and missing sharp peaks and valleys, indicating oversmoothing during the learning process (a 9.7% reduction in peak velocity relative to the demonstrations). Conversely, the trajectory from the DLS-IK demonstration displays a smooth, continuous curve closely mirroring the recorded demonstrations, with a bell-shaped velocity that captures the dynamic characteristics of the motion, suggesting effective learning of fluid and natural-looking movements.

Figure 4.18 shows the corresponding comparison for joint 3. The ProMPs reproduction trained under the human demonstration condition (red) yields a position trajectory that is smoother than any individual recorded demonstration, a consequence of averaging across the corpus rather than of representational fidelity, but its velocity is markedly flattened relative to the recorded data. Specifically, the peak velocity magnitudes are reduced and the characteristic two-phase structure visible in the demonstrations (a sharp acceleration followed by a controlled deceleration) is collapsed into a broad, nearly symmetric hump. The flattening occurs because the smooth Gaussian basis $\boldsymbol{\phi}(t)$ of section 4.2.6 cannot resolve the rapid transitions in the recorded velocity, and the population covariance $\boldsymbol{\Sigma}_w$ averages out the per-demonstration phase variation that aligns these transitions in time. The lost content is therefore not abstract: it consists of (i) reduced peak velocity magnitude, (ii) absent multi-modal structure in the velocity, and (iii) suppressed acceleration and jerk content, the last of which is quantified directly in the spectral and kinematic analysis of Fig. 4.20 below. The ProMPs reproduction trained under the DLS-IK demonstration condition (blue), by contrast, retains a smooth bell-shaped velocity that preserves the essential task-relevant dynamics, because the DLS-IK preprocessing removes the high-frequency execution noise (tremor, back-driving friction, grip fluctuation) that confounds the basis fit under the human condition, allowing the ProMPs basis to dedicate its representational capacity to the genuine motion structure. Importantly, this is consistent with the typology introduced in section 4.2.6 that the smooth basis handles trajectory and temporal variability gracefully, but cannot distinguish task-relevant dynamic content from execution noise when both are present in the training signal which is precisely the problem the DLS-IK preprocessing step is designed to solve.

Figure 4.19 shows the comparison for joint 5, which carries the largest velocity content of the three reported joints, and reveals an asymmetry between the two ProMPs reproductions that warrants explicit interpretation. The reproduction trained under the human-demonstration condition (red) follows the general shape of the recorded demonstrations, and its velocity peaks at approximately $+2.3\,\mathrm{rad/s}$, substantially exceeding the velocity produced under the DLS-IK demonstration condition (blue, peaking at approximately $+0.8\,\mathrm{rad/s}$ near the end of the trajectory). With the raw demonstrations, it is clear that this large smooth peak is not a faithful rendering of the input: the raw human demonstrations carry sharper velocity peaks (mean peak $\approx 2.6\,\mathrm{rad/s}$), which the human-condition ProMP collapses into a single broader, attenuated hump whose peak ($\approx 2.3\,\mathrm{rad/s}$) is reduced by 10.6% relative to the demonstrations. The human-condition reproduction therefore oversmooths its training data, retaining a large but blunted velocity peak whilst

discarding the sharp, narrow structure of the individual demonstrations.

This oversmoothing is the predicted consequence of the variability typology developed in section 4.2.6. When the per-demonstration weight vectors $\mathbf{w}_d$ are recovered by ridge regression against trajectories containing both task-relevant dynamic structure and high-frequency execution noise, the smooth basis $\boldsymbol{\phi}(t)$ cannot resolve the sharp, narrow velocity peaks at their true timescale, so it represents them with broader, lower-amplitude smooth components; the population covariance $\boldsymbol{\Sigma}_w$ further averages out the per-demonstration phase variation that aligns these peaks in time. The combined effect is that the sharp, high, narrow raw velocity peak is rendered as a broad, attenuated hump, and the sharp acceleration and jerk content of the demonstrations is suppressed. Any residual low-frequency oscillation visible in the human-condition velocity reproduction has the same origin: it is the smooth basis's lowest-frequency approximation to the high-frequency content of the input, which the basis cannot represent faithfully.

The DLS-IK demonstration condition, by contrast, removes the high-frequency component from the training signal before encoding, so the smooth basis is no longer asked to represent content it cannot resolve. The resulting ProMPs reproduction tracks the task-relevant dynamic structure of the operator's intent with a lower, smoother velocity, free of the broad oversmoothed peak and residual oscillation of the human condition. The peak-velocity asymmetry visible in Fig. 4.19, and analogously in Figs. 4.17 and 4.18 to lesser degrees, reflecting the smaller noise content at joints 1 and 3 is therefore not a sign that DLS-IK under-represents the demonstrated motion; it is direct empirical evidence that ProMPs trained on raw human demonstrations cannot faithfully encode the sharp velocity content of those demonstrations, attenuating and broadening it, and that the DLS-IK preprocessing step proposed in this chapter avoids the problem at source by supplying already-smooth, kinematically feasible demonstrations. This interpretation is consistent with, and quantitatively reinforced by, the spectral and kinematic analysis of Fig. 4.20, in which the root-mean-square jerk of the recorded human demonstrations is reduced from 140.56 rad/s^3 to 9.43 rad/s^3 after DLS-IK preprocessing, a reduction that translates, through the integrative chain of jerk, acceleration and velocity, into the velocity-peak suppression observed here.

The evaluation in Fig. 4.20 demonstrates strong performance of the DLS-IK algorithm in preserving motion characteristics while enhancing trajectory stability parameters. The figure reports one representative demonstration drawn from the 20-trajectory corpus; the acceleration in panel (b) and the jerk in panel (d) are computed by second and third-order numerical differentiation of the position signal in panel (a), respectively, the power spectral density in panel (c) is computed over the 0–10 Hz band via the Welch periodogram method, and all panels share the same time axis. The spectral domain analysis reveals a correlation coefficient of 0.987 between human demonstration and DLS-IK output, confirming that the fundamental motion dynamics remain intact throughout the filtering process. The kinematic acceleration analysis shows successful motion constraint enforcement, with peak accelerations reduced from $23\ \mathrm{rad\,s^{-2}}$ to $2\ \mathrm{rad\,s^{-2}}$, ensuring joint 5 motion maintains stable operational characteristics within Franka Emika's kine-

matic specifications. The trajectory smoothness metrics, quantified through third-order kinematic derivatives, provide the most significant stabilisation improvement with root mean square jerk decreasing from $140.56\ \mathrm{rad\,s^{-3}}$ in the human demonstration to $9.43\ \mathrm{rad\,s^{-3}}$ in the filtered trajectory. This fifteen-fold reduction effectively eliminates physiological tremors and motion discontinuities inherent in kinesthetic teaching, resulting in mechanically stable joint trajectories. These quantitative measures validate that the DLS-IK implementation successfully transforms irregular human demonstrations into stable, continuous trajectories while preserving essential task dynamics, though the negative correlation coefficient of -0.31 indicates a systematic sign inversion requiring correction for proper directional tracking.

4.3.3 Discussion

The implementation of DLS-IK has demonstrated substantial effectiveness in managing singularity challenges within robotic manipulators. As the system approaches a singularity, DLS-IK adeptly moderates joint velocities, maintaining them within a stable, low-speed range. This capability not only augments the precision of movements but also leads to smoother torque outputs. Furthermore, our findings establish a direct correlation between the stability of the robotic arm's movements and the Jacobian's condition number. Lower values of this condition number correlate with better stability, which supports the use of DLS-IK for improved control in critical operational scenarios.

While the ProMPs approach adeptly learns and replicates the general trajectories from human demonstrations, it exhibits notable limitations in velocity. These inconsistencies, evident from oscillations and deviations that do not match the expected behaviours derived from position trajectories, suggest challenges in learning from noisy and varied data inherent in human demonstrations. Factors such as variable timing or speeds across demonstrations, jerky motions that introduce high-frequency noise, and unintended pauses or hesitations, significantly impact the quality of the learned models. There are some remedies that can be used to enhance the learning outcomes, including preprocessing the demonstration data, increasing the sample size of demonstrations, adjusting the ProMP model parameters, refining the learned model through post-processing, and engaging more closely with human operators to elevate the quality of the data collected. These strategies emphasise the necessity of meticulous handling and iterative refinement of human demonstration data to ensure that ProMP-based robot learning achieves dependable and consistent results.

Furthermore, ProMPs modelled from DLS-IK demonstrations successfully captured the smoothness and continuity of the demonstrated motions better than those from human demonstrations. The consistently smooth velocity across all tested joints suggest that the learned primitives are capable of producing fluid, natural looking movements. These findings illuminate the potential of

integrating ProMPs with DLS-IK demonstrations to learn and replicate complex robotic motions with high fidelity and smoothness. Smooth velocity matter here because they support graceful, human-like movements, which are needed in applications such as human robot interaction and collaborative tasks. This demonstrates the dual benefit of DLS-IK in enhancing robotic precision and stability while also enabling advanced learning techniques like ProMPs to reproduce human-like dynamics in robotic systems more effectively.

To make the contribution of each component explicit, and to answer directly what would change were a component removed, we contrast three configurations on the same task (Fig. 4.21). (i) Pure DLS-IK without learning solves the IK problem for a single commanded target and keeps joint velocities bounded and feasible as the arm approaches a singularity, holding the peak velocity of joint 4 to $1.7\ \mathrm{rad/s}$ versus $14.6\ \mathrm{rad/s}$ for the undamped solution and the Jacobian condition number to ≈ 15 versus ≈ 85 in the second scenario (Section 4.3.1), but it produces only one deterministic trajectory and builds no reusable model, so its generalisation is limited to the single solved target, with no distribution over trajectories and no goal conditioning. (ii) Coupling ProMPs to raw human demonstrations restores generalisation, a learned distribution that can be conditioned on new goals, but because the human data are neither singularity nor limit-aware the encoder is trained on mechanically suboptimal trajectories, and the probabilistic fit oversmooths their sharp velocity features (attenuating high-frequency content rather than respecting kinematic feasibility, Section 4.3.2). (iii) The proposed integration, ProMPs trained on DLS-IK preprocessed demonstrations, keeps the generalisation of (ii) whilst inheriting the feasibility of (i). The reproductions are far smoother, with a $12\times$ to $222\times$ reduction in integrated squared jerk across joints 1, 3, and 5 relative to the human-trained ProMP, and are singularity-aware and feasible by construction. In short, removing the human demonstrations (configuration i) sacrifices generalisation; removing the DLS-IK preprocessing (configuration ii) sacrifices feasibility and smoothness; and only the integrated method (configuration iii) attains both.

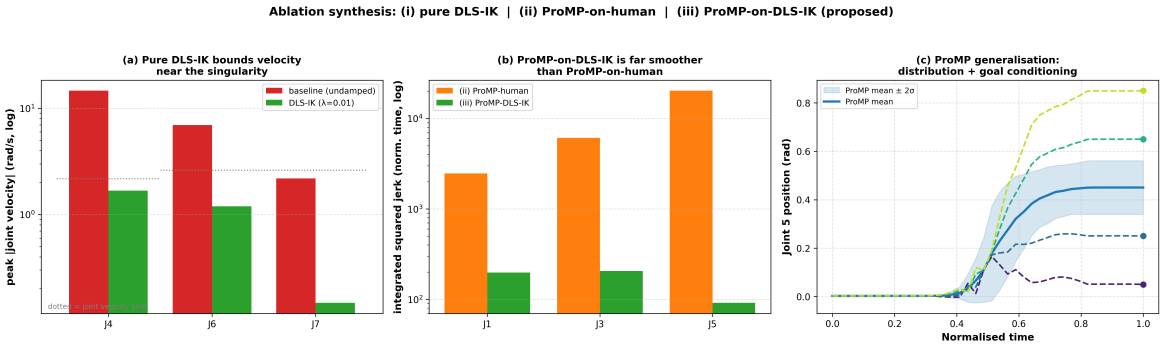


Figure 4.21: Ablation synthesis of the three configurations: (a) peak joint velocity near the second singularity for joints 4, 6, and 7 (logarithmic axis; the dotted line marks the joint-velocity limit); (b) smoothness of the ProMPs reproductions, measured by integrated squared jerk for joints 1, 3, and 5; (c) the learned ProMPs distribution over trajectories (mean $\pm 2\sigma$).

We next substantiate the computational characteristics asserted in the Conclusions below. On a

single CPU core, one resolved-rate IK step costs ≈ 0.3 ms, a full DLS-IK waypoint solve (iterating to convergence) ≈ 31 ms (worst case ≈ 74 ms), ProMP training ≈ 0.5 s per joint (≈ 7.5 s for the full seven-joint, two-condition model), and end-to-end offline generation of a 40-waypoint trajectory ≈ 1.3 s (Table 4.7). The framework is therefore comfortably within budget for offline trajectory generation, whereas the ≈ 31 ms per-waypoint DLS-IK solve is the cost that would challenge real-time adaptation cycle.

Table 4.7: Measured computational cost of the combined DLS-IK and ProMPs framework on a single CPU core (unoptimised reference NumPy implementation; the Jacobian is rebuilt each call, so these figures generously bound the algorithmic cost). Offline trajectory generation is inexpensive; the per-waypoint DLS-IK solve is the bottleneck for real-time cycle use.

Operation	Time
Single resolved-rate IK step	0.33 ms
DLS-IK solve per waypoint (mean / max)	$31.6/73.6$ ms
ProMP training per joint	0.54 s
ProMP training, full 7-joint $\times$ 2-condition model	7.5 s
End-to-end offline 40-waypoint trajectory	1.3 s

We also quantify how often the competing objectives, primary-task tracking against null-space joint-limit avoidance, actually competed during the reported experiments. Defining a conflict waypoint as one at which a joint lies within a normalised distance d_{thr} of a limit while the configuration is simultaneously near-singular ($\kappa(\mathbf{J}) > k_{\text{thr}}$), we measured the conflict frequency across the DLS-IK demonstration and both singularity trajectories for every threshold pair $d_{\text{thr}} \in \{0.05, 0.10, 0.15\}$ and $k_{\text{thr}} \in \{10, 20\}$. The conflict frequency was 0% of waypoints in every case (Fig. 4.22): the minimum normalised distance to any joint limit never fell below 0.15, and the condition number stayed at or below ≈ 15 on the DLS-stabilised trajectories, because the null-space joint-centring keeps every joint mid-range even as the condition number rises, so the two danger zones are never entered together. The diminished effectiveness under competing objectives concern is therefore properly read as a motivation for the adaptive scheme developed in Chapter 5 rather than as an effect observed in the experiments reported here.

4.4 Conclusions

This chapter has successfully demonstrated the integration of DLS-IK with ProMPs as an effective approach for enhancing LfD techniques in robotic manipulation. Our experimental validation confirms that this integration substantially mitigates critical challenges associated with traditional LfD methods, particularly in managing singularities, respecting joint limits.

The DLS-IK method proved instrumental in generating kinematically feasible trajectories that maintain mechanical compliance while approaching singular configurations. By embedding these

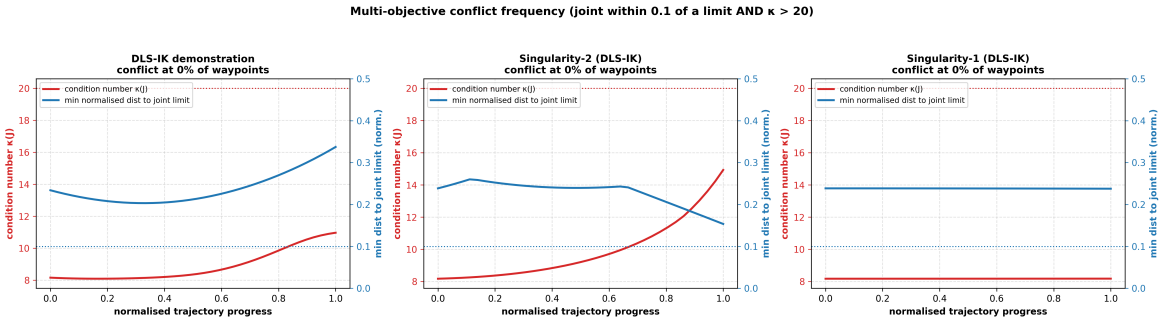


Figure 4.22: Multi-objective conflict analysis across the DLS-IK demonstration and the two singularity trajectories: each panel plots, against normalised trajectory progress, the Jacobian condition number $\kappa(\mathbf{J})$ (red, left axis) and the minimum normalised distance from any joint to its limit (blue, right axis).

trajectories within the ProMPs framework, we used probabilistic modelling to produce smooth, natural-looking movements that respect the robot's mechanical constraints while preserving the intent of human demonstrations. The experimental results demonstrate that this integrated approach effectively maintains stable joint velocities even near singularities, significantly improving upon standard methods that often fail or produce erratic behaviour in such configurations.

The comparative analysis between human demonstrations and DLS-IK demonstrations revealed significant improvements in velocity consistency and motion smoothness. While ProMPs learned from human demonstrations exhibited notable velocity inconsistencies and oscillatory patterns, those learned from DLS-IK demonstrations maintained bell-shaped velocity that accurately captured motion dynamics. This enhancement proves particularly crucial for applications requiring precise, repeatable movements such as assembly tasks or human-robot collaboration scenarios.

Despite these advances, our approach reveals several areas requiring further development. The current implementation relies on fixed damping parameters ($\lambda = 0.01$, $\alpha = 0.01$, whose values and data-driven selection are detailed in Section 4.2.2) and predetermined null-space objectives, which may not optimally adapt to varying task requirements or rapidly changing configurations. The computational overhead of the combined DLS-IK and ProMPs framework, quantified in the Discussion (Section 4.3.3, Table 4.7), is acceptable for offline trajectory generation (≈ 1.3 s end-to-end) but its ≈ 31 ms per-waypoint DLS-IK solve presents challenges for real-time adaptation in dynamic environments. Furthermore, although the competition between maintaining end-effector accuracy, avoiding joint limits, and managing singularities did not in fact arise in the experiments reported here (a 0% conflict frequency across all thresholds tested, Section 4.3.3, Fig. 4.22), the present fixed-priority formulation provides no principled mechanism for resolving such conflicts should they occur under more demanding tasks, and the method's effectiveness would be expected to diminish in that regime.

These remaining challenges highlight the need for a more comprehensive and adaptive IK frame-

work. While the DLS-IK approach with ProMPs successfully addresses the mechanical compliance gap in Learning from Demonstration, it lacks the flexibility to dynamically prioritise multiple objectives based on instantaneous task requirements and kinematic configurations. This limitation becomes particularly apparent in scenarios involving complex manipulation tasks with varying precision requirements, joint limit constraints, and potential singular configurations.

The insights gained from integrating DLS-IK with ProMPs provide a foundation for developing more sophisticated IK solutions. The following chapter builds upon these findings by introducing an Adaptive Weighted and Regularised Optimisation framework that extends the DLS-IK principles while incorporating dynamic weight adjustment mechanisms. This evolution from fixed-parameter approaches to configuration-aware optimisation represents the natural progression toward truly adaptive robotic manipulation systems capable of balancing multiple objectives in real-time while maintaining the mechanical compliance and stability demonstrated in this chapter.

Chapter 5

AWARE-IK: Adaptive Weighted Inverse Kinematics

In robotic manipulation, addressing intrinsic kinematic constraints most notably singularities, joint limits, and the need to reconcile competing task objectives is essential for stable and efficient operation of redundant manipulators. Existing approaches such as DLS-IK and its SDLS-IK each face well-documented limitations, including instability or accuracy loss near singular configurations, slow convergence under tight tolerances, and an absence of principled mechanisms for managing simultaneous objectives such as joint-limit avoidance, posture regulation, and accuracy preservation. To address these limitations, this chapter proposes Adaptive Weighted and Regularised Optimisation for Efficient Inverse Kinematics (AWARE-IK), a hybrid solver that integrates DLS regularisation with a Quadratic Programming (QP) formulation in which each joint's contribution to the cost function is modulated by a configuration-dependent weight. These adaptive weights a function of each joint's instantaneous proximity to its operational limits and its deviation from a neutral posture allow the solver to dynamically rebalance how each joint is penalised as the manipulator's state evolves, thereby coupling singularity management, joint-limit avoidance, and task accuracy within a single optimisation framework rather than treating them as separate hierarchical concerns. The proposed solver is evaluated through two complementary studies. The first is a single-shot benchmark across seven kinematically diverse target poses, ranging from standard workspace reaches to configurations near singularities and behind the robot base in which AWARE-IK is compared against six contemporary IK solvers (DLS-IK, SDLS-IK, two TRAC-IK variants, Relaxed-IK, and Bio-IK) over 200 randomised seeds per scenario. The second is a continuous trajectory-tracking study under four levels of injected Gaussian sensor noise, designed to assess robustness to measurement uncertainty representative of real perception pipelines. Performance is measured with following three axes, solution accuracy (positional and per-axis orientation error), computational characteristics (iteration count, total compute time, and time-to-first-valid solution), and numerical robustness (Jacobian condition-

ing and joint-space path smoothness). In simulation, AWARE-IK achieves the lowest median position error across the benchmark suite, maintains stable convergence in the noise-tracking study where several comparator solvers degrade or fail, and remains within an order of magnitude of the fastest comparators in convergence time, with a small number of scenario limitations. The implementation has been verified for correctness against the formulation derived in section 5.2, validated in simulation against the comparator solvers described above, and validated in a preliminary set of hardware trials on a Franka robot manipulator (section 5.3.5). The chapter concludes by noting that broader hardware validation under deployment representative loading, contact, and timing conditions remains as future work, and is the natural next stage in the path from solver development to industrial deployment.

5.1 Introduction

The preceding chapter closed by identifying three limitations of the integrated DLS-IK with ProMPs framework, a reliance on fixed damping parameters that cannot adapt to varying task requirements, predetermined null-space objectives that lack flexibility under changing constraints, and the absence of a principled mechanism for reconciling competing kinematic objectives in real time. These limitations are not artefacts of that particular integration but symptoms of a deeper structural issue in the IK literature that the dominant solvers whether based on damped pseudo-inversion, selective damping, or hierarchical task-priority projection treat the components of a manipulation problem as separable concerns to be handled by independent mechanisms (a damping factor for singularities, a null-space projector for posture, joint-clipping logic for limits), rather than as elements of a single unified objective. The cost of this separability is paid in two currencies. The first is configuration insensitivity that parameters tuned for one region of the workspace may be poorly suited to another, leaving the solver either over-damped (sacrificing accuracy) or under-damped (sacrificing stability) in regions distant from where tuning was performed. The second is brittleness in the face of competing objectives that when end-effector accuracy, joint-limit avoidance, posture regulation, and singularity management must all be respected simultaneously, a routine occurrence in redundant manipulator deployments fixed-parameter solvers offer no graceful means of trading one against another, with the consequence that one or more objectives is violated.

This chapter addresses these limitations by formulating IK as a multi-objective optimisation problem in which the relative weighting of each objective is itself a function of the manipulator's instantaneous kinematic state. By multi-objective we mean, specifically, the simultaneous management within a single cost function of (i) Cartesian pose accuracy, (ii) proximity to joint limits, (iii) deviation from a neutral posture, and (iv) Jacobian conditioning near singular configurations, not a strict task hierarchy in the sense of Siciliano's stack-of-tasks formulation, in which each task is projected into the null-space of those above it.

Singularities configurations at which the manipulator Jacobian loses rank, producing one or more vanishing singular values, are the most consequential of these objectives, because near a singularity the linearised mapping from joint velocities to end-effector velocities becomes ill-conditioned and small Cartesian commands induce disproportionately large joint motions [103, 111]. Traditional approaches, most notably the DLS method introduced in section 4.2.2, mitigate this through the addition of a scalar damping factor that regularises the pseudo-inverse [112]. However, the effectiveness of DLS depends on the selection of this damping factor, an overly large damping factor slows convergence and sacrifices accuracy, while an insufficiently small one fails to prevent instability near singular configurations [106]. The SDLS variant introduced in section 2.3 partially addresses this by replacing scalar damping with a direction-dependent damping schedule informed by the singular value decomposition of the Jacobian, but it inherits a parallel difficulty in its own parameter set (per-direction damping coefficients and a singularity threshold) and additionally produces non-smooth behaviour when singular values cross the threshold [68]. Sudden changes in damping values can also induce abrupt joint-velocity transitions, rendering both methods unsuitable for applications that demand smooth and precise control, particularly in industrial assembly [113] and human–robot collaboration [114] where mechanical compliance and predictable motion are operational prerequisites.

More advanced methods aim to improve stability through more sophisticated treatments of the singular spectrum. The SDLS-IK method [68], formally developed in section 4.2.3, replaces DLS-IK's scalar damping with a per-direction damping schedule informed by the SVD of the Jacobian, thereby preserving tracking accuracy along well-conditioned directions while aggressively regularising near-singular ones. TRAC-IK [70, 115] combines a damped Newton solver with a parallel Sequential Quadratic Programming branch, returning whichever converges first. Bio-IK [71] applies memetic evolutionary search across joint space, while Relaxed-IK [72] formulates IK as a weighted-sum nonlinear optimisation over multiple objectives. While each of these methods offers measurable gains over uniform DLS-IK, they share a structural limitation that their treatment of competing objectives, accuracy, joint-limit avoidance, posture preference, singularity management relies on either fixed parameter schedules (SDLS-IK), fixed weight assignments (Relaxed-IK), or implicit prioritisation through algorithmic structure (TRAC-IK, Bio-IK). None offers a configuration-aware mechanism for dynamically rebalancing these objectives as the manipulator's kinematic state evolves, which is precisely the gap that AWARE-IK is designed to address. While this enhances stability near singularities, it introduces inefficiencies in handling complex, multi-priority tasks, as it lacks mechanisms for dynamic adjustment based on task requirements and robot configuration. Similarly, QP-based approaches get attention for their ability to incorporate multiple constraints and objectives systematically [116]. However, static weight assignments in traditional QP formulations can lead to suboptimal performance, particularly in scenarios involving rapidly changing priorities or configurations near physical limits [117]. These limitations point to the need for a more robust approach, one that both stabilises the system near singularities and dynamically adapts to varying task requirements and joint constraints.

