Real-Time Kinematic Redundancy Resolution and Path Parameterization for Redundant Manipulators

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Wolfgang Wiedmeyer

Tag der mündlichen Prüfung:	03. Februar 2026
1. Referent:	Prof. Dr.-Ing. Tamim Asfour
2. Referent:	Prof. Dr.-Ing. Torsten Kröger

This thesis was made possible by the support and contributions of many people over the years, for which I want to express my gratitude.

First, I would like to thank my advisors Prof. Tamim Asfour and Prof. Torsten Kröger, who gave me the opportunity to work at IAR-IPR, allowed me to dive deep into interesting and challenging research topics, and supported me all the way to the finish line.

Thank you to all the colleagues and friends at IAR-IPR who helped me and created a fun and enriching work environment. I will miss the camaraderie with you. A big thank you to Dr. Xi Huang who assisted with the research on real-time path parameterization. The contribution would not have been possible without his implementation of the path pre-processing. Mark Weinreuter helped tremendously with the implementation of several important usability improvements for the Robot Learning Lab, and with overall code quality. My office colleague Michael Mende supported me in so many ways, especially in setting up the Robot Learning Lab infrastructure. Dr. Christoph Ledermann provided advice and encouragement. Dennis Hartmann and Dr. Denis Štogl gave quite a bit of starting aid when I was still a rookie at the institute.

I also had the pleasure to advise and work with several exceptionally motivated and talented students. They did initial investigations and work on all three parts of the thesis, most notably: Philipp Altoé looked into the closed-form multi-objective optimization for Inverse Kinematics, Leandro Piekarski do Nascimento explored analytical formulation for maximum path velocities and real-time-capable trajectory generation, and Moritz Behr, Iris Andrussow, Lukas Feldmann, and Raphael Wirth contributed to the automated submission processing and project designs for the Robot Learning Lab.

Finally, it would not have been possible to overcome all the challenges along the way without my family and friends outside of work. I greatly appreciate your loving support and thank you for being there.

Abstract

Real-Time Kinematic Redundancy Resolution and Path Parameterization for Redundant Manipulators

This thesis focuses on creating real-time-capable solutions for online robot motion planning problems. The development of a novel lab as an application for these algorithms is also part of this thesis. Online motion planning allows robots to react instantaneously to unexpected events and adapt their motions on the fly. Real-time motion planning addresses the additional safety and reliability requirements by ensuring that movements are planned deterministically within strict bounds on computation time and without unexpected failure.

Goal poses or full paths that the manipulator's end-effector should reach or traverse are mapped to the required joint angles by an Inverse Kinematics (IK) algorithm. A target pose for a robotic manipulator in Cartesian space is usually specified by six parameters representing position and orientation, which renders the IK problem under-constrained for anthropomorphic manipulators with seven degrees of freedom and requires redundancy resolution to determine a unique solution. The *first* contribution of this thesis is a real-time-capable redundancy resolution scheme for such manipulators. A closed-form optimization solution is derived that can consider multiple objectives: maximizing the distance to the joint limits while simultaneously minimizing joint velocities and accelerations. This optimization enables efficient and smooth following of a given Cartesian path, reducing required motion times.

Given a path consisting of waypoints in joint space, e. g., computed by an IK and/or path planning algorithm, a time-optimal velocity profile or path parameterization needs to be added to the path. These trajectories can then be executed by a robot joint controller to follow the path exactly and as quickly as possible. The *second* contribution is the development of such an online trajectory generation algorithm for paths that is suitable for real-time applications and redundant manipulators. Appropriate parts of time-parameterization methods, initially developed for offline planning, have been adopted, combined, and extended to provide a fast solution that respects given bounds on joint velocities and accelerations and has predictable computation times.

The *third* contribution is a remotely accessible lab with several industrial manipulators that is open to the public. Users can program the robots through a web interface or upload applications. The lab operates a safe and secure automated execution pipeline for

submitted source code to achieve a high usage rate of the robots while maintaining a low maintenance burden. Submissions are first evaluated in a simulation environment that mirrors the real robot cell setup and then executed by an available robot in the lab. As they are generally applicable to online and real-time motion planning for serial manipulators, the first two contributions are extensively evaluated separately. However, they are also essential building blocks for a motion planning framework in this remote lab. Motion inputs from the submitted source code need to be validated on the fly during motion generation to ensure the safety of the robots. Usage of the lab by different target groups is reported. All implemented contributions of this thesis are made available as free and open-source software.

Zusammenfassung

Echtzeitfähige kinematische Redundanzauflösung und Pfadparametrisierung für redundante Manipulatoren

Diese Arbeit konzentriert sich auf die Entwicklung echtzeitfähiger Lösungen für Probleme der Online-Roboterbewegungsplanung. Der Aufbau eines neuartigen Labors zur Anwendung dieser Algorithmen ist ebenfalls ein Bestandteil dieser Arbeit. Die Online-Bewegungsplanung ermöglicht es Robotern, prompt auf unerwartete Ereignisse zu reagieren und ihre Bewegungen während der Laufzeit anzupassen. Bei der echtzeitfähigen Bewegungsplanung werden zusätzliche Anforderungen an Sicherheit und Zuverlässigkeit berücksichtigt, indem sichergestellt wird, dass Bewegungen deterministisch innerhalb klar definierter zeitlicher Grenzen und ohne unerwartete Fehler geplant werden.

Zielposen oder vollständige Pfade, die der Endeffektor des Manipulators erreichen oder abfahren soll, werden mithilfe einer Inversen Kinematik (IK) in die erforderlichen Gelenkwinkel umgerechnet. Eine Zielpose für einen Roboterarm im kartesischen Raum wird üblicherweise durch sechs Parameter zur Darstellung von Position und Orientierung definiert, was das IK-Problem für anthropomorphe Manipulatoren mit sieben Freiheitsgraden unterbestimmt macht und eine Redundanzauflösung erfordert, um eine eindeutige Lösung zu finden. Der *erste* Beitrag dieser Arbeit ist ein echtzeitfähiges Redundanzauflösungsschema für solche Manipulatoren. Es wird eine Optimierungslösung in geschlossener Form entwickelt, die mehrere Ziele berücksichtigen kann: die Maximierung des Abstands zu den Gelenkgrenzen bei gleichzeitiger Minimierung der Gelenkgeschwindigkeiten und -beschleunigungen. Dies ermöglicht es, einem vorgegebenen kartesischen Pfad effizient und weniger ruckartig zu folgen und die erforderlichen Bewegungszeiten zu reduzieren.

Ein vorgegebener Pfad, bestehend aus Wegpunkten im Gelenkraum, z.B. berechnet durch einen IK- und/oder Pfadplanungsalgorithmus, erfordert die Ergänzung um ein zeitoptimales Geschwindigkeitsprofil oder eine Pfadparameterisierung. Diese berechneten Trajektorien können dann von einem Gelenkregler ausgeführt werden, um dem Pfad genau und so schnell wie möglich zu folgen. Der *zweite* Beitrag ist die Entwicklung eines solchen Online-Trajektoriengenerierungsalgorithmus für Pfade, passend für Echtzeitanwendungen und redundante Manipulatoren. Geeignete Teile von Zeitparameterisierungsmethoden, die ursprünglich für die Offline-Planung entwickelt wurden, wurden übernommen, kombiniert und erweitert, um eine schnelle Lösung zu bieten, die die vorgegebenen

Grenzen für Gelenkgeschwindigkeiten und -beschleunigungen einhält und vorhersehbare Berechnungszeiten aufweist.

Der *dritte* Beitrag ist ein aus der Ferne zugängliches Labor, das mit mehreren industriellen Manipulatoren ausgestattet ist und der Öffentlichkeit über das Internet zugänglich ist. Benutzer können die Roboter über eine Webschnittstelle programmieren oder Anwendungen hochladen. Das Labor wird von einer sicheren und automatisierten Ausführungspipeline für eingereichten Quellcode betrieben, die darauf abzielt, eine hohe Auslastung der Roboter zu erreichen, bei gleichzeitig geringem manuellem Überwachungsaufwand. Einsendungen werden zunächst in einer Simulationsumgebung evaluiert, die die reale Roboterzelle abbildet, und anschließend von einem verfügbaren Roboter im Labor ausgeführt. Da die Ansätze generell auf die Online- und Echtzeitbewegungsplanung für serielle Manipulatoren anwendbar sind, werden die ersten beiden Beiträge separat umfassend evaluiert, dienen jedoch auch als wichtige Bausteine des Bewegungsplanungsframeworks in diesem Fernlabor. Die Bewegungseingaben aus dem eingereichten Quellcode müssen während der Bewegungserzeugung validiert werden, um die Sicherheit der Roboter zu gewährleisten. Es wird auch über die Nutzung des Labors durch verschiedene Zielgruppen berichtet. Alle implementierten Beiträge dieser Arbeit stehen als freie und Open-Source-Software zur Verfügung.

Contents

CHAPTER 1

Introduction

Online motion planning, in contrast to offline planning, allows robots to react instantaneously to unexpected events, adapt their motions, or plan new motions on the fly. Real-time-capable motion planning additionally requires that movements are planned fast, within strict bounds on computation time without unexpected failure, and with deterministic results. Use cases range from sensor-based motion planning, safety-critical human-robot collaboration, to providing roboticists with flexible and reliable motion planning frameworks to develop new applications. The progress in this research field enables robots to be employed not only in fenced-off industrial environments with predefined and recurring motion sequences, but also to sense dynamically and reason about tasks in less structured environments and then safely plan and execute the required motions. Several contributions of the thesis at hand are aimed at this research field.

Concerning the type of robot, this thesis focuses on the common manipulator type of robot, which is a robotic arm fixed to a table or other base frame. The robot consists of several revolute joints connected via arm segments or links. Most manipulators used have six degrees of freedom (DoF). However, anthropomorphic manipulators are also available that mimic the kinematics of a human arm and have seven DoFs. Additionally, the common task for a robot is examined, which involves moving to specific goal poses in space, typically determined by human input or a task and path planner. Goal positions are usually given in Cartesian space for the robot's end-effector, e. g., a gripper, or any other tool. Alternatively, goal poses can be specified directly for the individual joints. Kinematics maps between the Cartesian space and the joint space. Inverse kinematics computes the joint angles corresponding to a given goal pose. Inverse kinematics is thus a fundamental operation, as robots are controlled in joint space, but target configurations are commonly given in Cartesian space. In addition to point-to-point movements, a path can also be defined with intermediate waypoints that the robot should traverse.

After the goal poses or the path are determined and optionally converted to joint space by the inverse kinematics, a trajectory can be generated, which adds timing information. Here, the goal is to find a velocity profile that allows the robot to move as fast as possible along the path within its kinematic constraints. In this part of the motion planning pipeline, the problem of time-optimal trajectory generation is addressed in this thesis. After this information is available, a joint position controller can sample the trajectory and control the individual joints to follow the given velocity profile, which is the actual execution part.

Currently, state-of-the-art low-level robot joint position controllers already operate in fixed control cycles like 1 ms. Current research in real-time-capable motion planning algorithms aims to run more complex and higher-level parts of a planning pipeline under the same or almost the same constraints. This will enable more powerful robot motion interfaces that, e. g., can reliably produce globally time-optimized movements from a start to a goal pose in Cartesian or task space with intermediate waypoints, even for manipulators with more than six DoFs. This thesis presents two contributions to the usual manipulator motion planning pipeline: inverse kinematics and trajectory generation. The contributions provide real-time-capable algorithms for these parts and address the research question how more capabilities from offline planning approaches can be transferred to real-time path following. The two contributions also facilitate the realization of a novel application, a remotely accessible robotics lab, which is the third contribution. In the remainder of this chapter, the contributions are introduced, then described in more detail in Section 1.1, and finally, Section 1.2 presents the outline of the thesis.

The *first* contribution of this thesis is a novel approach to redundancy resolution. A desired end-effector pose for a robotic manipulator in task space is usually specified by six parameters representing the position and orientation, which renders the inverse kinematics problem under-constrained for anthropomorphic manipulators with seven DoFs. Real-time-capable, straightforward algorithms exist for non-redundant manipulators to map a target pose to target joint positions. However, in the case of seven-DoF manipulators, the redundant DoF leads to an infinite number of solutions, which form the null-space or self-motion manifold. As part of the solution of the inverse kinematics problem, redundancy resolution is needed to select a suitable solution from the null-space. In this thesis, a novel multi-objective optimization approach to redundancy resolution is introduced that is real-time-capable. In contrast to existing real-time-capable redundancy solutions, our approach can smoothly and efficiently follow a given continuous Cartesian path.

Collision-free motion planning in robotics is usually split up into two steps:

1. Find a path using a path planning algorithm. Sampling-based path planners are commonly used. These define paths using a discrete set of waypoints, either in Cartesian space (task space) or in joint space.

2. Add a time parameterization to the joint space path using a trajectory generator. These trajectories can then be executed by a robot joint controller. The trajectory generator needs to be able to take the waypoints of the first step as input and produce a velocity profile that follows the waypoints exactly. Most current real-time and online trajectory generators (OTGs) are only able to generate trajectories between a given start and goal state without the possibility of specifying intermediate waypoints for a time-optimal velocity profile across all waypoints. This restriction means that they cannot be used to follow a given path; instead, they are intended for the generation of time-optimal point-to-point trajectories.

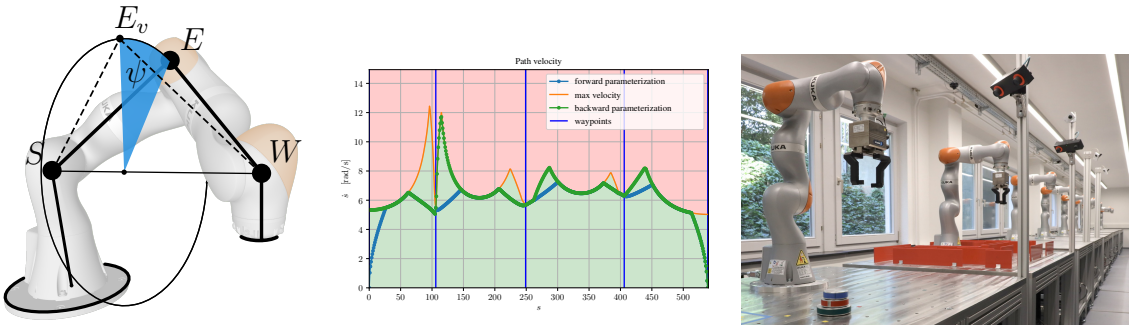
The *second* contribution of this thesis is a novel method for an OTG that can follow a given path consisting of waypoints and is suitable for serial manipulators with six DoFs or more.

The *third* contribution is the application of the developed methods as part of a remotely accessible lab for robotics researchers with up to ten manipulators available. The lab is operated by a safe and secure automated execution pipeline for submitted source code.

Although generally applicable to online and real-time motion planning for serial manipulators, the first two contributions are essential building blocks for a motion planning framework in this remote lab. Motion inputs from the submitted source code need to be validated on the fly during motion generation to ensure the safety of the robots. The lab hosts several projects, and users can develop their own applications or solutions to the given exercises using a simulation environment. They can then access the lab over the Internet and submit their source code, which is executed in the lab by an available robot.

1.1 Contributions

The three stated contributions of this work are summarized in Figure 1.1 and will be introduced in greater detail in the following.



(a) arm angle ψ (b) trajectory generation (c) Robot Learning Lab

Figure 1.1: The three main contributions of this thesis visualized. (a) shows the arm angle that is optimized for the redundancy resolution as part of the real-time inverse kinematics (adapted from Wiedmeyer et al. (2021)). (b) The path parameterization algorithm generates time-optimal path velocity profiles for following a set of waypoints. (c) And a view of the remotely accessible robotics testbed.

1.1.1 Multi-Objective Kinematic Redundancy Resolution

Employing a closed-form inverse kinematics solution for motion planning offers several benefits over conventional numerical methods, most notably significantly reduced computation times and improved suitability for real-time use. Considerable progress has been made in developing analytic inverse kinematics solutions for seven-DoF manipulators without joint offsets. Redundancy resolutions have been used to move individual joint angles away from their limits. Nevertheless, these closed-form approaches do not take any additional objectives into account.

In this thesis, a multi-objective real-time optimization-based approach is developed that can minimize joint velocities and accelerations, while still avoiding joint limits (Wiedmeyer et al., 2021). The method is explained in detail in Chapter 3 and only a summary is given here. The derived closed-form solution of the optimization problem only requires a single intuitive tuning parameter. A C++ implementation of the complete kinematics algorithm was created that covers the entire solution space. The evaluation of the null-space motion generation shows that our approach achieves faster motion times and smoother

arm motions along Cartesian paths in comparison to the state-of-the-art approach. Experimental results with a KUKA iiwa robot show that the complete solution algorithm is suitable for mapping given Cartesian waypoints to joint space configurations in real-time.

1.1.2 Time-Optimal Trajectory Generation for Path-Following

Fast offline methods have been proposed to solve the time-optimal trajectory generation problem. These methods are needed to follow a given joint space path with an arbitrary length, subject to kinematic constraints for an arbitrary number of DoFs. In contrast, real-time and online trajectory generators focus on point-to-point movements or short path segments. Our contribution bridges this gap and proposes a *real-time-capable time-optimal path parameterization* (*RT-TOPP*) algorithm. Chapter 4 details the contribution. In summary, the algorithm can handle a discrete set of path waypoints and reliably generate trajectories within hard constraints on the maximum computation time and the joint velocity and acceleration. This advancement will enable such a trajectory generator to run in a robot's control cycle, allowing for instant reactions to path changes or new path processing requests. The waypoints given in the Cartesian space (task space) can be pre-processed by the aforementioned contribution, the inverse kinematics algorithm (see Section 1.1.1). Parts of the TOPP methods, which were initially developed for offline planning, have been adopted, combined, and extended for a novel online and real-time concept. The developed algorithm was evaluated on a large set of randomly generated paths .

1.1.3 Automated Remote Developer Access to Robot Cells

This part of the dissertation details the development of the *KUKA Robot Learning Lab at KIT* – a remotely accessible robotics testbed open to the public. Chapter 5 describes the architecture and operation of the laboratory. The purpose of the laboratory is to make state-of-the-art industrial lightweight robots more accessible for both education and research. In this case, up to ten robot cells are available in parallel. Such expensive hardware is usually not available to students or less privileged researchers to conduct experiments. The design and operation of the Robot Learning Lab are described, and the challenges of making experimental robot cells remotely accessible are discussed. In particular, it is essential to guarantee safety and security while still allowing users as much freedom as possible in developing programs that control the robots. A novel, fully automated and efficient processing pipeline for experiments prepares the lab for a large number of users, has a low maintenance burden, and allows a high usage rate of the robots (Wiedmeyer et al., 2019). Easy-to-use web-based development environments are integrated, allowing the lab to be used with just a web browser. Web API access is also available for advanced users to upload their source code.

All implemented contributions of this thesis are made available as free and open-source software.

1.2 Structure of the Thesis

In the following, the remainder of this thesis is outlined by summarizing the content of each subsequent chapter.

Chapter 2 presents the current state of the art and summarizes previous work related to the main contributions of this thesis. First, the chapter discusses relevant preliminaries on motion planning. Subsequently, inverse kinematics approaches for seven-DoF robot manipulators are discussed with a focus on closed-form solutions and redundancy resolution schemes. The second part categorizes and compares trajectory generation techniques. Particular attention is paid to methods suitable for the development of an online and real-time approach to path following. Finally, a survey of remote robotics testbeds and related approaches is conducted to give an overview of the current state in the field of the third contribution of this thesis, the remotely accessible robotics lab.

The Chapters 3-5 describe the three core contributions of this thesis. Each chapter concludes with a summary.

Chapter 3 introduces a multi-objective optimization approach to redundancy resolution that integrates with a fully analytical inverse kinematics for seven-DoF manipulators and selects a suitable solution for the redundant DoF. The combined cost function and individual objective functions are detailed, including the closed-form solution to the optimization. The implementation and experiments are described, which examine the real-time capability of the algorithm and arm motions generated for Cartesian path following tasks.

Chapter 4 presents a trajectory generation algorithm for real-time path following. After an overview, the individual components of the method are described that are needed to find a continuous path representation mapped to a single dimension, which allows finding the maximum available path velocity in every path segment. The final parameterization algorithm is then detailed, which scans the path twice to find a time-optimal velocity profile. Experiments validate the implementation and runtime performance, and the algorithm output is inspected.

Chapter 5 introduces the Robot Learning Lab, a KIT facility that allows software developers to access the robots in the lab over the Internet. First, general design considerations are highlighted. The lab infrastructure, with an emphasis on automated submission processing and safety and security, is then described. It is demonstrated how users can interact with the lab, including moving the robots, and an overview and statistics on the lab's usage to date are provided.

Chapter 6 concludes the thesis by summarizing the most important contributions and results of the work. Limitations and open challenges are discussed. Further possible applications of the first two motion-planning contributions are outlined, and final remarks are given.

CHAPTER 2

State of the Art and Related Work

This chapter provides an overview of the current state of the art related to the topic of this thesis. The related work specifically relevant to the three main contributions of the thesis is discussed. For the first two contributions (inverse kinematics and trajectory generation), the literature survey targets methods that are promising starting points for a real-time-capable solution. Section 2.1 focuses on the inverse kinematics problem for 7-DoF manipulators and closed-form or partially analytic solutions to the problem. Section 2.2 discusses the challenge of time-optimal path parameterization, the different optimization approaches for fast offline solutions, and limitations of current online trajectory generation algorithms. Finally, Section 2.3 surveys remotely accessible robotics testbeds and their architecture, mainly to investigate if and how they provide safe, secure, and automated remote robot programming and motion planning capabilities. These are relevant criteria for the third contribution of this thesis, the remotely accessible robotics lab.

2.1 Closed-Form Inverse Kinematics for Redundant Manipulators

A target pose in task space is typically specified in six dimensions, making the Inverse Kinematics (IK) problem under-constrained for manipulators with seven degrees of freedom (DoF). This extra DoF introduces redundancy and results in infinitely many possible solutions, referred to as the null-space or self-motion manifold. Consequently, a redundancy resolution step is required within the IK algorithm to choose an appropriate solution from this null-space.

This section first discusses different approaches to IK, then highlights the advantages of a closed-form solution for real-time applications compared to differential methods, and finally describes the related work and advances to develop closed-form IK solutions for seven-DoF manipulators with integrated redundancy resolution.

Differential IK approaches, which rely on the pseudoinverse or quadratic programming, are usually employed to solve the IK problem. These approaches apply well to various kinematic structures and different levels of redundancy. They can also impose constraints on joint position, velocity, and acceleration (Flacco et al., 2012, 2015). Collision avoidance can also be considered by relaxing other kinematic constraints (Glass et al., 1995;

De Luca and Ferrajoli, 2008; Khatib et al., 2020), respectively, by ensuring minor deviations from a given nominal trajectory (Faroni et al., 2020). Classical approaches to IK and redundancy resolution are further described in the textbook by Siciliano et al. (2008), including generalized IK solvers using numerical methods. Ames et al. (2022) also explored the application of deep learning to generate IK solutions for redundant robots called IKFlow. The accuracy of IK solutions computed by IKFlow cannot compete with differential solvers or even closed-form solutions, but a large set of possible sample solutions from the redundancy space can be generated in a few milliseconds. IKFlow is also not able to provide an optimized redundancy resolution, nor to determine the complete solution space. It provides a set of possible IK solutions that are in the redundancy space.

Closed-form IK solutions for many non-redundant kinematic chains can nowadays be auto-generated by software solvers. Solvers like the open-source IKFast (Diankov, 2010) generate code from a robot description file, and the generated source code can be used as an IK solver for the specified kinematic chain. Ostermeier et al. (2025) recently improved the code generation speed of such an IK solver so that numerically stable closed-form solutions for many non-redundant manipulator types with six DoFs or less are derived in less than 50 µs. However, such solvers are not able to generate IK solutions for seven DoFs or more without locking joints to remove redundant DoFs. This means for a seven-DoF robot that the user has to select one joint, which is either kept locked or for which the user has to select a desired angle before IK computation. Therefore, the solvers cannot utilize redundancy and do not implement a scheme for resolving redundancy.

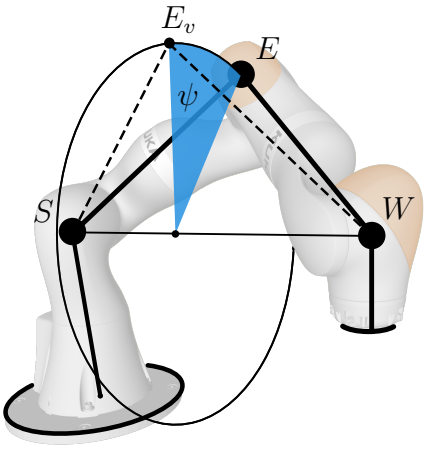


Figure 2.1: Redundancy is represented by the arm angle ψ, defined between the actual elbow E and the virtual elbow E_v in the reference plane, using the line from the shoulder S to the wrist W as the rotation axis. The figure is adapted from Wiedmeyer et al. (2021).

In contrast, closed-form IK solutions only exist for a few redundant robot types, like the popular non-offset 7-DoF anthropomorphic manipulator structure. Examples are the KUKA LBR and iiwa generation of robots. This robot type has no offsets in the shoulder, elbow, and wrist joints (see Figure 2.1 for an example structure). The individual axes on the shoulder and wrist intersect at one point. In a fully stretched vertical pose of the robot, a single line connects the center points of all joints.

There also exist robots with offsets in one, multiple, or all three areas of the shoulder, wrist, and elbow. An example is the Franka Emika robot. Ongoing research is focused

on deriving fully closed-form solutions for offset-type structures. The challenge lies especially in developing an IK algorithm that covers the whole solution space for arbitrary target poses in the workspace. For example, a recent work by Shao et al. (2024) proposes a solution that is suitable for incremental changes in the arm angle given by a collision avoidance metric and current joint states. However, the algorithm does not appear to be suitable for determining the entire solution space for a given target pose. It is thus not able to achieve joint position limit avoidance using a redundancy resolution scheme. The literature review in this section focuses on the non-offset type of seven-DoF manipulators, as the contribution of this thesis targets this type. The existing algorithms for this special type are more capable, especially concerning redundancy resolution, compared to those for the general type with potential offsets in the joints.

Advantages of Closed-Form Approaches

Analytic approaches for the non-offset manipulator type generally parametrize the elbow redundancy with an arm angle ψ that defines the relative rotation between the shoulder-elbow-wrist plane and a reference plane (Dahm and Joublin, 1997), as seen in Figure 2.1. The redundancy forms a circle on which the elbow can move, while wrist and shoulder positions do not change (assuming a fixed end-effector pose). Key benefits of closed-form solutions for robots with a single degree of redundancy, in comparison to traditional differential IK approaches, are as follows:

- An exact solution at the position level is returned in one call.
- The complete null-space given from joint position limits can be determined and explored using different optimization criteria.
- Computation times are much faster, and real-time requirements can be met more easily. This advantage is especially beneficial for online motion planning.
- A desired arm angle can be directly specified if needed. In addition, continuous switching between this particular solution and a general solution with an optimized arm angle is possible.
- Redundancy resolution approaches, e. g., with the objective of avoiding joint limits, can be intuitively specified in the null-space using the arm angle. They can also be fully analytic. It can be ensured that the optimized arm angle is reachable and within joint limits.
- Drift effects on the self-motion manifold do not need to be addressed, and cyclic motions can be executed in a deterministic fashion.

These advantages make an existing closed-form solution a well-suited starting point for the development of a general-purpose real-time IK algorithm for seven-DoF manipulators.

Development of a General-Purpose and Complete Solution

Closed-form solutions have been developed for specific use cases that do not require a complete solution that covers the full redundancy space. Most prominently, several existing analytic IK solutions can generate human-like motions (Asfour and Dillmann,

2003; Tondu, 2006; Wang and Artemiadis, 2013). The seven-DoF analytical IK approach was also used as part of an iterative redundancy resolution approach to replicate natural human arm postures in a more complex human arm musculoskeletal model (Li et al., 2022). These approaches are particularly interesting for human-robot interaction, as a more human-like behavior allows humans to anticipate the robot's movements better. This behavior can also make humans feel more comfortable in proximity to the robot. And these solutions can help transfer captured or learned human motions to the robot arm. However, these algorithms are not suitable for a general-purpose motion planning framework, as they do not fully take advantage of the available agility of the robot. Instead, a closed-form solution is needed that can operate within the full actual joint limits of the manipulator and that can handle all possible global configurations. See Figure 2.2 for a visual explanation of global configurations and Section 2.1.1 for a possible formal description.

Shimizu et al. (2008) formulated a closed-form IK solution for 7-DoF manipulators and introduced a method to map joint limits to the redundancy space. Joint limits are avoided by applying an analytical redundancy resolution scheme that determines the globally optimal arm angle, maximizing the separation from the joint limits. They define the arm angle using a reference plane that represents a non-redundant image of the manipulator with the third joint angle $\theta_3 = 0$. Oh et al. (2017) extend the approach to include self-collision avoidance. However, the redundancy resolution approach cannot be used to follow a continuous Cartesian path because it does not ensure that there are no jumps in the arm angle. Jumps would lead to jerky movements of the end-effector due to the decelerations required for the elbow to first move to its new position before the end-effector movement can continue. It is also possible that jumps result in reconfiguration motions, where the robot is forced to interrupt the Cartesian path following task to move to a new configuration, from which it is then able to continue following the path.

IK algorithms that control the global configuration are presented by Kuhlemann et al. (2016) and Li et al. (2018). Huber and Wollherr (2019a) formulate a different parameterization approach for the IK solution, which allows for efficient investigation of the task space manipulability for a large set of poses in parallel. Manipulability is an essential and frequently used optimization criterion. It measures the dexterity of the robot configuration at the target pose, and its optimization increases the agility and likelihood that the robot can move to another pose nearby.

Xu et al. (2016) use a dual arm angle parameterization to avoid algorithmic singularities. Algorithmic singularities can arise in arm angle parameterizations when the reference plane for the arm angle becomes undefined. This type of singularity occurs in configurations where the arbitrary vector and the shoulder-wrist vector used for the definition of the reference plane become collinear. Similar to kinematic singularities, large arm angle movements or other undesirable behavior can occur, including algorithm failure. Xu et al. (2016) resolve this issue by using two arbitrary vectors and by switching to the one where the reference plane can be constructed in the current pose. A later work by Elias and Wen (2024) further investigates algorithmic singularities for different seven-DoF robotic manipulators and their arm angle parameterizations. They propose a new type of parameterization that results in a singularity only along a half-line instead of a full line connecting the shoulder and wrist as in previous publications. This allows, e. g., to move the singularity below the robot's base. Moving it there makes it less likely that the algorithmic singularity is reached, but also not completely unavoidable, depending on the workspace and setup of the robot.

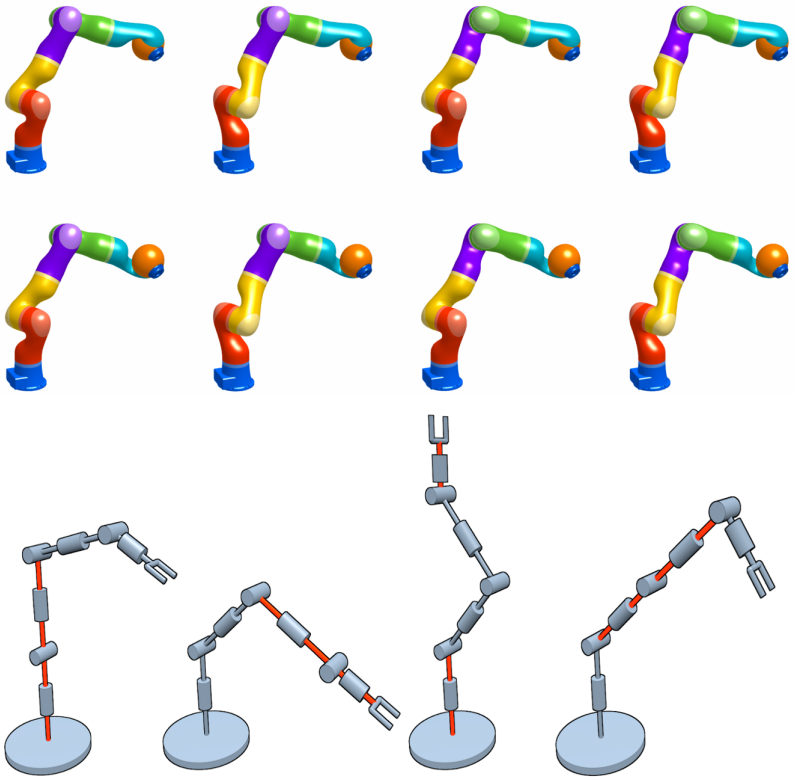


Figure 2.2: Examples for global configurations and kinematic singularities. The first two rows show eight different possible global configurations of a KUKA LBR IV reaching the same flange pose with a fixed arm angle. Links are colored to highlight the different combinations of joint orientations. The first three robot postures in the bottom row depict singularities where the redundancy is lost. The fourth posture (bottom right) is the type of singularity where, additionally, the end-effector motion becomes restricted. Aligned joint axes are colored in red. The figure is adapted from Kunze (2016), used with permission.

Although these works address essential issues of previous IK solutions like global configuration control and algorithmic singularities, challenges such as joint limit avoidance and kinematic singularities are not examined. Figure 2.2 shows the different types of kinematic singularities that can arise with the seven-DoF manipulator structure. These singularities pose challenges as the arm angle becomes undefined, or even further DoFs are lost, and the end-effector movement is restricted.

Finally, Faria et al. (2018) build on the previous work of Shimizu et al. (2008) and Kuhlemann et al. (2016), and describe an approach that can solve the global configuration manifold and considers joint limits and kinematic singularities. Algorithmic singularities are resolved as well. They also propose an analytic redundancy resolution approach that allows for continuous adaptation of the arm angle based on the previous arm angle to avoid joint limits and kinematic singularities. In summary, their IK solution effectively combines nearly all the beneficial properties of other related works while introducing novel and useful redundancy resolution features. Therefore, their approach to position-based IK with redundancy resolution is used as a starting point and a baseline for the IK algorithm presented in this thesis and its improved redundancy resolution.

Although the literature review above demonstrates steady progress in advancing closed-form IK methods, there is still no closed-form IK solution for 7-DoF manipulators that offers a fully analytical redundancy resolution while simultaneously accounting for multiple objectives beyond proximity to the previous arm configuration, joint limits, and singularity avoidance. In particular, the minimization of joint angle derivatives is not addressed, which would be beneficial to reduce motion times and increase the smoothness of arm motions. Instead, approaches using numerical methods have been proposed for this (Lee and Buss, 2006; Guigue et al., 2010; Flacco and Luca, 2015; Al Khudir and De Luca, 2018), or the combination of a closed-form IK parameterization with a numerical non-analytical optimization for redundancy resolution (Kunze, 2016; Zeng et al., 2018). Similarly, particle swarm optimization has been proposed by Tian et al. (2021), which also raises the issue of finding or providing suitable tuning and weighting parameter values for the optimization. Using a numerical gradient-based or related solver makes it easier to incorporate cost functions for different objectives into the optimization. Still, it has limited suitability for real-time applications due to unpredictable worst-case computation times for convergence and potential nondeterministic behavior.

A more recent work by Vu et al. (2023) combines a neural network with the parameterized analytical formulation of the IK with arm angle and global configuration. The neural network predicts the optimal arm angle and global configurations using two objectives: manipulability at the target pose and closeness to the previous joint configuration. The algorithm outperforms a classical general IK solver using damped least-squares in terms of computation time, success rate, and optimality of the solution. However, only average and not worst-case computation times are reported, which would be necessary to assess the real-time capability. Additionally, only point-to-point movements are investigated, and the experiments lack the performance evaluation along continuous Cartesian paths.

Furthermore, evaluations of the above-described IK solutions have almost exclusively been provided only in simulations and not with a real robot. With the only recent exception of Dou et al. (2022), the discussed evaluations of closed-form algorithms also lack an investigation of the null-space and evaluation of the arm angle motion therein along a trajectory. Software implementations of these IK algorithms have not been published. These shortcomings are addressed in the thesis at hand.

In the remainder of this state-of-the-art section, the IK algorithm from Faria et al. (2018) is summarized as it is used as a base for our algorithm. Three properties of the algorithm, which are also relevant for our contribution, are detailed: global configuration control to uniquely identify the branches of possible configurations, the mapping of joint limits and singularities to the null-space, and the proposed redundancy resolution scheme by the authors.

2.1.1 Global Configuration

For a given end-effector pose and arm angle, up to eight different global configurations ($2^3 = 8$) are possible, which need to be uniquely identified to provide a consistent IK solution (see also Figure 2.2). The three joints at the shoulder, elbow, and wrist are influencing the global configuration. For every one of these joints, two values and thus a binary state are possible (given by the sign of the joint angle). Some end-effector poses can be reached by different combinations of joint angle signs, e. g., $-30°$ at the shoulder and $30°$ at the wrist, and vice versa, $30°$ at the shoulder and $-30°$ at the wrist.

To describe the specific branch of IK solutions within the global configuration manifold, the global configuration parameter GC_k is defined. This parameter encodes the sign of the joint angles at the shoulder ($k = 2$), elbow ($k = 4$), and wrist ($k = 6$). It is expressed as

$$GC_k = \begin{cases} 1, & \text{if } \theta_i \geq 0 \\ -1, & \text{if } \theta_i < 0 \end{cases}, \ \forall i \in \{2, 4, 6\}\,. \tag{2.1}$$

The bi-directional mapping between joint angles and arm angle can then be expressed in terms of GC_k. For a non-redundant six-DoF manipulator or a seven-DoF manipulator with a fixed arm angle, an IK solution can then be uniquely identified. For the redundant case, this nomenclature helps first to determine the global configuration and then to correctly compute the available null-space in the respective global configuration branch.

2.1.2 Joint Limits and Singularities in the Null-Space

Due to projection of the joint limits and singularities of all joints to the null-space angle ψ, blocked intervals of the arm angle can be identified. The derivative of the joint angle, $\mathrm{d}\theta_i(\psi)/\mathrm{d}\psi$, is analyzed to detect singular configurations and to determine on which side of a joint limit a blocked range of the arm angle lies. These blocked intervals are not accessible because entering them would cause at least one joint limit to be exceeded or bring the robot into a singular configuration.

Once the blocked intervals have been identified individually for each joint i, there remain n_j feasible intervals $\Psi_{i,j}$ for the arm angle on the redundancy circle. The global feasible intervals Ψ_{all} are determined as

$$\Psi_{all} = \bigcap_{i=1}^{7} \Psi_i, \ \Psi_i = \bigcup_{j=1}^{n_j} \Psi_{i,j}\,. \tag{2.2}$$

For pivot joints ($i = 1, 3, 5, 7$), $\theta_i(\psi)$ exhibits a discontinuity at a singular arm angle ψ_{sing}. For these joints, a blocked interval $[\psi_{sing} - \delta, \ \psi_{sing} + \delta]$ with a tolerance margin δ must be created to avoid the indeterminate arm angle. The elbow joint ($i = 4$) is excluded from this analysis since it is the only joint that does not depend on the arm angle.

2.1.3 Redundancy Resolution

To resolve the redundancy, an arm angle from one of the global feasible intervals Ψ_{all} has to be selected. For smooth trajectory planning, there should be no jump in the arm angle. This is accomplished by selecting the new arm angle ψ_t depending on the preceding arm angle ψ_{t-1}. The new arm angle should be in the same feasible interval $\Psi_{all,m}$ as the previous arm angle ($\psi_{t-1} \in \ \Psi_{all,m}, \ \Psi_{all,m} = \left[\psi_m^l, \psi_m^u\right]$).

The new arm angle is obtained with

$$\psi_t = \psi_{t-1} + \mathrm{K}\left(\frac{\psi_m^u - \psi_m^l}{2}\right) \left[\mathrm{e}^{-\alpha\left(\frac{\psi_{t-1} - \psi_m^l}{\psi_m^u - \psi_m^l}\right)} - \mathrm{e}^{-\alpha\left(\frac{\psi_m^u - \psi_{t-1}}{\psi_m^u - \psi_m^l}\right)} \right]. \tag{2.3}$$

where $\mathrm{K} \in [0, 1]$ and $\alpha \in \mathbb{R}^+$. This metric pushes the arm angle to the middle of the feasible interval and away from the borders. K determines the strength of the repulsion from the borders, while α specifies how far from the boundaries this repulsion starts to act. By appropriately adjusting these two parameters, one can obtain smooth transitions between different arm angles. Even when the IK is repeatedly queried with the same goal pose, the arm angle is continuously adapted until it reaches the middle of the current feasible interval.

Dou et al. (2022) more recently proposed a similar analytical resolution formula with two tuning parameters, originally developed for a seven-DoF structure different from common manipulators and closer to the human arm, specifically to the human wrist movement mode. They did not use exponential functions to adapt the proximity to the previous arm angle. Using our notation, their arm angle is obtained by

$$\psi_t = \psi_{t-1} + \mathrm{K}\left(\frac{\psi_m^u - \psi_m^l}{2}\right)\left[\left(\frac{\psi_m^u - \psi_{t-1}}{\psi_m^u - \psi_m^l}\right)^\alpha - \left(\frac{\psi_{t-1} - \psi_m^l}{\psi_m^u - \psi_m^l}\right)^\alpha\right].$$

Unfortunately, they did not compare their redundancy resolution with that of Faria et al. (2018), and their evaluation lacks a thorough evaluation of the computation times for their complete IK algorithm. In this thesis, the full IK algorithm from Faria et al. (2018) is used as a base. It was developed for the same manipulator structure that is targeted in this thesis. Consequently, their redundancy resolution scheme is also an appropriate baseline.

2.1.4 Summary

There are two main classes of approaches to inverse kinematics and redundancy resolution for redundant manipulators: differential and closed-form. Differential IK solutions (e. g., using the pseudoinverse) generalize well to different kinematic structures and can enforce different kinematic joint space constraints or incorporate collision avoidance. In contrast, current closed-form solutions are only applicable to a specific manipulator type with a fixed number of DoFs, like the non-offset seven-DoF manipulator type targeted in this thesis, and the incorporation of differential constraints or collision avoidance is usually not given. However, a closed-form approach generally has much faster computation times and is well-suited as a starting point for a real-time IK solution. Exact and deterministic results can be directly obtained at the position level. The non-iterative nature of the method makes it possible to determine worst-case computation times without loss of precision. The redundancy can be parameterized by an arm angle that describes the elbow rotation on a circle (assuming a fixed end-effector).

Considerable progress has been made in developing a closed-form IK solution for the seven-DoF manipulator type without offsets in the joints. Algorithmic and kinematic singularities can be avoided, and the global configuration manifold can be uniquely identified. Joint limits are projected to the redundancy or null space. A redundancy resolution can be directly determined in this available null space by selecting a suitable arm angle. However, existing redundancy resolution methods can only move away from joint limits or singularities, and no further objectives are considered. This limitation can lead to jerky movements and inefficient trajectories when following Cartesian paths. Additional objectives known from differential methods, such as optimizing joint velocities or accelerations, are not considered. However, this would be beneficial for generating faster

and smoother movements. Furthermore, the movement of arm angle in the null space along such paths needs to be studied to evaluate and compare redundancy resolution approaches. Existing methods also require tuning several redundancy resolution parameters for a given scenario. Still, such intermediate offline parameter optimization is not possible during continuous online planning, and the redundancy resolution should generalize well to different applications.

This summary concludes the related work section on inverse kinematics and the description of the method by Faria et al. (2018). Their method, including their redundancy resolution in Eqn. (2.3), serves as a baseline for the IK algorithm developed in this thesis, as it is the most advanced approach in the related work. An open-source implementation of the complete algorithm was also created for this thesis. In the Experimental Section 3.3, a direct comparison is conducted.

2.2 Time-Optimal Path Parameterization

Collision-free motion planning is usually split up into (i) finding a path using a path planning algorithm and then (ii) adding a time parameterization, respectively a velocity profile, to the joint space path using a trajectory generator. A robot controller can then execute these trajectories given by the path parameterization. The second contribution of this thesis is the development of a real-time-capable trajectory generator that can follow a given path. This section provides an overview of the relevant related work in this area.