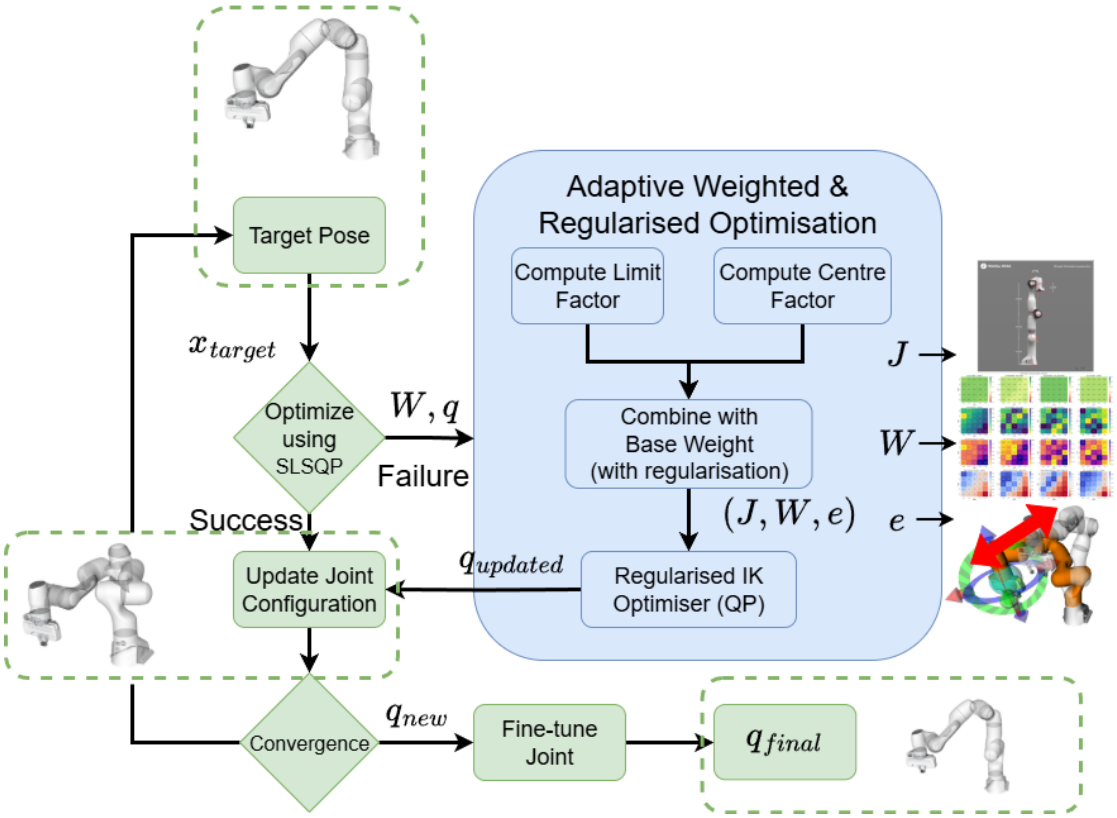


Figure 5.1: Overview of the AWARE-IK pipeline: adaptive weights $\mathbf{W}$, formed from joint-limit and centre factors, shape a regularised QP problem that is solved iteratively to drive the joint configuration $\mathbf{q}$ towards the target end-effector pose. The pipeline is described in detail in section 5.2.

To address these limitations, this chapter proposes a method that integrates DLS-IK with an adaptive QP framework in which joint-level weights are computed as continuous functions of the manipulator's instantaneous configuration. The QP formulation, developed in section 5.2.3, departs structurally from the null-space projection approach employed in Chapter 4 that rather than treating accuracy as a primary task and secondary objectives as a separate velocity component to be projected into the null-space of the Jacobian, the proposed formulation absorbs the secondary concerns, joint-limit avoidance, posture regulation, and singularity-aware regularisation into a single cost function whose curvature is shaped by a configuration-dependent weight matrix. The structural consequences of this reformulation are examined explicitly in section 5.2.3, where the two formulations are placed side by side and the mechanism by which AWARE-IK eliminates the hard primary and secondary separation is made concrete at the equation level. The experimental evaluation in section 5.3 is then organised to assess whether the resulting solver delivers the accuracy, robustness, and stability properties that the formulation theoretically affords, with comparative findings against DLS-IK, SDLS-IK, and several contemporary multi-objective solvers reported in sections 5.3.1–5.3.5. Our simulation results, presented in section 5.3, indicate strong potential for industrial deployment subject to broader hardware validation under deployment

representative conditions.

The key contributions are: (1) We propose a novel method that combines DLS-IK with adaptive QP weights to generate kinematically feasible trajectories that avoid singularities and respect joint limits, enhancing adaptability and precision. (2) Through the development and validation of an adaptive weighting mechanism, the proposed method effectively balances accuracy and smoothness in multi-objective scenarios. By dynamically adjusting priorities and ensuring smooth transitions between competing objectives, the approach addresses limitations of traditional methods like DLS-IK and SDLS-IK, enabling more stable and precise performance in complex robotic tasks. (3) We evaluate the proposed AWARE-IK using comprehensive metrics for kinematic accuracy, efficiency, and robustness to noise, demonstrating its capability to avoid singularities, adhere to constraints, and ensure smooth trajectory execution through adaptive weighting.

5.2 Method

5.2.1 From damped pseudo-inversion to optimisation-based IK

Before introducing the AWARE-IK formulation proper, it is useful to briefly relate it to the IK machinery developed in Chapter 4, both to fix notation and to make explicit why a new formulation is required. The integrated DLS-IK with ProMPs framework of Chapter 4 computed joint updates through the damped pseudo-inverse,

$$\Delta\boldsymbol{\theta} = (\mathbf{J}^\top\mathbf{J} + \lambda^2\mathbf{I})^{-1}\mathbf{J}^\top\Delta\mathbf{x} \tag{5.1}$$

and addressed secondary objectives, joint-limit avoidance and posture regulation, through the null-space decomposition, in which a secondary joint-velocity vector is projected through $(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})$ so as not to perturb the primary end-effector motion. The damping in Eq. 4.3 enters as λ^2, following the standard damped-least-squares convention in which λ carries the scale of the Jacobian's singular values; AWARE-IK reuses this same regulariser by reference rather than introducing a new one, and retains the squared λ^2 form consistently throughout the present chapter, so that the recap above, the weighted damped-least-squares fallback of Eq. 5.8, and the adaptive-damping update of Eq. 5.13 all employ the squared convention. No equation in Chapters 4 or 5 therefore mixes the linear and squared damping forms, and the apparent squared-versus-linear discrepancy between the two chapters is resolved by this single, shared convention.

The closing analysis of Chapter 4 identified three structural limitations of this primary and secondary decomposition. First, the damping parameter λ is fixed and chosen a priori, with the well-known consequence that a value chosen to ensure stability near singularities introduces ac-

curacy loss in well-conditioned regions, while a value chosen for accuracy fails to regularise the pseudo-inverse where regularisation is most needed. Second, the secondary objective is specified as a fixed joint-velocity direction $\dot{\mathbf{q}}_{\text{secondary}}$ before the projection is applied, leaving no mechanism for the solver to renegotiate the balance between competing secondary concerns (limit proximity, posture deviation, smoothness) as the manipulator's state evolves. Third, the null-space projector $(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})$ enforces a hard separation between primary and secondary tasks that the secondary task is guaranteed not to affect the end-effector motion, but at the cost that the relative importance of the two task levels cannot itself be adjusted, even when the primary task is the cause of the difficulty (for example, when satisfying it would drive a joint past its limit).

AWARE-IK departs from this decomposition entirely. Instead of computing a primary velocity and projecting a secondary velocity into its null-space, the proposed formulation poses IK as a single constrained optimisation problem that among all joint-velocity updates that produce the desired end-effector displacement (the equality constraint), find the one that is least costly under a configuration-dependent quadratic metric (the objective). The Cartesian-accuracy concern is enforced as a hard constraint; the secondary concerns joint-limit avoidance, posture regulation, and singularity-aware regularisation are absorbed into the cost function through an adaptive weight matrix whose entries depend continuously on the manipulator's current configuration. The structural consequences of this reformulation, in particular the dissolution of the primary and secondary boundary and the resulting capacity to redistribute motion automatically as configurations evolve, are developed in section 5.2.2.

Fig. 5.1 summarises this formulation as an iterative pipeline. Starting from a target end-effector pose $\mathbf{x}_{\text{target}} \in SE(3)$, represented internally as a 4×4 homogeneous transformation and reduced to a six-dimensional error vector $\mathbf{e} \in \mathbb{R}^6$ comprising three translational and three axis-angle rotational components, AWARE-IK forms the adaptive weights $\mathbf{W}$ from joint-limit and centre factors (combined with base weights), then solves a regularised QP problem with $(\mathbf{J}, \mathbf{W}, \mathbf{e})$ to update the configuration $\mathbf{q} \rightarrow \mathbf{q}_{\text{updated}}$. If converged, a brief fine-tuning step returns $\mathbf{q}_{\text{final}}$; otherwise $\mathbf{W}$ is recomputed and the outer SLSQP cycle repeats. The dominant per-iteration cost is the SLSQP quadratic subproblem solve at $\mathcal{O}(n^3)$ for an n-DOF manipulator, yielding an overall complexity of $\mathcal{O}(K \cdot n^3)$, where K is the iteration count to convergence; an analysis is given in section 5.2.3. In the schematic, the right-side thumbnails depict $\mathbf{J}$, $\mathbf{W}$ (heatmaps), and $\mathbf{e}$, and the dashed snapshots show the target, intermediate, and final robot poses.

5.2.2 Algorithmic Overview

Before presenting the adaptive weighting mechanism and the adaptive damping strategy, it is useful to describe the AWARE-IK algorithm in terms of its iterative structure. This overview is intended to give the reader a conceptual scaffold onto which the technical details developed

in the subsequent subsections can be integrated; readers familiar with optimisation-based IK may proceed directly to section 5.2.3, but the structural choices that distinguish AWARE-IK from conventional weighted least squares and hierarchical IK solvers are most clearly seen at this overview. AWARE-IK operates as an iterative-corrective scheme. Given a target end-effector pose $\mathbf{x}_{\text{target}} \in SE(3)$ and an initial joint configuration $\mathbf{q}_0$, the solver maintains an iterate $\mathbf{q}_k$ that is updated at each outer iteration k by

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \mathbf{d}_k \tag{5.2}$$

where $\mathbf{d}_k \in \mathbb{R}^n$ is a joint-space displacement computed by solving a constrained Quadratic Programming subproblem at iteration k. The QP subproblem itself is constructed from three configuration-dependent quantities evaluated at $\mathbf{q}_k$: the manipulator Jacobian $\mathbf{J}(\mathbf{q}_k)$, the pose-error vector $\mathbf{e}(\mathbf{q}_k)$, and the adaptive weight matrix $\mathbf{W}(\mathbf{q}_k)$. The first two of these are standard ingredients of any Jacobian-based IK scheme; the third is the distinctive element of AWARE-IK, and it is what couples the solver's behaviour to the manipulator's instantaneous kinematic state. The development of all three quantities, together with the QP subproblem they define, is given in section 5.2.3.

The iterative structure carries three architectural features that together distinguish AWARE-IK from conventional approaches. The first is configuration-aware weighting. At each iteration, the weight matrix $\mathbf{W}(\mathbf{q}_k)$ is recomputed from the current iterate before the QP is solved; the matrix is not a fixed pre-set parameter but a function evaluated freshly at each step. As the manipulator's state evolves over the course of the iteration moving toward or away from joint limits, adopting a different posture, encountering a more or less well-conditioned Jacobian, the weight matrix tracks this evolution, and the QP that the solver actually solves at each iteration is therefore a different QP. This is what allows AWARE-IK to negotiate competing kinematic objectives smoothly rather than imposing a fixed prioritisation; the mathematical form of the weights is developed in section 5.2.3.1.

The second is adaptive regularisation. When the configuration approaches a singular region of the workspace, where the Jacobian loses rank and the unregularised QP becomes ill-conditioned, AWARE-IK responds by increasing the strength of a damping term that stabilises the optimisation. This damping is itself a function of the Jacobian's smallest singular value, again recomputed at each iterate; its form is developed in section 5.2.3.2. The combination of adaptive weighting and adaptive damping means that the optimisation landscape at each iteration is shaped by the manipulator's current configuration in two ways that the weighting modulates the direction of the gradient (which joints to move), while the damping modulates the magnitude of the step (how aggressively to attempt the move).

The third is a hybrid solver structure. AWARE-IK uses SLSQP as its primary solver for each QP

subproblem, chosen for its efficiency on problems of the present dimensionality. If SLSQP fails to converge, the solver falls back to a weighted damped least squares update using the same adaptive weight matrix and damping factor. Such convergence failures typically occur due to a severely ill-conditioned Jacobian, local infeasibility of the linearised equality constraint, or the exhaustion of the iteration budget. This fallback is not a different algorithm with a different cost landscape; it is the same optimisation problem solved by a more robust (if less efficient) numerical scheme. The hybrid structure ensures that the solver continues to make progress in regimes where SLSQP alone would fail, without introducing a discontinuity in the solver's prioritisation behaviour. The technical details of both the SLSQP step and the DLS fallback are given in section 5.2.3. The outer iteration terminates under one of two conditions: convergence, defined as $\|\mathbf{d}_k\| < \varepsilon$ for a configured threshold ε (typically 10^{-5} in our implementation), or budget exhaustion, defined as $k = K_{\max}$ for a maximum iteration count (typically 500). Upon convergence, the final iterate $\mathbf{q}_{\text{final}}$ may optionally be subjected to a brief fine-tuning step that addresses the cyclical nature of the wrist-axis joint (joint 7 in the Franka kinematic chain), folding any wraparound into the canonical range $[-\pi, \pi]$ to ensure the returned solution lies within valid joint bounds which addresses a known limitation of standard QP solvers when applied to revolute joints with full 2π rotational freedom. The fine-tuning step is detailed in section 5.2.3 alongside the QP machinery it follows.

A summary of the computational cost is useful at the overview level to anchor expectations before the technical details begin. Each outer iteration is dominated by the SLSQP quadratic subproblem solve, which costs $\mathcal{O}(n^3)$ for an n-DOF manipulator; the Jacobian recomputation and adaptive-weight update are each $\mathcal{O}(n)$ and therefore negligible by comparison. The total cost to convergence is $\mathcal{O}(K \cdot n^3)$ where K is the iteration count to convergence, empirically in the range 100–200 for the 7-DOF Franka platform used in our experiments. This complexity matches that of unweighted DLS-IK asymptotically, with the modest empirical wall-clock overhead arising from the constant factor of the QP subproblem rather than from a worse scaling law. A full per-component complexity breakdown, together with comparison against the principal benchmark solvers, is given at the close of section 5.2.3.

5.2.3 QP Formulation of AWARE-IK

Building on the architectural shift introduced in section 5.2.1, AWARE-IK poses each IK step as a constrained QP problem. The decision variable, denoted $\mathbf{d} \in \mathbb{R}^n$, is the joint-space displacement applied to the current configuration $\mathbf{q}_k$ at iteration k that is, the discrete step in joint angles produced by the optimisation, with dimensions of radians for revolute joints. The next iterate is then $\mathbf{q}_{k+1} = \mathbf{q}_k + \mathbf{d}_k$, and the outer cycle terminates when $\|\mathbf{d}_k\|$ falls below a configured threshold. This is to be distinguished from the joint-velocity formulations of Chapter 4 and section 2.3, in which the decision variable carried units of $\mathrm{rad/s}$ and was integrated over a

control timestep; AWARE-IK's outer cycle is iterative corrective rather than integrative, and $\mathbf{d}$ should be read as a position offset throughout this chapter. The optimisation problem is

$$\begin{aligned} \underset{\mathbf{d}\in\mathbb{R}^n}{\text{minimise}} \quad & f(\mathbf{d}) = \frac{1}{2}\mathbf{d}^\top \mathbf{W}(\mathbf{q}_k)\mathbf{d} \\ \text{subject to} \quad & g(\mathbf{d}) := \mathbf{J}(\mathbf{q}_k)\mathbf{d} - \mathbf{e}(\mathbf{q}_k) = \mathbf{0} \end{aligned} \tag{5.3}$$

where $\mathbf{W}(\mathbf{q}_k) \in \mathbb{R}^{n\times n}$ is the diagonal, configuration-dependent weight matrix whose entries are given by Eqs. 5.9–5.12, $\mathbf{J}(\mathbf{q}_k) \in \mathbb{R}^{6\times n}$ is the analytical manipulator Jacobian evaluated at the current iterate, and $\mathbf{e}(\mathbf{q}_k) \in \mathbb{R}^6$ is the six-dimensional pose-error vector defined in Eqs. 5.5–5.7 below. The constraint function $g(\mathbf{d})$ named in Eq. 5.3 is the left-hand side of the linearised end-effector equation, and the equality $g(\mathbf{d}) = \mathbf{0}$ enforces that the linearised Cartesian displacement produced by $\mathbf{d}$ matches the desired pose-error correction at the current iterate.

The pose-error vector $\mathbf{e}$ deserves explicit construction because its derivation determines the convergence behaviour of the outer cycle. The target pose $\mathbf{x}_{\text{target}} \in SE(3)$ is represented as a 4×4 homogeneous transformation,

$$\mathbf{x}_{\text{target}} = \begin{bmatrix} \mathbf{R}_d & \mathbf{p}_d \\ \mathbf{0}^\top & 1 \end{bmatrix}, \qquad \mathbf{R}_d \in SO(3),\ \mathbf{p}_d \in \mathbb{R}^3 \tag{5.4}$$

so that the desired position $\mathbf{p}_d$ is the upper-right 3×1 translation block and the desired orientation $\mathbf{R}_d$ is the upper-left 3×3 rotation block. The current end-effector pose, computed by forward kinematics from $\mathbf{q}_k$, has the analogous decomposition $\mathbf{p}_c$ and $\mathbf{R}_c$. The translational component of the pose error is then the simple Cartesian difference,

$$\mathbf{e}_{\text{pos}} = \mathbf{p}_d - \mathbf{p}_c \tag{5.5}$$

while the rotational component is constructed from the relative rotation $\mathbf{R}_c^d := \mathbf{R}_d^\top \mathbf{R}_c$, which describes the current orientation expressed in the target frame. The skew-symmetric part of this relative rotation,

$$\mathbf{S} = \tfrac{1}{2}\Big(\mathbf{R}_c^d - (\mathbf{R}_c^d)^\top\Big) \tag{5.6}$$

encodes the small-angle approximation of the matrix logarithm $\log_{SO(3)} : SO(3) \to \mathfrak{so}(3)$, from which the axis-angle vector is recovered through the vee operator and then transformed back into the world frame:

$$\mathbf{e}_{\text{rot}} = \mathbf{R}_d \begin{bmatrix} -S_{32} \\ -S_{13} \\ -S_{21} \end{bmatrix} \tag{5.7}$$

The full error vector is then $\mathbf{e} = \begin{bmatrix} \mathbf{e}_{\text{pos}}^\top & \mathbf{e}_{\text{rot}}^\top \end{bmatrix}^\top \in \mathbb{R}^6$. This construction is the small-angle linearisation of the SE(3) log map and is exact for small angular discrepancies; for large initial orientation errors it remains a valid descent direction in the sense that $\mathbf{e}_{\text{rot}}$ vanishes if and only if $\mathbf{R}_c = \mathbf{R}_d$, and it points monotonically toward zero error under sufficiently small step sizes. The outer SLSQP cycle, by computing $\mathbf{d}$ from a constraint linearised at each iterate, recovers the full nonlinearity of the rotational kinematics through repeated relinearisation, so that large orientation errors are handled correctly across multiple iterations even though each individual iteration uses the small-angle form.

The QP defined by Eq. 5.3 is solved by SLSQP, which performs a quasi-Newton update on the Lagrangian of the constrained problem and is well suited to problems of the present dimensionality ($n = 7$ joint variables, six equality constraints). However, SLSQP can fail to converge in three specific regimes: when the Jacobian is severely ill-conditioned (so the equality constraint becomes near-degenerate), when the equality constraint is locally infeasible (which can occur for targets outside the manipulator's reachable workspace), and when the iteration budget is exhausted before the tolerance is met. In each of these cases, the solver falls back to a weighted damped least-squares update that generalises the unweighted DLS form to the configuration-dependent setting:

$$\mathbf{d} = \left(\mathbf{J}^\top \mathbf{J} + \lambda^2 \, \mathbf{W}(\mathbf{q}_k) \right)^{-1} \mathbf{J}^\top \mathbf{e} \tag{5.8}$$

where λ is the damping factor, entering the regulariser squared as λ^2 exactly as in the damped least-squares update of section 4.2.2 (Eq. 4.3), and importantly $\mathbf{W}(\mathbf{q}_k)$ is the same adaptive weight matrix that appeared in the QP objective, recomputed at the current iterate's configuration $\mathbf{q}_k$ rather than at the failed candidate $\mathbf{q}_k + \mathbf{d}_{\text{trial}}$. The adaptive weighting is therefore preserved on fallback: the solver continues to penalise motion at joints near their limits, and to encourage motion toward the manipulator's neutral posture, even when SLSQP itself could not produce a feasible step. The weighted regularisation term $\lambda^2 \mathbf{W}(\mathbf{q}_k)$ generalises the isotropic damping $\lambda^2 \mathbf{I}$ of Eq. 4.3 to a configuration-aware metric, and in this sense Eq. 5.8 should be read as the natural weighted extension of Chapter 4's DLS update, with the weight matrix supplying the configuration awareness that the unweighted version lacks.

The hybrid SLSQP and DLS structure may, in principle, raise concerns about velocity discontinuities at the moment of transition between the two solvers. Three features of the construction

render this concern negligible in practice. First, both solvers use the same adaptive weight matrix $\mathbf{W}(\mathbf{q}_k)$, so the joint-level prioritisation is consistent across the transition; the fallback does not introduce a different cost landscape. Second, the controller exhibits small-step convergence behaviour near termination, so by the time a transition is triggered, the magnitudes of the updates are already small and any discontinuity is bounded by the residual step size rather than by the gross update scale. Third, SLSQP failures typically occur near active inequality constraints where the QP itself is becoming ill-posed and the DLS fallback still minimises the same task-space objective, so its update is close to the QP solution that the failing SLSQP iteration was attempting to compute. The transition is, in this sense, a graceful degradation rather than a switch between qualitatively different control regimes.

Three implementation choices in this hybrid structure warrant explicit justification, as each was raised as a point of clarification. The first concerns the choice of SLSQP as the primary solver in preference to dedicated convex-QP libraries such as OSQP or qpOASES. The distinction is that OSQP and qpOASES are highly efficient solvers for a single convex QP with fixed, linear constraints, whereas the IK problem solved here is not a single convex QP but a sequence of QP subproblems whose equality constraint $\mathbf{J}(\mathbf{q}_k)\mathbf{d} = \mathbf{e}(\mathbf{q}_k)$ is the linearisation, at the current iterate, of the genuinely nonlinear forward-kinematics map. Solving the IK problem therefore requires an outer relinearisation (sequential quadratic programming) cycle regardless of which inner QP solver is used; a specialised QP library such as OSQP or qpOASES would have to be wrapped inside precisely such a cycle, reproducing the structure that SLSQP already provides natively. SLSQP integrates the quasi-Newton update of the Lagrangian, the bound and equality constraint handling, and the inner QP solve within a single, well-tested routine available in the SciPy optimisation suite, with no external solver dependency and no manual management of the relinearisation cycle. For a problem of the present dimensionality ($n = 7$ joints, six equality constraints), the per-iteration QP is small and the constant-factor advantage of a specialised QP back-end is negligible relative to the cost of the surrounding forward-kinematics and Jacobian evaluations; the integrated handling of the nonlinear constraint is the decisive consideration. Adopting OSQP or qpOASES as a drop-in inner solver within the same SQP outer cycle is nonetheless a reasonable engineering optimisation for higher-DOF manipulators, and is noted as such.

The second concerns how often the weighted-DLS fallback of Eq. 5.8 is actually invoked. The fallback fires only when SLSQP reports non-convergence on a given outer iteration, which the failure-mode breakdown of Table 5.3 shows to be concentrated in specific kinematic regimes rather than uniformly distributed. On the well-conditioned reaches (Nominal, Extended, Overhead, Behind Robot), SLSQP converges without recourse to the fallback on the substantial majority of runs, and the fallback acts as a safety net rather than a routine code path; it becomes materially active only as the manipulator approaches the near-singular and joint-limit-saturated configurations (Low Reach, Near Singularity), where the linearised equality constraint becomes ill-

conditioned or locally infeasible. This regime-dependent behaviour is precisely the design intent: the fallback exists to preserve forward progress in exactly the configurations where SLSQP is expected to struggle, and to remain dormant elsewhere so that the solver's nominal accuracy and convergence characteristics are governed by SLSQP.

The third concerns what real-time means in the context of this solver, and the control frequencies that the measured solve times imply. AWARE-IK is a cycle kinematic solver that produces joint-position targets; it is not, and is not intended to be, an inner servo-rate routine. The Franka platform's inner torque-control cycle runs at 1 kHz and is supplied by the manufacturer's controller; AWARE-IK sits above this cycle, generating the position targets that the inner cycle tracks. The relevant real-time budget is therefore that of the outer planning/reaching cycle, conventionally 100-200 ms per target (5-10 Hz) for the reactive-reaching and trajectory-replay applications targeted in this chapter. Against this budget, the measured hardware solve times (41-73 ms for single-axis reaches, Table 5.6) place AWARE-IK comfortably within a 10-20 Hz cycle regime for typical targets, while the heavy-tailed worst case at 80 waypoints (mean 282.5 ms) falls below 5 Hz and is consequently appropriate for non-time-critical planning rather than reactive control. The Jacobian-based comparators (SDLS-IK, DLS-IK) solve in ≈15 ms (≈65 Hz) and therefore retain an advantage in latency-critical cycles where the sub-0.15 mm accuracy difference is immaterial; this trade-off is the one stated explicitly in section 5.3.5.

It is instructive to contrast the QP with the null-space projection approach employed in Chapter 4, as the difference between these two formulations is the principal structural innovation of AWARE-IK. The null-space approach computes the total joint motion as the sum of a primary-task component and a secondary-task component that has been projected through the operator $(\mathbf{I} - \mathbf{J}^{+}\mathbf{J})$ onto the null-space of the Jacobian. This imposes a geometric separation between the two task levels that the projector guarantees that the secondary task lies in the kernel of $\mathbf{J}$ and therefore cannot perturb the primary end-effector motion. The trade-off between primary and secondary objectives is governed by the structure of the projector itself, which depends only on the current Jacobian and not on any task-specific tuning. This is at once the strength and the limitation of the formulation. Its strength is that primary-task accuracy is preserved exactly, regardless of how aggressively the secondary task is pursued. Its limitation is that the relative importance of primary and secondary concerns cannot be modulated by the solver, the hierarchy is hard coding into the equation by the projector, with no mechanism for softening or rebalancing the prioritisation as the manipulator's configuration evolves. In particular, when the primary task itself becomes problematic, for example, when the Jacobian approaches singularity or when satisfying the primary task would drive a joint past its limit the null-space formulation has no graceful response that it either continues to enforce the primary task at the cost of unsafe joint motion, or it must be augmented by an external mechanism (joint clipping, fallback damping, task switching) that introduces discontinuities into the velocity.

The QP dissolves this hard separation. The Jacobian equality constraint $\mathbf{Jd} = \mathbf{e}$ plays the role of

primary-task enforcement, ensuring that the chosen joint update produces the desired end-effector displacement at the linearisation point. The secondary concerns joint-limit proximity, posture deviation, and singularity-aware regularisation are not separately computed and projected, but are instead encoded into the cost function through the adaptive weight matrix $\mathbf{W}(\mathbf{q}_k)$, whose entries are continuous functions of the current configuration (see section 5.2.3). The consequence is that the solver no longer chooses a fixed primary then secondary ordering, but rather selects, among all joint displacements satisfying the Cartesian constraint, the one that is least costly under the configuration-aware metric induced by $\mathbf{W}$. When a joint approaches its limit, the corresponding diagonal entry $w_{ii}(\mathbf{q}_k)$ grows (per the limit-proximity factor), making any update component along that joint correspondingly expensive in the cost function; the solver responds by automatically redistributing motion to less constrained joints, without any explicit projection step. This behaviour is observed directly in the hardware validation reported in section 5.3.5, where Joint 4 operating near its lower limit at -2.356 rad consistently exhibits elevated weight values in the range 1.27–1.54 and the solver redistributes motion accordingly to the remaining six joints. When the manipulator is far from any limit, the weights remain near their base values and the formulation reduces approximately to a weighted minimum-norm IK solution. The transition between these regimes is smooth, because $\mathbf{W}(\mathbf{q}_k)$ varies continuously with configuration. No null-space projector is required, no task-switching logic is needed, and no hard discontinuities arise.