Sampling-based path planners are commonly used. These define paths using a discrete set of waypoints, either in Cartesian space (task space) or in joint space. Waypoints in Cartesian space can be transformed to joint space by inverse kinematics algorithms. The trajectory generator thus needs to be able to take these waypoints as input and produce a trajectory that follows the waypoints exactly. Most current real-time and online trajectory generators (OTGs) are only able to generate trajectories between a given start and goal state without the possibility of specifying intermediate waypoints. This restriction means that they cannot be used to follow a given path; instead, they are intended for the generation of point-to-point trajectories. The Reflexxes motion libraries (Kröger, 2011) and Ruckig (Berscheid and Kröger, 2021) are popular examples of this type of OTG.

In contrast, offline trajectory generation methods that can follow a given path have been developed. However, while some of the methods are computationally efficient, they are not usable for real-time applications with strict limits on the maximum computation time and additional reliability requirements. Additionally, it is usually assumed that the robot is at rest at the start and goal states, and the online trajectory planning case, where the robot is moving and has non-zero velocity in the current configuration, is not considered. Nevertheless, these offline methods are capable of respecting given kinematic and dynamic constraints. When generating a trajectory, different optimization criteria can be applied, such as minimizing the energy of the system or minimizing the execution time of the trajectory. The latter is selected for this work as it is usually the most important or most requested optimization criterion. The class of methods with this objective is referred to as *Time-Optimal Path Parameterization* (TOPP).

The real-time concept proposed in this thesis adopts techniques from offline TOPP methods. For this reason, the following literature survey details the state of the art in this

area, including jerk-constrained solutions. The TOPP problem received considerable attention from the research community, and initial breakthroughs were already achieved in the 1980s. Most notably, Bobrow et al. (1985) introduced a bang-bang algorithm as the first true solution to the TOPP problem for a general serial manipulator. A key contribution was to express the equations of motion in terms of a single path variable s and its time derivatives $\dot{s}, \ddot{s}$. s denotes the position along the path and $\dot{s}, \ddot{s}$ path velocity and acceleration. This formulation enables the projection of all N-DoF constraints onto a single path variable, and together with its derivatives, the time-optimal trajectory can be found in the resulting phase plane. This phase space has two dimensions (position and velocity) with first- and second-order constraints. Acceleration or torque is then the control variable or system input. Bobrow et al. (1985) defined a Maximum Velocity Curve (MVC) in the $(s, \dot{s})$ phase plane according to the manipulator's torque limits. The maximum velocity that still allows the robot to stay on the path at a given position is found using this curve. They also used this curve to find switching points at which decelerations transition to accelerations. Shin and McKay (1985) found the minimum-time solution at a desired resolution by using a grid in the phase plane.

The phase space becomes three-dimensional (position, velocity, and acceleration) with additional third-order constraints. Jerk or torque rate is the input here. The MVC becomes a surface in this space, as the maximum velocity also depends on the acceleration in this case. This is further explained in Section 2.2.1 when jerk-limited approaches are discussed. The remainder of this section first provides an overview of fast offline methods, i. e., methods that stand out by their computational efficiency and low computation times in their evaluations. It is also discussed whether these algorithms could be suitable for online planning and how close these methods are to a real-time application. Most of these methods fall into two main categories that describe two different principles in solving the optimization problem, which are further detailed. Finally, in Section 2.2.2, existing online and real-time algorithms and their suitability for path parameterization are discussed.

2.2.1 Fast Offline Methods

Over the last decades, research in the area of TOPP has focused on increasing the robustness of parameterization algorithms, reducing complexity, making them easier to implement, and especially lowering computation times. In the following survey, the following evaluation criteria are applied, which are essential to assess whether the given algorithm could be suitable as a starting point for an online and real-time approach:

- Is the algorithm fast? How does the computation time increase with path length or path discretization size? Are there hints on worst-case computation times?
- How complex is the algorithm or the individual parts of it? Are there any components where the expected computation time is hard to estimate, like a search algorithm?
- Is the time-optimality proven? How close is the algorithm output to the theoretically time-optimal solution?
- Was the algorithm well tested on a large and diverse set of paths? What is the success rate for the parameterization? Under what circumstances does the algorithm fail?

- Is it possible to handle non-zero start and goal configuration velocities with the algorithm, or can it be adapted for this use case? Can it handle a discrete set of waypoints as input?
- Does it have tuning parameters or other parameters that can make it harder to apply the algorithm to arbitrary paths without having to change parameters? Changing or tuning parameters would not be possible in an online planning scenario.

Algorithms are mainly evaluated using these questions, and prior work is highlighted where multiple criteria seem to be well fulfilled.

There are two prominent families of offline TOPP algorithms: methods based on Numerical Integration (NI) or based on Convex Optimization (CO). NI-based approaches rely on Pontryagin's Maximum Principle, which states that the time-optimal solution consists of alternating maximum acceleration and deceleration segments with switching points in between. Usually, the Maximum Velocity Curve or MVC is either explicitly computed or implicitly found at the needed path positions. The curve is used to find switching points or, if possible, to move with maximum path velocity along a path segment. NI algorithms can also belong to the category of Dynamic Programming (DP) methods, as they divide the minimum-time problem into a series of smaller sub-problems. These approaches are fast, but reliably finding switching points may be difficult and time-consuming, and creating a robust implementation of such an algorithm can be challenging. When the integration process requires the explicit selection of an appropriate optimal value around switching points and singularities, these approaches contribute insights into the properties of these points, how they may be found, and the conditions for them to arise.

CO-based approaches usually solve a large global optimization program that searches for optimal controls. Although these methods are simple to implement and robust, they are usually at least one order of magnitude slower than NI-based approaches. In contrast to DP-based approaches, the computation time is not roughly proportional to the number of data points in the s-domain and usually increases significantly more with increasing path complexity. The computation time also increases more than linearly with the number of constraints (Pham, 2019). Recent advances move towards a similar DP-based approach by splitting up the problem into smaller CO problems that can then be solved using linear programming solvers. These approaches benefit from reduced computation times and provide insight into finding the true globally time-optimal solutions that are not just local minima.

2.2.1.1 Numerical Integration and Dynamic Programming

Kunz and Stilman (2012); Kunz (2015) derive efficient and stable analytical expressions for the MVC using symmetric velocity and acceleration limits. They use line segments and circular blends to interpolate a given path consisting of waypoints. This approach does not yield a continuous path where all waypoints are on the path, but the robot moves near intermediate waypoints on blends. Their algorithm also requires a numerical search for switching points. Such a search requires stepping locally along the path, which can be time-consuming and does not allow limiting the number of scans of path segments to a certain amount. However, their open-source implementation has been proven to be reliable and efficient for offline planning. And their algorithm is the standard path param-

eterization method in the popular *MoveIt* motion planning framework (Sucan and Chitta, 2026) as part of ROS[1].

Pham (2014) provides a general and fast implementation that can also handle dynamic constraints. A thorough classification of dynamic singularities is also given. Nguyen and Pham (2016) extends the approach to SO(3) motions and applies them to spacecraft reorientation tasks, but this extension is also relevant for time parameterization of manipulator end-effector motions.

Dong and Stori (2006) present an algorithm that only requires two passes along the path, one forward and one backward. They do not compute the MVC directly, but use line-search to find switching points. This approach allows them to directly identify switching points during one of the two path scans without additional search steps along the path. During the backward pass, the velocity profile from the forward pass is used as an upper limit. Barnett and Gosselin (2020) build on their work and use a bisection search algorithm instead of the line search, which is more numerically stable. Their results show that their algorithm is the fastest implementation so far, considering only first- and second-order kinematic constraints, and it is also stable with long paths.

The works listed above are not able to handle third-order constraints like jerk. Tarkiainen and Shiller (1993) introduced an approach to handle general third-order constraints, but proper handling of singularities is missing. They suggest modifying the original path to avoid them, but this approach is not feasible for arbitrary path shapes and is not suitable for online planning. Pham and Pham (2017) further investigate the TOPP problem with third-order constraints. They provide insights into the different types of singularity that might arise and suggest an approach to connect two maximum jerk profiles with a minimum jerk profile, which is more efficient than previous approaches. While their algorithm is time-optimal, it is only able to handle third-order constraints and not lower-order constraints. Mattmüller and Gisler (2009) introduce a nearly time-optimal solution and show how different cases that arise during integration can be treated. They connect maximum acceleration profiles using backtracking and integrating with minimum jerk from both sides to achieve a smooth connection without jerk violations. This approach is slow because, in the case of large jerk violations, multiple backtracking steps are needed, and for every backtracking step, a full integration from both sides for the segment is executed. Additionally, such jerk violations are usually frequent along a path. Pham and Pham (2017) propose a solution for this based on Multiple Shooting, which appears to be faster. Dong et al. (2007) extend their previously published time-optimal feed rate optimization algorithm (Dong and Stori, 2006) with jerk constraints. They do the first two scans using maximum jerk (forward and backward) to achieve a continuous velocity profile. Velocity and acceleration constraints are enforced during both scans, while jerk constraints are ignored if no feasible acceleration can be achieved otherwise in a given optimization step. A third pass is added to add minimum jerk profiles that remove the acceleration discontinuities. Backtracking is used again when the minimum jerk profiles cannot be connected from the initial integration start positions. In conclusion, handling third-order constraints efficiently and reliably remains a subject of ongoing research in the TOPP field, even for offline methods.

The work of Wang et al. (2021) is an example of incorporating more complex third-order constraints, such as the torque rate and Coulomb viscous friction, in the dynamics model.

[1] Documentation is available at https://moveit.picknik.ai/main/api/html/namespacetrajectory__processing.html.

Their approximate time-optimal algorithm is more efficient than previous work for offline planning. Still, it also shows that finding the true time-optimal solution under various third-order and dynamics constraints is not yet achieved or sufficiently understood. The recent work by Kiemel and Kröger (2024) should also be noted for its alternative approach. They propose an iterative scheme to compute feasible target accelerations for intermediate waypoints and use Ruckig (Berscheid and Kröger, 2021) to compute intermediate trajectories, which are jerk-limited. To improve and achieve accurate path tracking, the path must be scanned several times iteratively, which is not suitable for a real-time approach. Unfortunately, the evaluation lacks an investigation of the computational efficiency of the algorithm.

2.2.1.2 Convex or Non-linear Optimization

Verscheure et al. (2009) provide the reference work for solving TOPP as a large convex optimization program with restrictions on torque, velocity, and acceleration. They introduce a direct transcription method that achieves global optimality by solving a second-order cone program.

Alternatively, the problem can be split into a set of small linear programs that can be solved recursively by scanning the path similarly to the NI approach. These methods can still be classified into the CO category, as a CO solver is still required, but these approaches could also be categorized into the dynamic programming class. TOPP-RA by Pham and Pham (2018) is a popular example of this approach. Reachable and controllable sets are computed using reachability analysis, and dynamic constraints are supported. The asymptotic optimality of this algorithm has been shown by Spasojevic et al. (2019) when the distance between discretization points tends to zero. However, the computation times are slower than reported by the algorithm of Barnett and Gosselin (2020). Consolini et al. (2019) present a faster solution than Pham and Pham (2018) that requires fewer linear programs per path node, and more efficient solver methods are proposed. Similarly, Nagy and Vajk (2019) also provide a faster solution, using a special sequential solver. Especially for longer paths with several thousand discretization points, their implementation provides a runtime performance of around 10 ms with approximately linear increase in computation time depending on the path length. However, their algorithm cannot handle restrictive torque constraints. It is also not shown how these algorithms can be extended to third-order constraints, especially without an increase in computation time of several orders of magnitude. Further work in this direction is discussed next.

Palleschi et al. (2019) formulate a global optimization based on Verscheure et al. (2009) with the added support of jerk constraints. While the resulting non-linear programming problem is non-convex, they manage to achieve a convex relaxation using McCormick envelopes. A related work from Debrouwere et al. (2013) uses sequential convex programming and introduces jerk constraints as a difference of convex functions. Unfortunately, Palleschi et al. (2019) do not address how their work compares to Debrouwere et al. (2013), and they deferred further investigation of computational efficiency and its improvement using different solvers to future work. Ji et al. (2023) conduct more in-depth evaluations and compare different CO approaches in this area. They manage to consider system dynamics and reach near time-optimality in their algorithm while retaining low computation times compared to other algorithms of this class.

Ma et al. (2021) transform the jerk constraints into linear acceleration constraints and introduce them to the convex optimization. The approach is basically an extension of TOPP-RA (Pham and Pham, 2018) using a constant path jerk in phase space, which does not allow the imposing of direct restrictions on joint jerks, but reduces them. The trajectory at the start and goal positions is not jerk-restricted. Inspired by sampling-based path planning algorithms, Huang et al. (2023) use custom sampling strategies to build a tree of feasible nodes in the phase space. They aim to extend the TOPP-RA algorithm with jerk constraints, thereby avoiding the usual issues of finding switching points or eliminating jerk-violating segments. Computation times are hard to predict with this approach, but they show in their evaluation that their algorithm is only roughly five times slower than TOPP-RA. Wang et al. (2023) achieve a jerk-limited joint trajectory by also building on TOPP-RA and adding two more path scans to eliminate maximum and minimum jerk violations. Dynamics and torque constraints are supported as well. Similar to jerk-limited NI solutions, iterative integrations and switching point search algorithms are needed, which make the computation time complexity dependent on the path shape. The computation times are higher than the trajectory durations, indicating that incorporating jerk constraints and robot dynamics remains complex and time-consuming for such offline methods, despite recent advances.

A different group of methods uses non-linear optimization solvers to support third-order constraints and various techniques to increase computational efficiency and stability. The resulting optimization problem is then usually not convex anymore. Lin et al. (2021) use non-linear optimization to include jerk constraints. Although not real-time-capable due to iterative optimization components, the short computation times are still notable for less restrictive jerk limitations. However, similarly to Kunz and Stilman (2012), blending is used around the waypoints. This approach results in paths that approximate straight lines, but intermediate waypoints are no longer hit. The robot only moves near these waypoints on a blend. Wittmann and Rixen (2022) use a different approach to make the non-linear optimization more efficient by using zero-clamped cubic splines and introducing their analytical gradients to the optimization problem. Their benchmark shows average computation times for their parameterization, which are twice as high as those in Kunz and Stilman (2012). This result does not allow their approach to compete with the most efficient second-order NI-based methods. Still, it shows an interesting avenue for adding jerk limits in a computationally efficient manner to the optimization problem that can be solved using a standard non-linear solver like sequential quadratic programming (SQP). They assume approximate linear time complexity with respect to the number of waypoints and degrees of freedom. From their evaluation, it is unclear whether the initial guesses needed for the segment durations and virtual waypoint locations generalize to arbitrary path shapes and constraints, and how they influence the computation times. Additionally, their spline formulation is likely not applicable to non-zero start and goal velocities for an online planning application.

Although global motion planning methods that optimize a collision-free path together with the trajectory generation are out of the scope of this literature review, Nvidia's cuRobo (Sundaralingam et al., 2023a,b) can still be mentioned as they manage to generate jerk-minimized fast trajectories and use parallel GPU-accelerated solvers for a significant reduction in computation times compared to similar previous works. They note that their approach can be extended to work with non-zero start and goal states for online planning. But the usual unpredictable increase in computation time on more complex problems, which is common to global optimization using general solvers, is still present.

2.2.2 Online and Real-Time Approaches

As already mentioned, popular OTG methods like the Reflexxes motion libraries (Kröger, 2011) and Ruckig (Berscheid and Kröger, 2021)[2] usually generate point-to-point trajectories in joint space or on SE(3) (Huber and Wollherr, 2019b), e. g., for the robot's end-effector, and cannot be used for path following. However, there are a few exceptions to this rule. Lange and Albu-Schäffer (2015) present an online method for jerk-limited path-accurate trajectory generation; however, this approach is not intended for a given path consisting of waypoints and with arbitrary path length. Rather, it generates a trajectory that is as close as possible to an original robot program and synchronized with it by back-tracking and scaling the given commanded trajectory. It improves the path tracking on small online path changes while ensuring that kinematic constraints are respected.

There also exist approaches that scale or filter a given reference trajectory online, possibly also in real-time, in case the original trajectory becomes infeasible for the robot constraints or to respect additional constraints like jerk or torque (Faroni et al., 2021). Task scaling techniques behave similarly, where, e. g., the robot motion is slowed down if a human enters a predefined safety zone; see also speed and separation monitoring techniques in the technical specification ISO TS 15066 (International Organisation of Standardisation, 2016). These methods assume that not only a path but also a velocity profile was computed beforehand, and they try to stay close to the original path, so only the velocity profile is adapted at best. This type of technique is not the focus point, as only geometric information about the path, such as discrete waypoints or setpoints, is known before trajectory generation in the problem that is addressed in this thesis. The actual generation of initial velocity profiles in an online and real-time manner is the research question at hand, which would allow the path to be changed during runtime, react to those path changes, and follow the new path exactly. However, Zanchettin and Lacevic (2022) with the use of the algorithm from Casalino et al. (2016) move towards a path parameterization approach also in this human-robot collaboration application area. They use a bang-bang-type controller with path acceleration as control input and compute path-constrained braking trajectories for every trajectory state to check if the robot can be safely stopped and thus, if it is still a safe path state.

Kim and Croft (2019) introduce a concept that uses TOPP methods for locally near time-optimal online trajectory generation, but only for short path segments. The algorithm considers velocity and torque limits and runs in a control cycle of 6 ms. The method interpolates between the trajectories generated separately for each segment of the path, focusing on reducing time-consuming dynamics computations. They also assumed that the trajectories for every path segment can be simplified by only three phases, namely acceleration, maintenance, and deceleration, which does not apply to arbitrary paths with longer or more complex path segments, e. g., with several corners.

Zhang et al. (2024) use a receding horizon trajectory generation method to achieve near time-optimal path tracking under second-order kinematic constraints. The goal is to react to online path changes. Like Kim and Croft (2019), they employ trapezoidal velocity profiles. Only batches of two waypoints are processed, so a local trajectory is computed

[2]There is also the proprietary version Ruckig Pro available, which supports intermediate waypoints and jerk limits. However, their optimization does not guarantee time-optimality, only that the iteratively improved trajectory will be faster than stopping at each intermediate waypoint. For more information, see `https://docs.ruckig.com`.

to reach the next waypoint within a control cycle. The next direction switch independently for each axis is taken into account with the assumption that zero velocity needs to be reached at these points, or in the absence of a direction switch point, it is ensured that zero velocity can be reached at the goal state. In this way, the algorithm does not support non-zero velocity at the goal state. Oscillating behavior arises from this approach because frequent acceleration changes occur with such a short planning horizon. The experiments show four slow movements along smooth paths with small velocity and acceleration bounds. Guarantees are not given that the OTG always finds a valid near-time-optimal trajectory if one exists, but it is an interesting approach to splitting up a path into segments and sub-problems for TOPP.

He et al. (2022) propose a nearly time-optimal switching trajectory index coordination framework that utilizes control law methods and manages to synchronize several axes. On a test trajectory, one iteration of the method is shown to be real-time capable to run in one control cycle. They evaluated on a test system with two axes using entirely geometrically predefined continuous paths and several tuning parameters. Lin et al. (2023) build upon this previous work and define a control invariant set, which ensures that a feasible control input is always available to keep the system within velocity, acceleration, and torque bounds. They use a local greedy search within this set to find a time-optimal trajectory. The resulting linear programming problem can be solved iteratively in every control cycle and is validated on a seven-DoF robot, at least for a sample path. The backup control strategy theoretically ensures that the system is driven into an admissible area below the MVC as quickly as possible, from which the time-optimal trajectory can be further constructed along switching points. However, in practice, a maximum of N prediction steps along the path must be selected to ensure a computation bound. This makes it challenging to ensure that the prediction horizon is generally large enough so that an existing solution is always found. In the same way, a time horizon for the state flow map needs to be selected. Still, it is a promising real-time time-optimal planning approach under kinodynamic constraints. Further evaluations, e. g., using a large set of randomly generated paths, are needed to investigate if constraints are always respected, the feasible time-optimal trajectory is always found, how computation times behave, and if online path changes can be processed.

2.2.3 Summary

Robot path parameterization methods transform a given N-DoF path consisting of waypoints into a continuous path described with a single path variable and its derivatives, also called phase space. Then, a velocity profile for the path is generated in the phase space, which can be mapped back to a time-optimal trajectory in joint space. Offline time-optimal path parameterization methods create a trajectory for the whole path before execution and cannot be directly used for online and real-time trajectory generation. They cannot adhere to upper bounds on computation time and usually also do not support a start or goal state where the robot is not at rest. However, as research in this field has increasingly focused on reducing computational complexity and increasing efficiency, fast and robust algorithms have been developed. Most notably, in the class of numerical integration, methods based on dynamic programming can find a valid trajectory with only two integration passes along the path and identify switching points for the path acceleration on the fly. Similarly, in the branch of convex optimization approaches, complexity was significantly reduced by splitting the problem into small linear programs (LPs), further reducing the number of necessary LPs to be solved, or improving the solver efficiency.

The above-described advances hold for first-order and second-order kinematic constraints like joint velocity and acceleration bounds, and in a few cases for dynamic constraints like torque bounds. Supporting efficiently and reliably third-order constraints like jerk or torque rate is still ongoing research, even for offline methods.

Most real-time online trajectory generation methods target point-to-point movements without intermediate waypoints. There exists a gap between offline and online methods, and only a few works address the problem of online trajectory generation for paths consisting of waypoints without prior timing information, e. g., generated by a path planning algorithm. Only a short planning horizon is usually considered, with little information on the remaining path. Global feasibility cannot be ensured in practice, and the quality of the velocity profile suffers. Nevertheless, some methods can achieve near globally time-optimal results for given example paths and can process online path changes, or ensure the robot can be stopped on the path. Bringing advances from offline methods to the online and real-time application area could help to reduce the gap further and provide improved parameterization results with larger planning horizons and increased reliability.

2.3 Remote Robotics Testbeds

This section details the related work relevant to the third contribution of this thesis, the remotely accessible Robot Learning Lab. The laboratory aims to make modern industrial manipulators accessible over the Internet to develop and test robotics software applications.

Despite continuously decreasing costs, robots are still expensive hardware. This is especially true for state-of-the-art industrial lightweight robots with force/torque sensing capabilities in every joint. To utilize robots for various applications, research facilities and universities establish general-purpose robot cells equipped with additional sensors. Maintaining these experimental robot cells demands both expertise and time. In addition, software interfaces are typically designed for a particular application, which leads to extra effort when the robot cell is adapted for other kinds of experiments. Safety measures, such as workspace restrictions and collision detection algorithms, are often insufficient, necessitating constant monitoring of experiments to prevent hardware damage.