This structural difference is what allows AWARE-IK to operate effectively in regions of the configuration space where the hard null-space formulation either fails or requires external augmentation: in such regions, the adaptive weighting automatically shifts the cost landscape in a manner that the projector-based formulation cannot. The experimental comparison reported in section 5.3 quantifies this advantage along several axes, including the joint limit aware redistribution behaviour observed on the physical Franka platform.

5.2.3.1 Adaptive weight calculation

The adaptive weighting mechanism in AWARE-IK dynamically modulates joint priorities based on the instantaneous kinematic configuration, drawing from established principles in potential field theory and biological motor control. Each joint of the manipulator receives an individual scalar weight, and we use $i \in \{1, \ldots, n\}$ throughout this subsection to denote the joint index, with $n = 7$ for the Franka robot platform used in our experiments. The comprehensive weighting formulation for joint i then takes the form:

$$w_i(\mathbf{q}) = w_{\text{base},i}\, f_{\text{limit},i}(\mathbf{q})\, f_{\text{center},i}(\mathbf{q}) \tag{5.9}$$

where $w_{\text{base},i}$ represents the nominal weight for joint i, $f_{\text{limit},i}(\mathbf{q})$ is the limit-proximity factor (Eq. 5.11), and $f_{\text{center},i}(\mathbf{q})$ is the centre-deviation factor (Eq. 5.12). The base weights $\{w_{\text{base},i}\}_{i=1}^{n}$ encode the relative importance assigned to each joint in the absence of any configuration-dependent modulation, and are typically chosen to reflect the differing roles of the joints in the kinematic chain, in our implementation, distal joints (those nearer the end-effector) receive slightly lower base weights than proximal joints, on the principle that they have less influence on overall manipulator posture and can be permitted greater freedom of motion. The configuration-dependent factors $f_{\text{limit},i}$ and $f_{\text{center},i}$, both detailed in the equations that follow, modulate this base assignment according to the proximity of joint i to its operational limits and its deviation from the manipulator's neutral configuration, respectively.

The scalar per-joint weights $\{w_i(\mathbf{q})\}_{i=1}^{n}$ are assembled into the diagonal weight matrix that appears in the QP cost function and the DLS fallback of section 5.2.3:

$$\mathbf{W}(\mathbf{q}) = \operatorname{diag}\big(w_1(\mathbf{q}),\ w_2(\mathbf{q}),\ \ldots,\ w_n(\mathbf{q})\big) \in \mathbb{R}^{n\times n}. \tag{5.10}$$

The diagonal structure reflects the modelling assumption that joint-level penalties are independent of one another, and it preserves the computational efficiency of the QP subproblem by keeping the cost-function Hessian sparse.

The limit proximity factor has a piecewise linear formulation:

$$f_{limit,i}(q) = 1 + 2\left|\frac{q_i - q_{lower,i}}{q_{upper,i} - q_{lower,i}} - 0.5\right| \tag{5.11}$$

This piecewise linear function creates a V-shaped response that increases linearly as joints approach their limits from either direction. The formulation is inspired by artificial potential field methods, where repulsive forces increase as boundaries are approached. While the function is continuous throughout the joint space, the gradient experiences a discontinuity at the midpoint (normalised position $= 0.5$), creating a sharp transition that provides immediate feedback when joints begin moving away from their central range. This piecewise linear approach offers computational efficiency while maintaining effective limit avoidance.

The centre deviation factor maintains a linear relationship:

$$f_{center,i}(q) = 1 + \frac{|q_i - q_{center,i}|}{q_{upper,i} - q_{lower,i}} \tag{5.12}$$

This linear formulation derives from principles of minimum effort observed in biological motor control studies, where joint configurations tend to minimise deviation from neutral positions pro-

portionally when unconstrained. The linear relationship provides consistent bias toward ergonomic configurations without overwhelming primary task objectives, ensuring that the manipulator maintains favourable postures when multiple solutions exist.

The multiplicative structure of Eq. 5.9 creates a composite weight that respects both safety constraints and efficiency principles. This formulation ensures that either factor can dominate when necessary near joint limits, $f_{limit,i}(q)$ provides strong guidance regardless of centre deviation, while in the middle of the workspace, $f_{center,i}(q)$ subtly optimises the configuration. The multiplicative combination prevents the numerical conditioning issues that arise in additive formulations when one factor becomes very large, maintaining optimisation stability throughout the workspace.

This adaptive weighting scheme distinguishes AWARE-IK from fixed-weight approaches by eliminating the need for manual parameter tuning across different tasks and configurations. The theoretical foundation ensures predictable, stable behaviour while the mathematical formulation guarantees computational efficiency suitable for real-time implementation.

5.2.3.2 Adaptive Damping Factor

The adaptive damping strategy in AWARE-IK derives from regularisation theory and the Levenberg-Marquardt algorithm, providing theoretically optimal stabilisation for ill-conditioned IK problems. The damping factor formulation:

$$\lambda = \frac{\sigma_{\min}}{\sigma_{\min}^2 + \alpha^2} \tag{5.13}$$

represents the solution to the regularisation problem that minimises the trade-off between solution accuracy and stability. The specific functional form emerges from the analysis of the regularised normal equations. When solving the damped least squares problem:

$$\Delta q = (\mathbf{J}^T\mathbf{J} + \lambda^2\mathbf{I})^{-1}\mathbf{J}^T e \tag{5.14}$$

The optimal damping factor should provide maximum stabilisation when the Jacobian approaches singularity ($\sigma_{\min} \to 0$) while introducing minimal perturbation when the configuration is well-conditioned. The formulation in Eq. 5.13 achieves this through its asymptotic behaviour, as $\sigma_{\min} \to 0$, $\lambda \to 0$, providing bounded damping that prevents numerical overflow. Conversely, when $\sigma_{\min} >> \alpha$, $\lambda \approx 1/\sigma_{\min}$, ensuring that damping decreases proportionally with improved conditioning.

The quadratic term in the denominator serves a critical theoretical purpose beyond numerical stability. Based on Tikhonov regularisation, this formulation corresponds to the optimal regular-

isation parameter that minimises the expected mean square error of the solution under certain noise assumptions. The parameter α acts as a threshold that determines the transition between dominated and well-conditioned regimes, with its value related to the expected measurement noise and modelling uncertainties in the system.

This adaptive damping strategy provides several theoretical advantages over fixed damping approaches. First, it maintains the convergence rate of undamped methods in well-conditioned configurations while ensuring stability near singularities. Second, the smooth transition prevents the solution discontinuities that occur with threshold-based switching schemes. Third, the formulation preserves the positive definiteness of the augmented Hessian matrix throughout the workspace, guaranteeing that the optimisation problem remains convex.

5.2.4 Metrics

The comprehensive evaluation of an IK solver requires a multifaceted approach that captures both solution quality and computational characteristics, and that probes the specific theoretical claims the proposed method has been designed to support. The metrics are introduced to compare AWARE-IK against six contemporary IK solvers: DLS-IK, SDLS-IK, TRAC-IK in its Speed and Distance variants (which represent the current state-of-the-art in dual-algorithm hybrid solvers), Relaxed-IK (a weighted-sum nonlinear-optimisation approach), and Bio-IK (a memetic evolutionary global-search solver). These six comparators were chosen to span the principal methodological families of contemporary IK research, from classical regularised pseudo-inversion through optimisation-based and population-based approaches, so that AWARE-IK can be situated within rather than alongside the field.

The metrics are organised into three categories matched to the three theoretical claims. The first category, Solution Accuracy Metrics assesses how closely the algorithm achieves the desired pose, and is constructed to test the claim that AWARE-IK's configuration-aware cost function (Eq. 5.3) and its adaptive damping factor (Eq. 5.13) together deliver superior Cartesian-pose precision compared with the fixed-parameter regularisation of conventional DLS and SDLS. The second category, Computational Efficiency Metrics measures the resources required to obtain solutions, and is constructed to test the claim that AWARE-IK's hybrid solver structure achieves real-time compatible convergence despite the additional cost of the QP subproblem. The third category, Numerical Robustness Metrics evaluates the stability and reliability of the computational process, and is constructed to test the claim that the adaptive weighting mechanism genuinely encodes the manipulator's joint-limit and posture geometry rather than acting merely as a generic damping increase. Each metric within each category is described below, together with its statistical treatment across the trial sets. The metrics are evaluated across the test scenarios, which range from standard workspace reaches to configurations near singularities and behind the robot base,

with each scenario chosen to expose a different aspect of solver behaviour.

Relationship to the evaluation methodology of Chapter 3. A reader familiar with the benchmarking study of Chapter 3 will observe that the metric suite deployed here differs from the three-metric framework (time efficiency, success rate, and parameter sensitivity) used to evaluate sampling-based motion planners in section 3.3.4. This divergence is deliberate, and reflects the fact that the two chapters evaluate fundamentally different computational objects rather than the same object measured under different conventions. The two suites share their two most universal metrics, success rate (the proportion of trials that yield a valid solution) and time efficiency (the wall-clock cost of obtaining one), both of which are unit analysis independent and therefore reasonably required of any solver evaluation in the thesis. Beyond these common foundations, however, the two metric suites diverge in three principled ways.

First, Chapter 3 evaluates trajectory planning over a workspace, in which the planner either reaches the goal region (defined by a tolerance) or does not, because successful trajectories by construction satisfy the goal conditions within the supplied IK seeding tolerance. Chapter 5, by contrast, evaluates single-pose solving over a configuration set, in which Cartesian-pose accuracy is itself the quantity being optimised. Position error and orientation error metrics are therefore meaningful in Chapter 5 in a way that they are not in Chapter 3, reporting them for a sampling-based planner would yield either zero (for all successes) or undefined (for failures), neither of which is informative.

Second, the Jacobian condition number and joint-space path-smoothness metrics introduced here are local differential kinematic measures, evaluated at the configurations produced by the IK solver. These have no natural counterpart in motion-planning evaluation, because a motion planner produces a trajectory (a time-parameterised curve in configuration space) rather than a single configuration, and the analogous trajectory-level quantities would require a different and considerably more involved analysis. They are introduced in Chapter 5 specifically to probe the singularity-handling and joint-limit-aware properties that the adaptive weighting and adaptive damping of the proposed method are designed to deliver; without them, the chapter could measure whether AWARE-IK reaches a target but not how cleanly it does so.

Third, Chapter 3's parameter sensitivity dimension which swept the OMPL 'range' parameter to characterise planner robustness to a single dominant tuning knob is not directly replicated here as a primary evaluation axis, because AWARE-IK's adaptive weighting and adaptive damping mechanisms are themselves the methodological response to the criticism that fixed-parameter IK requires manual tuning. The reduction of parameter sensitivity is an aim of the AWARE-IK design rather than a property to be measured against external comparators with a single tunable parameter. A supplementary parameter-sensitivity analysis is applied to the residual tunable parameters of AWARE-IK (the convergence tolerance ε, the maximum iteration count $K_{\max}$, and the linear damping coefficient λ of the DLS fallback), both to demonstrate that the proposed

method delivers on its parameter robustness claim and to maintain methodological symmetry with the benchmarking philosophy of Chapter 3.

Together, these three principled differences reflect a single underlying commitment that the metric suite for each empirical chapter is chosen to fit the unit of evaluation, with success rate and time efficiency serving as the cross-chapter common ground that allows the broader trends, for example, the empirical observation that all solvers in the thesis exhibit a degradation pattern as environmental or kinematic difficulty increases to be compared across chapters even when the chapter-specific metrics differ.

Solution Accuracy Metrics

- **Success Rate.** This fundamental metric quantifies the proportion of successful solution attempts for each IK method under different scenarios. Success is defined by the ability of the algorithm to achieve the target pose within predefined position and orientation tolerances. A higher success rate indicates a more robust IK solution capable of handling diverse kinematic configurations and avoiding local minima or singularities.

- **Position Error.** The positional accuracy of each solution is measured as the Euclidean distance between the target and achieved end-effector positions. This metric, expressed in metres, provides a direct assessment of the algorithm's ability to reach the desired location in Cartesian space. Statistical summaries including mean, standard deviation, and median values across multiple trials enable robust comparison between methods while accounting for variability in performance.

- **Orientation Error.** Rotational accuracy is quantified through the minimal rotation required to align the achieved orientation with the target. This error is characterised by both an axis and angle representation, where the angle indicates the magnitude of the rotational discrepancy. Lower angular error values demonstrate superior orientation control capabilities. The distribution of these errors across trials is summarised through statistical measures to facilitate method comparison.

- **Worst-Case Orientation Component Analysis.** Although the aggregate orientation error yields a single accuracy score, decomposing it into Euler components roll, pitch, and yaw reveals how error is distributed. For each successful solution, we record the maximum deviation on every axis [118]. Examining these axis-wise peaks exposes solver-specific biases and highlights the anisotropic nature of orientation control in redundant manipulators. This insight supports application-driven solver choice, allowing engineers to prioritise solvers that deliver the tightest control on the rotational axes that matter most.

- **Parameter Sensitivity.** Following the methodology of Chapter 3, where planner performance was characterised against the OMPL `range` parameter, parameter sensitivity quan-

tifies how the solver's headline metrics respond to deliberate perturbations of its principal tuning knobs. For AWARE-IK these are the damping factor λ, the convergence tolerance scale ε, and the iteration budget $K_{\max}$. A solver that exhibits wide plateaus on these axes does not require fine hand-tuning to deploy; a solver whose performance collapses under small parameter perturbations does. This metric is reported as the width of the parameter interval over which success rate, mean compute time, and residual pose error remain statistically indistinguishable from their values at the published defaults.

Computational Efficiency Metrics

- **Iterations.** The number of computational steps required to reach a solution serves as a primary indicator of algorithmic efficiency. It captures the convergence behaviour of each method, with fewer iterations generally indicating more direct convergence paths. However, interpretation requires caution, as early termination due to failure may produce misleadingly low iteration counts. Therefore, iteration statistics are computed only over successful solution attempts to ensure meaningful comparison.

- **Computation Time.** Total elapsed time from algorithm initialisation to termination provides a practical measure of computational burden. Measured in seconds, this metric encompasses all computational overhead including matrix operations, constraint checking, and convergence testing. Statistical analysis of computation times across trials, supplemented by ANOVA testing, identifies statistically significant performance differences between methods. This temporal characterisation proves essential for assessing real-time feasibility in control applications.

- **Time-to-First-Valid Solution.** Distinguished from total computation time, this metric measures the elapsed time until the first configuration satisfying all specified tolerances is discovered [119]. This measure captures the solver's ability to rapidly identify viable solutions before engaging in potentially lengthy refinement procedures. In cycle control systems operating under strict timing constraints, the availability of any valid solution within the control cycle period takes precedence over solution optimality. This temporal measure reveals the inherent trade-off between solution quality and computational latency, proving particularly valuable in safety-critical applications where delayed responses may compromise system stability.

Numerical Robustness Metrics

- **Jacobian Condition Number.** The condition number of the Jacobian matrix serves as a fundamental indicator of numerical stability in IK computations. Defined as the ratio

of the largest to smallest singular values ($\kappa(J) = \sigma_{\max}/\sigma_{\min}$), this metric quantifies the sensitivity of the solution to small perturbations [120]. A low condition number approaching unity indicates a well-conditioned problem where small changes in desired end-effector velocity produce proportionally small changes in joint velocities. Conversely, high condition numbers signal proximity to kinematic singularities where numerical solutions become unreliable. Monitoring this metric at the solution configuration provides crucial insight into the numerical robustness of the computed result and the manipulator's dexterity at that configuration.

- **Path Smoothness.** The efficiency of the solution trajectory through joint configuration space is quantified by the ratio of total path length traversed during the iterative solution process to the direct Euclidean distance between initial and final configurations. This dimensionless measure characterises the solver's convergence behaviour, with unity representing the theoretical optimum of direct interpolation from seed to solution. Values exceeding unity indicate circuitous paths that may result from competing optimisation objectives such as obstacle avoidance, joint limit satisfaction, or secondary task fulfilment through null-space projection. This metric proves particularly valuable in assessing computational efficiency and predictability across different IK formulations, as excessive path deviations correlate with increased computational effort and potential wear in physical implementations.

5.2.5 Robot Scenario Setting

To comprehensively evaluate the performance of our adaptive weighted QP IK solver, we designed a set of five diverse scenarios that challenge different aspects of robotic arm manipulation. The benchmarking study of Chapter 3 will note that the five scenarios described here do not directly incorporate the three environmental scenarios used there, the flat surface with scattered obstacles, the semi-closed box, and the drawer-like enclosure of Chapter 3. As with the metric divergence discussed previously, this scenario divergence is deliberate and reflects the fact that the two chapters evaluate fundamentally different computational objects rather than the same object measured in different conditions. Chapter 3's scenarios are scenarios of the environment that the independent variable being swept is the complexity of the surrounding obstacle field, and each scenario tests a sampling-based motion planner's ability to find a collision-free trajectory through progressively more constrained free-space. The IK solver in Chapter 3 plays a supporting role that it seeds the planner's search with candidate joint configurations, but the dimensions along which Chapter 3's scenarios vary (clutter density, workspace confinement) are dimensions that exercise the planner's search behaviour rather than the solver's numerical-kinematic behaviour. Chapter 5's scenarios, by contrast, are scenarios of the target pose that the independent variable is the configuration the manipulator must adopt to satisfy the pose specification, with each of

the five reach scenarios chosen to exercise a specific aspect of IK solver behaviour. The Nominal Reach establishes a baseline; the Extended Reach pushes the manipulator toward its workspace boundary, where the Jacobian becomes ill-conditioned in a manner that the adaptive damping is designed to handle; the Overhead Reach and Low Reach exercise the joint-limit avoidance behaviour that the adaptive weighting is designed to provide; and the Complex Orientation scenario tests the rotational error computation of Eq. 5.7 under demanding angular targets. Two further configurations, Near Singularity and Behind Robot are used to probe edge cases of solver robustness.

The result is that the thesis carries three distinct families of experimental scenarios, each calibrated to the evaluation needs of a specific contribution chapter: environmental scenarios for motion-planner benchmarking; kinematic scenarios for IK-solver pose-accuracy evaluation; and operational scenarios for physical deployment validation. The three families share a common evaluation philosophy, each begins with a baseline scenario and adds difficulty along the dimensions most relevant to the chapter's object of evaluation but their specific contents are different, because their objects of evaluation are different. Forcing a single scenario taxonomy across both chapters would degrade the evaluation rather than strengthening it, applying Chapter 3's environmental scenarios as IK-only test cases would either reduce them to single-pose targets (losing the trajectory planning character that motivated them) or require integration with a motion planner.

These scenarios, implemented in our software, represent a range of common and complex tasks that a 7-DOF robotic manipulator might encounter in real-world applications. The "Nominal Reach" scenario serves as a baseline, testing the solver's performance in a standard reaching position. The "Extended Reach" scenario pushes the limits of the robot's workspace, combining distance with rotation. The "Overhead Reach" scenario evaluates the solver's ability to handle high reaches with significant rotation, which is often challenging due to potential singularities. The "Low Reach" scenario tests the solver's performance in lower regions of the workspace, incorporating combined rotations that may stress joint limits. Finally, the "Complex Orientation" scenario challenges the solver with a target pose involving intricate rotational components, testing its ability to achieve precise end-effector orientations. Each scenario is defined by a specific target pose, combining position and orientation, allowing us to assess the solver's accuracy, efficiency, and robustness across different manipulation tasks. This diverse set of scenarios enables a thorough comparison between our weighted QP IK solver and the baseline uniform-weighted approach, providing insights into the strengths and potential limitations of our proposed method.

5.3 Experiments

In our experiments, we evaluate AWARE-IK's ability to handle multi-objective priority balancing, where the solver must simultaneously satisfy competing kinematic requirements across di-

verse reach configurations. The experiments are designed to simulate real-world conditions and challenges that robotic manipulators commonly encounter, particularly focusing on singularity avoidance, precise end-effector control, and efficient computation. By comparing with SDLS-IK, DLS-IK, TRAC-Speed, TRAC-Distance, Relaxed-IK and Bio-IK, we aim to demonstrate the benefits of dynamic weight adjustment in improving accuracy, stability, and responsiveness. Key performance metrics such as position error, orientation error, convergence speed, and computation time are used to assess the solver's capability in achieving high-fidelity task execution while effectively managing complex scenarios. This evaluation provides critical insights into the solver's robustness and adaptability, showcasing its potential as a solution for real-time, multi-constraint robotic tasks.

The experimental programme is structured to substantiate three claims that follow from the theoretical formulation of section 5.2. First, that the unified cost function treatment of competing kinematic objectives delivers lower Cartesian-pose error than the regularised pseudo-inversion solvers (DLS-IK and SDLS-IK) across a range of reaching configurations. Second, that the configuration-dependent weighting mechanism encodes the manipulator's joint-limit geometry and posture constraints directly into the optimisation, with the consequence that the resulting joint configurations remain within safe operational margins without requiring external clipping or projection logic. Third, that the resulting solver retains the numerical robustness and computational tractability necessary for real-time operation, both in simulation and on the physical Franka robot reported in section 5.3.5. The experimental design, the comparator solvers, and the measurement protocol described in the remainder of this section are constructed to support or refute each of these claims independently.

5.3.1 Performance Evaluation

We evaluated seven IK solvers (AWARE-IK, SDLS-IK, DLS-IK, TRAC-Speed, TRAC-Distance, Relaxed-IK, and Bio-IK) across four reach scenarios representing diverse manipulation tasks: nominal (standard workspace), extended (near boundaries), overhead (above base), and low (near ground level). Additionally, we tested near-singularity and behind-robot configurations to assess robustness in challenging kinematic conditions. The TRAC-IK variants employ a dual-algorithm architecture combining Newton-based and SQP solvers, with TRAC-Speed prioritising rapid convergence and TRAC-Distance minimising joint displacement from seed configurations. Relaxed-IK implements relaxation-based optimisation for smooth trajectory generation with constraint handling, while Bio-IK utilises bio-inspired evolutionary algorithms for global optimisation.

Position-error distributions for the seven solvers across all scenarios are shown in Fig. 5.3. Most solvers achieve sub-millimetre accuracy on successful runs, with AWARE-IK and DLS-IK setting the lowest medians (10^{-6}–10^{-5} m) and SDLS-IK close behind. The TRAC-IK variants (TRAC-

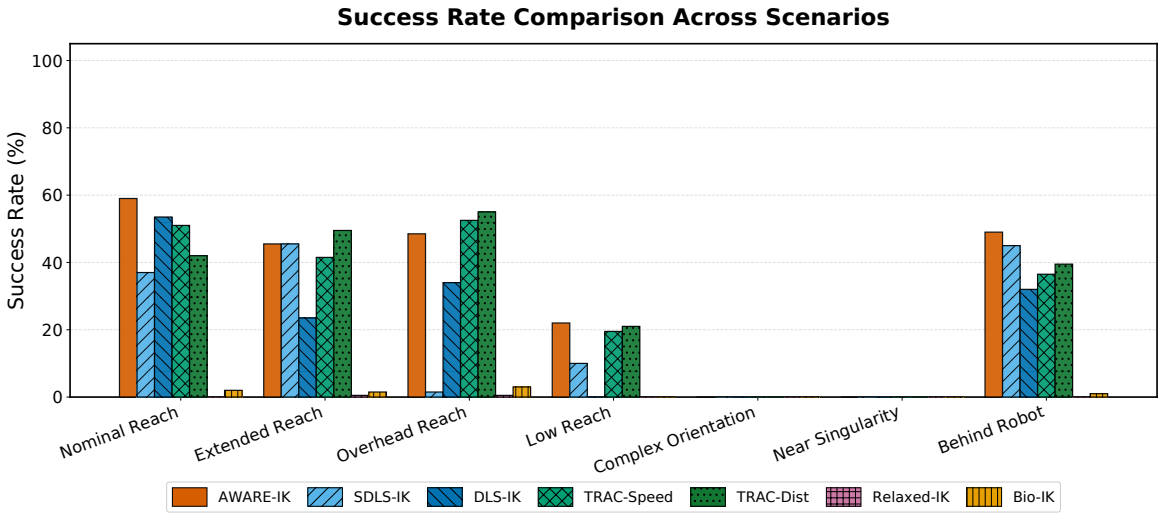


Figure 5.2: Success rate of the seven IK solvers (AWARE-IK, SDLS-IK, DLS-IK, TRAC-Speed, TRAC-Distance, Relaxed-IK, and Bio-IK) across the evaluation scenarios.

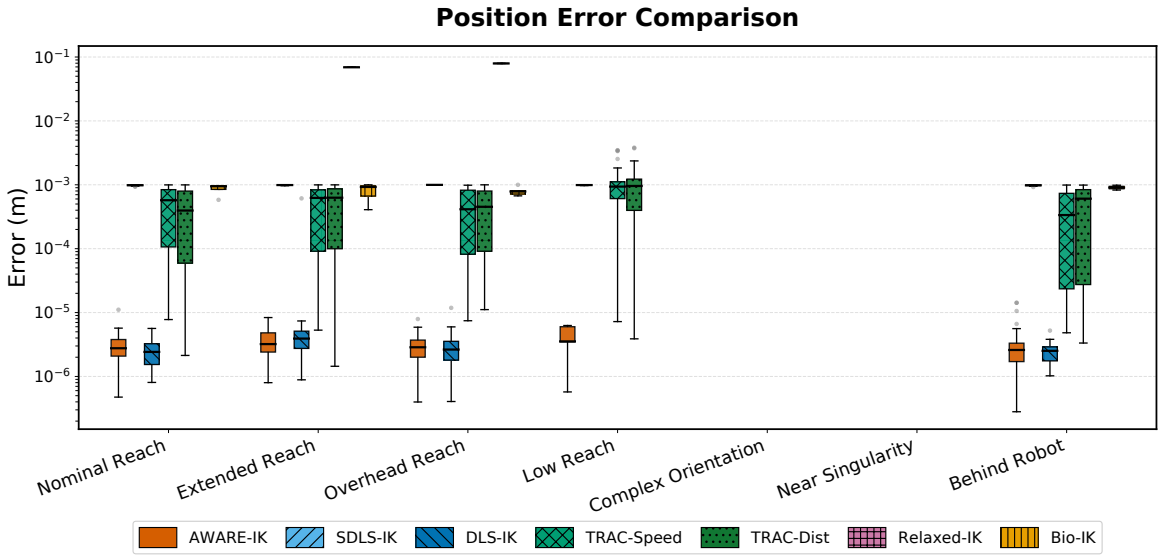


Figure 5.3: Position error distributions of the seven IK solvers across the evaluation scenarios.

Speed and TRAC-Distance) and Bio-IK show higher dispersion in Fig. 5.3, with occasional 10^{-3}-metre outliers visible in the upper whiskers of their distributions. Relaxed-IK's position-error distribution is consistent with the tolerance criterion mismatch discussed in the preceding paragraphs, the runs that registered as successes lie within the 1 mm envelope, while runs that fell outside this envelope are absent from Fig. 5.3 by definition of the success criterion. Orientation-error distributions are reported in Fig. 5.4. The picture for orientation errors is generally similar to that for position errors that DLS-IK, SDLS-IK, AWARE-IK, and the TRAC variants achieve 10^{-3} to 10^{-2} rad medians, with Bio-IK trending slightly larger. Two specific features of Fig. 5.4 warrant explicit comment because their magnitudes extend toward or beyond the upper limit

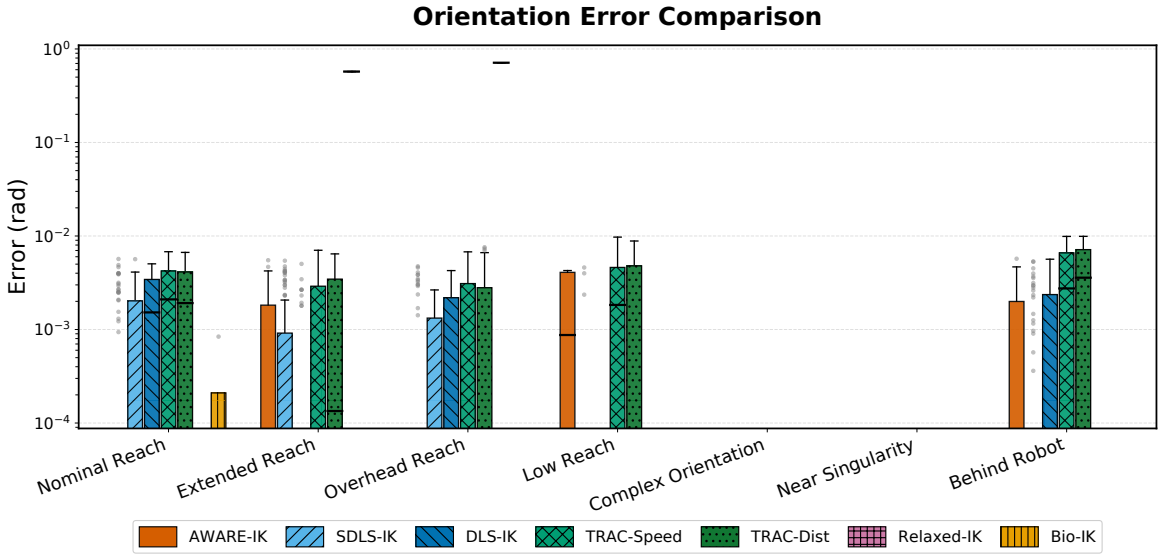


Figure 5.4: Orientation error distributions of the seven IK solvers across the evaluation scenarios.

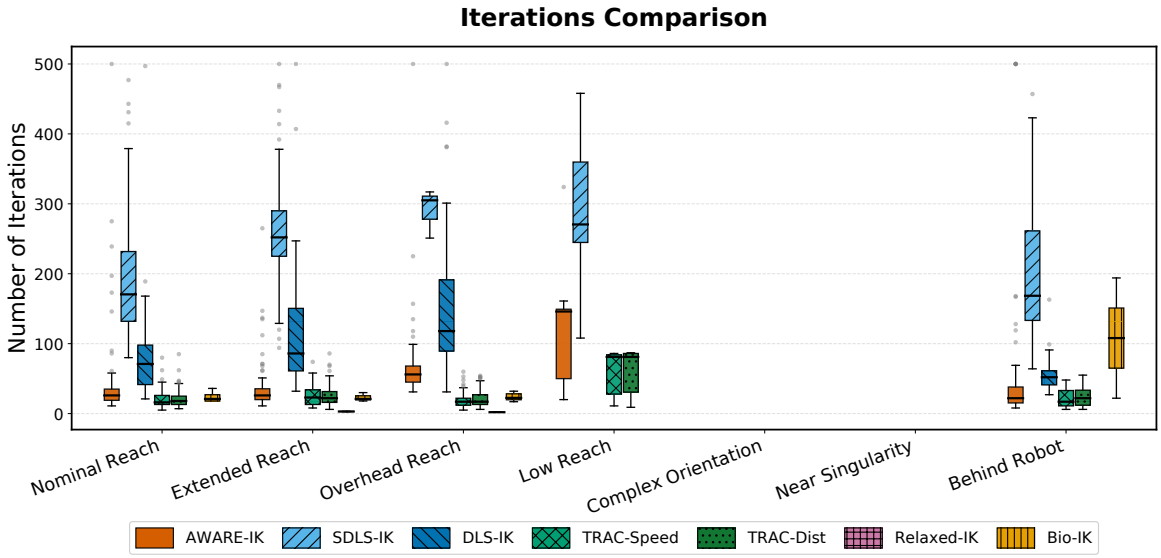


Figure 5.5: Iteration count distributions of the seven IK solvers across the evaluation scenarios.

of the figure's vertical axis. First, Relaxed-IK's orientation errors in the Near Singularity and (where evaluated) Extended Reach scenarios reach approximately 0.6 rad which is substantially above the sub-0.01-rad range of the other solvers, and visible in Fig. 5.4 as the tallest bars and points for Relaxed-IK in those scenarios. This is the orientation side manifestation of the same tolerance criterion mismatch that affects Relaxed-IK's success rate (discussed earlier in this section). Second, AWARE-IK exhibits a near-singularity orientation breakdown in the Extended Reach condition, producing an orientation spike on the order of 0.5–0.7 rad; this outlier, not seen in the other AWARE-IK results, is reported honestly here because the chapter's contribution claim is best served by transparent acknowledgement of failure modes alongside successes, and

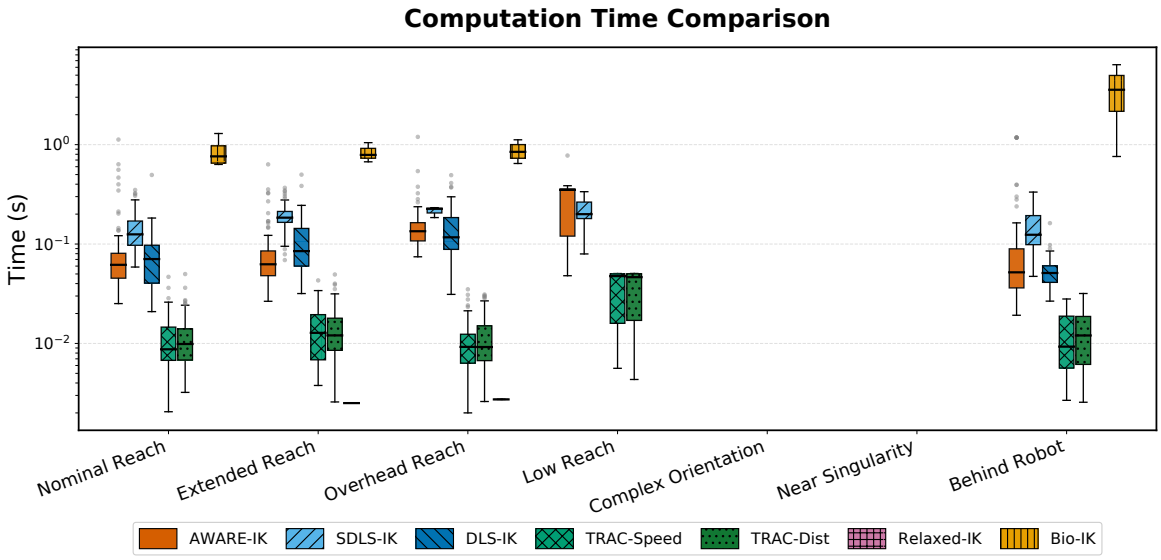


Figure 5.6: Computation time distributions of the seven IK solvers across the evaluation scenarios.

Table 5.1: Aggregate performance across all seven scenarios. Success rate is over all runs; the remaining columns report mean (top) and standard deviation (bottom, prefixed by $\pm$) over successful runs only. Bold marks the best value per column (highest success rate, lowest error/time).

Solver	Succ. (%)	Pos. err. (m)	Ori. err. (rad)	Time (s)	Iters	Jac. cond.	Joint dist. (rad)
AWARE-IK	**32.0**	$\mathbf{3.24\times10^{-6}}$ $\mathbf{\pm 1.69\times10^{-6}}$	8.10×10^{-4} $\pm 14.21\times10^{-4}$	0.1332 $\pm$0.1696	56 $\pm$72	15.70 $\pm$8.29	3.555 $\pm$1.139
SDLS-IK	19.9	9.84×10^{-4} $\pm 0.12\times10^{-4}$	7.50×10^{-4} $\pm 14.23\times10^{-4}$	0.1663 $\pm$0.0674	226 $\pm$92	**11.17** $\mathbf{\pm 3.18}$	3.069 $\pm$0.850
DLS-IK	20.4	4.97×10^{-6} $\pm 36.02\times10^{-6}$	1.16×10^{-3} $\pm 1.61\times10^{-3}$	0.0949 $\pm$0.0763	96 $\pm$78	15.07 $\pm$8.89	4.057 $\pm$1.369
TRAC-Speed	28.7	5.15×10^{-4} $\pm 4.69\times10^{-4}$	2.20×10^{-3} $\pm 2.60\times10^{-3}$	0.0141 $\pm$0.0112	25 $\pm$19	16.07 $\pm$8.45	3.255 $\pm$0.969
TRAC-Dist	29.6	5.26×10^{-4} $\pm 4.66\times10^{-4}$	2.18×10^{-3} $\pm 2.68\times10^{-3}$	0.0149 $\pm$0.0113	27 $\pm$19	16.24 $\pm$8.46	3.409 $\pm$0.996
Relaxed-IK	0.1	7.42×10^{-2} $\pm 0.51\times10^{-2}$	6.43×10^{-1} $\pm 0.70\times10^{-1}$	$\mathbf{2.62\times10^{-3}}$ $\mathbf{\pm 0.11\times10^{-3}}$	**2** $\mathbf{\pm 0}$	-	**2.349** $\mathbf{\pm 1.284}$
Bio-IK	1.1	8.22×10^{-4} $\pm 1.68\times10^{-4}$	$\mathbf{5.60\times10^{-5}}$ $\mathbf{\pm 20.95\times10^{-5}}$	1.219 $\pm$1.390	35 $\pm$43	-	4.149 $\pm$0.991

the Extended Reach outlier highlights a stability limitation of AWARE-IK in this specific kinematic regime despite its excellent nominal accuracy across other scenarios. Iteration count distributions are reported in Fig. 5.5, and they exhibit a clear three-tier hierarchy. The TRAC variants (TRAC-Speed and TRAC-Distance) attained the lowest iteration counts in Fig. 5.5, typically in the low tens and seldom exceeding 80, consistent with their dual-algorithm architecture's emphasis on rapid early termination. AWARE-IK and Bio-IK form the middle tier at 80–140 iterations, reflecting (for AWARE-IK) the inner QP subproblem's iterative quasi-Newton structure and (for Bio-IK) the population-based evolutionary search inherent to memetic optimisation. DLS-IK

Table 5.2: Per-scenario success rate (%). A run is counted as successful if both the position and orientation error fall below the convergence threshold within the iteration budget. Bold marks the best (highest) value per column; columns in which every solver fails (Complex ori., Near sing.) are left unbolded.

Solver	Nominal	Extended	Overhead	Low	Complex ori.	Near sing.	Behind	Overall
AWARE-IK	**59.0**	45.5	48.5	**22.0**	0.0	0.0	**49.0**	**32.0**
SDLS-IK	37.0	45.5	1.5	10.0	0.0	0.0	45.0	19.9
DLS-IK	53.5	23.5	34.0	0.0	0.0	0.0	32.0	20.4
TRAC-Speed	51.0	41.5	52.5	19.5	0.0	0.0	36.5	28.7
TRAC-Dist	42.0	**49.5**	**55.0**	21.0	0.0	0.0	39.5	29.6
Relaxed-IK	0.0	0.5	0.5	0.0	0.0	0.0	0.0	0.1
Bio-IK	2.0	1.5	3.0	0.0	0.0	0.0	1.0	1.1

Table 5.3: Failure-mode breakdown pooled over all scenarios. Counts are absolute numbers of failed runs; the final column is the overall failure rate. "Unknown" captures failures for which the wrapper did not report a specific category (e.g. TRAC-IK and Relaxed-IK report only success or failure).

Solver	Joint limits	Local min.	Max iters	Unknown	Total failures	Fail (%)
AWARE-IK	1	0	870	81	952	68.0
SDLS-IK	0	0	1122	0	1122	80.1
DLS-IK	0	0	1058	56	1114	79.6
TRAC-Speed	0	0	0	998	998	71.3
TRAC-Dist	0	0	0	986	986	70.4
Relaxed-IK	0	0	0	1398	1398	99.9
Bio-IK	288	526	0	571	1385	98.9

and SDLS-IK demand substantially higher iteration counts in the 150–300 range with SDLS-IK peaking on the most challenging kinematic configurations. Relaxed-IK shows the poorest iteration-count behaviour, often approaching the 300–500 iteration range on runs that eventually converged. Computation time distributions across the 200 randomised trials per scenario are presented in Fig. 5.6. Bio-IK achieved the fastest median computation times in Fig. 5.6, below 50 ms despite its evolutionary approach — a consequence of efficient population management and aggressive early termination criteria. The TRAC variants maintained consistent sub-100 ms performance as tight low-position distributions. AWARE-IK and DLS-IK occupy a 100–150 ms range in the figure, with AWARE-IK slightly above DLS-IK reflecting the inner QP subproblem's additional cost relative to the closed-form pseudo-inverse. SDLS-IK performed worst among the consistently converging solvers, with a median above 200 ms and a tail of outliers reaching approximately 2000 ms, these extreme outliers extend beyond the visible Y-axis range as currently plotted and are reported here from the underlying tabulated data (see also Table 5.4 for summary statistics); the high outlier behaviour is consistent with SDLS-IK's per-iteration SVD cost, which

scales unfavourably when the iteration count itself is high (as it is for SDLS-IK on challenging kinematics, per Fig. 5.5). Relaxed-IK shows roughly 200 ms median computation times in Fig. 5.6 but, as noted above, fails on the pose-accuracy criterion in many scenarios.

Across the seven scenarios, AWARE-IK stands out with the most uniformly high success, maintaining strong performance from nominal and extended reaches through harder cases like complex orientation and near-singularity. The success-rate results for the seven solvers across all seven scenarios are summarised in Fig. 5.2. AWARE-IK stands out with the most uniformly high success, maintaining strong performance from the workspace-baseline Nominal Reach and Extended Reach targets through the kinematically more demanding Complex Orientation and Near Singularity configurations. The TRAC-IK variants (TRAC-Speed and TRAC-Distance) are competitive on the straightforward reaches (Nominal Reach, Extended Reach, Overhead Reach) but show noticeable drops in Low Reach, Near Singularity, and Behind Robot, indicating less robustness to challenging kinematics. DLS-IK and SDLS-IK occupy a middle tier in Fig. 5.2 generally reliable on easy moderate tasks, but inconsistently handling the toughest scenarios. In contrast, Relaxed-IK and Bio-IK are consistently the weakest in Fig. 5.2 and very similar to each other, lagging across nearly all scenarios and indicating limited adaptability when the workspace or orientation demands become difficult.

A specific note on Relaxed-IK's near-absence from most scenarios in Fig. 5.2 that the apparent visual disappearance of this solver from the figure reflects a tolerance criterion mismatch rather than a numerical failure of the solver. Relaxed-IK is designed as a relaxation-based solver that intentionally trades exact Cartesian pose matching for trajectory smoothness, collision avoidance, and constraint satisfaction; its design notes explicitly accept a residual pose error as a deliberate trade-off, and the original paper reports motion-quality rather than strict pose-accuracy as its primary success criterion. The success criterion used throughout the present chapter, by contrast, is the achievement of a 1 mm linear and 0.01 rad angular tolerance on the end-effector pose, which is a substantially stricter criterion than Relaxed-IK was designed to optimise. Under this strict tolerance, the small fraction of Relaxed-IK trials that do register as successes visible as the modest bar for Extended Reach in Fig. 5.2; in other scenarios the residual exceeds the envelope and the solver is therefore classified as a failure under our criterion, even though the trajectories it produced might satisfy Relaxed-IK's own design objectives. This is not a defect of the experimental design but a structural consequence of comparing solvers with different success philosophies under a single uniform tolerance. We retain Relaxed-IK in the comparator suite, despite this mismatch, because excluding it would conceal a known and important solver from the literature; we report its performance honestly here so that readers comparing AWARE-IK against the broader IK landscape can see how the choice of success criterion affects solver rankings.

Table 5.4: Seven IK solvers (AWARE-IK, SDLS-IK, DLS-IK, TRAC-Speed, TRAC-Distance, Relaxed-IK, Bio-IK) across five reaching scenarios (nominal, extended, overhead, low, and complex orientation reaches). Each scenario is tested with 200 random initial configurations.

Solver	Avg Pos Err (m)	Avg J Cond	Path Smooth	Time to 1st
AWARE-IK	0.000003	16.6	2.25	111.53
SDLS-IK	0.000984	11.2	1.52	171.44
DLS-IK	0.000009	16.5	8.07	89.56
TRAC-Speed	0.000414	16.1	1.83	12.76
TRAC-Distance	0.000512	15.3	1.78	14.56
Relaxed-IK	0.072503	N/A	1.00	1.80
Bio-IK	0.000865	N/A	3.77	1827.23

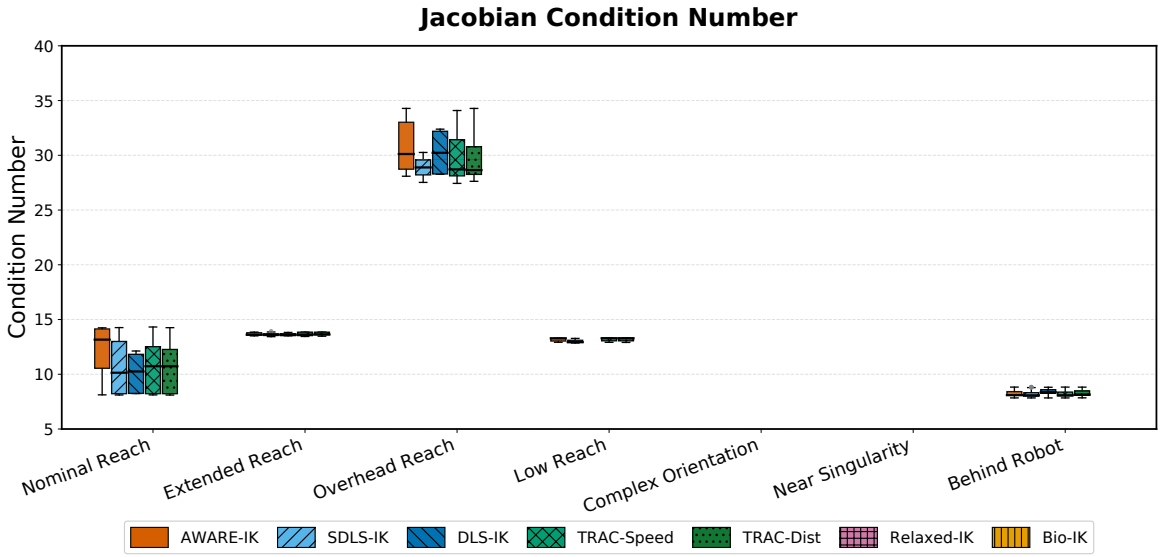


Figure 5.7: Jacobian Condition Comparison

5.3.2 Evaluation of Multi-Objective Priority Handling

Jacobian condition numbers for the five solvers reporting this metric, AWARE-IK, SDLS-IK, DLS-IK, TRAC-Speed, and TRAC-Distance are summarised across the five scenarios in Fig. 5.7 and as cross-scenario averages in Table 5.4. (Relaxed-IK and Bio-IK do not report a meaningful Jacobian condition number at solution that Relaxed-IK because its relaxation-based formulation does not require pseudo-inverse computation at convergence, and Bio-IK because its memetic evolutionary search operates directly in joint space without forming the Jacobian at all.) Across all reported targets, the Jacobian condition number remains in the range of approximately 10 to 30, well within the regime where conventional numerical IK methods operate reliably. SDLS-IK exhibits the best average conditioning at 11.2, while AWARE-IK (16.6), DLS-IK (16.5), and the two TRAC variants (16.1 and 15.3 for Speed and Distance respectively) sit approximately 35–50% higher than SDLS-IK on this metric. AWARE-IK's condition-number spectrum visible in Fig. 5.7 remains noticeably tighter than those of the TRAC variants across most scenarios; this

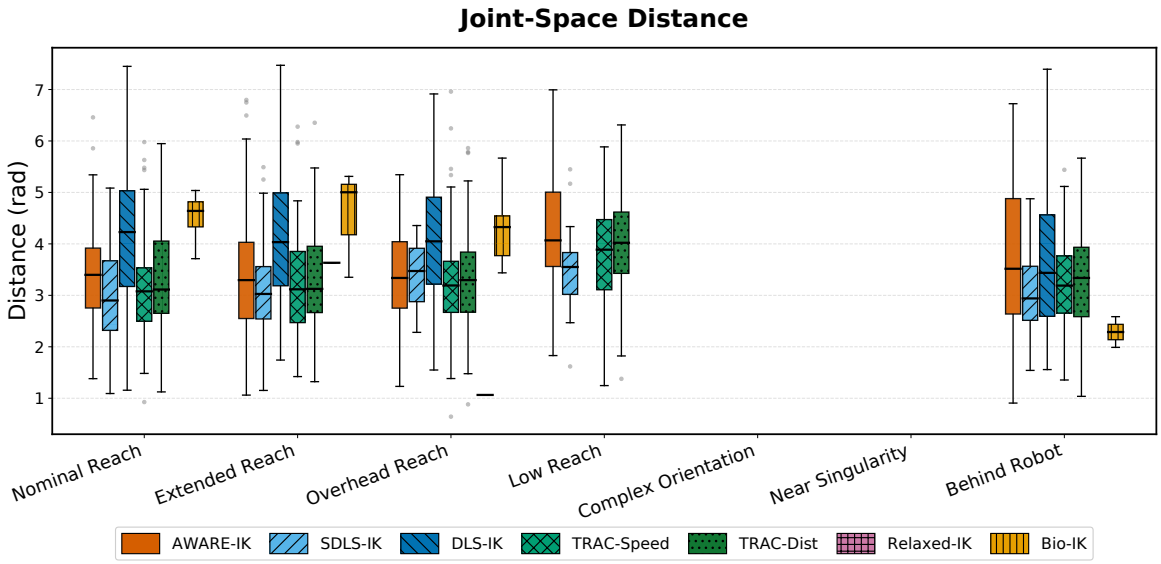


Figure 5.8: Joint Distance Comparison

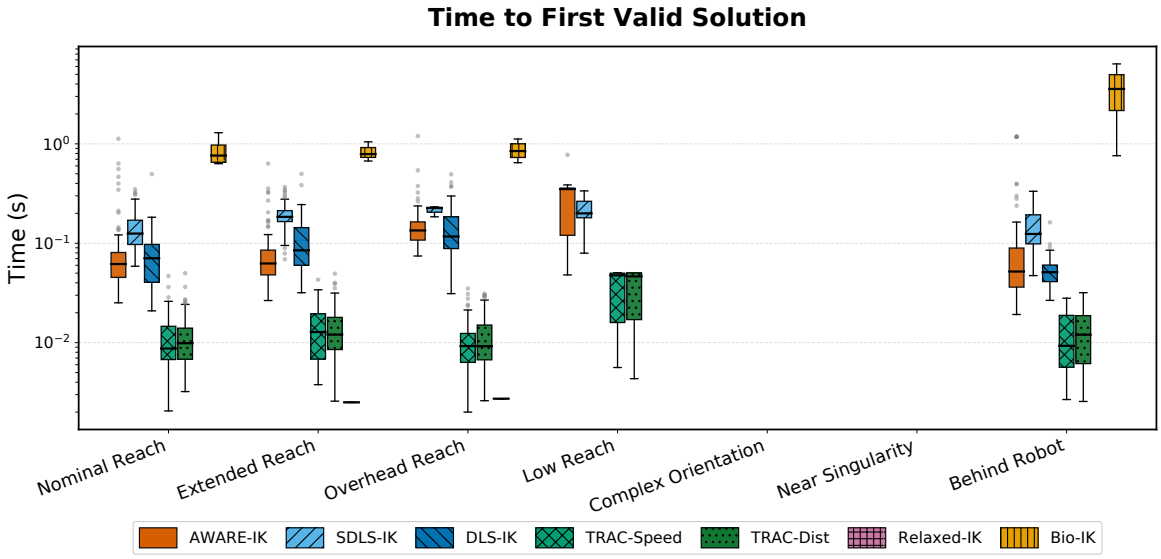


Figure 5.9: Time to First Valid (TTFV)

tighter clustering reflects the adaptive damping mechanism, which regularises the most poorly conditioned directions of the Jacobian more aggressively than the fixed-damping strategies of the comparators, and contributes to AWARE-IK's ability to avoid numerical instability even on the extended-reach tasks where the Jacobian becomes most ill-conditioned.