For these reasons, it is common for only select students to be able to use such robot cells within the scope of long-term projects like theses. For smaller student and research projects, the effort is usually not invested in setting up the cell for the project, or long-term research projects have prioritized access. Most students need to resort to simulation, just as researchers without sufficient funding do. However, current simulation frameworks do not sufficiently reflect real-world behavior, especially when manipulating objects or emulating sensors (Collins et al., 2019). A simulation-only approach prevents researchers from validating their methods with real-world experiments. On a larger scale, the concept of Crowd Research (Vaish et al., 2017) has been introduced to provide more open access to research experiences, create tools for university researchers and diverse volunteers to collaborate in the field of Computer Science, and produce top-tier scientific results and publications. Easy and open access to robotic hardware is needed to bring this concept to the field of robotics and allow roboticists to conduct experiments, benchmark, and compare results. Unified and standardized benchmarking platforms are required to compare

methods and make results easier to reproduce. A remotely accessible robotics lab is a possible approach to address this demand.

From a student's perspective, exploring the foundations of robotics by controlling a real robot that interacts with the real world can be a significant motivational factor in education, allowing one to gain experience with robotic hardware. Few may be able to participate in multi-day workshops or competitions where they are introduced to configuring or programming a professional-grade robot, like, e. g., it was done in the Roboterfabrik project (Haddadin et al., 2019). In a university practical course, select students may be able to test their developed project on a robot. In secondary school education, it has been shown that a remote lab, where students could test their programs with an educational robot, significantly improves learning outcomes compared to a pure simulation environment (Gao et al., 2023).

Many, if not most, university students who focus on robotics as part of their curriculum never work with a professional robot during their studies. The same is true for online robotics courses or MOOCs (Massive Open Online Courses), where usually only a simulator is provided for exercises, or students have to create their own small experimental setups at home (Corke et al., 2016; Morimoto et al., 2020). However, it would be very beneficial for students to gain some practical experience with robots before starting their careers in the robotics industry. They would learn not only how to work with industrial-grade robotic hardware, but also how to handle real sensor data. It also helps to demonstrate the relevance of the theoretical content of the lecture. The same applies to training courses for the industry workforce. A remotely accessible robotics lab with automated experiment execution in a safe environment can address these issues.

In the following sections, a brief overview of remote access testbeds is provided that have achieved success in their respective domains, such as robotics research or education, and categorize them by the type of robotic hardware they utilize: stationary robots and mobile robots. This survey excludes remotely accessible robotics labs that offer only teleoperation or telepresence capabilities, as such setups are not appropriate for software development purposes. In some cases, remote access to a robotics lab system is provided mainly by standard means such as VPN access and remote desktop sessions, supervised by a lab operator (Kumar et al., 2024). These works are also not further detailed because the focus here is on advances in remote robotics lab technology.

Benchmarks and one-time competitions are also not included if code submissions are only uploaded (e. g., as an archive or Docker container) via a standard off-the-shelf web form or API without novel remote access and remote programming methods, and manually executed by an operator on a robotic system without automated evaluation and direct user feedback (Zhou et al., 2023). Such platforms typically have only one specific setup for the competition or benchmark. They are not designed to be flexible for different experimental setups or to provide a general-purpose robotics platform base for various research directions or educational exercises. However, a few cloud benchmark competitions, such as those for grasping and manipulation, provided similar advances. Some are useful for remote robotics testbeds, such as unified and pre-configured offline development environments, standardized testing scenarios for specific tasks, and procedures to at least partially automate experiment execution, data collection, and ranking of results on a leaderboard (Liu et al., 2021; Gürtler et al., 2022). As an example, the Real Robot Challenge utilized an established cluster system framework to manage user submissions and submission distribution onto their eight manipulation platforms, each equipped with three fingers and

small objects that can be grasped. They provided SSH Shell access to the cluster for container upload and download links to retrieve recorded data (Bauer et al., 2022).

The focus here is on remote labs that allow users to develop their own software programs, test, and iterate with real robotics hardware in the loop over the Internet, and that have been online and operational at least at some point in the past. Novel methods for remote robot programming, automation to minimize manual intervention, and safety and security measures to protect lab infrastructure are highlighted.

2.3.1 Stationary Robot Testbeds

Some remote robotics labs have been developed that use a stationary-mounted robotic manipulator, such as the industrial robots used in this thesis as part of the Robot Learning Lab. The Telerobot (Trevelyan, 2004), an initial work on the Web from 1994 to 2004, and the RACT lab (Casini et al., 2008) each employ an industrial robot, while Balestrino et al. (2009), Goldstain et al. (2007), Jara et al. (2011), and Šafarič et al. (2005) use an educational robotic manipulator. Trevelyan (2004) and Balestrino et al. (2009) allow students to program the robot for pick-and-place tasks with the help of a graphical user interface. The setup of Goldstain et al. (2007) also supports the upload of code files generated from a simulator. In addition, a visual servoing task is provided by Casini et al. (2008). Jara et al. (2011) also make it possible to use robot-specific code-like instructions through the web interface.

Although these remote robotics labs were pioneering work at their time, they lacked the flexibility to conduct research experiments. They do not provide the ability to submit programs in a general-purpose programming language, and the feature set is almost entirely limited to what is needed to teleoperate the robots in the lab. Except for the work of Šafarič et al. (2005), only basic safety needs are addressed for specific projects that were developed for the labs. Safety is not treated in a general and holistic way, and the authors do not present safety concepts that apply to different experimental setups.

2.3.2 Mobile Robot Testbeds

In recent years, the development of remote robotics labs with stationary robots has ceased, and mobile robot testbeds have become popular, especially with single or multiple small wheeled robots. The PR2 Remote Lab (Pitzer et al., 2012) gave access to a PR2 personal robot, and RPN (Casañ et al., 2015) provided a NAO humanoid for remote access and web-based programming. Gao et al. (2023) let users control a small four-wheeler robot with a planar arm to pick up cubes in a game field. The Robotarium (Pickem et al., 2017), Duckietown autonomous driving miniature setup (Tani et al., 2020), SyRoTek (Kulich et al., 2013), Robotnačka (Petrovi and Balogh, 2012) and Remrob (Krūmiņš et al., 2024; Schumann et al., 2023) deploy small wheeled robots. The RemRob lab features several separate robot cells, each with one mobile robot, allowing users to work in parallel with a robot. In contrast, the Robotarium offers a single large area where a fleet of mobile robots can be controlled. The PR2 Remote Lab (Pitzer et al., 2012) focused on research and development, while the RPN (Casañ et al., 2015), SyRoTek (Kulich et al., 2013), Robotnačka (Petrovi and Balogh, 2012), Gao et al. (2023), and Remrob (Krūmiņš et al., 2024) target online education at the school or university level. The Duckietown automated

lab (Tani et al., 2020) and Robotarium (Pickem et al., 2017) were used both in teaching and research.

The Robotarium allows users to develop their algorithms in MATLAB or Python using their custom simulation. Registered users can then submit their scripts utilizing the lab's website. Submissions are automatically run in a queue, and recorded videos and data are available online after the experiment finishes (Wilson et al., 2020). The Duckietown challenge server provides benchmark results for different challenge specifications and setups and allows users to view results. Gao et al. (2023) provided a simulation with a Python-based programming interface and some way to manually start remote experiments and view a live video feed. Remrob (Krūmiņš et al., 2024) has a more sophisticated system to allow multiple students to work in parallel with a simulation and/or a real robot in one of the multiple robot cells in the lab, as the lab is intended for a massive open online course (MOOC) to give learners hands-on experience with ROS. They used hardware-accelerated desktop environments running in Docker containers on the lab's server. They streamed a remote VNC session to the user's browser. Hence, users had remote access to a ROS development machine, and capabilities were available to save sessions, load configurations for different projects, and access a live video stream of the robot cell. The sessions are distributed using time slots in a booking system, and an admin interface for monitoring is available to the lab's operator.

Of these mobile robot testbeds, only the Robotarium has safety measures that prevent damage to the robots, even from untrustworthy or malicious users (Wilson et al., 2021). Their safety protocol is specifically for a fleet of small wheeled robots operating in a wall-enclosed arena. These safety challenges for swarm robotics differ significantly from those for industrial manipulators. In addition, the risks of expensive hardware damage are much higher with high-end industrial manipulators than with low-cost mobile robots in comparison. In contrast to the testbeds described above with stationary robots, almost all mobile robot labs allow local execution of submitted user code, but the security risks of code compromising local machines are not sufficiently or not at all addressed, especially regarding network security. The Robotarium (Wilson et al., 2021), Duckietown lab (Tani et al., 2020), and Remrob (Krūmiņš et al., 2024) at least make use of containerization to provide a consistent local code execution environment. The Remrob lab had to enable additional system privileges for the user's desktop environments, which creates a significant additional attack surface on the lab's host server and infrastructure. In addition, users have unrestricted access to the lab network. In general, misconfigurations in robotics labs can expose sensor data or robot control interfaces to the public Internet (DeMarinis et al., 2019). A remote lab requires not only secured and well-defined Internet-connected interfaces, such as web APIs or browser-based interfaces, but also has the additional challenge of shielding the lab's internal network with robot and sensor interfaces from the executed submitted user software. The submitted source code must be executed in a sandbox with access only to the robot interfaces and sensors allocated to it. Additionally, the interfaces must be secured to prevent misuse or malicious attacks on the lab infrastructure.

For the mobile robot system, automated reset of the physical environment after an experiment is finished mainly involves moving the robots to a starting position and providing charging capabilities, which are also not always available (e. g., in the Remrob lab). If the environment can be changed or there is some manipulation or interaction capability, automation of experiment execution and the subsequent reset becomes more complex. Some labs require intervention by an operator. Gao et al. (2023) only solved this for their

specific game field setup, where the cubes, which should be picked up, are four-wheeled robots themselves and can move to a starting position.

Several sensor network testbeds exist that also provide mobile robots, most notably the FIT IoT-LAB (Adjih et al., 2015). However, their research focuses on networking solutions in an IoT context and does not target robotics. There is also the Construct Sim[3], a commercial provider of robotics and ROS online training courses for industry, individuals, and campuses (Tellez, 2016). They offer a web-based development environment that mimics a minimal and simplified desktop workspace with a ROS development environment. At a subscription fee, a mix of mobile robots and manipulators in different setups is available, which are also customized for enterprise clients. After booking a time slot, users can connect from the web environment to one of the robots and work with the robot for the allocated time. Their system is proprietary and closed-source software, and cannot be used or adapted outside their platform with a custom development environment or code upload. This restriction makes it unsuitable for research applications or for designing advanced university courses that could also be made available to the public.

2.3.3 Summary

A robotics testbed, which is remotely accessible over the Internet, has been proven to be useful in education and research domains. Students at school or university usually do not have access to expensive professional or industry-grade robots as part of their curriculum, even in a robotics specialization. Access to real robotics hardware in education can be significantly increased by a remote lab and scaled to several users in parallel if several robots are available and the experiment execution is automated, safe, and secure. Remote access to a real robot can increase motivation, and it makes experimenting and applying the learned theory possible with a real system. Researchers can evaluate algorithms on such a testbed, and the testbed can be used as a benchmark platform. Roboticists, who previously were only able to test in simulation or had to invest time and money to create their own test setup, can then remotely run experiments and achieve a sim-to-real transfer of their research. In recent years, several remote labs have become available that provide access to small wheeled robots for education or research fields such as swarm robotics or autonomous driving. To the best of our knowledge, no current remote lab with several modern industrial-grade manipulators is accessible for research and education, which would be helpful to teach robotics basics and conduct research in manipulation and industrial automation, among other applications. The additional requirements to automate, safeguard, and secure such a testbed are not covered by the current state of the art. Especially, the requirement for a novel motion planning interface needs to be addressed, which prevents damage to the hardware due to misuse while still allowing fast, reactive, and flexible motion generation. Easy remote programming access is required for students, and an automated submission processing system is needed for several robot cells in parallel. For research, the lab software and hardware framework must enable hosting projects of arbitrary complexity and serve as a development platform that provides software development capabilities established in the field.

[3] https://www.theconstruct.ai

CHAPTER 3

Real-Time Inverse Kinematics for Seven-DoF Manipulators

In this chapter, we present a novel multi-objective optimization approach to redundancy resolution. The method aims to minimize the joint velocities and accelerations while maximizing the distance to the joint limits simultaneously. The aim is to facilitate faster and smoother motions along Cartesian paths while not compromising reachability. A closed-form solution to the optimization is obtained, which can be conveniently adjusted using a single timing parameter. A free and open-source software implementation of the complete kinematics algorithm is provided alongside this thesis.

Publications Related to this Chapter
This chapter is based on the following previously published journal paper, which introduced the used method and described the experiments:

- **Wolfgang Wiedmeyer**, Philipp Altoé, Jonathan Auberle, Christoph Ledermann, and Torsten Kröger. A real-time-capable closed-form multi-objective redundancy resolution scheme for seven-dof serial manipulators. *IEEE Robotics and Automation Letters*, 6(2):431–438, 2021.

The chapter is structured as follows. The proposed multi-objective redundancy resolution scheme is detailed in Section 3.1, including the derivation of a closed-form solution to the optimization problem. Section 3.2 describes the implementation of the complete IK algorithm and its integration into a motion planning framework. The experimental validation of the approach is reported in Section 3.3. First, the computational efficiency of the algorithm is evaluated. Second, the null-space motion along given Cartesian paths and its effects on trajectory parameterization are studied in comparison to the state-of-the-art redundancy resolution method described in Section 2.1.3. Finally, experiments are conducted with a KUKA iiwa robot. Conclusions are summarized in Section 3.4.

3.1 Multi-Objective Optimization as Redundancy Resolution

The redundancy resolution described in Section 2.1.3 tries to maximize reachability by pushing away from the joint limits. This objective is often chosen for motion planning. It would be desirable to integrate additional objectives into the redundancy resolution, such as minimizing joint velocities and accelerations. Moradi and Lee (2005) use the minimization of the arm angle movement as the optimization objective, but this does not necessarily lead to reduced joint velocities. The reachability objective generally requires increased joint velocities because the push-off from the joint limits forces a perpetual adjustment of the arm angle. This behavior usually results in a decreased end-effector velocity and longer trajectory execution times.

To justify optimizing joint velocities within the redundancy resolution, consider the time derivative of the joint angle at time step t:

$$\frac{\mathrm{d}\theta_i(t)}{\mathrm{d}t} = \dot{\theta}_i(t) = \frac{\partial \theta_i(\boldsymbol{P}(t), \psi(t))}{\partial \boldsymbol{P}(t)} \cdot \dot{\boldsymbol{P}}(t) + \frac{\partial \theta_i(\boldsymbol{P}(t), \psi(t))}{\partial \psi(t)} \dot{\psi}(t) \ . \tag{3.1}$$

$\boldsymbol{P}(t)$ is the path consisting of end-effector poses and $\dot{\boldsymbol{P}}(t)$ denotes the end-effector velocity. In this chapter, θ_i refers to the angle of joint i. The first term in Eqn. (3.1) represents the necessary change in the joint angle to account for the end-effector movement $\Delta \boldsymbol{P}$ to reach the goal pose $\boldsymbol{P}_t$, while the arm angle is kept constant. The second term is the added joint movement to reach the variable arm angle ψ. An optimization can select the arm angles at each time step, so that the second term minimizes the overall joint angle derivative. Suppose we assume that the joint velocity is bounded. In that case, a higher end-effector velocity can be achieved because, through the minimization together with the second term, the value of the first term can be increased. A subsequent time-optimal path parameterization can take advantage of the results of this optimization to achieve a shorter execution time for the given path, which we show in the evaluation (see Section 3.3).

The behavior can be influenced by changing the tuning parameters in Eqn. (2.3) from the existing redundancy resolution by Faria et al. (2018), but this approach is not very intuitive and only allows rough adjustments. It is also challenging to find parameters that work well across a larger set of different paths in the task space.

It would be advantageous to directly include the minimization of joint velocities as an objective for the redundancy resolution. In addition to joint velocity optimization, further time derivatives of the joint angle, such as acceleration, could also be minimized in such a closed-form redundancy resolution approach. Optimizing joint accelerations results in smoother trajectories. Execution times for trajectories are also shortened when expensive joint decelerations are avoided, and high joint velocities are preserved.

To provide sufficient flexibility for online motion planning, redundancy must be resolved locally for each individual pose, relying only on information about robot configurations from earlier time steps.

To address these requirements, a multi-objective optimization approach is selected that combines joint limit avoidance with the minimization of joint angle derivatives. Optimizing joint velocities by minimizing joint angle changes during path planning allows

subsequent time-optimal trajectory generation to achieve a shorter execution time for the trajectory, which we show in the evaluation (see Section 3.3). Optimizing joint accelerations leads to reduced joint torques, which can lead to smoother trajectories. Optimization of joint jerks can also be included in the overall optimization scheme if it could be beneficial for an application. But usually, only jerk limitation and not jerk minimization is required.

3.1.1 Combined Cost Function As Weighted Sum

The goal of the optimization is to determine an arm angle ψ_t that minimizes the discrete-time cost function f:

$$\psi_t = \underset{\psi}{\operatorname{argmin}} \left\{ f\left(\boldsymbol{\theta}_t(\psi), \boldsymbol{\theta}_{t-1}, \boldsymbol{\theta}_{t-2}\right) \right\} \ , \tag{3.2}$$

using joint angles $\boldsymbol{\theta}$ from previous time steps.

The cost function f is selected as

$$\begin{gathered} f = w_l \, f_l + w_v \, f_v + w_a \, f_a \, , \\ \text{with } w_l = w_{dis}, \ w_v = \left(1 - w_{dis}\right), \ w_a = s_a \left(1 - w_{dis}\right), \ s_a \in \{0, 1\} \, . \end{gathered} \tag{3.3}$$

f_v and f_a are the objective functions for joint velocities and accelerations, f_l for the repulsion from the joint limits. Individual objective functions are described in the next section.

w_{dis} is a dynamic weight and is defined as

$$w_{dis} = \frac{\left| \psi_{t-1} - \left(\psi_m^u + \psi_m^l\right)/2 \right|}{\left(\psi_m^u - \psi_m^l\right)/2}, \quad w_{dis} \in [0, 1] \, . \tag{3.4}$$

Depending on the distance of the previous arm angle ψ_{t-1} from the limits of the current interval, w_{dis} causes a more substantial weighting of the repulsion objective f_l, or the joint derivative objectives f_v and f_a are more weighted. This adaptive weighting guarantees that, when the arm angle is already near its bounds, f_v and f_a do not drive it even closer to those limits. Conversely, f_l is prevented from exerting a damping influence on the self-motion when the arm angle lies near the center of the admissible interval.

s_a in w_a makes it possible to enable or disable the inclusion of f_a, since it is optional. f_v must always be used together with f_a because minimization of joint accelerations alone can cause instabilities: If repulsion from the joint limits forces the acceleration of the joints, then f_a will prevent the braking of the respective joints until a joint limit is approached again. f_v enforces deceleration of the joints and prevents such runaway behavior.

w_v, w_a and w_l are linear in w_{dis}. Optionally, the tuning parameters $d_{(\cdot)}$ could be used to change the shape of the weighting functions and thus influence the relative weighting between the objectives: $(w_v)^{d_v}$, $(w_a)^{d_a}$, $(w_l)^{d_l}$. Although this can improve performance, these parameters are not used in this thesis to keep the set of necessary tuning parameters small.

3.1.2 Objective Functions

Quadratic cost functions are selected for the different objectives. They are particularly suitable for obtaining a closed-form solution because their derivatives are linear. Due to the quadratic form, large joint angle derivatives are weighted disproportionately higher, which is advantageous when aggregating across all seven joints. The objective functions are likewise normalized to range from zero to one, which facilitates their comparison.

For the reachability objective, a possible approach would be to reuse the redundancy resolution from Faria et al. (2018) described in Section 2.1.3. However, this would reintroduce the two parameters K and α, which are difficult to tune. Instead, the reachability objective is chosen as

$$f_l(\psi) = \left(\frac{\psi - \left(\psi_m^u + \psi_m^l\right)/2}{\left(\psi_m^u - \psi_m^l\right)/2} \right)^2 . \tag{3.5}$$

This objective forces an aggressive push-off from the lower border ψ_m^l and the upper border ψ_m^u of the currently feasible interval towards the middle. If all arm angles are possible for a goal pose and there are no blocked intervals, then this objective is ignored ($w_{dis} = 0$).

Joint velocities are minimized using

$$f_v(\psi) = \frac{1}{7} \sum_{i=1}^{7} \left(\frac{\dot{\theta}_{i,t}(\psi)}{\dot{\theta}_{i,max}} \right)^2 \approx \frac{1}{7} \sum_{i=1}^{7} \left(\frac{\theta_{i,t}(\psi) - \theta_{i,t-1}}{\dot{\theta}_{i,max}\, \Delta t} \right)^2 , \tag{3.6}$$

where $\dot{\theta}_{i,max}$ denotes the joint velocity limit of the joint i. The joint velocity $\dot{\theta}_{i,t}(\psi)$ is approximated by using a finite backward difference with the sampling time Δt. This approximation requires the assumption that Δt is sufficiently small.

The term f_a, used to minimize joint accelerations, is formulated accordingly based on the second-order finite backward differences of the individual joint angles and the corresponding joint acceleration limits $\ddot{\theta}_{i,max}$:

$$f_a(\psi) = \frac{1}{7} \sum_{i=1}^{7} \left(\frac{\theta_{i,t}(\psi) - 2\,\theta_{i,t-1} + \theta_{i,t-2}}{\ddot{\theta}_{i,max}\, (\Delta t)^2} \right)^2 . \tag{3.7}$$

Please note that, under these objectives, the previous arm angle ψ_{t-1} is not explicitly required to lie within a feasible interval at the goal pose. Nonetheless, this condition should still be maintained along continuous paths in order to prevent unintended jumps across blocked intervals (e. g., singularities). Should ψ_{t-1} not be in a feasible interval (e. g., because the current pose and the goal pose are too far apart), then as a fallback solution, the optimization is ignored, and the arm angle is set to the middle of the closest feasible range.