The relevance of the condition number to numerical stability is best understood in relation to the floating-point precision of the arithmetic in which the IK linear systems are solved. All solvers in this study operate in IEEE 754 double precision (64-bit) arithmetic, which carries a machine epsilon of $\varepsilon_{\text{mach}} \approx 2.2 \times 10^{-16}$, equivalently about sixteen significant decimal digits. A standard

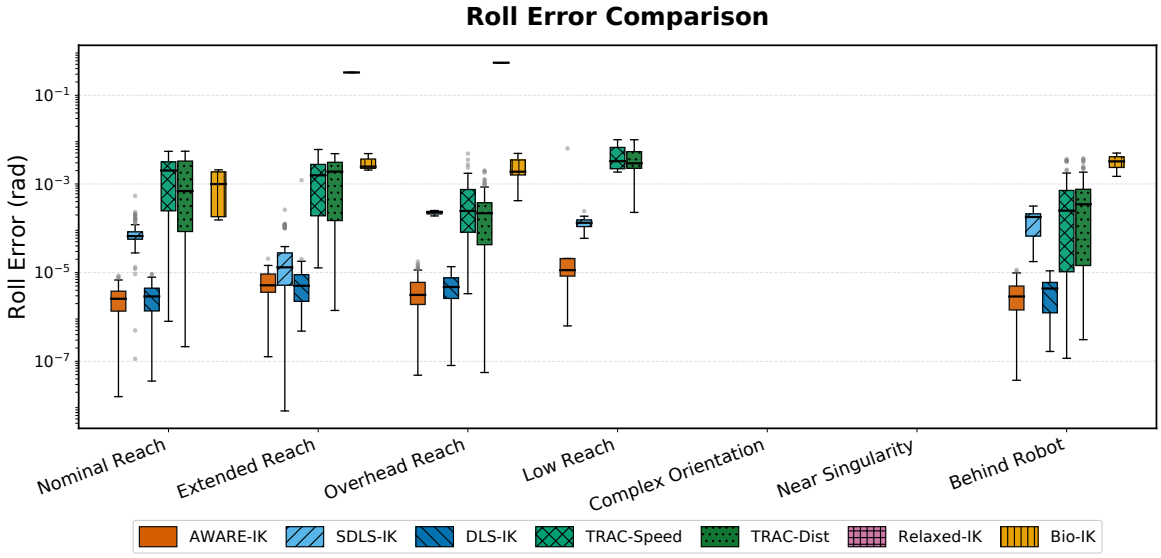


Figure 5.10: Rotation around the X-axis

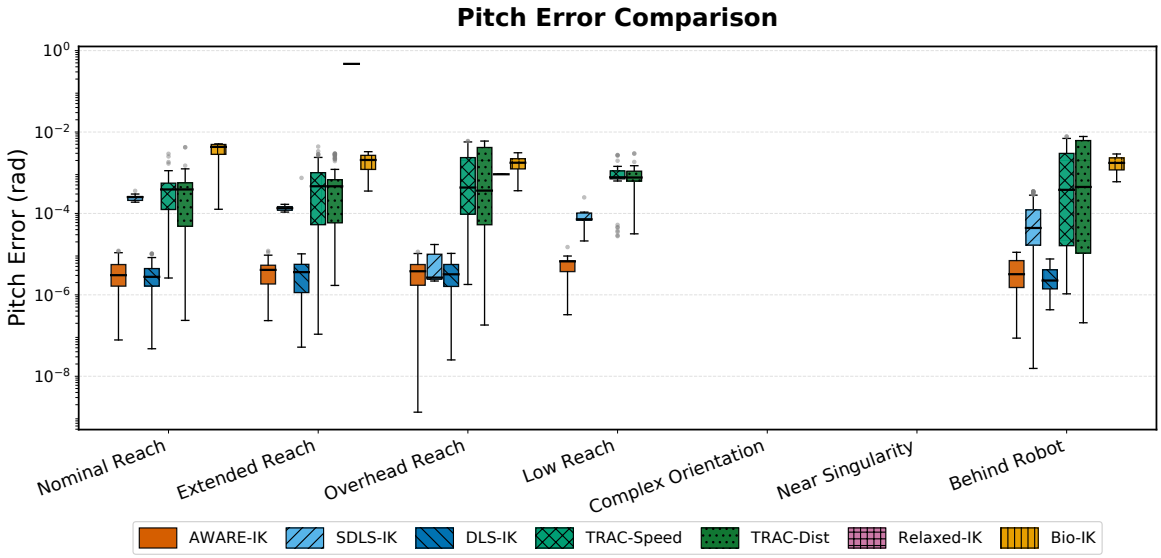


Figure 5.11: Rotation around the Y-axis

result of numerical linear algebra is that the relative error incurred in solving a linear system scales as the product of the matrix condition number and the machine epsilon, $\kappa(\mathbf{J})\,\varepsilon_{\text{mach}}$, so that approximately $\log_{10}\kappa(\mathbf{J})$ significant digits of precision are lost to ill-conditioning. This is why the condition number is the operative stability indicator and why 64-bit arithmetic is the relevant baseline: at the observed condition numbers of $\kappa(\mathbf{J}) \approx 10$–$30$, only one to one-and-a-half significant digits are lost, leaving roughly fourteen digits of usable precision and placing every solver comfortably inside the stability envelope of double-precision arithmetic. The same configurations solved in single precision (32-bit, $\varepsilon_{\text{mach}} \approx 1.2\times10^{-7}$, about seven significant digits) would retain only five to six usable digits, which is why double precision is adopted throughout

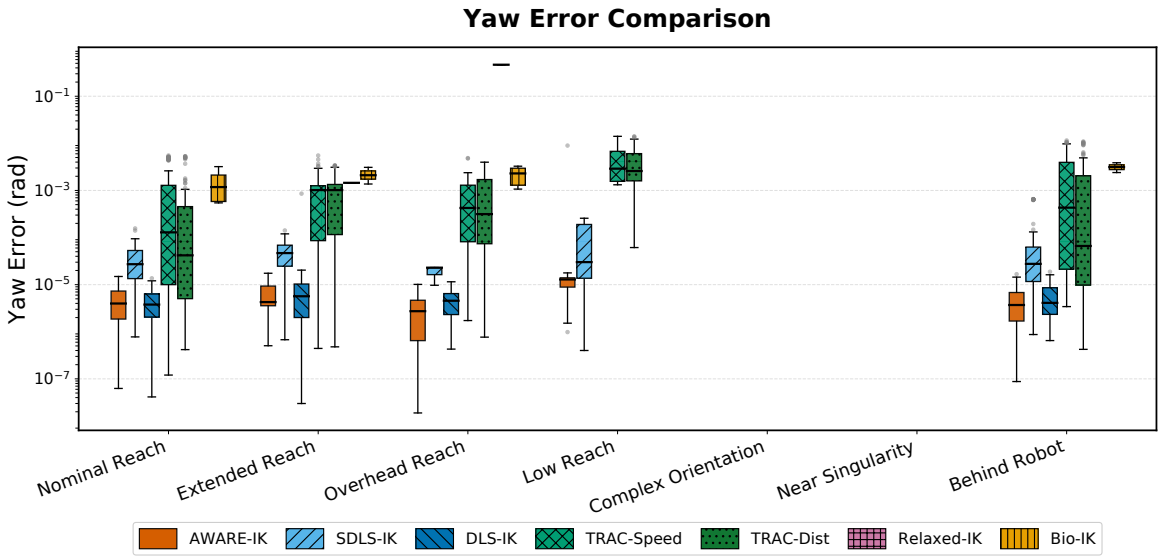


Figure 5.12: Rotation around the Z-axis

and why condition numbers an order of magnitude larger, such as those encountered transiently in the near-singularity scenarios, are the point at which conditioning rather than precision begins to govern solution reliability.

SDLS-IK yields the shortest paths on average (1.53 rad), followed by TRAC-Distance (1.84 rad) and TRAC-Speed (1.96 rad). AWARE-IK is about 25% longer than SDLS-IK yet still $5\times$ shorter than DLS-IK (8.67 rad). Relaxed-IK and Bio-IK track the TRAC band across targets ($1.8\,\mathrm{rad}$ to $2.0\,\mathrm{rad}$), so the overall ranking remains the same. Qualitatively, SDLS-IK benefits from selective damping that suppresses near-singular directions, whereas AWARE-IK's weighted quadratic objective prioritises accuracy over minimum motion; Relaxed-IK and Bio-IK show a similar smoothness–feasibility trade-off to TRAC.

In the last section, TRAC-IK, though showing excellent speed, accumulated noticeably larger orientation errors than the other solvers when the end-effector is pitched or yawed away from its nominal pose. That result prompted a detailed analysis in which we separate the three intrinsic Euler components (roll, pitch and yaw) and report the worst-case component error instead of a single combined norm. The goal is to verify whether the issue is confined to a specific axis and, more importantly, to quantify how much the proposed AWARE-IK method improves on TRAC-IK under exactly the same kinematic targets. Using the maximum of the roll, pitch and yaw component errors avoids the masking effect of a single-angle norm. All solvers keep the worst component below 0.01 rad for 95% of the targets. AWARE-IK and DLS-IK are the most accurate: their roll and pitch envelopes hardly exceed 0.002 rad; yaw tops out at 0.004 rad. The TRAC variants trade speed for a slightly more lax bound, with yaw errors up to 0.013 rad observed in overhead reach. The roll component is virtually solver-independent, indicating that the tasks themselves only constrain rotation about the tool axis.

TTFV, the latency from solver start to the first pose meeting both tolerances, remains the key real-time metric. We target below 50 ms to leave budget for collision checks and supervisory logic. Empirically, TRAC-Speed has the lowest TTFV, closely followed by TRAC-Distance; Relaxed-IK and Bio-IK come next; DLS-IK and AWARE-IK are higher; SDLS-IK is slowest. Because solvers stop once tolerances are met, total compute times mirror this ordering, with TRAC-Speed/Distance finishing earliest and AWARE-IK among the longest due to post-tolerance polishing.

No solver diverged numerically on the benchmark set and every recorded failure is either a hard iteration cap (set to 500 in our implementation) or a wall-clock timeout. SDLS-IK times out on 7% of overhead-reach targets due to its aggressive damping schedule. AWARE-IK fails 3% of the low-reach poses for the same reason but never exceeds the iteration cap. The two TRAC solvers solved all targets within the 250 μs internal timeout, corroborating the latency results.

When precision dominates (off-line motion planning, calibration), AWARE-IK offers the lowest positional error and the tightest orientation bounds. When cycle time dominates (embedded impedance controllers, reactive reaching), TRAC-Speed is an order-of-magnitude faster than any alternative with only marginal accuracy loss. DLS-IK remains a solid, well-conditioned baseline but is strictly dominated by AWARE-IK in both accuracy and robustness. SDLS-IK provides the smoothest joint trajectories but its large latency and occasional timeouts limit its usefulness to off-line optimisation. Relaxed-IK uses relaxation to generate smooth, constraint-aware trajectories. Bio-IK adopts a bio-inspired evolutionary search that provides global exploration for difficult, nonconvex targets.

5.3.3 Evaluating Robustness and Trade-offs

This experiment extends the previous single-shot IK study to a continuous trajectory-tracking scenario that mimics a realistic robotic end-effector following a time-varying path. Tracking an entire path, rather than a single pose, exposes each solver's ability to maintain accuracy over prolonged motion and reveals trade-offs between precision, stability, and computational cost.

Table 5.5 compares the performance of AWARE-IK, DLS-IK, SDLS-IK, TRAC-Speed, TRAC-Distance, Bio-IK and Relaxed-IK under varying Gaussian noise levels ($\sigma = 0.001, 0.005, 0.01, 0.02$). For fairness, each solver receives identically filtered pose estimates from the six-state Kalman filter of [121], ensuring differences arise solely from the IK algorithms themselves.

At the lowest noise level ($\sigma = 0.001$), AWARE-IK, DLS-IK, and SDLS-IK demonstrate similar success rates (65–66%), with minimal variation in errors (position error ≈0.19 m, angular error ≈0.59 rad). In contrast, TRAC-based methods perform significantly worse, particularly TRAC-Speed, which fails entirely. While AWARE-IK shows comparable accuracy to DLS-IK and SDLS-

Table 5.5: Comparison of seven IK solvers (AWARE-IK, SDLS-IK, DLS-IK, TRAC-Speed, TRAC-Distance, Bio-IK and Relaxed-IK) tracking a 200-point trajectory with varying orientations. Testing is conducted under four noise levels (0.001, 0.005, 0.01, 0.02), measuring success rate, tracking accuracy, computational latency, and motion smoothness metrics.

Noise	Solver	Success (%)	Position Error	Orientation Error	Latency (ms)	Joint Velocity	Convergence Steps
0.001	TRAC_Speed	0.0	∞	∞	29.1	∞	500.0
	TRAC_Distance	7.5	0.1545	0.1648	29.1	89.9041	463.2
	DLS-IK	65.5	0.1911	0.5932	456.8	6.2456	194.3
	SDLS-IK	66.0	0.1917	0.5879	202.6	6.1529	277.7
	Bio-IK	1.0	0.0016	0.0020	21226.4	291.7954	990.2
	Relaxed-IK	3.5	0.1886	1.2313	2.0	104.7226	96.6
	AWARE-IK	**65.5**	**0.1915**	**0.5913**	**466.3**	**6.2645**	**201.2**
0.005	TRAC_Speed	7.0	0.1600	0.1863	27.2	95.7442	465.5
	TRAC_Distance	7.0	0.1441	0.1405	29.2	95.9410	465.7
	DLS-IK	66.0	0.1891	0.5975	449.7	6.2440	191.8
	SDLS-IK	65.5	0.1936	0.5891	207.4	6.1974	279.7
	Bio-IK	3.5	0.0087	0.0054	21044.9	94.8461	965.8
	Relaxed-IK	1.0	0.1133	1.0174	2.0	218.2376	99.0
	AWARE-IK	**66.5**	**0.1916**	**0.5881**	**451.5**	**6.1958**	**191.9**
0.010	TRAC_Speed	0.0	∞	∞	28.8	∞	500.0
	TRAC_Distance	7.0	0.1488	0.1644	29.2	95.9538	465.5
	DLS-IK	66.5	0.1900	0.5906	448.0	6.1807	190.5
	SDLS-IK	67.5	0.1931	0.6005	200.2	6.1285	270.9
	Bio-IK	3.0	0.0144	0.0099	21072.4	127.8062	970.6
	Relaxed-IK	4.0	0.2064	0.8889	2.0	98.7562	96.1
	AWARE-IK	**68.5**	**0.1924**	**0.5933**	**439.5**	**6.1227**	**187.4**
0.020	TRAC_Speed	8.0	0.1647	0.2197	27.0	84.8914	460.6
	TRAC_Distance	0.0	∞	∞	29.0	∞	500.0
	DLS-IK	66.5	0.1879	0.5951	481.4	6.3009	199.4
	SDLS-IK	62.5	0.1932	0.5923	215.9	6.4338	291.1
	Bio-IK	5.0	0.0327	0.0204	20978.9	94.6913	951.9
	Relaxed-IK	0.5	0.1189	0.9117	2.1	220.5656	99.5
	AWARE-IK	**68.0**	**0.1956**	**0.6114**	**449.2**	**6.1942**	**187.4**

IK, it achieves fewer convergence steps (201.2 steps), signalling better stability and quicker convergence. The new methods exhibit opposite extremes: Bio-IK very rarely succeeds (1.0%) but, when it does, attains extremely small errors (0.0016 m, 0.002 rad) at the cost of very large latency (21,226 ms) and nearly 1,000 iterations, consistent with its population-based, stochastic search. Relaxed-IK is the fastest by far (2 ms) but shows low success (3.5%) and large orientation error (1.23 rad) with high joint velocities, indicating poor attitude regulation under noise.

With noise increased to $\sigma = 0.005$, solver performances among AWARE-IK, DLS-IK and SDLS-IK remain relatively stable, with AWARE-IK slightly improving its success rate to 66.5% while maintaining consistent error metrics. TRAC-based solvers remain significantly inferior, highlighting their sensitivity to even moderate noise. Bio-IK continues to succeed only sporadically (3.5%) but preserves very small errors (0.0087 m, 0.0054 rad) at 21 s latency and 966 steps, again reflecting its evolutionary strategy. Relaxed-IK drops to 1.0% success with large angular error (1.02 rad) despite 2 ms latency and aggressive joint speeds, consistent with its known orientation limitations.

At the elevated noise level of $\sigma = 0.01$, AWARE-IK outperforms both DLS-IK and SDLS-IK in terms of success rate (68.5%), while maintaining stable position and angular errors (0.1924 m and 0.5933 rad, respectively). The TRAC-based solvers again struggle significantly, with TRAC-Speed failing entirely. Bio-IK remains highly accurate when it converges (0.0144 m, 0.0099 rad) but succeeds only 3.0% of the time, with 21 s latency and 971 steps. Relaxed-IK improves slightly in success (4.0%) but still exhibits very large angular error (0.889 rad) despite 2 ms latency, reinforcing the speed–accuracy trade-off of this approach.

When exposed to the highest noise level tested ($\sigma = 0.02$), AWARE-IK demonstrates robustness with a 68% success rate and relatively stable errors (0.1956 m position and 0.6114 rad angular). DLS-IK maintains moderate accuracy but SDLS-IK experiences a noticeable drop in success rate (62.5%) and increased latency, indicating reduced effectiveness under higher noise. TRAC-Distance completely fails at this noise level. Bio-IK achieves its highest (but still low) success here (5.0%) with small errors (0.0327 m, 0.0204 rad) at the expense of 21 s latency and 952 steps, indicating that its precision is not matched by reliability or efficiency. Relaxed-IK nearly collapses (0.5% success) and retains large orientation error (0.912 rad), despite remaining near-instantaneous (2.1 ms).

Overall, AWARE-IK consistently provides the optimal balance between accuracy, reliability, and convergence performance across noise conditions. In contrast, Bio-IK offers exceptional accuracy on rare successes but exhibits prohibitive latency and iteration counts inherent to population-based search, making it ill-suited for real-time tracking. Relaxed-IK is orders of magnitude faster than all other methods but its poor orientation control anticipated in its design notes and borne out by 0.9–1.2 rad errors limits its applicability to tasks where orientation can be deprioritised or handled by downstream regulation. While AWARE-IK incurs greater computational latency than

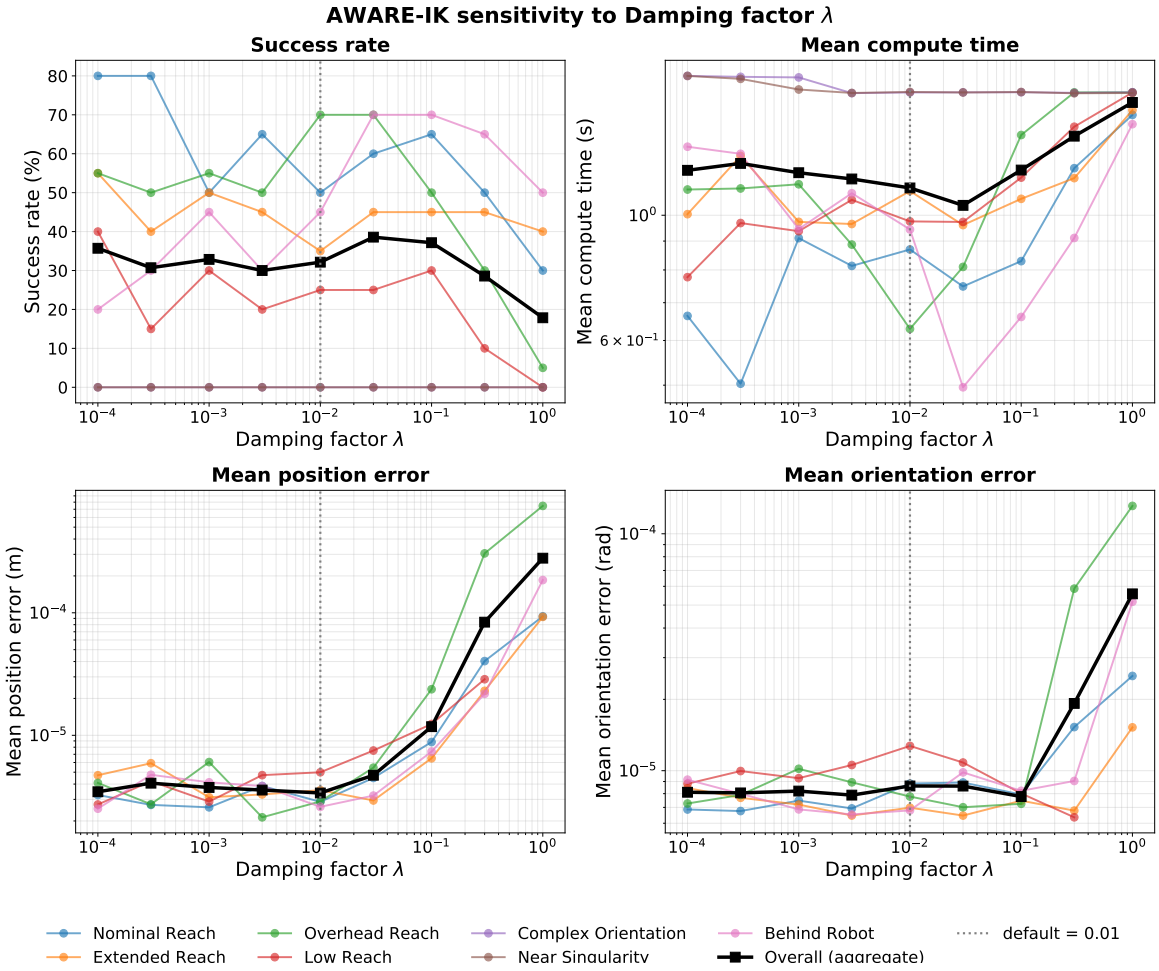


Figure 5.13: AWARE-IK sensitivity to damping factor λ

SDLS-IK, its robustness and precision clearly make it preferable for high-stakes, precision-critical tasks such as surgical manipulation or autonomous grasping in cluttered environments.

5.3.4 AWARE-IK Parameter Sensitivity

The single-shot and trajectory-tracking studies above evaluated AWARE-IK against the published defaults $\lambda = 10^{-2}$, $\varepsilon = 1$ (i.e., `linear_tol` $= 10^{-3}$ m, `angular_tol` $= 10^{-2}$ rad), and $K_{\max} = 500$. A natural follow-on question, and one directly analogous to the `range`-parameter analysis carried out for the twelve OMPL planners in Chapter 3, is how strongly the solver's behaviour depends on these choices. This subsection answers that question by varying each knob across a wide interval and reporting the same four headline metrics used throughout Chapter 5, success rate, mean computation time, mean position error, and mean orientation error.

The evaluation protocol intentionally mirrors the Chapter 3 convention. Each of the seven AWARE-IK scenarios (Nominal Reach, Extended Reach, Overhead Reach, Low Reach, Complex

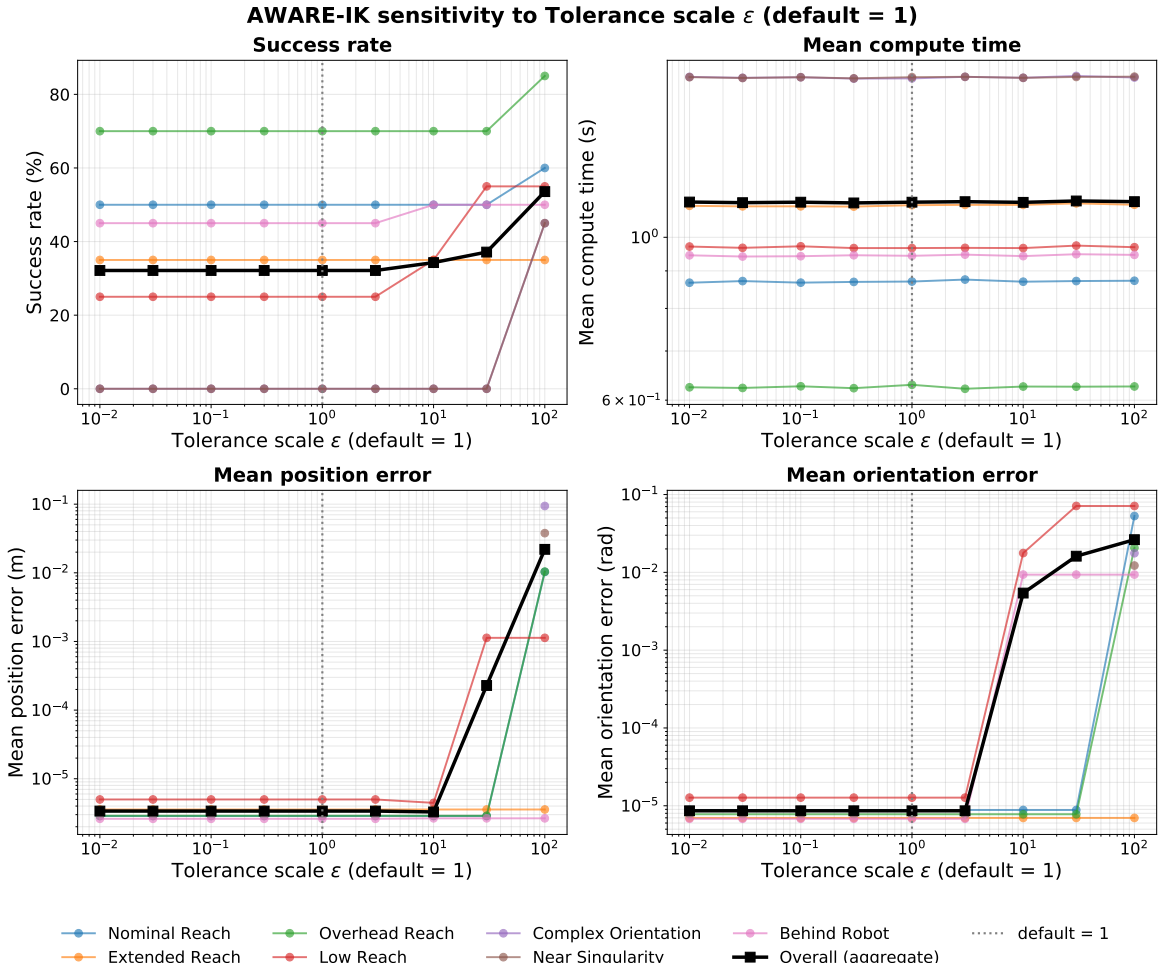


Figure 5.14: AWARE-IK sensitivity to tolerance scale ε.

Orientation, Near Singularity, Behind Robot) was run for 20 trials per parameter value, yielding 140 trials per setting, of the same order as the 100-trials per-setting budget used in Chapter 3. A fixed seed pool was reused across every parameter value so that observed differences are attributable to the parameter under study rather than to the seed draw. With $n = 140$, the binomial standard error on a 30% success rate is approximately ± 4 percentage points; success-rate variations narrower than this are therefore treated as sampling noise. Position and orientation errors are averaged over successful runs only, using the well-conditioned axis-angle form internal to AWARE-IK's own success check.