It may also occur that, even when only sparse waypoints are available, the previous arm angle still lies within a feasible range. However, f_v and f_a reach large values that exceed the specified joint velocity and acceleration limits due to the considerable distance between the target poses. In this case, these two objectives cannot be trusted to deliver optimal arm angles, and only f_l is used to find the optimal arm angle.

3.1.3 Closed-Form Solution

To find a minimal solution to the multi-objective cost function in Eqn. (3.3), the individual objective functions (3.5), (3.6), and (3.7) are first minimized separately. The overall optimal arm angle can then be expressed using the terms of the individual solutions for the objective functions.

The reachability objective $f_l(\psi)$ in Eqn. (3.5) has the trivial optimal solution

$$\psi_t^l = \frac{\psi_m^u + \psi_m^l}{2} \,. \tag{3.8}$$

To obtain a closed-form solution in the discrete-time domain for $f_v(\psi)$ and $f_a(\psi)$, one must determine an expression for the joint angle variation $\Delta\theta_i(\psi)$ between two time steps that can be parameterized by the arm angle.

Using Eqn. (3.1) and with the help of finite differences, the joint angle difference can be locally linearized with respect to the arm angle:

$$\begin{aligned} \Delta\theta_i(\psi) &= \Delta_{\boldsymbol{P}}\theta_i + \underbrace{\left.\frac{\partial\theta_i}{\partial\psi}\right|_{\boldsymbol{P}_t,\psi_{t-1}}}_{:=\partial_\psi\theta_i} (\psi - \psi_{t-1}) \;, \\ \Delta_{\boldsymbol{P}}\theta_i &= \theta_{i,\boldsymbol{P}_t,\psi_{t-1}} - \theta_{i,\boldsymbol{P}_{t-1},\psi_{t-1}} \,. \end{aligned} \tag{3.9}$$

This approximation is sufficiently accurate down to the acceleration level as long as Δt and $\Delta\boldsymbol{P}$ are small. This assumption holds if the redundancy resolution approach is used to determine the target joint angles within a single control cycle or if the Cartesian goal path $\boldsymbol{P}(t)$ is sampled in small steps. If the optimal arm angle ψ_t needs to be computed with higher precision, finite backwards differences with a higher-order accuracy can be easily incorporated. The derivative $\partial_\psi\theta_i$ exists for all joints, except for the fourth (see Section 2.1.2). The fourth joint does not influence the arm angle, and thus $\partial_\psi\theta_4 = 0$.

The optimal arm angle with respect to velocity ψ_t^v is the solution of $\mathrm{d}f_v/\mathrm{d}\psi = 0$. We obtain

$$\begin{aligned} \psi_t^v &= -\frac{\sum_{i=1}^{7} \left[a_{v,i}\, \partial_\psi\theta_i \left(\Delta_{\boldsymbol{P}}\theta_i - \partial_\psi\theta_i\, \psi_{t-1}\right)\right]}{\sum_{i=1}^{7} \left[a_{v,i} \left(\partial_\psi\theta_i\right)^2\right]} \;, \\ \text{with} \qquad a_{v,i} &= \frac{1}{7\left(\dot{\theta}_{i,max}\right)^2} \;, \end{aligned} \tag{3.10}$$

using Eqs. (3.6) and (3.9). And the optimal arm angle from the acceleration objective f_a is

$$\begin{aligned} \psi_t^a &= -\frac{\sum_{i=1}^{7} \left[a_{a,i}\, \partial_\psi\theta_i \left(\Delta_{\boldsymbol{P}}\theta_i - \partial_\psi\theta_i\, \psi_{t-1} - \Delta\theta_{i,t-1}\right)\right]}{\sum_{i=1}^{7} \left[a_{a,i} \left(\partial_\psi\theta_i\right)^2\right]} \;, \\ \text{with } a_{a,i} &= \frac{1}{7\left(\ddot{\theta}_{i,max}\right)^2}, \; \Delta\theta_{i,t-1} = \theta_{i,t-1} - \theta_{i,t-2} \,. \end{aligned} \tag{3.11}$$

Finally, the overall cost function f in Eqn. (3.3) is solved with $\mathrm{d}f/\mathrm{d}\psi = 0$ and the optimal arm angle ψ_t is

$$\begin{aligned} \psi_t = \frac{\tilde{w}_v \psi_{t,n}^v + \tilde{w}_a \psi_{t,n}^a + \tilde{w}_l \psi_t^l}{\tilde{w}_v \psi_{t,d}^v + \tilde{w}_a \psi_{t,d}^a + \tilde{w}_l}, \text{ with } \tilde{w}_v = \frac{w_v}{(\Delta t)^2}, \\ \tilde{w}_a = \frac{w_a}{(\Delta t)^4}, \; \tilde{w}_l = \frac{4\, w_l}{\left(\psi_m^u - \psi_m^l\right)^2}. \end{aligned} \tag{3.12}$$

The indices n and d denote the numerators and denominators of ψ_t^v and ψ_t^a in Eqs. (3.10) and (3.11). The weights w_v and w_a are scaled with Δt, so the choice of Δt has a significant influence on the weighting of f_v and f_a.

Δt can be chosen as the desired execution time for the given position delta if this value is known when calling the IK. However, Δt is typically not available during IK computation because trajectory generation is performed in two stages: First, the Cartesian path is sampled using the IK and then a trajectory is obtained via a path parameterization algorithm. In such a planning pipeline, Δt serves as the sole tuning parameter in the optimization. If set to a small value, the objectives f_v and f_a are assigned a greater weight compared to f_l. It should not be chosen too small, otherwise f_v and f_a are no longer guaranteed to be normalized within the interval from 0 to 1. The repulsive effect from the joint limits will only become substantial when the arm angle is already very close to the boundaries. If set to a larger value, the heavier weight of f_l can cause substantial changes in the arm angle. As a general rule, Δt should be set to the maximum time difference between two sampling steps that the robot is expected to take.

In general, there are two options for the selection of Δt. If a Cartesian trajectory parameterization is used, then Δt can be set to the desired execution time for the time step. If a high Cartesian end-effector velocity is demanded and thus a small Δt, then f_v and f_a are automatically assigned a greater weight than f_l.

3.2 Implementation

The algorithm was implemented in C++ as a standalone library with Boost and Eigen as dependencies[1]. We refrained from an implementation that runs on the graphics processing unit (GPU) to ensure portability, as no benefits were expected for the intended use case. As Huber and Wollherr (2019a) show, a GPU-based implementation is only faster when a large set of poses is evaluated in parallel. Instead, our implementation is meant to be used to sequentially sample a path where the solution for the current position depends on the solution for the previous position.

Building on this library, we additionally offer a plugin for the *MoveIt* motion planning framework (Sucan and Chitta, 2026), which can be used as a drop-in replacement for the standard kinematics solver. The plugin integrates with MoveIt's collision avoidance. A readme is provided with instructions on how to use the library and its interface either directly in a C++ project, a ROS project, or how to replace the default MoveIt kinematics plugin with our implementation, and some practical tips on how to set the needed parameters and determine a good value for the tuning parameter. Both the library and the plugin

[1] Source code, examples, and documentation are available at https://github.com/KITrobotics/rll_sdk/tree/master/kinematics.

are part of the SDK for the KUKA Robot Learning Lab at KIT (Wiedmeyer et al., 2019), the remotely accessible robotics development platform described in Chapter 5. The motion planning interface of the lab leverages the MoveIt plugin. It enables users to plan point-to-point (PTP) motions or to follow Cartesian paths, and also provides the ability to define a preferred arm angle at the target pose.

The kinematics implementation includes the forward kinematics with arm angle computation and the complete IK solution with both the redundancy resolution from Faria et al. (2018) (see Section 2.1) and our approach (see Section 3.1). The library has an easy-to-use API that only requires a goal pose and joint angles from previous states as a seed state. Beyond returning various error codes, the library can also issue warnings when a solution is close to a kinematic singularity. Parameters and IK solving strategies can be modified during runtime. 7-DoF manipulators without joint offsets are supported.

Additionally, three different control modes for the global configuration can be specified (see Section 2.1.1):

- KEEP: A solution is only searched within the current global configuration. This is the preferred mode for path following.
- CLOSEST: If the seed state is near multiple global configurations, the algorithm searches for solutions within each of those configurations and returns them in order of their proximity to the seed state. If no solution is found in this subset, the search is extended to all global configurations.
- ALL: Same as above, except all global configurations are searched by default.

Computing solutions in several global configurations is beneficial for PTP motions, and it allows covering the complete solution space. In addition, a solution in a different global configuration is often closer to the seed state than a solution from the closest feasible arm angle interval in the current global configuration. The coefficients used in the mapping formulas between joint space and null-space are computed in advance and then reused to determine solutions across different global configurations, thereby improving the algorithm's efficiency.

3.3 Experiments

In the following, the runtime performance of the IK implementation is assessed, and the previously described redundancy resolution methods are compared based on two Cartesian path-following tasks. A KUKA LBR iiwa 7 R800 lightweight robot (KUKA AG, 2026) is employed for experimental validation. The robot's limb lengths, as well as its joint position and velocity limits, are defined by the manufacturer. Since limits on joint accelerations are not given, we adopt the following symmetric bounds:

$$\ddot{\boldsymbol{\theta}}_{\max} = (5.44\ 5.44\ 5.56\ 7.22\ 7.78\ 10.0\ 10.0)\ \left[\mathrm{rad/s^2}\right] = -\ddot{\boldsymbol{\theta}}_{\min} \tag{3.13}$$

As a safety offset, the joint position limits are reduced by $1°$. Velocity and acceleration limits are set to 40% of their maximum values.

3.3.1 Runtime Performance

To cover all potential corner cases, every redundancy resolution strategy and global configuration control mode in the implementation were evaluated using one billion random and feasible seed states and goal poses. In every instance, the library succeeded in producing a valid solution that respects the joint limits.

Table 3.1: Computation times [µs] of the full IK algorithm for different combinations of global configuration modes and redundancy resolution methods. Adapted from Wiedmeyer et al. (2021).

Global configuration	Redundancy Resolution	Mean	Std. Deviation	Maximum
KEEP	Faria et al. (2018)	2.754	0.369	6.079
KEEP	multi-objective	3.119	0.391	6.155
CLOSEST	multi-objective	3.569	1.468	27.57
ALL	multi-objective	20.89	2.997	37.06

Computation times were evaluated using a set of 100 000 randomly generated, feasible input seed states and goal poses. All experiments were performed on a machine running Ubuntu 18.04[2] (Intel i7-7700K at 4.20GHz, 32GB RAM). The results are listed in Table 3.1. The reported computation times refer to the complete IK algorithm, which includes projecting joint limits and singularities into the null space, as well as computing the feasible intervals of the arm angle (see Section 2.1.2). Our multi-objective solution in Eqn. (3.12) is only slightly slower than the single-objective approach from Faria et al. (2018) in Eqn. (2.3), and the global configuration mode CLOSEST is more efficient than ALL.

3.3.2 Null-Space Motion

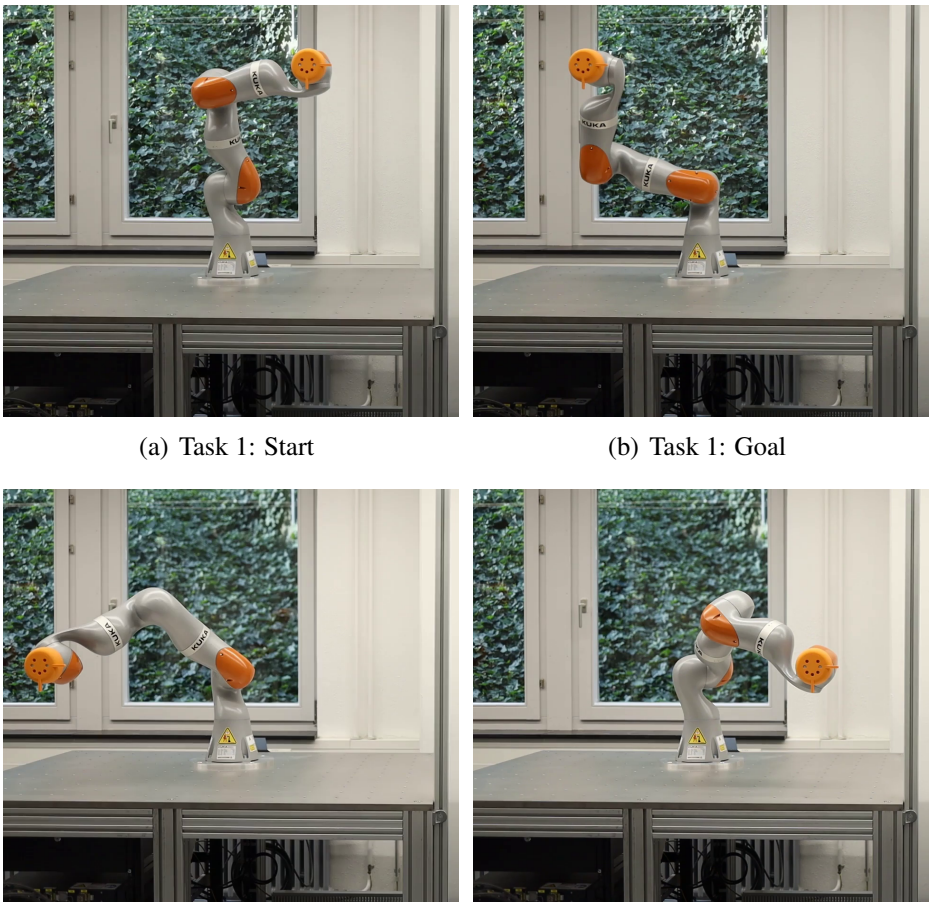
Two Cartesian path-following tasks are chosen to analyze the impact of the redundancy resolution methods. Both tasks are specified at the end-effector flange, and the end-effector orientation remains constant along the path. The paths are discretized with a sampling interval of $\Delta s = 0.001$ m. After determining the joint angles with the IK for every sampled pose, the approach by Kunz and Stilman (2012) is used to generate a time-optimal trajectory with limits on joint velocities and accelerations. In our approach, Δt is therefore not chosen as the desired motion duration at each time step, but instead as a single fixed, optimized value. This corresponds to the typical use case described in Section 3.1.3.

The first task is an example of a path that requires a large arm angle change to avoid joint limits to reach the goal pose. The start configuration is

$$\boldsymbol{\theta}_{0,1} = (-0.24, 0.38, 1.66, -1.42, 1.22, -1.36, -1.75) \ [\text{rad}] \,,$$

equal to the flange position $\boldsymbol{P}_{0,1} = (0.35, 0.35, 0.78)$ [m]. The requested motion is a linear path parallel to the y-axis in the negative direction with a length of 0.68 m (see also Figure 3.1).

[2]The Linux kernel was patched with the Preempt-RT real-time patch.



(a) Task 1: Start (b) Task 1: Goal

(c) Task 2: Start (d) Task 2: Goal

Figure 3.1: The start and goal poses of the two test tasks. Task 1 performs an end-effector motion from right to left and in Task 2 the robot moves from left to right. The pictures are screenshots from the accompanying video which shows the motions for examination[3].

The parameters K and α for the redundancy resolution of Faria et al. (2018) in Eqn. (2.3) and Δt in Eqn. (3.12) for our approach were optimized in simulations using this task. The parameters that resulted in the shortest motion durations along the path were selected.

The gain K was adjusted in increments of 0.001, the parameter α in increments of 0.1, and the time step Δt in increments of 0.01 s. The optimal parameters are $K = 0.016$, $\alpha = 4.3$ for the redundancy resolution by Faria et al. (2018) and $\Delta t = 0.04$s for our approach. To optimize motion durations, we found that K typically has to be selected very small in order to restrict the maximum border push-off within a single sampling step. On the other hand, α should be selected relatively large to facilitate a continuous adaptation of the arm angle. This tuning is necessary to keep the overall border push-off along the path large enough to avoid hitting joint limits.

Figure 3.2(a) illustrates the null-space along the path and depicts the arm angle motions for the various redundancy resolution methods. The brightness of the colors represents the distances between the IK solutions in joint space, providing an impression of how far apart these solutions are. Thanks to the aggressive limit avoidance objective in Eqn. (3.5), the multi-objective solution can maintain enough distance to the borders, especially at the goal pose (at 100% path progression). The solution that includes acceleration minimization

($s_a = 1$) yields a smoother progression of the arm angle than the solution that does not incorporate the acceleration objective.

In contrast, the approach from Faria et al. (2018) has the least smooth arm angle profile, especially close to the start pose, and it manages to stay only slightly above the lower border at the goal pose. The tuning of the parameters for minimal motion times provides just enough push-off from the limits to reach the goal. If these parameters were applied to a path where the arm angle interval boundaries increase more steeply, the push-off would probably be insufficient to reach the target without passing into a blocked interval. This behavior makes it challenging to choose parameters that generally provide enough repulsion from the limits without significantly increasing the motion times and sacrificing the smoothness of the trajectory. Therefore, with their method, it is difficult to identify parameters that perform reliably across a broader variety of paths in task space. Our method however ensures sufficient push-off close to the borders, without the need to specifically tune the amount of repulsion. The dynamic weighting with w_{dis} in Eqn. (3.3) automatically makes the reachability objective the primary objective close to the borders, while still avoiding jumps in the progression of the arm angle with the help of the weighted sum with the velocity and acceleration objectives.

Table 3.2: Motion times for both test tasks using different redundancy resolution schemes. Adapted from Wiedmeyer et al. (2021).

Task 1		Task 2	
method	T [s]	method	T [s]
Faria et al. (2018)	3.563	$\psi = \text{const.}$	3.313
multi-obj. $s_a = 1$	2.666	Faria et al. (2018)	3.936
multi-obj. $s_a = 0$	2.758	multi-obj.	3.035

The motion times T are listed in Table 3.2. The optimization without acceleration objective leads to a 22.6% faster execution time than Faria et al. (2018). Adding the acceleration minimization makes the optimization 25.2% faster than the resolution from Faria et al. (2018). The end-effector velocities resulting from the different methods are shown in Figure 3.3(a). Only the multi-objective optimization that includes the acceleration objective shows a monotonically increasing profile during the acceleration phase, with only a minor decline just before the deceleration phase begins. Moreover, the acceleration objective smooths the velocity profile and avoids the sharp edges observed in the other two profiles. The method proposed by Faria et al. (2018) cannot produce a null-space motion that increases the end-effector velocity near the end of the trajectory in the same manner as the multi-objective optimization approach.

The second task illustrates paths for which a wide range of arm angles is possible, and the target pose can still be reached without adjusting the arm angle along the path. The robot starts in the configuration $\boldsymbol{\theta}_{0,2} = (-0.86, 0.91, 0.0, -1.21, -1.15, -0.99, 0.90)$ [rad], which is the flange position $\boldsymbol{P}_{0,2} = (0.55, -0.50, 0.38)$ [m]. The requested motion is a linear path parallel to the y-axis in the positive direction with a length of 1.0 m (see Figure 3.1). The tuning parameters that were optimized for Task 1 are reused. Typically, the tuning parameters are optimized only once using a path that requires a significant change in arm angle to reach the goal (as seen in Task 1) to ensure sufficient reachability for a particular setup and application. Then the same parameters are used during operation of the robot,

as repeatedly optimizing them for different requested paths during runtime would be too time-consuming and not suitable for online planning. Using the same set of parameters, we examine how well they transfer to a task that features a different null-space structure.

In the null-space of task 2, a blocked interval initially reaches into the upper area at the start of the path and later extends into the lower area at the end of the path (see Figure 3.2(b)). This path property causes deviations and sudden drop-offs of the end-effector velocity for the Faria et al. (2018) solution due to the limit change at the beginning and end of the trajectory (see Figure 3.3(b)), which leads to a jerky movement of the end-effector. Such jerky motions due to significant arm angle changes in short time periods would also be undesirable in human-robot collaboration scenarios as they happen unexpected and cannot be anticipated. In contrast, the multi-objective resolution delivers smooth results. Concluding from Table 3.2, the multi-objective redundancy resolution is 22.9% faster than the one from Faria et al. (2018). To demonstrate that the multi-objective resolution extends beyond merely maintaining a constant arm angle, the redundancy resolution with a constant arm angle is also examined. In particular near the end of the path, the multi-objective solution identifies arm angles that raise the end-effector velocity, leading to an 8.39% reduction in execution time compared to using a fixed arm angle. This demonstrates that our redundancy resolution method can yield improved performance even in situations where adjusting the arm angle is not required to avoid joint limits.

In addition to the two test cases presented, various other paths were evaluated, and similar improvements could be observed. Two of the additional test cases, a triangle movement and a path following task parallel to the x-axis, are shown in the accompanying video of this chapter[3]. For very few paths, where the feasible arm angle interval was narrow, the multi-objective scheme was less than 10% slower than the method from Faria et al. (2018). However, a larger overall distance was achieved from the boundaries of the arm angle interval. The multi-objective approach was faster in almost all test cases. Especially when a sudden change in the arm angle borders leads to an unnecessarily large repulsion with the Faria et al. (2018) approach, the multi-objective scheme achieved motion times that were up to 58% shorter by maintaining a smoother profile of the arm angle along the path. Finally, every test case was successfully executed with the KUKA iiwa. The generated trajectories were processed by a custom ROS trajectory following controller that uses the Reflexxes Type IV motion library (Kröger, 2011) to enforce jerk limits[4]. The controller utilizes our ROS hardware interface to send position commands to the robot via the real-time-capable KUKA FRI network interface at a rate of 1 kHz.

The linked video[3] presents the robot trajectories for four test scenarios, each executed with both redundancy resolution strategies. In addition to the two experiments discussed above, it also includes two additional cases: a triangular motion and a path-tracking task parallel to the x-axis. The side-by-side comparison in the video highlights that our method achieves shorter motion times. Observing the movements of the end-effector and elbow, the smoother arm motion of our method becomes evident compared to the more jerky motions of the Faria et al. (2018) solution.

We further observed that, since both redundancy resolution methods operate locally and lack information about the structure of the null-space at upcoming poses, they have diffi-

[3]The video is available at https://youtu.be/XQ8eCKQv8cA. Alternatively, a copy of the video can be found at https://rll.ipr.iar.kit.edu/videos/ik.mp4 .