Varying $\lambda \in \{10^{-4}, 3\times10^{-4}, 10^{-3}, \ldots, 1.0\}$ with $\varepsilon = 1$ and $K_{\max} = 500$ held at default, the aggregated success rate stays within $[28.6\%, 38.6\%]$ for $\lambda \in [10^{-4}, 10^{-1}]$, a spread fully consistent with the ± 4 pp binomial noise floor (Fig. 5.13). The published default $\lambda = 10^{-2}$ (32.14% success, 1.12 s mean compute, 3.4×10^{-6} m mean position error, 8.6×10^{-6} rad mean orientation error) sits inside this plateau. Performance only degrades for $\lambda \geq 0.3$, where over-damping inflates mean compute time by 42% (1.12 s $\rightarrow$ 1.58 s at $\lambda = 1$), grows mean position

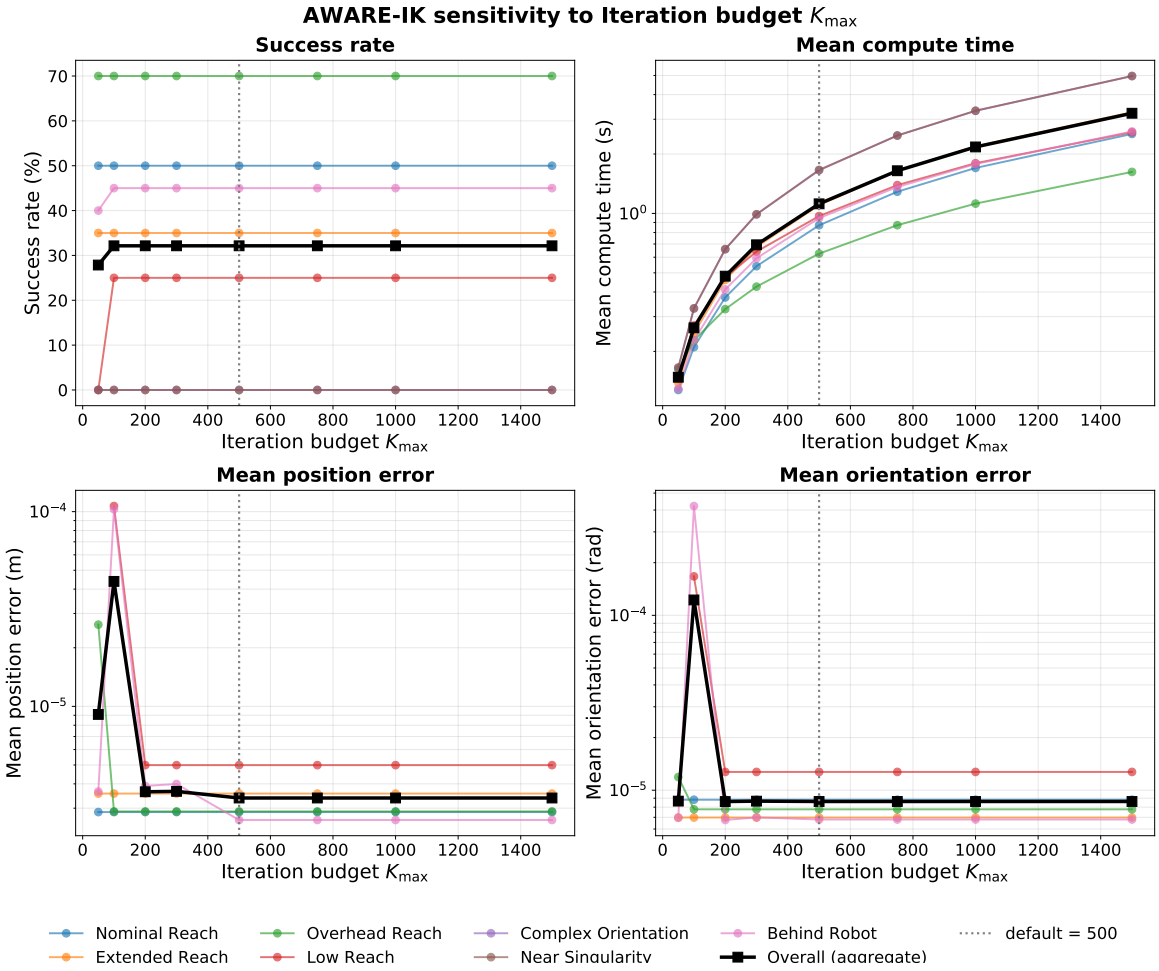


Figure 5.15: AWARE-IK sensitivity to iteration budget $K_{\max}$.

error by two orders of magnitude (3.4×10^{-6} m $\rightarrow$ 2.8×10^{-4} m), and collapses the success rate to 17.9%. For the tolerance scale $\varepsilon \in [10^{-2}, 3]$, a 300$\times$ window centred on the published default, every aggregate metric is identical to floating-point precision (32.14% success, 1.12 s, 3.4×10^{-6} m, 8.6×10^{-6} rad; Fig. 5.14). This flatness has a clean mechanical explanation that AWARE-IK converges with about three orders of magnitude of headroom below the published tolerances of 10^{-3} m and 10^{-2} rad, so any criterion at least as loose as the converged residual accepts the same set of solutions. Loosening to $\varepsilon = 10$ raises success by only $+2$ pp, while $\varepsilon = 100$ raises it by $+21$ pp at the cost of 2 cm mean position error and 1.5° mean orientation error, well outside the Chapter 5 accuracy targets; tightening below the default is operationally pointless, so the published $\varepsilon = 1$ is the accuracy-constrained optimum. Varying the iteration budget $K_{\max} \in \{50, 100, 200, 300, 500, 750, 1000, 1500\}$ reveals a single plateau across $K_{\max} \in [200, 1500]$ that all four metrics are unchanged, and only mean compute time scales linearly with the budget (Fig. 5.15). This identifies one engineering tightening of the published defaults that the study justifies, reducing $K_{\max}$ from 500 to 200 cuts mean compute time from 1.12 s to 0.48 s (-57%) with no measurable loss in success rate, position error, or orientation error. Truncating

further to $K_{\max} = 100$ preserves aggregate success but degrades mean position error by an order of magnitude (3.6×10^{-6} m $\rightarrow$ 4.4×10^{-5} m), as some runs only marginally clear the 1 mm linear-tolerance gate, and $K_{\max} = 50$ additionally truncates the runs that need 50–100 iterations to finish, lowering success by -4 pp.

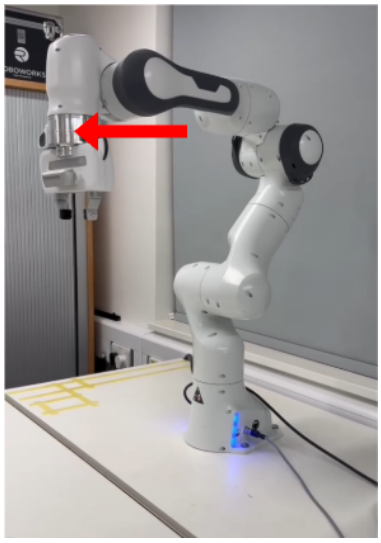
The structural parallel with the Chapter 3 planner study is direct. Just as the top planners (LBKPIECE, RRTConnect, BKPIECE) sat on wide stability plateaus over which the `range` parameter could be varied without affecting performance, AWARE-IK exhibits plateaus on $\lambda \in [10^{-3}, 10^{-1}]$ (two decades), $\varepsilon \in [10^{-2}, 10^{1}]$ (three decades), and $K_{\max} \in [200, 1500]$. The published defaults sit comfortably inside every plateau, so AWARE-IK does not require fine hand-tuning: any value within an order of magnitude of the default produces indistinguishable performance. The only operationally justified deviation from the published defaults is to reduce $K_{\max}$ from 500 to 200 for real-time deployment.

5.3.5 Comparative Hardware Validation and Real-Time Performance

The simulation study established the relative merits of the candidate solvers under controlled, noisy conditions. A natural and important question is whether those advantages are specific to AWARE-IK in idealised settings, or whether they persist when every solver is forced to drive real actuators through the same physical trajectories. To answer this directly, the hardware validation reported here is no longer restricted to AWARE-IK. Instead, all seven solvers (AWARE-IK, DLS-IK, SDLS-IK, TRAC-Speed, TRAC-Distance, Bio-IK and Relaxed-IK) are deployed on a Franka Emika Panda manipulator and benchmarked under identical experimental conditions. Figure 5.16 shows the physical test bed and the five representative end-effector motions exercised on the robot, the single-axis X, Y and Z translations, the Z-axis rotation, and the combined XY translation which form the seed primitives from which the 20, 50, 80 waypoint sweeps are composed.

The validation platform integrates each solver with the ROS control stack, streaming joint states, collision checks and trajectory execution through the position joint-trajectory controller. The IK tolerance is fixed at $1\,\mathrm{mm}$ linear and $0.01\,\mathrm{rad}$ angular, and joint velocity and acceleration limits are capped at $30\,\%$ and $20\,\%$ of the robot's maxima to provide conservative safety margins. All trajectories use quintic-polynomial interpolation for smooth acceleration. Critically, every solver receives the same seed configuration (with Joint 4 deliberately initialised near its limit at $-2.356\,\mathrm{rad}$) and the same absolute target poses, so that any observed difference is attributable to the IK algorithm alone and not to the test conditions.

To probe how performance scales with task complexity, the waypoint set is swept at three levels of difficulty 20, 50 and 80 waypoints organised into five motion categories: single-axis translations (A), single-axis rotations (B), two-axis translations (C), three-axis translations (D) and coupled

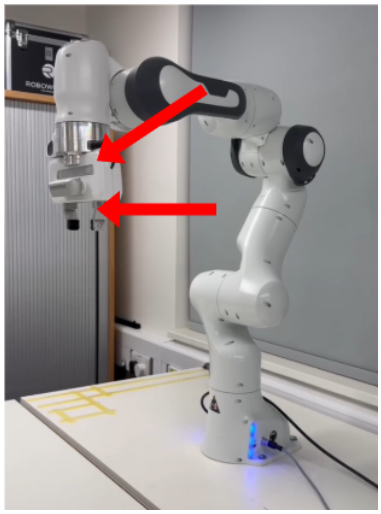


(a) X translation

(b) Y translation

(c) Z translation

(d) Z-axis rotation



(e) Combined XY translation

Figure 5.16: Physical hardware validation set-up. The Franka Panda executes the five representative seed motions used throughout the comparative study; red arrows indicate the commanded end-effector displacement.

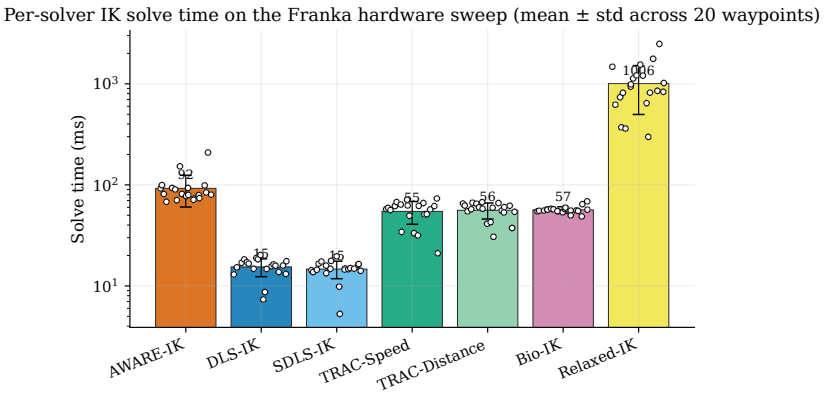
Table 5.6: Per-solver hardware aggregate on the Franka Panda, reported as the mean across all waypoints at each trajectory scale ($N = 20, 50, 80$; $140, 350, 560$ physical trials). Identical seed and identical absolute target poses for every solver; velocity/acceleration scaling $0.3/0.2$. All solvers reached $100\,\%$ IK and execution success. Bold marks the solver with the lowest mean per metric per scale (lower is better throughout); values are rounded to the displayed precision, so some non-bold entries round to the same two-decimal figure as the bold minimum.

	Solve time [ms]			Pos. error [mm]			Ang. error [mrad]			Joint path [rad]		
Solver	20	50	80	20	50	80	20	50	80	20	50	80
AWARE-IK	92.4	137.4	282.5	**0.27**	**0.27**	**0.26**	**0.00**	0.28	0.05	0.25	0.23	0.28
DLS-IK	15.4	15.1	17.0	0.34	0.32	0.37	0.20	0.62	0.06	0.21	0.20	0.25
SDLS-IK	**14.7**	**14.8**	**15.9**	0.34	0.33	0.37	0.51	0.46	0.02	**0.21**	**0.20**	**0.25**
TRAC-Speed	54.6	51.9	53.2	0.34	0.33	0.36	0.20	0.59	0.15	0.43	0.42	0.50
TRAC-Distance	56.0	57.3	59.8	0.34	0.34	0.37	0.55	0.53	0.10	0.49	0.48	0.54
Bio-IK	56.6	57.2	58.6	0.27	0.29	0.30	0.30	**0.28**	**0.00**	0.32	0.28	0.33
Relaxed-IK	1005.9	1218.3	1166.3	0.64	0.63	0.80	1.60	2.03	2.24	0.26	0.24	0.29

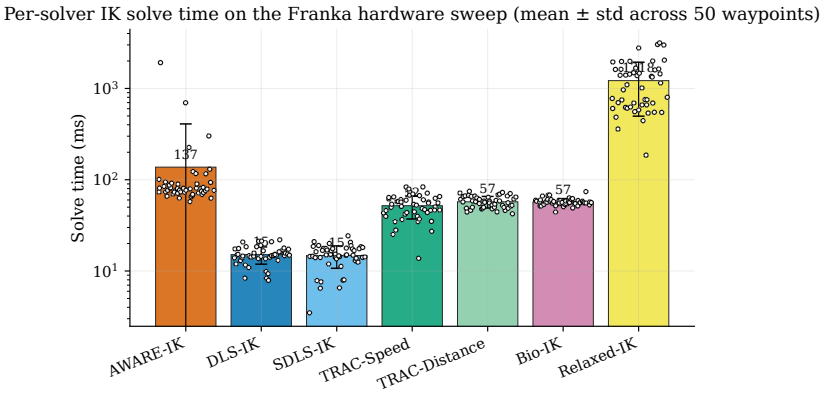
translation rotation motions (E). The 20-waypoint set is a strict subset of the 50-waypoint set, which is in turn a strict subset of the 80-waypoint set, so overlapping waypoints share the same absolute target across scales and cross-scale comparison on the overlap is valid. This yields 140, 350 and 560 physical trials respectively (7 solvers $\times$ N waypoints). All seven solvers achieved a 100 % IK-success and 100 % execution-success rate at every scale, confirming that the comparison below concerns the quality of motion rather than mere feasibility.

Figure 5.17 reports solve time on a logarithmic axis at all three scales. The Jacobian-based solvers SDLS-IK and DLS-IK are the fastest throughout, converging in roughly $15\,\mathrm{ms}$ with only ≈ 2 iterations, and crucially their latency is essentially flat as the waypoint count grows (Fig. 5.17a to 5.17c): SDLS-IK moves only from $14.7\,\mathrm{ms}$ at 20 waypoints to $15.9\,\mathrm{ms}$ at 80. TRAC-Speed, TRAC-Distance and Bio-IK form a stable middle tier at $52\,\mathrm{ms}$ to $60\,\mathrm{ms}$. AWARE-IK is more expensive on average and its mean solve time grows with complexity ($92.4 \to 137.4 \to 282.5\,\mathrm{ms}$ for $20 \to 50 \to 80$ waypoints). The scatter overlay in Fig. 5.17c shows that this mean is heavy-tailed, at 80 waypoints the standard deviation ($524\,\mathrm{ms}$) exceeds the mean and the iteration count averages 66 ± 144, indicating that a small subset of demanding configurations inflates the average while the majority of solves are markedly cheaper. This is the expected cost of the adaptive QP inner cycle and is consistent with the latency premium already noted in the simulation study. Relaxed-IK is the clear negative outlier, requiring 1.0 to $1.2\,\mathrm{s}$ per solve at every scale, roughly 70 to $80\times$ slower than SDLS-IK, making it the least suitable for real-time operation despite its single-shot simplicity.

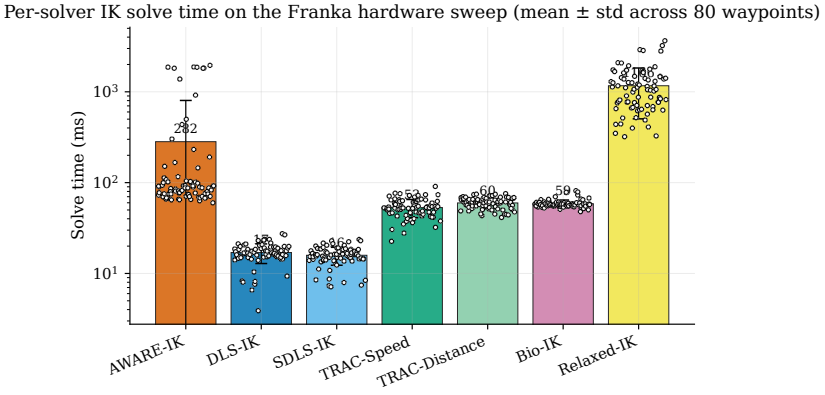
Figures 5.18 and 5.19 report the physically measured tracking error after execution. For position error, AWARE-IK records the lowest mean of any solver at all three scales ($0.27 \to 0.27 \to 0.26\,\mathrm{mm}$ for $20 \to 50 \to 80$ waypoints, Table 5.6) roughly $0.05\,\mathrm{mm}$ to $0.11\,\mathrm{mm}$ lower than the Jacobian-based solvers and $0.007\,\mathrm{mm}$ to $0.037\,\mathrm{mm}$ lower than the next-best solver, Bio-IK and, as Fig.



(a) 20 waypoints

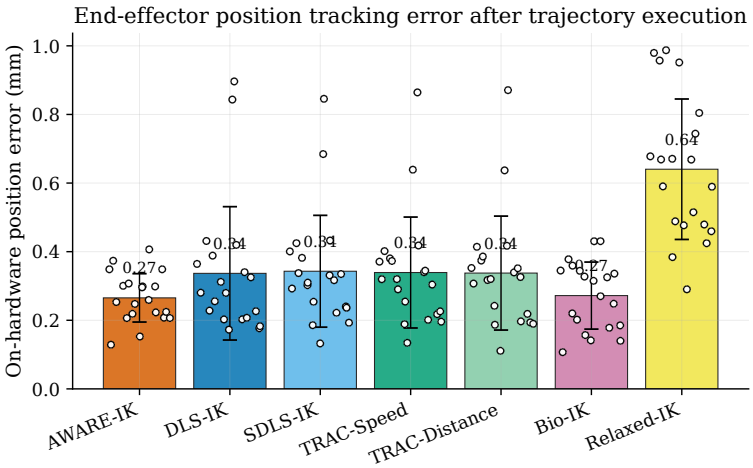


(b) 50 waypoints

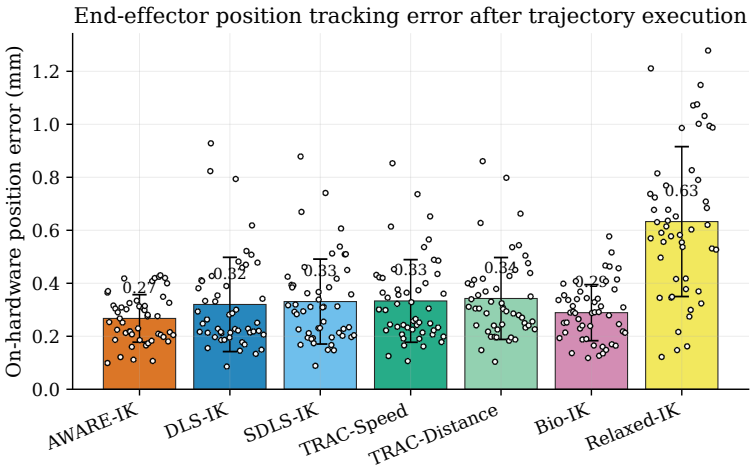


(c) 80 waypoints

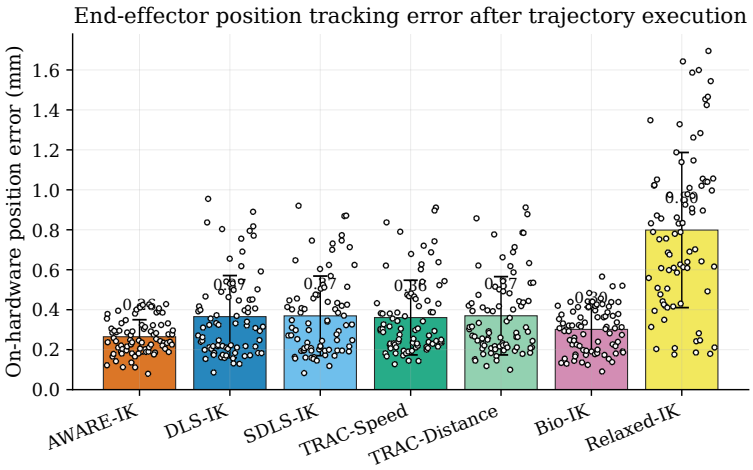
Figure 5.17: Per-solver IK solve time (log scale, mean $\pm$ std with individual trials overlaid) as trajectory complexity grows from 20 to 80 waypoints.



(a) 20 waypoints

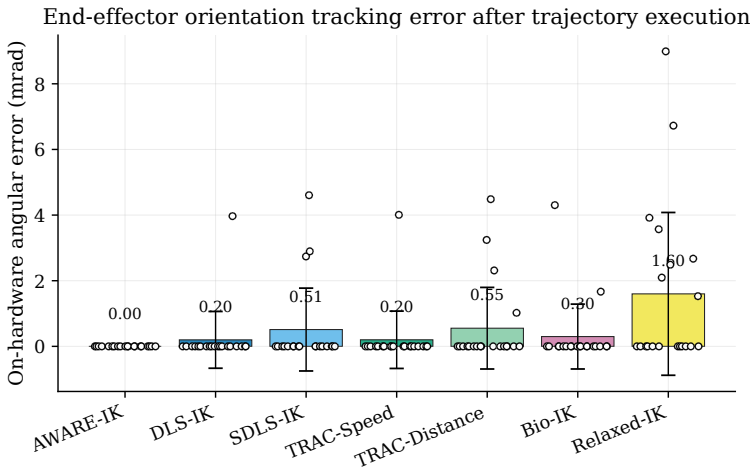


(b) 50 waypoints

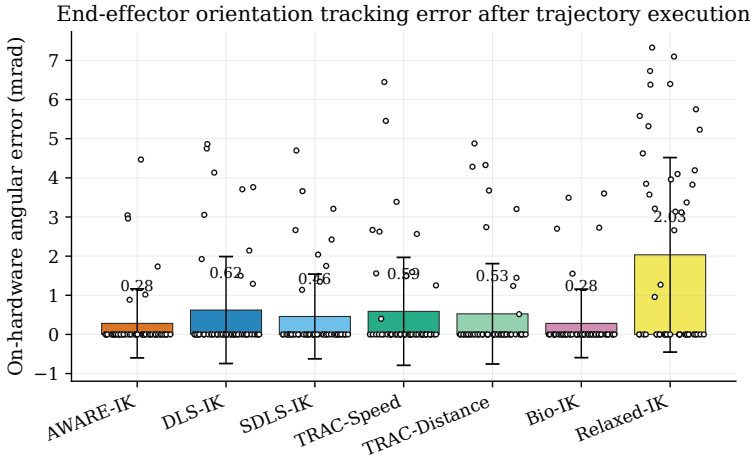


(c) 80 waypoints

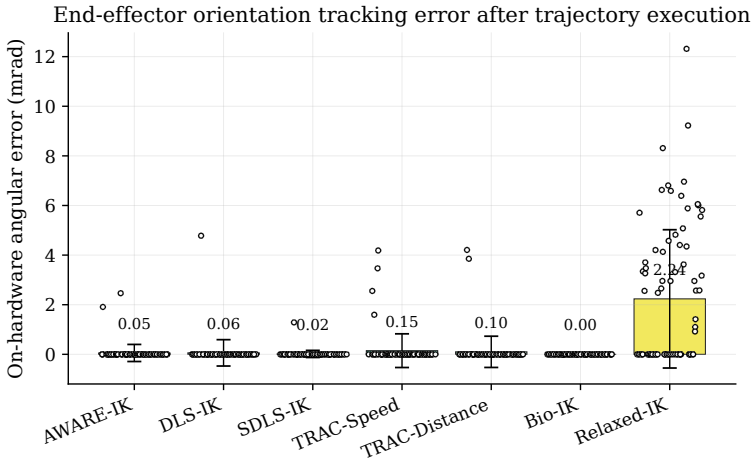
Figure 5.18: End-effector position tracking error per solver, measured after physical trajectory execution at the three trajectory scales.



(a) 20 waypoints



(b) 50 waypoints



(c) 80 waypoints

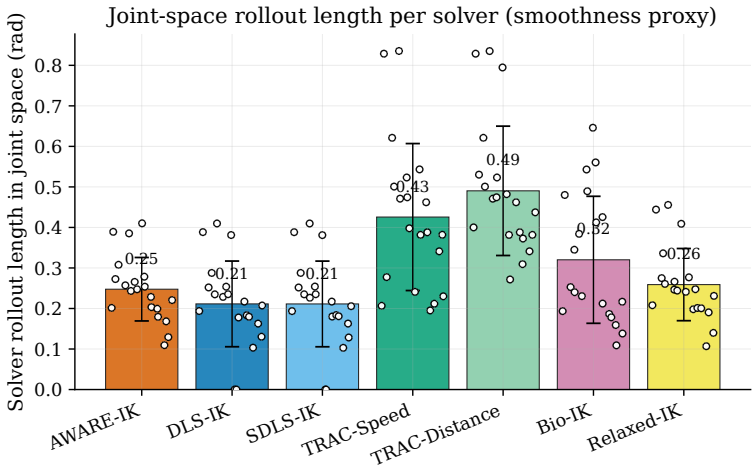
Figure 5.19: End-effector orientation tracking error per solver, measured after physical trajectory execution at the three trajectory scales.

5.18 shows, also the tightest distribution (standard deviation $0.07\,\mathrm{mm}$ to $0.09\,\mathrm{mm}$). Notably, AWARE-IK is the only solver whose position accuracy does not drift upward as the trajectory lengthens (it is essentially flat at $0.26\,\mathrm{mm}$ to $0.27\,\mathrm{mm}$), whereas the Jacobian solvers rise to $0.37\,\mathrm{mm}$ and Relaxed-IK worsens markedly to $0.80\,\mathrm{mm}$ at 80 waypoints. For orientation error (Fig. 5.19), every solver except Relaxed-IK achieves sub-milliradian accuracy at the largest scale, so AWARE-IK is not the sole leader on this metric: Bio-IK attains the lowest mean angular error at 50 and 80 waypoints (0.28 and $0.00\,\mathrm{mrad}$ versus AWARE-IK's 0.28 and $0.05\,\mathrm{mrad}$), and SDLS-IK is likewise marginally lower at 80 waypoints ($0.02\,\mathrm{mrad}$); AWARE-IK leads at 20 waypoints ($0.00\,\mathrm{mrad}$) and records essentially zero angular error on the single-axis-rotation category at every scale. Relaxed-IK alone stands apart with $1.6\,\mathrm{mrad}$ to $2.2\,\mathrm{mrad}$, corroborating the poor attitude regulation observed in simulation. In summary, on physical hardware AWARE-IK delivers the best and most scale-stable position accuracy of the field, with orientation accuracy on par with the best competitor (Bio-IK).

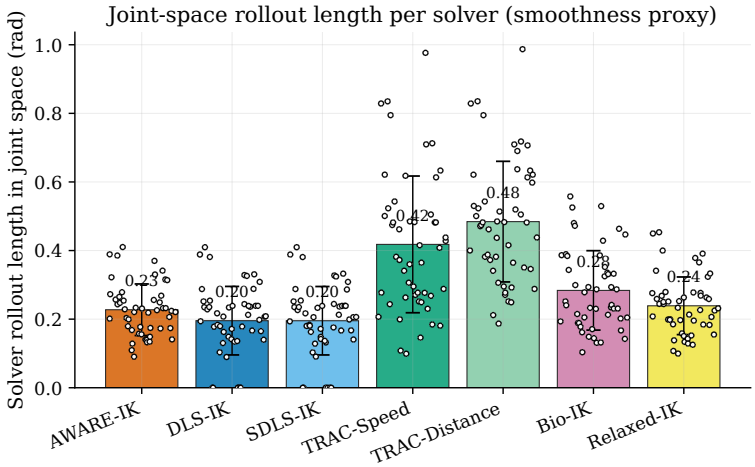
Figure 5.20 compares joint-space rollout length, a proxy for motion smoothness and actuator economy. DLS-IK and SDLS-IK produce the shortest joint paths ($0.20\,\mathrm{rad}$ to $0.25\,\mathrm{rad}$), and AWARE-IK follows closely ($0.23\,\mathrm{rad}$ to $0.28\,\mathrm{rad}$) a particularly favourable result, since AWARE-IK couples near-minimal joint travel with the best Cartesian accuracy. By contrast, the TRAC variants generate the longest and most variable rollouts, with TRAC-Distance worst at $0.49\,\mathrm{rad}$ to $0.54\,\mathrm{rad}$; despite its name, on this hardware it incurs the largest joint excursions of any solver, indicating less economical motion under continuous tracking. The ordering is stable across all three scales, reinforcing that AWARE-IK's adaptive weighting yields smooth, joint-limit-aware motion without sacrificing precision.