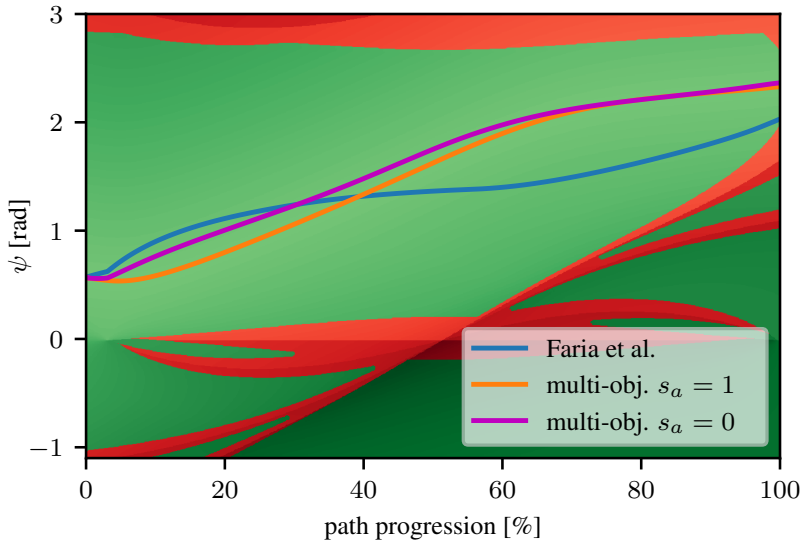
[4]For better trajectory tracking with the jerk-limited controller, the acceleration limits are scaled with the factor 0.16. Thus, execution times differ from the initial trajectory generation in Table 3.2.

culty handling paths where a new null-space boundary suddenly emerges near the current arm angle. Suppose that the arm angle motion of one of the redundancy resolution methods was accidentally moving towards such an emerging border. In that case, it might be possible that a valid solution is not found. For these invalid solutions, the prior arm angle falls within a blocked interval at the subsequent pose, so the redundancy resolution must choose an arm angle from another feasible interval. This case leads to a jump in the joint angles and prevents continuous path following. The other redundancy resolution method may be moving away from a new upcoming limit, which would return a valid solution, but this would be coincidental rather than intentional.

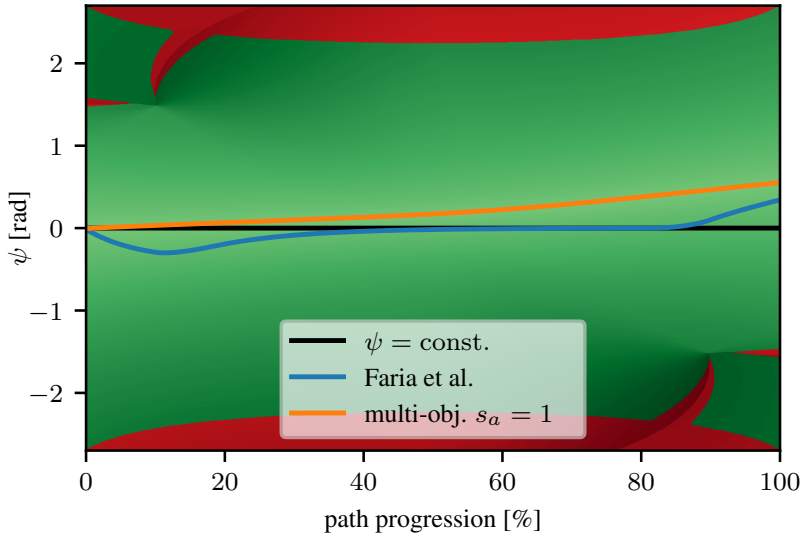
3.4 Summary

The chapter presented a redundancy resolution scheme as part of a fully analytic and deterministic IK solution that enables faster motion times along given Cartesian paths compared to the existing closed-form solution from the literature. Using a KUKA iiwa, this was examined by analyzing the arm's motion in the null-space. The multi-objective optimization strategy for redundancy resolution enables an improved time parameterization of continuous Cartesian paths, while simultaneously maintaining high reachability by avoiding joint limits. By incorporating information about changes in joint angles into the redundancy resolution scheme, joint velocities and accelerations are optimized, resulting in higher end-effector velocities and smoother trajectories. With the help of dynamic weights and a single tuning parameter, an overall optimal solution is computed for a given pose.

The runtime analysis shows low computation times, making the complete IK algorithm well-suited for integration into a real-time online motion planning controller or for rapid sampling of Cartesian paths. The redundancy resolution method is independent of the trajectory parameterization algorithm used, so it can easily be integrated into an existing motion planning pipeline. To further enable practical use of the complete IK algorithm, a free and open-source software implementation is provided.

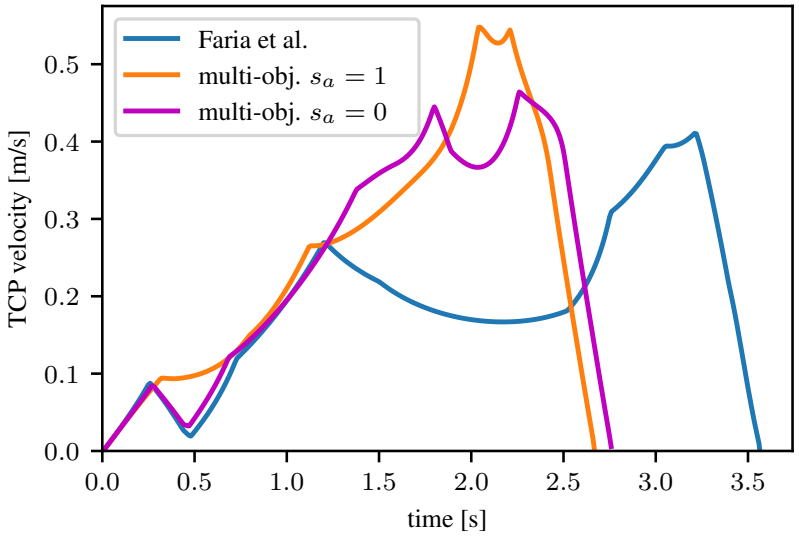


(a) Task 1

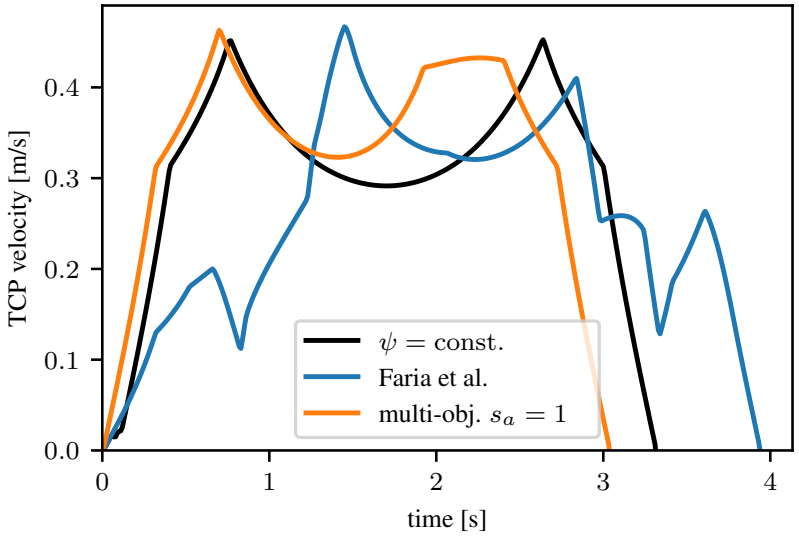


(b) Task 2

Figure 3.2: Null-space and arm angle motions therein along both test task paths using several redundancy resolution approaches. The robot moves from left to right, beginning at 0% path progression and ending at 100% path progression. Green areas represent the space of feasible arm angles, and blocked intervals are marked red. The color brightness $S_t(\psi)$ visualizes the distance of the IK solution $\boldsymbol{\theta}(\psi)$ from the solution $\boldsymbol{\theta}(\psi_t)$ belonging to the multi-objective redundancy resolution with $s_a = 1$. The distance is determined by summing up the individual joint angle differences ($S_t(\psi) = \sum_{i=1}^{7} |\theta_i(\psi) - \theta_i(\psi_t)|$). The darker the color, the greater the distance. The figure is adapted from Wiedmeyer et al. (2021).



(a) Task 1



(b) Task 2

Figure 3.3: End-effector (flange) velocities for the two test tasks when applying different redundancy resolution methods. The figure is adapted from Wiedmeyer et al. (2021).

CHAPTER 4

Real-Time Time-Optimal Path Parameterization

Offline trajectory generation methods have been developed that can follow a given path. However, while some methods are fast, they are not suitable for real-time applications with strict limits on computation time and required reliability. These offline methods can respect the given kinematic and dynamic constraints. Minimizing the trajectory's execution time is typically the most important optimization objective, and this work focuses on it. The class of methods with this objective is referred to as *Time-Optimal Path Parameterization* (TOPP). This chapter details the development of a real-time-capable TOPP algorithm called RT-TOPP.

Section 4.1 gives an algorithm overview. The algorithm takes a set of waypoints as input and maps them to a generalized, differentiable, one-dimensional path coordinate in the phase plane. Section 4.2 shows how the constraints of the joint space are imposed to derive an analytical maximum velocity curve (MVC), which defines the maximum possible velocity that keeps the robot on the path and within the joint constraints. The time-optimal trajectory expressed in the one-dimensional coordinate is computed using the MVC and two integration passes. Eventually, it is mapped back to the joint space, as described in Section 4.3. Special care is taken in the integration process to ensure robustness, even in corner cases, and to ensure that kinematic constraints are respected exactly. Finally, Sections 4.4 and 4.5 present the open-source implementation and an extensive evaluation using randomly generated test paths.

4.1 Algorithm Overview

The RT-TOPP algorithm takes a set of waypoints in joint space as input and produces a time-optimal trajectory that follows the path given by the waypoints and respects specified joint space constraints. To retain real-time capability with short cycle times, only a fixed maximum path length can be used for trajectory generation in each control cycle of the robot controller. The algorithm design supports this use case. Figure 4.1 shows the complete input and output parameters. For seven-DoF manipulators without joint offsets, waypoints given in Cartesian space can be converted to joint space using the second

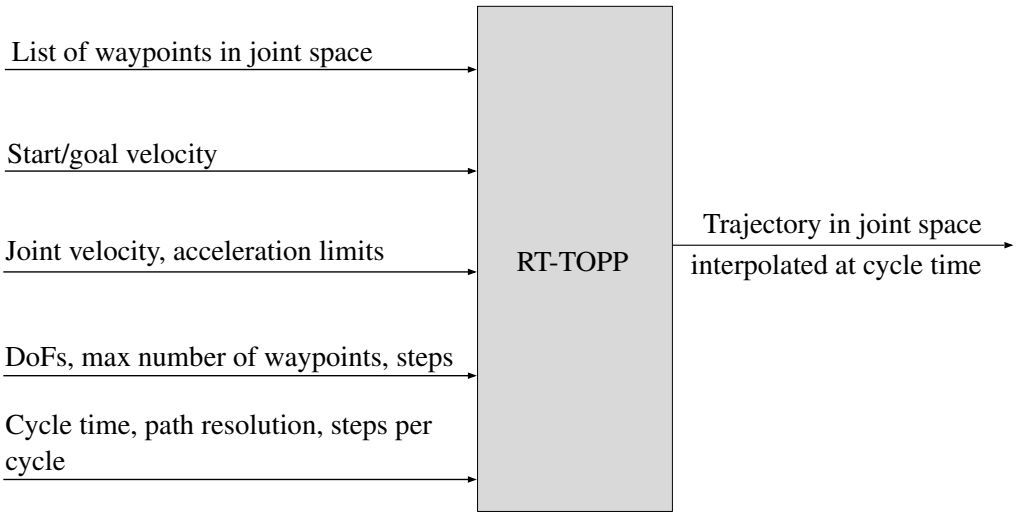


Figure 4.1: The input and output parameters of the path parameterization algorithm RT-TOPP.

contribution of this thesis, the closed-form Inverse Kinematics (see Chapter 3). Existing closed-form IK implementations, such as IKFast (Diankov, 2010), can retain real-time capability for non-redundant manipulators.

The algorithm supports a specified velocity for the first waypoint, which corresponds to the robot's current state. A goal velocity can also be selected for the final waypoint, and the algorithm attempts to reach it if it can be achieved while staying on the path.

The path resolution is the parameter that determines the distance between the grid points, which are generated when the path is discretized. The product of path resolution and integration or path scanning steps gives the path length. The cycle time and steps per cycle should determine how much time the algorithm has between control cycles of the robot controller and how many integration steps can be executed in this time. The evaluation in Section 4.5.2 helps determine the steps per cycle by benchmarking on the target platform. Since the real-time-capable implementation of the algorithm does not support dynamic memory allocation, the maximum number of waypoints that can occur must be specified. Similarly, the maximum path length that can be processed in one run is limited by the path resolution and the maximum number of scanning steps along the path.

An approach in the category of numerical integration (NI) and dynamic programming (DP) was selected as the base for the underlying algorithm. Approaches based on Convex Optimization (CO), that solve a single large optimization problem, are unsuitable for a real-time algorithm. Their computation times are significantly slower than those of their DP- or NI-based counterparts, making them generally less suitable candidates for online planning. And it is impossible to reliably predict computation times for a given path and number of degrees of freedom and constraints. Imposing an upper barrier for computation time does not seem possible without sacrificing optimality. Fast algorithms that solve small LP problems could be adopted for a real-time solution. Still, we refrained from that as it appears that these algorithms have less prospect to be extendable to third-order constraints while retaining efficient computations. This leaves us with the DP- or NI-based solutions requiring the fewest calculations per discretized path grid point.

Figure 4.2 shows an exemplary parameterization of our algorithm in the phase plane with path position s on the x-axis and path velocity $\dot{s}$ on the y-axis. The backward parameter-

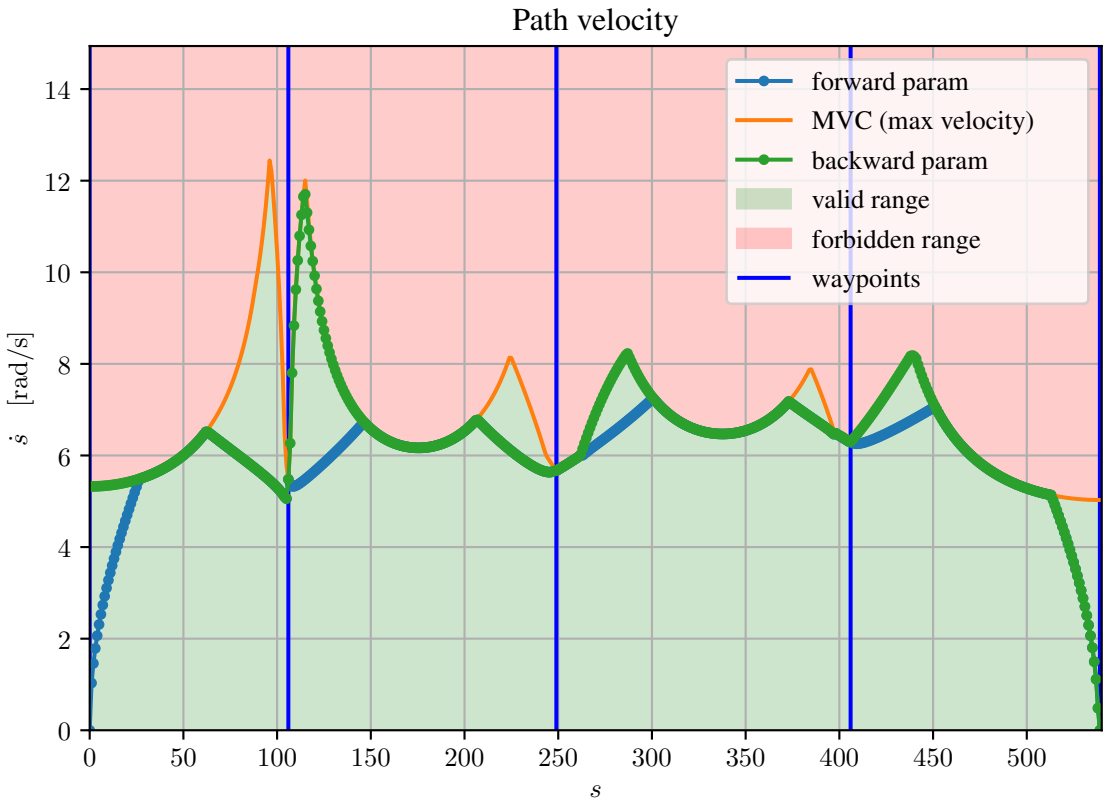


Figure 4.2: The Maximum Velocity Curve (MVC) and a complete time parameterization with forward and backward scan of our algorithm. The MVC separates the valid path velocity range from the invalid range for the path velocity, where joint velocity or acceleration constraint violations would occur. The backward parameterization is executed first, starting at the goal position, and defines the upper limit for the forward parameterization starting at the initial path position 0. Vertical blue lines mark the positions of the initially commanded waypoints along the continuous path.

ization is the first path scan. It accelerates backwards to the MVC, moves along it where possible, and jumps to lower MVC values where otherwise the forbidden range would be entered. The forward parameterization is the second path scan and leverages the backward scan as an upper limit curve to produce the final path parameterization.

More specifically, the basis for our algorithm is the combination and extension of the works by Barnett and Gosselin (2020) and Kunz and Stilman (2012). This approach ensures that only two integration passes along the path are needed, and no additional local search for acceleration switching points along the path is required. Switching points are identified directly during the path scanning process. The path-scanning process is modified to precisely meet acceleration constraints and make the algorithm more robust. Further improvements allow for upper barriers on integration steps and subsequently impose real-time constraints. Analytical expressions for the MVC allow for the determination of (singular) switching points in a fast and precise manner. In contrast to the derivations of Kunz and Stilman (2012), asymmetric acceleration limits are supported. This allows specifying acceleration limits computed from a dynamics model of the robot and given torque limits. A preliminary version of the MVC and the core integration process was de-

veloped in the supervised student thesis by do Nascimento (2021), and later improved and extended to meet real-time requirements and to pass all tests conducted in the evaluation in Section 4.5.

The analytical MVC replaces the bisection search used by Barnett and Gosselin (2020). The bisection search accounts for dynamic constraints like torque, but it is not real-time-capable, as it only aborts when the convergence criterion is reached; thus, the number of search steps is not limited, nor is an exact solution returned. Minor errors in MVC computations in the phase plane can lead to significant errors in joint space, especially concerning acceleration limit violations. Such a search algorithm for the MVC also has the risk that an area above the MVC is wrongly identified as a valid range. However, the two-pass integration scan cannot reach such an isolated area, and it does not fall within the overall valid range for first- and second-order constraints. Using an analytical MVC with an exact calculation of the acceleration at the switching point further helps mitigate these issues. Ideally, the algorithm should converge to the time-optimal solution when the integration step size is minimized. The backward integration along the path is executed first, then sampling is implemented as a forward integration along the path. When the backward integration step size is increased, limit violations can occur when sampling occurs in smaller steps between integration steps. The backward integration step size becomes a tuning parameter to weigh the accuracy of the time-optimal solution against the algorithm's runtime.

The necessary pre-processing to achieve an efficient and suitable one-dimensional path representation from a set of waypoints is described in Section 4.1.1. It was developed in collaboration with Xi Huang. More details on the real-time capability and requirements, such as the completeness of the algorithm, are given in Section 4.1.2.

4.1.1 Waypoints Pre-Processing

Our TOPP approach requires a pre-processing of the waypoints to achieve a continuous path representation that is differentiable and can be expressed with a single path variable. Cubic splines with sites defined at the input waypoints are used. These allow for a smooth C^2-continuous path representation. Furthermore, with the applied formulation of the spline interpolation, we can solve for the partial derivatives at an arbitrary point as a grid point between the waypoints and use them to derive the velocity and acceleration constraints in the phase plane or map back from the phase plane to joint space at such a grid point. Details on the maximum velocity constraints forming the MVC are given in Section 4.2.

Although formulating the path variable s as the arc length is intuitive, it is advantageous to formulate it with joint displacements. This formulation can correctly deal with the cases where the robot adjusts its orientation or a redundant robot executes a null-space motion, i. e., when the end-effector does not exhibit significant displacement. The one-dimensional path variable is formulated with the expression

$$\mathrm{d}s = \|\mathrm{d}\boldsymbol{q}\|, \tag{4.1}$$

where $\boldsymbol{q}$ refers to the robot configuration in joint space. The key idea of the formulation is to encode the robot's movement in a one-dimensional coordinate s. Given a value of s and its derivatives, we can retrieve the corresponding joint position and its derivatives. Using

the path variable as the base coordinate, we interpolate the input waypoints with a spline. In the numerical integration, the partial derivatives of the spline, e. g., $\boldsymbol{q}'(s) = \frac{\partial \boldsymbol{q}(s)}{\partial s}$ in the joint space, allow us to determine the constraints on the path velocity and acceleration given by the joint space kinematic constraints along the path.

The interpolated spline is finely discretized with a fixed path resolution to have approximately the same distance between grid points. Usually in related work, a fixed number of grid points is used in spline interpolation. However, this more straightforward approach cannot be applied to arbitrary path lengths during online planning. It would lead to significant variance in the distance between grid points, and the resulting discretization resolution would not be guaranteed to suffice for sampling, as path velocities may vary significantly between grid points. During path scans, constraints and velocities are computed only at the grid points and are assumed to be constant between them. A sufficiently small path resolution allows the final trajectory to be sampled with sufficient precision. The post-processing is discussed further in Section 4.3.4. First, a basic path projection from the given waypoints enables determining the basic spline coefficients and performing a basic interpolation with a refinement of the coefficients. L2 distance is used to check the distance between grid points, and a second and final interpolation to correct the initial guess, where needed, concludes the pre-processing. If a waypoint is between two grid points, the grid points are rearranged to allow a slightly different resolution around the waypoint. The procedure ensures that the initially specified waypoints are still included in the final interpolation.

Two types of boundary conditions are used for the splines. If the start or goal velocity is zero, a natural spline is applied where the second-order partial derivatives $\boldsymbol{q}''(s) = 0$ at the start or goal waypoint. The second boundary condition variant considers the specified velocity vector at the start and the goal pose. The first-order derivative is set to the normalized velocity vector at the start and goal, ensuring that the spline points in the movement direction at both. This keeps the path projection formulation valid for the online planning case. It also reduces the risk of introducing a sharp path corner near the start or goal during pre-processing, which may lead to an infeasible parameterization problem with the specified start/goal velocities.