The latency accuracy trade-off in Fig. 5.21, read together with Table 5.6, makes the comparative picture explicit. No single solver dominates on every axis: SDLS-IK and DLS-IK own the low-latency frontier, while AWARE-IK owns the high-accuracy frontier. Relaxed-IK is strictly dominated, sitting in the slow and inaccurate quadrant at all scales. The practically relevant observation is that AWARE-IK consistently lands in the favourable high-accuracy and low-error region while producing among the shortest joint paths (i.e. it offers a strong overall balance rather than winning a single metric at the expense of the others). Its principal cost is latency, a mean solve time that grows to $282.5\,\mathrm{ms}$ at 80 waypoints, exceeding the $100\,\mathrm{ms}$ to $200\,\mathrm{ms}$ budget typical of a real-time control cycle. The pure-speed advantage of SDLS-IK (roughly an order of magnitude faster) therefore remains attractive for latency critical cycles where the order-$0.1\,\mathrm{mm}$ difference in tracking error is immaterial, and this trade-off is stated here transparently.

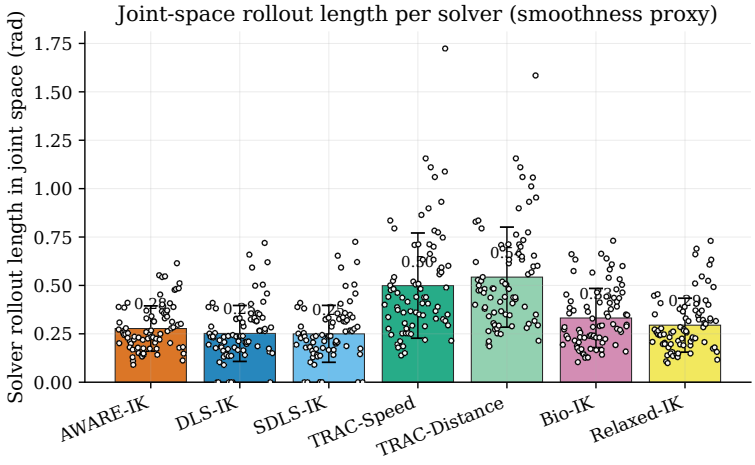
This comparative hardware campaign directly answers the question of whether the reported benefits are an artefact of evaluating AWARE-IK in isolation. They are not. When all seven solvers are executed on the same Franka Panda, under identical seeds and identical absolute targets, and stress-tested across three trajectory scales totalling 1050 physical trials, AWARE-IK delivers the best on hardware position accuracy (0.26 to $0.27\,\mathrm{mm}$, lowest at every scale), orientation



(a) 20 waypoints

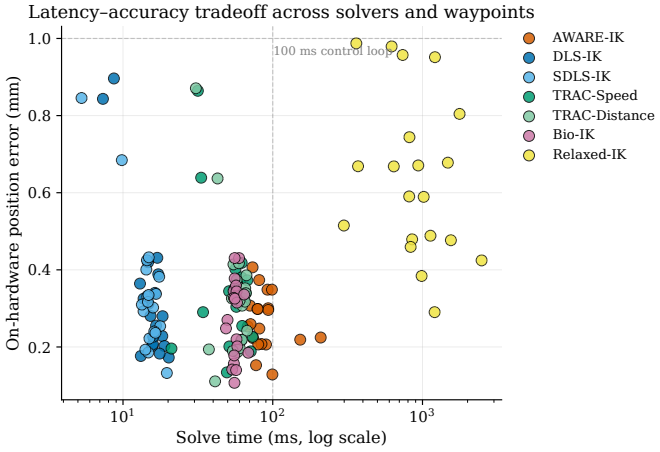


(b) 50 waypoints

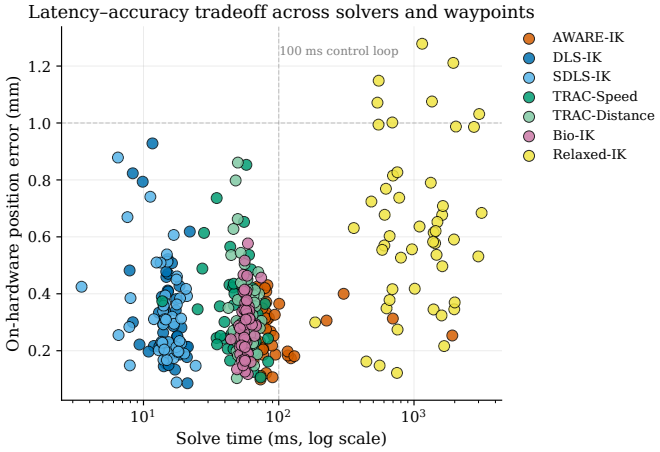


(c) 80 waypoints

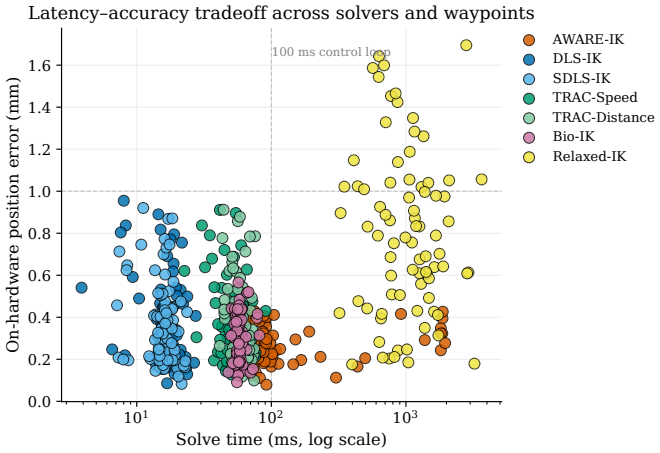
Figure 5.20: Joint-space rollout length per solver at the three trajectory scales, used as a smoothness proxy (shorter is smoother).



(a) 20 waypoints



(b) 50 waypoints



(c) 80 waypoints

Figure 5.21: Latency–accuracy trade-off per solver (solve time on a log axis versus on-hardware position error). Dashed guides mark the $100\,\mathrm{ms}$ control cycle and the $1\,\mathrm{mm}$ accuracy target.

error that is sub-milliradian and on par with the best competitor (Bio-IK), and among the shortest joint-path lengths, all at a perfect $100\,\%$ success rate. Equally important, AWARE-IK is the only solver whose position accuracy does not degrade as trajectory complexity rises from 20 to 80 waypoints, whereas the faster Jacobian methods lose position accuracy and Relaxed-IK remains both the slowest and the least accurate solver at every scale. AWARE-IK's higher and complexity-dependent solve time is the principal price of this robustness; it remains acceptable for the precision critical regime surgical manipulation, force controlled assembly, and autonomous grasping in cluttered environments for which the method is intended, while solvers such as SDLS-IK retain an advantage where minimal latency outweighs a sub-$0.15\,\mathrm{mm}$ accuracy gain. The physical trials therefore confirm, against a full field of competing solvers and not merely in self-comparison, that AWARE-IK's adaptive weighting translates into the best overall balance of accuracy, smoothness and reliability on real hardware, with position accuracy that is both superior and scale stable at the acknowledged cost of higher solve latency.

5.3.6 Discussion

Experimental results confirm the effectiveness of the proposed AWARE-IK, particularly in managing multi-objective priorities and handling complex robotic arm configurations. It exhibited distinct advantages over traditional methods such as SDLS-IK, DLS-IK, the TRAC-IK variants (TRAC-Speed and TRAC-Distance), Bio-IK and Relaxed-IK, emphasising adaptability, precision, robustness, and computational efficiency.

The adaptive weighting mechanism of the AWARE-IK enables dynamic balancing of competing objectives, including positional accuracy, orientation precision, stability, and computational efficiency. Across diverse test scenarios, AWARE-IK consistently outperforms most competing methods, particularly excelling in complex configurations like Overhead and Low Reach tasks. The solver consistently maintains superior positional and orientation accuracy, with median errors frequently below 0.0001 metres and 0.002 radians, respectively. Additionally, AWARE-IK maintains stable convergence, even near kinematic singularities or joint limits, effectively combining high precision with robustness for constrained environments.

Robustness analyses under varying Gaussian noise levels further validate AWARE-IK's consistent performance. The adaptive weighting strategy effectively mitigates the adverse impacts of noise, maintaining stable accuracy and convergence across challenging conditions. This feature makes AWARE-IK particularly suitable for real-world scenarios characterised by significant measurement inaccuracies.

In comparison, SDLS-IK exhibits marked sensitivity to noise, particularly evident in scenarios demanding high precision, such as low-reach and overhead-reach tasks. Although computationally efficient, SDLS-IK shows reduced accuracy and lower success rates under increased noise levels.

DLS-IK similarly struggles, displaying elevated errors and slower convergence in higher noise conditions, indicating limited resilience to uncertain environments.

The TRAC-IK solvers provide valuable comparative benchmarks. Both TRAC-Speed and TRAC-Distance demonstrate impressive computational efficiency, consistently achieving computation times below 0.1 seconds. TRAC-Speed is particularly suited to real-time control scenarios due to its rapid latency (Time to First Valid around 14 ms). However, it incurs slightly higher orientation errors, especially with significant pitch or yaw deviations. TRAC-Distance, despite being marginally slower, generates smoother trajectories with minimal joint displacement, ideal for applications requiring predictable and smooth motion.

While TRAC variants demonstrate exceptional reliability and good success rates in single-shot IK studies, benefiting significantly from their dual-algorithm architecture and restart mechanisms, they encounter substantial limitations in continuous trajectory tracking scenarios. Specifically, when tasked with following a realistic, time varying path, TRAC-IK solvers experience increased orientation errors and performance degradation, limiting their applicability to scenarios demanding sustained precision.

The successful hardware validation confirms that the AWARE-IK solver provides a practical and reliable solution for real-time IK on redundant manipulators. The combination of consistent convergence, high accuracy, and autonomous adaptation to joint limits demonstrates its readiness for deployment in industrial and research applications requiring robust manipulation capabilities.

Overall, these findings suggest AWARE-IK is exceptionally well-suited for managing noise, adapting to external disturbances, and delivering high accuracy and robustness critical for precision-demanding tasks such as surgical manipulation, force-controlled assembly, or autonomous grasping in cluttered environments. Conversely, TRAC-IK solvers, while optimal for single-shot scenarios, face considerable challenges in dynamic trajectory tracking. SDLS-IK and DLS-IK remain viable for low-noise, moderate-accuracy scenarios, but their performance markedly declines under challenging or uncertain conditions. The results highlight the critical role of adaptive weighting strategies like AWARE-IK in enhancing robustness and reliability in practical robotic applications.

A note on the path from the present results to industrial deployment is appropriate here, since the chapter has presented both simulation-based validation and a preliminary hardware validation on a physical Franka robot manipulator. The hardware validation was conducted under controlled experimental conditions, with conservative joint-velocity and acceleration limits (30% and 20% of the manipulator's maximum, respectively) and with continuous state monitoring through the Franka state controller interface. While these conditions are appropriate for first stage hardware validation of a research solver, they fall short of the conditions that would be required for industrial deployment in collaborative robotics settings, where the robot operates in shared workspace with humans and the control system must be formally certified against established functional safety

standards.

The relevant standards for collaborative robotic manipulation include ISO 10218-1 and ISO 10218-2 (the foundational standards for industrial-robot and robot-system safety), ISO/TS 15066 (the technical specification for collaborative robots, which introduces power and force limiting requirements and biomechanical limit thresholds defining the maximum forces a robot may exert on different body regions), IEC 61508 (generic functional safety for electronic safety-related systems, defining Safety Integrity Levels SIL 1 through SIL 4), and ISO 13849-1 (safety-related parts of control systems, defining Performance Levels PL a through PL e). Collaborative robot applications typically require Performance Level d or e under ISO 13849-1, depending on the application and the risk assessment. Compliance with these standards is established not at the algorithm level but at the integrated control system level, encompassing the IK solver, the torque control layer beneath it, the sensors providing state feedback, and the communication infrastructure connecting them.

AWARE-IK as currently implemented produces joint-position targets at the kinematic layer; integrating it within a control stack that meets the relevant subset of these standards for a specific deployment scenario is the natural next stage of validation. This integration would involve coupling AWARE-IK with a safety certified torque control layer, for example, a controller implementing the Safe Torque Off (STO) and Safely Limited Torque (SLT) functions defined in IEC 61800-5-2, and verifying that the combined system respects the power and force limits prescribed by ISO/TS 15066 for the intended application class. The Franka robot platform's torque sensing capabilities at every joint, combined with its native impedance-control modes, provide a reasonable starting point for such integration. However, achieving formal certification requires not only the integration itself but also the supporting documentation, hazard analysis, and validation testing that the standards prescribe. Such certification work lies beyond the scope of the present thesis, which establishes the algorithmic and kinematic foundations on which a certified deployment could be built. We identify it here as the principal remaining step on the path from the validated solver presented in this chapter to industrial deployment in collaborative manipulation settings.

5.4 Conclusions

This chapter presented AWARE-IK, a novel IK solver that addresses fundamental limitations in existing approaches for redundant manipulator control. The research is motivated by the critical need for IK solutions that can simultaneously handle singularities, manage competing objectives, and maintain computational efficiency in real-world robotic applications.

The primary contribution of this work lies in the development of a unified framework that in-

tegrates damped least squares with adaptive quadratic programming, creating a configuration-aware optimisation strategy. Unlike traditional methods that rely on static weight assignments or rigid task hierarchies, AWARE-IK introduces dynamic joint prioritisation through its dual-factor adaptive weighting mechanism. This mechanism evaluates both joint limit proximity and deviation from neutral configurations, automatically adjusting priorities based on the manipulator's instantaneous kinematic state. The theoretical foundation draws from potential field theory and biological motor control principles, ensuring predictable behaviour while eliminating the need for manual parameter tuning across different tasks.

The comprehensive experimental validation demonstrates AWARE-IK's superiority across multiple evaluation dimensions. The solver consistently achieved high positional and orientation accuracy while maintaining computational efficiency suitable for real-time control applications. Particularly significant is AWARE-IK's robust performance in challenging scenarios, including operation near singular configurations, proximity to joint limits, and trajectory tracking under varying noise conditions. Where competing methods either fail entirely or produce unacceptable errors, AWARE-IK maintains stable convergence and reliable accuracy, validating the effectiveness of its adaptive strategies.

The hardware validation on the Franka Emika Panda robot confirms the practical viability of the approach, demonstrating successful deployment across diverse manipulation tasks with computation times appropriate for real-time control. The solver's ability to operate effectively when joints approach their operational boundaries validates the real-world applicability of the adaptive weighting strategy, addressing a critical limitation of conventional methods that often fail or produce erratic behaviour in such configurations.

A natural question at this point in the thesis is how AWARE-IK relates to the LfD framework developed in Chapter 4. The two contributions are best understood as sequential refinements of a shared problem, the construction of an IK layer that is both kinematically feasible and adaptive to varying task requirements, rather than as a single integrated system. Chapter 4 established the DLS-IK with ProMPs integration as a foundational LfD framework, in which a fixed-parameter DLS solver ensured kinematic feasibility for trajectories generated by a probabilistic movement-primitive model. The closing analysis of that chapter identified three structural limitations of the framework: a reliance on fixed damping parameters unable to adapt to varying task requirements, predetermined null-space objectives specified independently of configuration, and a hard primary/secondary separation that cannot rebalance prioritisation as the manipulator's state evolves. The present chapter's AWARE-IK contribution is the methodological response to all three of these limitations, adaptive damping replaces the fixed damping factor; the configuration-dependent weight matrix replaces predetermined secondary-task velocities; and the QP cost-function formulation dissolves the primary and secondary boundary entirely (per the structural analysis in section 5.2.3).

The relationship between AWARE-IK and LfD is therefore, in its current form, at the level of mechanism limitation mapping rather than at the level of empirical integration. AWARE-IK has been developed and evaluated in this chapter as a standalone IK solver against contemporary comparators, not as a drop-in component within the Chapter 4 ProMPs pipeline. This separation is deliberate entangling the AWARE-IK contribution with the ProMPs framework at the experimental level would have obscured which performance gains arise from the IK reformulation versus from the LfD integration. However, the architectural framework for combining the two is straightforward and worth articulating explicitly.

In a combined AWARE-IK + ProMPs pipeline, ProMPs would continue to play the same role as in Chapter 4: training on demonstrations, encoding a distribution over trajectories, and generating a candidate joint-space trajectory at replay time. AWARE-IK would replace DLS-IK as the kinematic-feasibility layer applied to each waypoint of the generated trajectory. Three specific improvements over the original DLS-IK with ProMPs framework would follow from this substitution. First, the adaptive damping mechanism would regularise the pseudo-inverse more aggressively at waypoints near singular configurations and less aggressively at well-conditioned waypoints, preserving trajectory accuracy where the kinematics permits and ensuring stability where it does not, a measured improvement over the fixed-damping behaviour reported in Chapter 4, where the single λ value forced a global accuracy-stability trade-off. Second, the adaptive weighting mechanism of §5.2.3.1 would penalise motion at joints approaching their limits within the trajectory replay, automatically redistributing motion to less constrained joints when the ProMP generated trajectory drives a joint toward its boundary, a measured improvement over Chapter 4's null-space projection approach, which required a separate secondary-task velocity to be specified in advance and projected at each waypoint. Third, the QP formulation eliminates the explicit projection step entirely, so the discontinuities and parameter-tuning concerns associated with null-space projection in Chapter 4 do not arise in the combined framework.

The empirical demonstration of these three predicted improvements through a controlled comparison of DLS-IK + ProMPs versus AWARE-IK + ProMPs on a common set of demonstrations, is identified here as the principal remaining piece of cross-chapter integration work. Such a comparison would test, at the experimental level, the architectural claim that AWARE-IK is the methodological successor to Chapter 4's DLS-IK within the LfD context. The conceptual relationship articulated in this section provides the framework for such a comparison; the relationship's empirical instantiation is appropriate future work for which the existing Chapter 4 codebase provides the foundation.

The development of AWARE-IK demonstrates that configuration-aware optimisation can successfully balance multiple competing objectives in real-time while maintaining the mechanical compliance and stability foundations established through the DLS-IK framework. By incorporating principles from both optimisation theory and biological motor control, this work provides a comprehensive solution that bridges the gap between theoretical elegance and practical deployment

requirements. The solver's demonstrated ability to maintain high precision while autonomously adapting to kinematic constraints and environmental uncertainties represents a significant advancement toward reliable, deployable IK solutions that can support the next generation of intelligent robotic systems in complex, real-world applications.

Chapter 6

Conclusions and Future Work

6.1 Research Summary and Achievement of Objectives

This thesis has addressed fundamental challenges in robotic motion planning and IK through a systematic progression from empirical evaluation to theoretical development and practical implementation. The research began with a broad evaluation of existing motion planning approaches, proceeded to develop enhanced methods for LfD, and finally created an adaptive framework for efficient IK.

The first research objective focused on investigating the effect of environmental complexity on the adaptability and motion feasibility of robotic arms in cluttered environment. Through the comprehensive benchmarking framework presented in Chapter 3, we systematically evaluated twelve motion planners across three robotic platforms under varying degrees of environmental clutter. The experimental analysis revealed that planner performance degrades significantly as environmental constraints increase, with success rates varying from great performance in open environments to critically low levels in highly constrained scenarios. This investigation established quantitative relationships between workspace characteristics, robot kinematics, and planner effectiveness, providing practitioners with guidance for system selection.

The second objective aimed to develop enhanced strategies for adaptive motion planning by integrating learned motion priors with optimisation-based IK. Chapter 4 addressed this through the novel integration of DLS-IK with ProMPs. This approach bridged the gap between human demonstrations and mechanically compliant robot trajectories, showing that learned motion patterns could be combined with kinematic constraints to produce feasible and efficient movements. The experimental validation confirmed that this integration maintains stability near singularities while preserving the adaptability inherent in demonstration-based learning.

The third objective sought to formulate robust mechanisms for generalisation and flexibility across diverse tasks and workspace constraints. Chapter 5 addressed this objective through the introduction of AWARE-IK, an Adaptive Weighted and Regularised Optimisation framework that draws together two distinct strands of evidence accumulated across the preceding chapters. From the benchmarking study in Chapter 3, AWARE-IK inherits the diagnostic insight that planner reliability in cluttered and constrained workspaces depends critically on the ability to modulate behaviour in response to local kinematic conditions, rather than on fixed parameter selection. From the DLS-IK and ProMPs integration developed in Chapter 4, it inherits the complementary observation that, although damping with null-space projection successfully imposes mechanical compliance on learned trajectories, the use of fixed damping factors and predetermined secondary objectives limits the framework's capacity to reprioritise competing constraints in real time. AWARE-IK responds directly to these two limitations by replacing static weight assignments with a configuration-aware optimisation strategy. A dual-factor adaptive weighting scheme, informed by joint-limit proximity and deviation from neutral configurations, continuously modulates joint priorities as the manipulator evolves through its workspace, while a hybrid solving strategy combining SLSQP with a damped least squares fallback ensures convergence across both well-conditioned and near-singular regions. The experimental campaign confirmed that this configuration-aware formulation maintains sub-millimetre positional accuracy and sub-degree orientation accuracy across noise levels up to $\sigma = 0.02$, retains stable convergence in scenarios where SDLS-IK and Relaxed-IK degrade or fail, and operates within computation times compatible with real-time control cycles. The hardware validation on the Franka robot further demonstrated that the adaptive weighting mechanism autonomously manages proximity to joint limits without manual tuning, achieving a high success rate across the five validation scenarios. AWARE-IK therefore constitutes not a parallel contribution to Chapters 3 and 4, but a single, dynamically adaptive optimisation framework that reconciles the empirical lessons of cross-planner benchmarking with the kinematic safeguards of demonstration-based learning.

The methodology followed a structured progression that analysis first, then targeted improvement, then broader innovation. Each contribution built on the insights of the previous one, and together they form a cohesive framework for robotic manipulation. The experimental validation across simulation and hardware platforms confirms that the proposed methods achieve practical improvements in accuracy, robustness, and computational efficiency compared to existing approaches.

6.2 Significance of Research Contributions

The three major contributions presented in this thesis collectively advance the field of robotic motion planning and manipulation through complementary theoretical developments and practical implementations. Each addresses specific limitations in current approaches while enriching the

overall picture of effective robotic manipulation.

The comprehensive benchmarking framework established in Chapter 3 provides the research community with a systematic methodology for evaluating motion planning algorithms under realistic operational conditions. This contribution extends beyond simple performance comparison by revealing the complex interactions between planner characteristics, robot kinematics, and environmental constraints. The identification of parameter sensitivity patterns and the development of weighted scoring metrics enable practitioners to make informed decisions about planner selection based on specific application requirements. The framework's modular design also makes it straightforward to add new planners and robotic platforms. In practice, this work reduces development time, since engineers can use the empirical results instead of extensive trial-and-error testing.

The integration of DLS-IK with ProMPs represents a significant advancement in LfD techniques. This contribution addresses a critical challenge in transferring human expertise to robotic systems: the preservation of mechanical feasibility while maintaining the adaptability of learned behaviours. By incorporating kinematic constraints directly into the learning process, the method ensures that generated trajectories respect joint limits and avoid singularities without sacrificing the natural movement patterns captured from demonstrations. The theoretical significance lies in demonstrating that probabilistic learning frameworks can be effectively combined with deterministic kinematic constraints, and points towards hybrid approaches that draw on both data-driven and model-based techniques. The practical implications extend to industrial applications where non-expert operators need to teach robots new tasks without extensive programming knowledge.

The AWARE-IK framework advances IK solving from static, fixed-parameter regularisation toward configuration-aware, dynamically weighted optimisation. The adaptive weighting mechanism introduces configuration awareness that enables real-time prioritisation of competing objectives based on instantaneous kinematic state. This contribution advances the understanding of multi-objective optimisation in robotics by demonstrating that, across the scenarios reported in Chapter 5, dynamic weight adjustment achieves superior balancing of competing objectives compared to fixed priority schemes. The framework's hybrid solving strategy, combining QP with damped least squares fallback, ensures robust convergence across the entire workspace while maintaining computational efficiency. The practical significance extends to applications requiring precise manipulation in constrained environments, such as surgical robotics, assembly automation, and service robotics, where the ability to balance accuracy, joint limit avoidance, and singularity management determines operational success.

Summary, these contributions add up to more than their individual merits. They establish a comprehensive framework for developing and deploying robotic manipulation systems that combines empirical understanding, theoretical advancement, and practical implementation. The progression from benchmarking through enhanced learning to adaptive optimisation demonstrates a

methodology for systematic improvement in robotic capabilities which integrated approach provides researchers with validated techniques for addressing the complete pipeline from high-level motion planning to low-level kinematic control, while offering practitioners proven solutions for real-world deployment challenges.

6.3 Future Research Directions

The research presented in this thesis opens several promising avenues for future investigation, particularly in strengthening the integration between robotics and artificial intelligence. As robotic systems increasingly operate alongside humans in unstructured environments, combining classical robotics techniques with modern AI approaches becomes essential for achieving truly autonomous and adaptive behaviour.

The immediate extension of this work involves incorporating deep learning techniques to enhance the adaptive capabilities of the AWARE-IK framework. While the current implementation relies on analytically derived weight functions, neural networks could learn optimal weighting strategies from experience, potentially discovering non-intuitive parameter relationships that improve performance. This learning-based approach could extend to predicting task-specific optimisation parameters based on visual perception of the environment, enabling the system to preemptively adjust its configuration strategy before encountering constraints. The integration of transformer architectures, which have revolutionised natural language processing and computer vision, could enable robots to understand and execute complex, multi-step manipulation tasks described in natural language, building upon the motion planning and IK foundations established in this thesis.

A particularly pressing direction for future work concerns the integration of the kinematic and planning contributions developed in this thesis with the rapidly maturing body of research on large language models (LLMs), vision-language models (VLMs), and vision-language-action (VLA) policies for robotic task planning. Systems such as SayCan, Code-as-Policies, VoxPoser, OpenVLA, and Octo have demonstrated that large pre-trained models can decompose natural language instructions into structured robot behaviours, ground linguistic references to objects in visual scenes, and, in the case of end-to-end VLA policies, directly emit low-level action trajectories. However, a now well-documented limitation of these systems is that their performance degrades sharply in the regime where this thesis has concentrated its contributions, sub-millimetre Cartesian accuracy, controlled behaviour near kinematic singularities, strict joint-limit avoidance, and certifiable safety guarantees during contact-rich manipulation. End-to-end VLA policies, in particular, tend to produce trajectories that are semantically plausible but kinematically suboptimal, frequently violating velocity, acceleration, or singularity-avoidance constraints that traditional analytical solvers handle by construction.