4.1.2 Enabling Real-Time Capabilities

This section discusses the design choices made to enable real-time capability to the TOPP algorithm. At the algorithm's core, a two-pass integration along the path is performed, ensuring that, for full parameterization, each grid point is passed twice and no further iterative optimization steps are needed. For forward integration, a single step for sampling is sufficient, or a full scan is executed in the same way as the initial backward scan to precompute a complete trajectory. To stay within real-time constraints, only a part of the given path may be optimized to produce a locally time-optimal trajectory for the path segment. This procedure is necessary when the full path provided by the commanded waypoints cannot be parameterized in the given cycle time t_{cycle}. From a given t_{cycle} as an upper limit, the maximum allowed number of integration steps along the path must be derived.

This can be accomplished by providing users with a benchmark from which the worst-case execution time t_{step} for one integration step, including the pre-processing, on a given

system can be determined. See the evaluation in Section 4.5.2 for an example of such a benchmark. From this benchmark, the maximum trajectory length or the maximum number of integration steps n_{steps} can be derived for the target system:

$$n_{steps} = \frac{t_{cycle}}{2t_{step}}, \tag{4.2}$$

if a full parameterization is executed. In this way, the segment of the path to be optimized can be restricted, and an upper limit for the algorithm computation time can be enforced. Usually, time integration is used for NI- or DP-based approaches, which would make this approach impossible. The backward scan could not be done first to establish a safe upper velocity limit curve, which is needed to ensure that the robot does not accelerate from the current state into a region from which it cannot decelerate anymore on the path. It needs to be stepped locally instead of time integration along the path backwards from a maximum path position for the backward scan. To make this feasible, we borrow a reformulation of the optimization problem from CO-based methods, which is further detailed in Section 4.3.

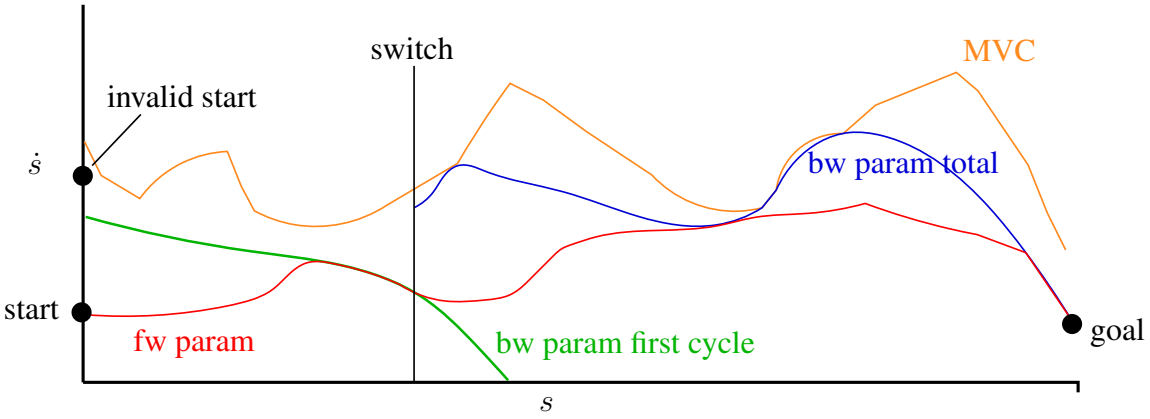


Figure 4.3: Concept for the path scanning process across multiple control cycles. In the first control cycle, a backward parameterization (green) is executed from the maximum path position given by n_{steps} and starting with zero velocity. In subsequent control cycles, the global backward parameterization (blue) is incrementally computed starting from the goal state. The forward parameterization (red) first uses the green backward parameterization as an upper limit. It can smoothly switch to the total backward param as soon as it is available for the current path position.

Executing the backward scan first also allows for only one sampling step in the forward integration, which is sufficient for an application in which the whole trajectory does not need to be known beforehand. Only the next sampling step is needed to command the position and velocity for the next control cycle to a joint controller. This allows for further reduction of computation time compared to existing offline methods. Figure 4.3 illustrates a possible way how the parameterization could work in the first control cycle and subsequent control cycles without having to repeatedly optimize the same path segments, which would heavily reduce the algorithm efficiency. This scenario assumes that the commanded waypoints do not change across control cycles. If they do change, the parameterization restarts with the first control cycle. If the total backward param is not yet available for sampling the next control cycle, the forward param moves along the bw param from the

first cycle if the forward param has already accelerated to the "bw param first cycle". In the worst case, the robot brakes on the path until it can accelerate to the global backward parameterization. This ensures that the robot never reaches an unsafe state along the path from which it cannot recover.

Another requirement for real-time capability is determinism. Given the same set of input parameters to the algorithm (see Figure 4.1), the results are reproducible, meaning the computed trajectory is always the same. The algorithm has no components that introduce randomness in the results, and we focus on numerical robustness. The completeness of the algorithm also needs to be ensured. The TOPP problem can be physically infeasible when the start state velocity is too high to keep the robot on the requested path. In this case, it can be switched to a time-optimal braking trajectory that is not path-constrained. The robot brakes as fast as possible until a velocity is reached at which the parameterization problem is feasible again. This braking trajectory can be computed in parallel using Reflexxes (Kröger, 2011) in the joint space using the velocity interface with zero goal velocity for all joints. The algorithm is still able to reach the maximum feasible velocity at the goal state if the goal state velocity was specified too high to be reached by the forward scan, but is below the MVC. This is useful for applications where the goal velocity does not need to be reached exactly. The pre-processing fits a spline to the start and goal states, which points in the direction of the velocity vectors at start and goal (if the velocities are non-zero). This allows us to evaluate $\dot{s} = \|\dot{\boldsymbol{q}}\|$, where $\dot{\boldsymbol{q}}$ is the given joint state at start or goal.

To be complete, the TOPP algorithm is only allowed to fail in one of the known infeasible cases. All allowed abort criteria that characterize the TOPP problem as infeasible are as follows:

1. Start or goal state velocity is above the MVC when projected onto the path. The algorithm aborts immediately, either in the backward or forward scan, as the velocities are physically infeasible under the given constraints. The switch to a path-independent braking trajectory is needed.

2. A minimum $\dot{s}$ is enforced (slightly above zero to cope with numerical integration errors). If one integration pass stalls at this minimum value for two steps in s, the problem is infeasible. This seems to happen at very sharp path edges. It can be completely avoided by the pre-processing enforcing C^2-continuity with sufficient path smoothness and always having a displacement in at least one DoF in every grid point. Our testing confirms this.

3. Forward or backward integration ends below the start or goal state, but the start and goal velocity are below the MVC. When the backward scan ends below the start velocity, the start velocity is above the safe zone, and the switch to the path-independent braking trajectory is needed, as in the first abort criteria. When the forward scan ends below the goal velocity, the algorithm is still able to reach the maximum possible velocity at the goal.

The algorithm needs to return a solution if one exists and, otherwise, it needs to return a failure in a finite time. The procedure of Shen et al. (2018) is used as a reference to verify the completeness of a TOPP algorithm. Shen et al. (2018) similarly formulate the above conditions, and prove sufficiently that they are sufficient for completeness and that their algorithm returns a solution if the above conditions are not given. They used a similar NI-based approach for torque and velocity constraints. An in-depth investigation of the

completeness of their proof and its applicability to our work is left for future work, but we also found no further failure conditions than the above in our work. The algorithm also needs to be able to correctly handle all possible cases of singularities. They usually arise when one or multiple of the first path derivatives approximate zero, i. e., at least one joint does not move between path poses. This is further discussed in Section 4.3.2 and shown by our testing in Section 4.5.

As already mentioned, a requirement for the real-time capability of the NI algorithm is that a time-optimal parameterization without constraint violation along the path is achieved by only two integration passes, and no unknown number of further optimization passes is needed. However, islands can theoretically occur in the phase plane, see Dong and Stori (2006) for a discussion. These are infeasible regions contained in feasible areas. Then, more than two passes are needed to remove constraint violations. The TOPP literature is unclear regarding the conditions for these islands to occur in practice, as they only appear rarely, except in special applications where certain velocity ranges should be avoided. Cartesian kinematic or other complex constraints could be a source for islands. It needs to be investigated if islands can appear for a manipulator when using splines with C^2-continuity, no complex constraints are present, and our analytical MVC derivation is used. So far in our comprehensive testing, no island did occur. Enforcing C^2-continuity for paths, considering only joint space constraints, and our analytical and robust MVC derivation apparently help to mitigate this issue.

4.2 Analytical Derivation of Path Constraints: Maximum Velocity Curve

As already mentioned, the algorithm requires a Maximum Velocity Curve (MVC) as an upper limit for the path velocity in phase space. The MVC ensures, during the integration scan along the path, that only velocities are considered for which no constraints are violated and valid acceleration ranges exist. The MVC represents the absolute "static" maximum velocity for every path position and does not depend on time or previous or next states around a given path position; it only depends on the given constraints and the path shape. The following sections detail the analytical derivation of the MVC when arbitrary, and thus also asymmetric joint velocity and acceleration limits are given. The derivations also hold for symmetric limits as a special case of arbitrary limits. However, if the given acceleration limits are known to be symmetric, directly using the simplified derivations for symmetric limits by Kunz and Stilman (2012) is significantly computationally more efficient.

This section outlines the completely analytical expressions for the constraints on the path velocity that arise from joint space constraints on velocity and acceleration. Using the partial derivatives $\boldsymbol{q}'(s) = \frac{\partial \boldsymbol{q}(s)}{\partial s}$, $\boldsymbol{q}''(s) = \frac{\partial^2 \boldsymbol{q}(s)}{\partial s^2}$, joint configuration and the path representation can be related. Joint velocity $\dot{\boldsymbol{q}}$ (first order) and acceleration $\ddot{\boldsymbol{q}}$ (second order) in terms of the path variable s follow as

$$\dot{\boldsymbol{q}}(s) = \boldsymbol{q}'(s)\dot{s}\,, \tag{4.3}$$

$$\ddot{\boldsymbol{q}}(s) = \boldsymbol{q}''(s)\dot{s}^2 + \boldsymbol{q}'(s)\ddot{s}\,. \tag{4.4}$$

The following sections describing the derivation of second-order constraints apply a similar solution process, e. g., summarizing the scheme for second-order joint constraints on the maximum path velocity:

1. Solve for minimum and maximum $\ddot{s}$.
2. $\ddot{s}_{min} \leq \ddot{s}_{max}$ must always hold, so $\ddot{s}_{max} - \ddot{s}_{min} \geq 0$.
3. Remove $\min$ and $\max$ operators: the inequality must hold for all joint combinations.
4. Solve for $\dot{s}$, using case differentiation where necessary.

In this section, the index i of q_i refers to joint $i \in \{1, ..., N\}$, where N is the number of joints. The general cases with arbitrary path derivative values are described, along with the simpler case where some partial path derivatives are equal to zero.

When considering first-order and second-order constraints, the path velocity is influenced by joint velocity limits and acceleration limits as seen in Eqs. (4.3) and (4.4). The following sections outline how the overall maximum path velocity constrained by these two types of limits can be obtained. The derivation steps by Kunz and Stilman (2012) are used as a template and extended to handle asymmetric joint constraints.

4.2.1 Joint Velocity Limits

If only joint velocity limits are considered, from Eqn. (4.3) follows

$$\begin{aligned} &\forall i \in \{1, .., N\} \\ \dot{q}_{min,i} \leq q_i'(s)\dot{s} &\leq \dot{q}_{max,i} \ . \end{aligned} \tag{4.5}$$

Eqn. (4.5) are the maximum joint velocities to keep the end-effector on the path and stay within the joint velocity limits $\dot{q}_{min,i}$ and $\dot{q}_{max,i}$. When $q_i'(s) = 0$, the path velocity $\dot{s}$ does not matter because the inequality is automatically satisfied. In all other cases, the following definitions must apply to guarantee that every joint remains within its limits:

$$\dot{q}_{absmax,i} = \begin{cases} \dot{q}_{max,i}, & \text{if } q_i'(s) > 0 \\ \dot{q}_{min,i}, & \text{if } q_i'(s) < 0 \end{cases} \tag{4.6}$$

$$\dot{s}^*_{max} = \min_i \frac{\dot{q}_{absmax,i}}{q_i'(s)} \tag{4.7}$$

We assume that $\dot{q}_{min,i} < 0$ and $\dot{q}_{max,i} > 0$ always hold. $\dot{s} \geq 0$ is also given, since returning on the path is not allowed. The general solution with asymmetric velocity constraints is then

$$\dot{s} \leq \dot{s}^*_{max} = \dot{s}^{vel}_{max} \ . \tag{4.8}$$

$\dot{s}^{vel}_{max}$ gives the first-order MVC, the MVC constructed only from velocity constraints.

4.2.2 Joint Acceleration Limits

From Eqn. (4.4), it follows that $\dot{s}$ also depends on acceleration values. Starting with the valid joint acceleration ranges

$$\begin{aligned} &\forall i \in \{1, .., N\} \\ \ddot{q}_{min,i} \leq q_i''(s)\dot{s}^2 + q_i'(s)\ddot{s} &\leq \ddot{q}_{max,i} \end{aligned} \tag{4.9}$$

As in Eqn. (4.7), minimum and maximum acceleration values must be simultaneously valid across all joints. Assuming that $q_i'(s) \neq 0$:

$$\ddot{q}_{absmin,i} = \begin{cases} \ddot{q}_{min,i}, & \text{if } q_i'(s) > 0 \\ \ddot{q}_{max,i}, & \text{if } q_i'(s) < 0 \end{cases} \tag{4.10}$$

$$\ddot{q}_{absmax,i} = \begin{cases} \ddot{q}_{max,i}, & \text{if } q_i'(s) > 0 \\ \ddot{q}_{min,i}, & \text{if } q_i'(s) < 0 \end{cases} \tag{4.11}$$

$$\ddot{s}^*_{min} = \max_i \frac{\ddot{q}_{absmin,i} - q_i''(s)\dot{s}^2}{q_i'(s)} \tag{4.12}$$

$$\ddot{s}^*_{max} = \min_i \frac{\ddot{q}_{absmax,i} - q_i''(s)\dot{s}^2}{q_i'(s)} \tag{4.13}$$

Since $\ddot{s}^*_{min} \leq \ddot{s}^*_{max}$ must always hold, further restrictions are placed on $\dot{s}$:

$$\min_i \frac{\ddot{q}_{absmax,i} - q_i''(s)\dot{s}^2}{q_i'(s)} - \max_i \frac{\ddot{q}_{absmin,i} - q_i''(s)\dot{s}^2}{q_i'(s)} \geq 0 \tag{4.14}$$

As done by Kunz and Stilman (2012), $\min$ and $\max$ functions can be eliminated by requiring that the inequality holds for all joint combinations

$$\begin{aligned} &\forall i, j \in \{1, .., N\}, i \neq j \\ &\frac{\ddot{q}_{absmax,i} - q_i''(s)\dot{s}^2}{q_i'(s)} - \frac{\ddot{q}_{absmin,j} - q_j''(s)\dot{s}^2}{q_j'(s)} \geq 0 \\ &\forall i, j \in \{1, .., N\}, i \neq j \\ &\underbrace{-\left(\frac{q_i''(s)}{q_i'(s)} - \frac{q_j''(s)}{q_j'(s)}\right)}_{a}\dot{s}^2 + \underbrace{\left(\frac{\ddot{q}_{absmax,i}}{q_i'(s)} - \frac{\ddot{q}_{absmin,j}}{q_j'(s)}\right)}_{b} \geq 0 \end{aligned} \tag{4.15}$$

Setting Eqn. (4.15) to zero, an upper bound for $\dot{s}$ can be determined. The equation then has the form of a set of parabolas. Iterating over all joint combinations, the solution follows as

$$\begin{aligned} &\forall i, j \in \{1, .., N\}, i \neq j, q_i'(s) \neq 0, q_j'(s) \neq 0 \\ &\dot{s} \leq \dot{s}^{acc,*}_{max} = \min_{i,j} \begin{cases} \sqrt{\frac{\left(\frac{\ddot{q}_{absmax,i}}{q_i'(s)} - \frac{\ddot{q}_{absmin,j}}{q_j'(s)}\right)}{\left(\frac{q_i''(s)}{q_i'(s)} - \frac{q_j''(s)}{q_j'(s)}\right)}}, & \text{if } (b \geq 0 \text{ and } a > 0) \text{ or } (b < 0 \text{ and } a < 0) \\ \infty, & \text{else .} \end{cases} \end{aligned} \tag{4.16}$$

If $q_i'(s) = 0$, Eqn. (4.9) can be simplified to

$$\ddot{q}_{min,i} \leq q_i''(s)\dot{s}^2 \leq \ddot{q}_{max,i} \, . \tag{4.17}$$

The maximum value is determined in this case as

$$\begin{aligned} &\forall i \in \{1, .., N\}, q_i'(s) = 0 \\ &\dot{s}^{acc,**}_{max} = \min_i \begin{cases} \sqrt{\frac{\ddot{q}_{min,i}}{q_i''(s)}}, & \text{if } q_i''(s) < 0 \text{ and } \ddot{q}_{min,i} < 0 \\ \sqrt{\frac{\ddot{q}_{max,i}}{q_i''(s)}}, & \text{if } q_i''(s) > 0 \text{ and } \ddot{q}_{max,i} > 0 \\ \infty, & \text{else .} \end{cases} \end{aligned} \tag{4.18}$$

When iterating over all joints, ∞ is always chosen in the case differentiation if for the given joint i, no upper bound condition is active in the given case. The joint is thus ignored in the selection of the minimum value among all computed joint upper bound values.

By merging all cases involving joint acceleration constraints, the new bounds are as follows:

$$\dot{s} \leq \dot{s}_{max}^{acc} = \min\left\{\dot{s}_{max}^{acc,*}, \dot{s}_{max}^{acc,**}\right\} \tag{4.19}$$

In this work, we generally assume $\ddot{q}_{min,i} < 0,\ \ddot{q}_{max,i} > 0\ \forall i \in \{1, .., N\}$, although other cases are already included in Eqn. (4.18). If it is possible that $\ddot{q}_{min,i} > 0$, e. g., due to dynamics, it needs to be verified if the derivations still hold or if a lower bound that might result in a $\dot{s}_{min} > 0$ needs to be added. Further investigation of these cases and their full inclusion is left for future work.

4.2.3 Overall Path Velocity Limits

The overall allowed maximum path velocity when traversing the path and considering first- and second-order constraints together is given by

$$\dot{s} \leq \dot{s}_{max} = \min\left\{\dot{s}_{max}^{vel}, \dot{s}_{max}^{acc}\right\} \tag{4.20}$$

This only covers upper bounds; zero velocity is always considered admissible, and thus the lower path velocity bound is zero velocity at all times. In the implementation, a small positive value near zero is enforced for numerical stability, so that the velocity cannot become a small negative value, e. g., due to small numerical integration errors. In the same way, a small epsilon value below $\dot{s}_{max}^{vel}$ and $\dot{s}_{max}^{acc}$ is used to ensure constraints are not overstepped.

Compared to Kunz and Stilman (2012), where only the case of symmetric joint space bounds is considered, more comparisons between joint combinations are necessary, and thus the computation of the MVC is slower and computationally more expensive. For the sake of consistency, we only use our own derivations from above, regardless of whether symmetric or asymmetric limits are present. However, a future extension would be to revert to the solution of Kunz and Stilman (2012) in the presence of symmetric limits, which would significantly reduce the computational burden in these cases.

4.3 Trajectory Generation: Two-Pass Path Scan

With the MVC formulation of the previous section, it is possible to calculate for an arbitrary path coordinate its velocity limit such that none of the joint constraints is violated. In this section, the actual parameterization algorithm is described, which relies on two integration passes along the path to compute a time-optimal velocity profile. First, the problem description is given in Section 4.3.1, where a transformation of the optimization problem, adopted from CO-based solutions, is used to allow scanning the path directly locally without the need for time integration. Later in Section 4.3.2, the specifics of the integration optimization are presented. Finally, extension to third-order constraints and proper trajectory sampling are discussed in Sections 4.3.3 and 4.3.4.

4.3.1 Problem Description

Here, the optimization problem to be solved is formulated. Given the constraints, the time to traverse the path needs to be minimized:

$$\begin{aligned}
\min_{T,s} \quad & T = \int_0^T dt && \text{(4.21a)}\\
\text{subject to:} \quad & s\left(0\right) = 0, && \text{(4.21b)}\\
& s\left(T\right) = s_{end}, && \text{(4.21c)}\\
& \dot{s}\left(0\right) = \dot{s}_0, && \text{(4.21d)}\\
& \dot{s}\left(T\right) = \dot{s}_T, && \text{(4.21e)}\\
& \dot{s}\left(t\right) \geq 0, && \text{(4.21f)}\\
& \dot{s}_{min}\left(t\right) \leq \dot{s}\left(t\right) \leq \dot{s}_{max}\left(t\right), && \text{(4.21g)}\\
& \ddot{s}_{min}\left(t\right) \leq \ddot{s}\left(t\right) \leq \ddot{s}_{max}\left(t\right), && \text{(4.21h)}\\
& \text{for } t \in \left[0, T\right].
\end{aligned}$$

The objective function can be reformulated in terms of s by changing the integration variable and selecting the path acceleration as the optimization variable.

$$T = \int_0^T dt = \int_{s(0)}^{s(T)} \frac{1}{\dot{s}} ds \tag{4.22}$$

$$\begin{aligned}
\min_{\ddot{s}} \quad & T = \int_{s(0)}^{s(T)} \frac{1}{\dot{s}} ds && \text{(4.23a)}\\
\text{subject to:} \quad & s\left(0\right) = 0, && \text{(4.23b)}\\
& s\left(T\right) = s_{end}, && \text{(4.23c)}\\
& \dot{s}\left(0\right) = \dot{s}_0, && \text{(4.23d)}\\
& \dot{s}\left(T\right) = \dot{s}_T, && \text{(4.23e)}\\
& \dot{s}\left(s\right) \geq 0, && \text{(4.23f)}\\
& \dot{s}_{min}\left(s\right) \leq \dot{s}\left(s\right) \leq \dot{s}_{max}\left(s\right), && \text{(4.23g)}\\
& \ddot{s}_{min}\left(s\right) \leq \ddot{s}\left(s\right) \leq \ddot{s}_{max}\left(s\right), && \text{(4.23h)}\\
& \text{for } s \in \left[0, s_{end}\right].
\end{aligned}$$

This formulation is based on what was proposed by Verscheure et al. (2009) and Palleschi et al. (2019). The proposed substitutions are

$$a\left(s\right) = \ddot{s} \tag{4.24}$$

$$b\left(s\right) = \dot{s}^2 \tag{4.25}$$

with the additional constraint

$$b'\left(s\right) = 2a\left(s\right) \tag{4.26}$$

due to

$$\dot{b}\left(s\right) = b'\left(s\right)\dot{s} \tag{4.27}$$

From this description, three conclusions are possible. The first one is that a convex formulation of the problem can be obtained by applying suitable variable substitutions such

that the objective function becomes convex and all constraints are linear. The second one is that in a discretized version, the global optimum results from choosing the maximum available path velocity in every step. The third one is that time integration is no longer necessary and the time-optimal solution can be found by directly stepping locally along the path.