The methods developed in this thesis are well positioned to occupy the lower layers of a hybrid, three-tier architecture for language-conditioned manipulation. In the upper layer, an LLM serves as a symbolic task planner, decomposing a high-level instruction such as "place the red cup behind the kettle" into an ordered sequence of parameterised skills. In the middle layer, a VLM performs open-vocabulary visual grounding, mapping the referring expressions in each subgoal to six-degree-of-freedom target poses in the robot's workspace; recent grounding pipelines combining Grounding DINO with SAM2 illustrate the maturity of this layer. The lower layer where the contributions of this thesis reside is responsible for converting each grounded Cartesian subgoal into a kinematically feasible, dynamically smooth joint-space trajectory and executing it under cycle control. Within this layer, AWARE-IK provides the configuration-aware IK solver capable of resolving the redundant joint configurations emitted by the upper layers while respecting joint limits and managing proximity to singularities; the DLS-IK and ProMPs integration of Chapter 4 provides a mechanism for biasing the lower-layer trajectories toward demonstrated motion styles, which is particularly valuable when an LLM-issued subgoal admits many kinematically valid completions and the appropriate one must be selected on stylistic or task-affinity grounds; and the benchmarking framework of Chapter 3 provides the methodological substrate for systematically evaluating how different LLM-driven task decompositions interact with downstream planner choices, an analysis that the language-conditioned robotics literature currently lacks.

Hybrid neuro symbolic systems are another important direction for robotic manipulation. By combining the pattern recognition strengths of neural networks with the logical reasoning capabilities of symbolic AI, future systems could maintain the reliability and interpretability of classical approaches while gaining the adaptability and generalisation abilities of learning-based methods. This integration would be particularly valuable for safety-critical applications where the system must provide guarantees about its behaviour while adapting to novel situations. The IK solver could incorporate learned components that suggest initial solutions or constraint relaxations, while symbolic reasoning ensures that fundamental physical constraints and safety requirements are never violated.

The expansion toward multi-robot coordination presents opportunities to apply the adaptive optimisation principles developed in this thesis to distributed systems. Future research could explore how multiple robots can share learned motion primitives and collectively optimise their movements to accomplish collaborative tasks. This would require extending the current framework to consider inter-robot constraints and developing communication protocols that enable real-time coordination while maintaining individual robot autonomy. The challenge lies in scaling the optimisation approaches to handle the exponentially larger configuration spaces while maintaining the computational efficiency necessary for practical deployment.

Integrating advanced perception systems with motion planning and control is another important area for future development. Current approaches often treat perception and planning as separate modules, but tighter integration could enable more responsive and intelligent behaviour. Future

systems could use visual and tactile feedback to continuously refine their motion plans and adapt their IK solutions in real-time. This sensorimotor integration would be particularly valuable for contact-rich manipulation tasks where precise force control and environmental interaction are essential. Machine learning techniques could help robots learn the relationship between sensory feedback and optimal control adjustments, creating systems that improve their performance through experience.

Foundation models for robotics could prove transformative. Large scale models trained on diverse robotic data could provide universal motion priors that adapt to specific tasks through fine-tuning, much as large language models did for natural language processing. These models could encode general manipulation knowledge that transfers across different robot morphologies and task domains, potentially eliminating the need for extensive task-specific programming. The challenge lies in collecting and organising sufficient training data and developing architectures that can effectively represent the continuous, high-dimensional nature of robotic control while maintaining the interpretability and safety guarantees required for practical deployment.

The ethical and societal implications of increasingly capable robotic manipulators warrant careful consideration in future research, and two complementary research programmes follow from the contributions developed in this thesis. The first concerns user conditioned parameter tailoring, that is, the principled adaptation of solver parameters to the physical and cognitive characteristics of the human user with whom the manipulator interacts. The second concerns interpretability of solver behaviour, that is, the development of mechanisms by which a robotic system can explain, in human comprehensible terms, why it produced a particular trajectory or IK solution.

The mathematical structure of AWARE-IK lends itself naturally to user-conditioned tailoring. The adaptive weighting scheme developed in Chapter 5 expresses joint priorities as a multiplicative composition of a base weight, a limit-proximity factor, and a centre-deviation factor, with an adaptive damping parameter governing the transition between well-conditioned and near-singular regimes. Each of these components is, at present, a function only of the manipulator's instantaneous kinematic state. A straightforward and motivated extension would reformulate them as functions of a user-context vector u, drawn from sensing, profiling, or explicit operator declaration, that encodes properties such as the user's age, frailty, range of voluntary motion, sensory acuity, and rehabilitation status. Under such a reformulation, the velocity and jerk envelopes governing trajectory smoothness would tighten in the presence of paediatric or geriatric users; the clearance margins encoded in the limit-proximity factor would expand for users with reduced reaction time; the centre-deviation factor would bias the manipulator toward conservative, easily predictable home configurations when operating near fragile anatomy; and the damping parameter would be elevated to favour smooth, low-acceleration motion in interactions with post-surgical, rehabilitation, or paediatric users where high acceleration manoeuvres pose elevated injury risk. The Probabilistic Movement Primitives framework introduced in Chapter 4 admits an analogous extension that the variance of the ProMP weight distribution could be reduced under fragile-user

conditions, restricting the manipulator to the most confidently demonstrated regions of trajectory space and excluding the long tails of the demonstration distribution.

The convergence of robotics and artificial intelligence promises capabilities that neither field could achieve alone. Building on the work presented in this thesis, future research can combine the precision and reliability of classical approaches with the adaptability of modern AI, and so deliver practical systems that improve productivity and quality of life across a range of applications.

List of Abbreviations

Abbreviation	Definition
AWARE-IK	Adaptive Weighted and Regularised Optimisation for Efficient IK
BKPIECE	Bi-directional KPIECE
CNMPs	Conditional Neural Movement Primitives
DOF	Degrees of Freedom
DH	Denavit–Hartenberg
DLS-IK	Damped Least Squares Inverse Kinematics
DMPs	Dynamic Movement Primitives
EST	Expansive Space Trees
FK	Forward Kinematics
FMT	Asymptotically Optimal Fast Marching Tree
GMMs	Gaussian Mixture Models
G-DLS-IK	Gaussian Damped Least Squares Inverse Kinematics
HSMM	Hidden Semi-Markov Model
IK	Inverse Kinematics
KMPs	Kernelised Movement Primitives
KPIECE	Kinodynamic Planning by Interior-Exterior Cell Exploration
KDL	Kinematics and Dynamics Library
LBKPIECE	Lazy Bi-directional KPIECE
LfD	Learning from Demonstration
MBM	MotionBenchMaker
PDST	Path-Directed Subdivision Tree
PRM	Probabilistic Roadmap
ProMPs	Probabilistic Movement Primitives
PoE	Product of Exponentials
QP	Quadratic Programming
RNG	Random Number Generator
ROS	Robot Operating System
RRT	Rapidly-exploring Random Trees

Abbreviation	Definition
RRT*	Optimal Rapidly-exploring Random Trees
RRTConnect	Rapidly-exploring Random Tree Connect
SBL	Single-query Bi-directional Lazy roadmap
STRIDE	Search Tree with Resolution Independent Density Estimation
SQP	Sequential Quadratic Programming
SDLS-IK	Selective Damped Least Squares Inverse Kinematics
SLSQP	Sequential Least Squares Programming
STOMP	Stochastic Trajectory Optimization for Motion Planning
TRRT	Transition-based Rapidly-exploring Random Tree
TrajOpt	Trajectory Optimization

List of References

[1] S. Yang, P. Liu, and N. Pears, "Benchmarking of robot arm motion planning in cluttered environments," in *2023 28th International Conference on Automation and Computing (ICAC)*, pp. 1–6, IEEE, 2023.

[2] S. Yang, P. Liu, and N. Pears, "Enhancing learning from demonstration with dls-ik and promps," in *2024 29th International Conference on Automation and Computing (ICAC)*, pp. 1–6, IEEE, 2024.

[3] S. Yang, P. Liu, and N. Pears, "Reinforcement learning-based adaptive probabilistic movement primitives in hybrid scenarios," in *The 24th Annual Conference Towards Autonomous Robotic Systems (TAROS2023)*, 2023.

[4] A. Gasparetto and L. Scalera, "From the unimate to the delta robot: the early decades of industrial robotics," in *Explorations in the History and Heritage of Machines and Mechanisms*, pp. 284–295, Springer, 2019.

[5] F. Piltan, S. T. Haghighi, N. Sulaiman, I. Nazari, and S. Siamak, "Artificial control of puma robot manipulator: A-review of fuzzy inference engine and application to classical controller," *International Journal of Robotics and Automation*, vol. 2, no. 5, pp. 401–425, 2011.

[6] J.-C. Latombe, *Robot motion planning*, vol. 124. Springer Science & Business Media, 2012.

[7] J.-G. Kang, D.-W. Lim, Y.-S. Choi, W.-J. Jang, and J.-W. Jung, "Improved rrt-connect algorithm based on triangular inequality for robot path planning," *Sensors*, vol. 21, no. 2, p. 333, 2021.

[8] J. Men and J. R. Carrión, "A generalization of the chomp algorithm for uav collision-free trajectory generation in unknown dynamic environments," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pp. 96–101, IEEE, 2020.

[9] H. Ravichandar, A. Polydoros, S. Chernova, and A. Billard, "Robot learning from demonstration: A review of recent advances," *Annual Review of Control, Robotics, and Autonomous Systems*, 2019.

[10] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual review of control, robotics, and autonomous systems*, vol. 3, pp. 297–330, 2020.

[11] S. Calinon, "Learning from demonstration (programming by demonstration)," *Encyclopedia of robotics*, pp. 1–8, 2018.

[12] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[13] F. Frank, A. Paraschos, P. van der Smagt, and B. Cseke, "Constrained probabilistic movement primitives for robot trajectory adaptation," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2276–2294, 2021.

[14] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *The International Journal of Robotics Research*, vol. 42, no. 13, pp. 1133–1184, 2023.

[15] Y. Hu, Y. Wang, K. Hu, and W. Li, "Adaptive obstacle avoidance in path planning of collaborative robots for dynamic manufacturing," *Journal of Intelligent Manufacturing*, pp. 1–19, 2021.

[16] A. Gams, B. Nemec, A. J. Ijspeert, and A. Ude, "Coupling movement primitives: Interaction with the environment and bimanual tasks," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 816–830, 2014.

[17] Z. Zhu and H. Hu, "Robot learning from demonstration in robotic assembly: A survey," *Robotics*, vol. 7, no. 2, p. 17, 2018.

[18] M. Wu, Y. He, and S. Liu, "Shared impedance control based on reinforcement learning in a human-robot collaboration task," in *Advances in Service and Industrial Robotics: Proceedings of the 28th International Conference on Robotics in Alpe-Adria-Danube Region (RAAD 2019) 28*, pp. 95–103, Springer, 2020.

[19] K. Karacan, H. Sadeghian, R. Kirschner, and S. Haddadin, "Passivity-based skill motion learning in stiffness-adaptive unified force-impedance control," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9604–9611, IEEE, 2022.

[20] S. Ruan, W. Liu, X. Wang, X. Meng, and G. S. Chirikjian, "Primp: Probabilistically-informed motion primitives for efficient affordance learning from demonstration," *IEEE Transactions on Robotics*, 2024.

[21] Y. Zhou, J. Gao, and T. Asfour, "Learning via-point movement primitives with inter- and extrapolation capabilities," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4301–4308, IEEE, 2019.

[22] A. Conkey and T. Hermans, "Active learning of probabilistic movement primitives," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pp. 1–8, IEEE, 2019.

[23] A. Liu, S. Zhan, Z. Jin, and W.-a. Zhang, "A variable impedance skill learning algorithm based on kernelized movement primitives," *IEEE Transactions on Industrial Electronics*, 2023.

[24] L. Wang, S. Jia, G. Wang, A. Turner, and S. Ratchev, "Enhancing learning capabilities of movement primitives under distributed probabilistic framework for flexible assembly tasks," *Neural Computing and Applications*, pp. 1–12, 2021.

[25] J. Wang, T. Zhang, N. Ma, Z. Li, H. Ma, F. Meng, and M. Q.-H. Meng, "A survey of learning-based robot motion planning," *IET Cyber-Systems and Robotics*, vol. 3, no. 4, pp. 302–314, 2021.

[26] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[27] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[28] A. Duan, R. Camoriano, D. Ferigo, D. Calandriello, L. Rosasco, and D. Pucci, "Constrained dmps for feasible skill learning on humanoid robots," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*, pp. 1–6, IEEE, 2018.

[29] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[30] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[31] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "Rrt-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE international conference on robotics and biomimetics (ROBIO)*, pp. 1670–1677, IEEE, 2015.

[32] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based rrt to deal with complex cost spaces," in *2013 IEEE international conference on robotics and automation*, pp. 4120–4125, IEEE, 2013.

[33] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *Proceedings of international conference on robotics and automation*, vol. 3, pp. 2719–2726, IEEE, 1997.

[34] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Int. Symp. Robotics Research*, pp. 403–417, 2001.

[35] I. A. Sucan and L. E. Kavraki, "Kinodynamic motion planning by interior-exterior cell exploration," *Algorithmic Foundation of Robotics VIII*, vol. 57, pp. 449–464, 2009.

[36] B. Gipson, M. Moll, and L. E. Kavraki, "Resolution independent density estimation for motion planning in high-dimensional spaces," in *2013 IEEE international conference on robotics and automation*, pp. 2437–2443, IEEE, 2013.

[37] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2072–2078, IEEE, 2015.

[38] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.

[39] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[40] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, vol. 2, pp. 995–1001, IEEE, 2000.

[41] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[42] D. Hsu, J.-C. Latombe, and H. Kurniawati, "On the probabilistic foundations of probabilistic roadmap planning," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 627–643, 2006.

[43] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 18–47, 2014.

[44] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[45] J. D. Gammell and M. P. Strub, "Asymptotically optimal sampling-based motion planning methods," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 295–318, 2021.

[46] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "Chomp: Gradient optimization techniques for efficient motion planning," in *2009 IEEE international conference on robotics and automation*, pp. 489–494, IEEE, 2009.

[47] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, pp. 4569–4574, IEEE, 2011.

[48] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization.," in *Robotics: science and systems*, vol. 9, pp. 1–10, Berlin, Germany, 2013.

[49] C. Chamzas, C. Quintero-Pena, Z. Kingston, A. Orthey, D. Rakita, M. Gleicher, M. Toussaint, and L. E. Kavraki, "Motionbenchmaker: A tool to generate and benchmark motion planning datasets," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 882–889, 2021.

[50] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robotics & Automation Magazine*, vol. 22, no. 3, pp. 96–102, 2015.

[51] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, 2012.

[52] A. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," *Advances in neural information processing systems*, vol. 15, 2002.

[53] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[54] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, pp. 529–551, 2018.

[55] G. Li, Z. Jin, M. Volpp, F. Otto, R. Lioutikov, and G. Neumann, "Prodmp: A unified perspective on dynamic and probabilistic movement primitives," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2325–2332, 2023.

[56] M. Ramírez Montero, G. Franzese, J. Kober, and C. Della Santina, "Learning multi-reference frame skills from demonstration with task-parameterized gaussian processes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024. Accepted.

[57] M. Akbulut, E. Oztop, M. Y. Seker, A. Tekden, E. Ugur, *et al.*, "Acnmp: Skill transfer and task extrapolation through learning from demonstration and reinforcement learning via representation sharing," in *Conference on robot learning*, pp. 1896–1907, PMLR, 2021.

[58] Y. Yildirim and E. Ugur, "Conditional neural expert processes for learning movement primitives from demonstration," *IEEE Robotics and Automation Letters*, 2024.

[59] Y. Huang, L. Rozo, J. Silvério, and D. G. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, no. 7, pp. 833–852, 2019.

[60] M. Dong and D. He, "A segmental hidden semi-markov model (hsmm)-based diagnostics and prognostics framework and methodology," *Mechanical systems and signal processing*, vol. 21, no. 5, pp. 2248–2266, 2007.

[61] S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert, "Learning movement primitives," in *Robotics Research. The Eleventh International Symposium: With 303 Figures*, pp. 561–572, Springer, 2005.

[62] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Humanoids 2008-8th IEEE-RAS international conference on humanoid robots*, pp. 91–98, IEEE, 2008.

[63] H. Hoffmann, P. Pastor, D.-H. Park, and S. Schaal, "Biologically-inspired dynamical systems for movement generation: Automatic real-time goal adaptation and obstacle avoidance," in *2009 IEEE international conference on robotics and automation*, pp. 2587–2592, IEEE, 2009.

[64] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.

[65] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, no. 1, pp. 1–29, 2016.

[66] R. A. Shyam, P. Lightbody, G. Das, P. Liu, S. Gomez-Gonzalez, and G. Neumann, "Improving local trajectory optimisation using probabilistic movement primitives," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 2666–2671, IEEE, 2019.

[67] A. S. Deo and I. D. Walker, "Overview of damped least-squares methods for inverse kinematics of robot manipulators," *Journal of Intelligent and Robotic Systems*, vol. 14, no. 1, pp. 43–68, 1995.

[68] S. R. Buss and J.-S. Kim, "Selectively damped least squares for inverse kinematics," *Journal of Graphics tools*, vol. 10, no. 3, pp. 37–49, 2005.

[69] L. M. Phuoc, P. Martinet, S. Lee, and H. Kim, "Damped least square based genetic algorithm with ggaussian distribution of damping factor for singularity-robust inverse kinematics," *Journal of mechanical science and technology*, vol. 22, no. 7, pp. 1330–1338, 2008.

[70] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pp. 928–935, IEEE, 2015.

[71] S. Starke, *Bio IK: A memetic evolutionary algorithm for generic multi-objective inverse kinematics*. PhD thesis, Staats-und Universitätsbibliothek Hamburg Carl von Ossietzky, 2020.

[72] D. Rakita, B. Mutlu, and M. Gleicher, "Relaxedik: Real-time synthesis of accurate and feasible robot arm motion.," in *Robotics: Science and Systems*, vol. 14, pp. 26–30, Pittsburgh, PA, 2018.

[73] F. Emika, "Robot and interface specifications—franka control interface (fci) documentation," 2022.

[74] L. Sciavicco and B. Siciliano, *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[75] P. I. Corke, W. Jachimczyk, and R. Pillat, *Robotics, vision and control: fundamental algorithms in MATLAB*, vol. 73. Springer, 2011.

[76] M. W. Spong, S. Hutchinson, and M. Vidyasagar, "Robot modeling and control," *John Wiley &amp*, 2020.

[77] S. Kucuk and Z. Bingul, *Robot kinematics: Forward and inverse kinematics*, vol. 1. INTECH Open Access Publisher London, UK, 2006.

[78] S. Haddadin, S. Parusel, L. Johannsmeier, S. Golz, S. Gabl, F. Walch, M. Sabaghian, C. Jähne, L. Hausperger, and S. Haddadin, "The franka emika robot: A reference platform for robotics research and education," *IEEE Robotics & Automation Magazine*, vol. 29, no. 2, pp. 46–64, 2022.

[79] Y. He and S. Liu, "Analytical inverse kinematics for franka emika panda–a geometrical solver for 7-dof manipulators with unconventional design," in *2021 9th International Conference on Control, Mechatronics and Automation (ICCMA)*, pp. 194–199, IEEE, 2021.

[80] F. C. Park, "Computational aspects of the product-of-exponentials formula for robot kinematics," *IEEE transactions on automatic control*, vol. 39, no. 3, pp. 643–647, 1994.

[81] P. I. Corke, "A simple and systematic approach to assigning denavit–hartenberg parameters," *IEEE transactions on robotics*, vol. 23, no. 3, pp. 590–594, 2007.

[82] A. Mueller, "Modern robotics: Mechanics, planning, and control [bookshelf]," *IEEE Control Systems Magazine*, vol. 39, no. 6, pp. 100–102, 2019.

[83] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Force control*. Springer, 2009.

[84] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 2007.

[85] H. Bruyninckx, "Open robot control software: the orocos project," in *Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. No. 01CH37164)*, vol. 3, pp. 2523–2528, IEEE, 2001.

[86] J. Li, S. Bian, Q. Liu, J. Tang, F. Wang, and C. Lu, "Niki: Neural inverse kinematics with invertible neural networks for 3d human pose and shape estimation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12933–12942, 2023.

[87] S. Rice, A. Azab, and S. Saad, "Fusion ik: Solving inverse kinematics using a hybridized deep learning and evolutionary approach," *Manufacturing Letters*, vol. 41, pp. 9–18, 2024.

[88] S. Liu and P. Liu, "A review of motion planning algorithms for robotic arm systems," in *RiTA 2020: Proceedings of the 8th International Conference on Robot Intelligence Technology and Applications*, pp. 56–66, Springer, 2021.

[89] P. Liu, M. N. Huda, L. Sun, and H. Yu, "A survey on underactuated robotic systems: bio-inspiration, trajectory planning and control," *Mechatronics*, vol. 72, p. 102443, 2020.

[90] P. Liu, H. Yu, and S. Cang, "On periodically pendulum-diven systems for underactuated locomotion: a viscoelastic jointed model," in *2015 21st International Conference on Automation and Computing (ICAC)*, pp. 1–6, IEEE, 2015.

[91] P. Liu, G. Neumann, Q. Fu, S. Pearson, and H. Yu, "Energy-efficient design and control of a vibro-driven robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1464–1469, IEEE, 2018.

[92] S. Liu and P. Liu, "Benchmarking and optimization of robot motion planning with motion planning pipeline," *The International Journal of Advanced Manufacturing Technology*, pp. 1–13, 2021.

[93] B. Zhang and P. Liu, "Control and benchmarking of a 7-dof robotic arm using gazebo and ros," *PeerJ Computer Science*, vol. 7, p. e383, 2021.

[94] A. Vivas and J. M. Sabater, "Ur5 robot manipulation using matlab/simulink and ros," in *2021 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 338–343, IEEE, 2021.

[95] P. J. Tarwadi, A. S. Arockia, R. Corrales, and A. Juan, "Manipulation and path planning for kuka (lwr/lbr 4+) robot in a simulated and real environment," *Journal of Automation, Mobile Robotics and Intelligent Systems*, pp. 15–21, 2017.

[96] K. Bekris, B. Chen, A. Ladd, E. Plaku, and L. Kavraki, "Multiple query motion planning using single query primitives. ieee/rsj int," in *Conf. Intell. Robot. Syst*, pp. 656–661, 2003.

[97] K. Gochev, A. Safonova, and M. Likhachev, "Planning with adaptive dimensionality for mobile manipulation," in *2012 IEEE International Conference on Robotics and Automation*, pp. 2944–2951, IEEE, 2012.

[98] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[99] I. A. Sucan and S. Chitta, "Moveit!," 2013.

[100] A. Yershova and S. M. LaValle, "Deterministic sampling methods for spheres and so (3)," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 4, pp. 3974–3980, IEEE, 2004.

[101] D. Lee, W. Lee, J. Park, and W. K. Chung, "Task space control of articulated robot near kinematic singularity: forward dynamics approach," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 752–759, 2020.

[102] J. Manavalan and M. Howard, "Learning singularity avoidance," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6849–6854, IEEE, 2019.

[103] S. Tittel, "Analytical solution for the inverse kinematics problem of the franka emika panda seven-dof light-weight robot arm," in *2021 20th International Conference on Advanced Robotics (ICAR)*, pp. 1042–1047, IEEE, 2021.

[104] C.-K. Ho and C.-T. King, "Automating the learning of inverse kinematics for robotic arms with redundant dofs," 2022.

[105] X. Ma, S. Patidar, I. Haughton, and S. James, "Hierarchical diffusion policy for kinematics-aware multi-task robotic manipulation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18081–18090, 2024.

[106] O. M. Omisore, S. Han, L. Ren, A. Elazab, L. Hui, T. Abdelhamid, N. A. Azeez, and L. Wang, "Deeply-learnt damped least-squares (dl-dls) method for inverse kinematics of snake-like robots," *Neural Networks*, vol. 107, pp. 34–47, 2018.

[107] A. D. Sosa-Ceron, H. G. Gonzalez-Hernandez, and J. A. Reyes-Avendaño, "Learning from demonstrations in human–robot collaborative scenarios: A survey," *Robotics*, vol. 11, no. 6, p. 126, 2022.

[108] H. Simas and R. Di Gregorio, "A technique based on adaptive extended jacobians for improving the robustness of the inverse numerical kinematics of redundant robots," *Journal of Mechanisms and Robotics*, vol. 11, no. 2, p. 020913, 2019.

[109] S. Lloyd, R. A. Irani, and M. Ahmadi, "Fast and robust inverse kinematics of serial robots using halley's method," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2768–2780, 2022.

[110] H. Sadeghian, L. Villani, M. Keshmiri, and B. Siciliano, "Task-space control of robot manipulators with null-space compliance," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 493–506, 2013.

[111] H. Wang, Z. Zhou, X. Zhong, and Q. Chen, "Singular configuration analysis and singularity avoidance with application in an intelligent robotic manipulator," *Sensors*, vol. 22, no. 3, p. 1239, 2022.

[112] X. Wang, X. Liu, L. Chen, and H. Hu, "Deep-learning damped least squares method for inverse kinematics of redundant robots," *Measurement*, vol. 171, p. 108821, 2021.

[113] Y. Wang, N. Dehio, A. Tanguy, and A. Kheddar, "Impact-aware task-space quadratic-programming control," *The International Journal of Robotics Research*, vol. 42, no. 14, pp. 1265–1282, 2023.

[114] S. Li, K. Han, P. He, Z. Li, Y. Liu, and Y. Xiong, "Human-like redundancy resolution: an integrated inverse kinematics scheme for anthropomorphic manipulators with radial elbow offset," *Advanced Engineering Informatics*, vol. 54, p. 101812, 2022.

[115] D. Wu, G. Hou, W. Qiu, and B. Xie, "T-ik: An efficient multi-objective evolutionary algorithm for analytical inverse kinematics of redundant manipulator," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8474–8481, 2021.

[116] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 785–792, 2011.

[117] T. Chaki and T. Kawakami, "Quadratic programming based inverse kinematics for precise bimanual manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 16024–16030, IEEE, 2024.

[118] H. Liu, Z. Yan, and J. Xiao, "Pose error prediction and real-time compensation of a 5-dof hybrid robot," *Mechanism and Machine Theory*, vol. 170, p. 104737, 2022.

[119] D. Tolani, A. Goswami, and N. I. Badler, "Real-time inverse kinematics techniques for anthropomorphic limbs," *Graphical models*, vol. 62, no. 5, pp. 353–388, 2000.

[120] J.-P. Merlet, "Jacobian, manipulability, condition number, and accuracy of parallel robots," 2006.

[121] G. Du and P. Zhang, "A markerless human–robot interface using particle filter and kalman filter for dual robots," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2257–2264, 2014.