4.3.2 Integration Optimization

An overview of the specifics of the integration process is given in Section 4.1.2. First, the integration method for scanning the path has to be selected. There are generally two options: equitemporal with a fixed integration time or equidistant in the phase space with a fixed distance between path samples.

Time integration is the standard approach for numerical integration methods. It has the following properties:

- It can be used directly for sampling at the required cycle time for the position controller, so direct computation of the required time-discrete information is given. It is the preferred choice for forward integration, as it can then be used directly for sampling without the need for further post-processing; see Section 4.3.4.
- Backward integration cannot be performed first for a real-time solution, as it has to always start at the goal position, with no option to predict the number of required integration steps. It is not possible to have a fixed number of integration steps known beforehand, and starting from an intermediate position on the path is also not possible.
- The current path segment cannot be directly determined, only after integrating up to the path position. At least first-order constraints should then be applied to the velocity at the next position using constraints computed at the current position, which leads to inaccuracies. A remedy may be to apply first-order constraints twice, at the current and at the next position, or to integrate several times until the velocity is sufficiently reduced that no constraints are violated.

Methods in the convex optimization category usually apply local integration or stepping along the path. Local integration has these properties:

- The backward integration can be done first, as a maximum number of grid points to scan over can be used, and the current path segment is directly determined.
- It is not directly compatible with time sampling for a position controller. Time-stamps need to be computed from path positions and then interpolated for the needed controller time rate, which is difficult to do without constraint violations for interpolated intermediate values.
- An implementation with a fixed number of grid points is easy. However, this approach does not generalize well to different paths for online generation, as the actual distance between sampling positions will vary greatly. Thus, a certain number of grid points will be too sparse for some paths, which causes issues for the post-processing, as not enough information is available for interpolation with sufficient accuracy.

- Adapting the grid point number depending on the shape of the path or the path length is difficult. Standard spline interpolation does not provide a direct method to compute the path length.

However, our custom spline interpolation for the pre-processing is able to achieve approximately equidistant segment length (see Section 4.1.1). The computed grid points from the pre-processing can be directly used for local integration, which resolves the issues of the last two items.

The procedure for local integration with equidistant sampling is described in the next section. Local integration was implemented first, for a full backward and forward parameterization. This allows for the generation of a complete time-optimal trajectory. Then, a full local backward parameterization, usable together with forward time integration for sampling, was implemented. The sampling method has remaining issues, but when these are fixed, the need for post-processing is avoided as the sampling time can be identical to the integration time step. An issue with the combination of forward time integration and backward local integration is that the backward integration is then evaluated between grid points to determine limits for the forward integration, so a fitting interpolation scheme has to be selected for the velocity curve resulting from the backward integration. Linear interpolation is chosen here as the velocity curve is piecewise linear due to constant acceleration. This is further detailed in Section 4.3.4.

The remainder of this section discusses several important ingredients and additions to the path parameterization or optimization algorithm to ensure time-optimality, reliability, and constraint-respecting trajectory output.

Equidistant formulas

The path velocity at the next path state can be computed using the variable substitutions from Section 4.3.1, which are applied by approaches based on convex optimization. Verscheure et al. (2009) discretized Eqn. (4.26) by a first-order approximation using the direct transcription method. In this way, the velocities s_i, s_{i+1} at the current grid point i and the next grid point $i+1$ relate to the acceleration $\ddot{s}_i$ with

$$b'(s) \approx \frac{\dot{s}_{i+1}^2 - \dot{s}_i^2}{s_{i+1} - s_i} = 2\ddot{s}_i \,. \tag{4.28}$$

In this work, the path acceleration serves as both the system input and free variable in each integration step, usually either the maximum or the minimum path acceleration. For the forward integration, Eqn. (4.28) can be reordered, and using the selected path acceleration, the next path velocity is determined with

$$\dot{s}_{i+1} = \sqrt{\dot{s}_i^2 + 2\left(s_{i+1} - s_i\right)\ddot{s}_i} \,. \tag{4.29}$$

When the velocity is clipped to the MVC velocity or reduced to the velocity from the backwards scan during the forward scan, the actual acceleration at the grid point can be determined with

$$\ddot{s}_i = \frac{\dot{s}_i^2 - \dot{s}_{i-1}^2}{2\left(s_i - s_{i-1}\right)} \,. \tag{4.30}$$

Maximum and minimum path accelerations

As the main system input (for every integration step), monitoring the acceleration values is required to avoid violations of the joint acceleration limits.

These values are the minimum and maximum path accelerations that allow the robot to follow the path without violating the available accelerations in each DoF. The constraint inequality can be rewritten according to Pham (2014) as

$$\boldsymbol{a}(s)\,\ddot{s} + \boldsymbol{b}(s)\,\dot{s}^2 + \boldsymbol{c}(s) \leq 0 \tag{4.31}$$

with

$$\boldsymbol{a}(s) = \begin{bmatrix} \boldsymbol{q'(s)} \\ -\boldsymbol{q'(s)} \end{bmatrix}, \boldsymbol{b}(s) = \begin{bmatrix} \boldsymbol{q''(s)} \\ -\boldsymbol{q''(s)} \end{bmatrix}, \boldsymbol{c}(s) = \begin{bmatrix} -\ddot{\boldsymbol{q}}_{\boldsymbol{max}} \\ \ddot{\boldsymbol{q}}_{\boldsymbol{min}} \end{bmatrix} . \tag{4.32}$$

The minimum and maximum allowed path accelerations across all N DoFs ($\forall j \in \{1, .., N\}$) follow as

$$\ddot{s}_{min} = \max_j \left\{ \frac{-c_j(s) - b_j(s)\,\dot{s}^2}{a_j(s)} \mid a_j(s) < 0 \right\} \tag{4.33}$$

and

$$\ddot{s}_{max} = \min_j \left\{ \frac{-c_j(s) - b_j(s)\,\dot{s}^2}{a_j(s)} \mid a_j(s) > 0 \right\} . \tag{4.34}$$

During the forward and backward passes along the path, the current path velocity is computed in the previous integration step. $\ddot{s}_{min} < \ddot{s}_{max}$ must always be ensured during a path scan, as $\ddot{s}_{min} > \ddot{s}_{max}$ indicates that an invalid path velocity state was reached unexpectedly. Barnett and Gosselin (2020) used the check $\ddot{s}_{max} < \ddot{s}_{min}$ to verify if an area above the MVC was reached, and only then triggered the bisection search to find the maximum feasible velocity. We did not adopt their $\ddot{s}_{max} < \ddot{s}_{min}$ check, as this turned out to be a non-stable solution. Instead, the MVC is computed in every step of the first scan using our analytical formulation to ensure that both first- and second-order constraints on the velocity are respected. We noticed that it is possible to reach island areas above the second-order MVC where $\ddot{s}_{max} > \ddot{s}_{min}$ again, and then the check would fail to detect an inadmissible path state.

An acceleration singularity occurs when at least one value in $\boldsymbol{a}(s)$ is close to zero. The non-zero values will always dominate as accelerations in the singular DoFs converge to infinite values. This allows us to integrate *through* singularities without issues, as all valid acceleration ranges are considered. The pre-processing ensures that at least one value is non-zero, so at least one degree of freedom will dominate with a finite acceleration value, and a displacement with non-zero velocity is ensured for at least one DoF.

When integrating forward, the path is traversed in the same direction as the robot will move along the final trajectory, so the computed path acceleration values reflect the real acceleration values for the final trajectory. This is not the case for the backwards scan. Here, the acceleration computed at grid point i is used to determine the velocity at path position $i-1$. Without further measures to improve the merging of forward and backward scans, this will lead to discrepancies and acceleration limit violations. To the best of our knowledge, this issue is not sufficiently treated in related work on offline NI-based methods. Especially for braking segments, where the robot moves along the velocity profile originally computed by the backwards scan, the actual available minimum acceleration

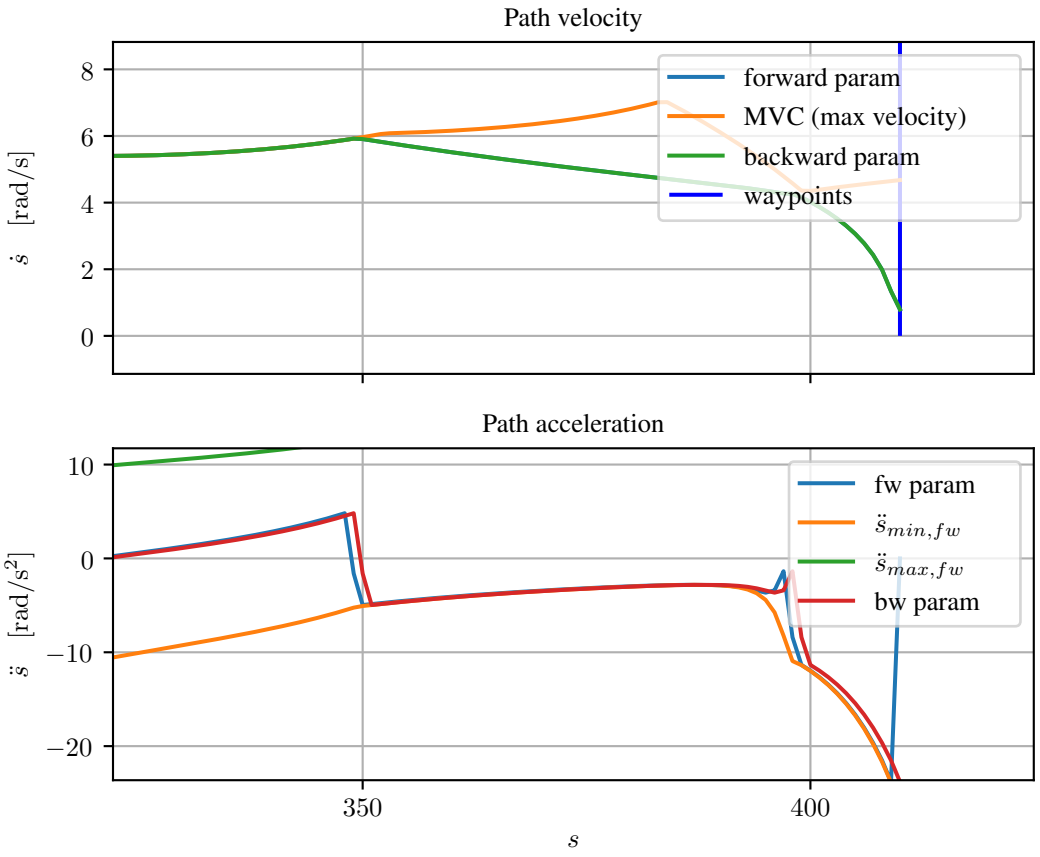


Figure 4.4: Braking trajectory: bringing forward and backward pass together. The forward parameterization is able to decelerate along the velocity profile from the backwards parameterization while moving along the minimum acceleration $\ddot{s}_{min,fw}$ computed in the forward scan (the forward param in blue is thus identical to the backward param in green in the path velocity plot). The improved backwards integration allows for the generation of a time-optimal braking segment without constraint violations.

is usually larger than what is assumed during the backwards scan. This discrepancy is due to the robot moving from a larger velocity, where available acceleration ranges are more restricted, to a smaller velocity during braking, where path acceleration ranges become greater. The backwards scan moves here from small to larger velocity and wrongly assumes the less restricted acceleration range at the smaller velocity to compute the next velocity. The acceleration ranges at every grid point during the backwards scan are shifted by one integration step compared to the forwards scan.

We tried to resolve this issue during the forward scan by looking ahead and switching earlier to $\ddot{s}_{min}$ or the smaller velocity to anticipate a braking segment, but this led to oscillating behavior or minimum acceleration violations. Instead, we integrate twice in every integration step during the backwards scan. First, $\ddot{s}_{min,i}$ in grid point i is used to determine the path state $i-1$ (including limiting the path velocity to the MVC if it was reached). Then using $\ddot{s}_{min,i-1}$ from this temporary state,

$$\ddot{s}_{min,i,final} = \max\left\{\ddot{s}_{min,i}, \ddot{s}_{min,i-1}\right\} \ . \tag{4.35}$$

$\ddot{s}_{min,i,final}$ is then used for backwards integration to reach the next grid point. This approach allows for resolving the shift between forward and backwards acceleration profiles. Figure 4.4 shows a braking trajectory segment towards the end of a path. The forward parameterization is able to switch one step earlier to the minimum path acceleration $\ddot{s}_{min,fw}$ at grid point 350 and is then able to follow it exactly while braking along the velocity profile from the backwards pass. The effect of Eqn. (4.35) can also be observed in the last path section, right of grid point 400 in the path acceleration plot, where the backwards parameterization in red stays a little over $\ddot{s}_{min,fw}$ as it considers the larger minimum acceleration at the next grid point.

Addressing Singularities on the MVC

As already described in Section 4.2, the MVC is defined both by first-order velocity constraints and second-order acceleration constraints. Segments of the path, where the MVC is only defined by first-order constraints, are those where the robot is able to move along the MVC. If it is possible during the parameterization to accelerate to these segments, the robot moves with at least one DoF at its velocity limit as long as the available acceleration ranges allow it to follow the MVC profile. The case is different for MVC segments dominated by second-order constraints. Usually, the acceleration limits become a singular value on these segments. The challenge for the time-optimal parameterization is then not only to touch these MVC segments without overstepping into inadmissible velocity ranges, but also to exactly hit the singular acceleration in these points at the same time.

The parameterization in the introductory Figure 4.2 has such a point at around grid point 100. These points are common, and commonly, the forward scan decelerates along the backwards parameterization to the singular point, assumes the singular acceleration in the grid point, and then accelerates afterwards again. Figure 4.5 shows such a singular point on the second-order MVC where the algorithm integrates successfully through it. Right to the singular point, the backwards parameterization is repeatedly clipped to the MVC, which explains why the acceleration from the backwards parameterization is mostly outside the valid acceleration range. Left of it, the backwards scan has generated a valid segment, which the forward scan uses to decelerate to the singular point.

Only another check during the backwards scan is introduced for these singularities to verify that a positive path velocity can be reached when integrating forward from these points with the singular acceleration. In rare cases, it is possible that $\dot{s} < 0$ when integrating forward, which would be an inadmissible state. Thus, the robot cannot be stopped starting from these points without acceleration constraint violations, as assuming $\dot{s} = 0$ in the next forward grid point would require a different acceleration than the singular one. The singular point is then classified as not admissible, and the path velocity is set to zero in these points to avoid the singularity and to stop the robot one grid point earlier.

Further details of the algorithm are commented and explained in the source code of the *passLocal()* method, which contains the implementation of the local forward and backward integration algorithm.

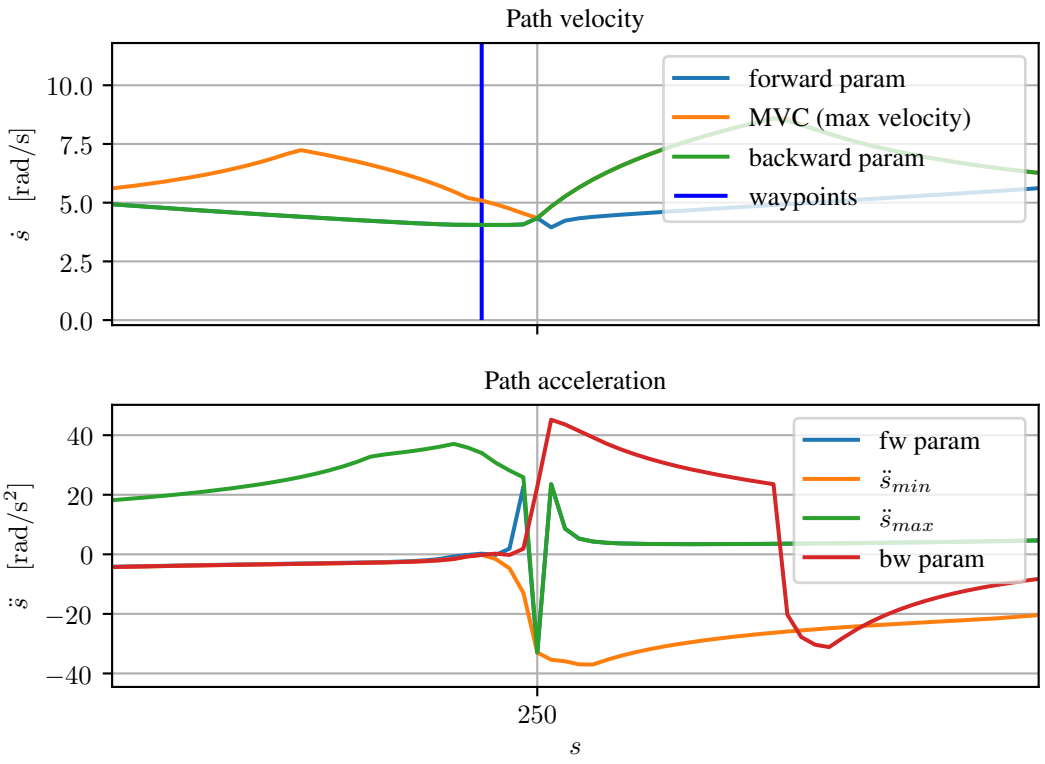


Figure 4.5: Singularity example on second-order MVC. The algorithm is able to integrate through a singularity at grid point 250, where $\ddot{s}_{min}$ and $\ddot{s}_{max}$ converge to a single value. The forward parameterization is able to exactly hit this singular acceleration value at the MVC while following the backwards parameterization curve up to this point.

4.3.3 Extension to Jerk Constraints: A Starting Point

While the development of a complete real-time TOPP algorithm with first-, second-, and third-order kinematic constraints is beyond the scope of this thesis, it is nonetheless a goal of our contribution to be a promising starting point for an extension to jerk constraints. Jerk constraints are needed for smooth trajectories and for applying the algorithm to a real robot, as acceleration jumps with infinite jerk cannot be handled by joint position controllers without losing path tracking accuracy or increasing wear of the mechanical parts. Motor-side oscillations can happen where the position controllers become increasingly unstable when movements repeatedly include large jerks. Commercial real-time control interfaces usually monitor the jerk, and if oscillations in the acceleration profile become too frequent or too extreme, a safety stop is executed, and the robot program is aborted. The above-described concept only handles first- and second-order kinematic constraints. Here, we want to summarize how the concept can be extended to third-order constraints like the jerk.

For the NI approach, maximum jerk instead of extremal acceleration is the system input for the first two integration passes. In the same way as acceleration constraints are relaxed during the backwards scan to adhere to velocity constraints, jerk constraints have the least priority and are ignored if otherwise acceleration or velocity constraints cannot be met. Assuming a solution is available to compute maximum and minimum velocity-

acceleration surfaces during the backwards and forward scan, in addition to the MVC for first- and second-order constraints, a velocity profile is available after the two scans that respects first-, second-, and third-order constraints in most segments. Our two-pass scan approach is based on Barnett and Gosselin (2020), who in turn adopted the two-pass procedure from Dong and Stori (2006). Dong et al. (2007) extended the two-pass approach with jerk as already summarized in Section 2.2.1.1, which can be used as a base. They used minimum jerk integrations in a third pass to remove remaining jerk violations. However, this is effectively not a single pass. Instead, backtracking is used repeatedly for every jerk violating segment until jerk constraints are respected. They start integrating with minimum jerk from both sides where the maximum profiles intersect. When the profiles cannot be connected with continuous acceleration from the initial integration start point, they go back to points further away from the intersection on both sides and start integrating again. This strategy can not only be slow, but also not real-time capable, as there is no maximum for the backtracking steps. A finite number of integrations per path segment needs to be guaranteed for real-time capability. The alternative method based on Multiple Shooting by Pham and Pham (2017) also only slowly converges for more restrictive jerk constraints. Please note that these jerk switching points can happen frequently along a path. Other NI-based solutions use similar strategies. Such iterative approaches are a challenge and likely not suitable for a real-time algorithm. So an open question is how to connect two maximum jerk profiles with a minimum jerk profile in order to remove jerk violations. To do this efficiently is also an open issue for offline methods. The task is to find a strategy that identifies these jerk switching points in a fast and real-time manner. Even if it is only a heuristic for a nearly time-optimal approximation, it still needs to ensure that the profiles starting in these points always connect in a tangent switch point that has no jerk violation.

Another challenge is to extend the analytical MVC formulation with jerk. The phase space with third-order constraints has three dimensions, and there are minimum and maximum acceleration surfaces depending on the path velocity for the third-order case, so extremal acceleration and velocity are coupled for a given path position. Considering and imposing jerk constraints, path velocity, acceleration, and jerk are bounded by

$$\begin{aligned} &\forall i \in \{1, .., N\} \\ \dddot{q}_{min,i} \leq q_i'''(s)\dot{s}^3 + 3q_i''(s)\dot{s}\ddot{s} + q_i'(s)\dddot{s} &\leq \dddot{q}_{max,i} \,. \end{aligned} \tag{4.36}$$

As can be seen in Eqn. (4.36), the joint jerk limits impose restrictions on path velocity, acceleration, and jerk. Using the same derivation process for the analytical MVC, a partial decoupling is possible, but likely not completely, for velocity and acceleration in such a cubic equation. The maximum velocity from the first- and second-order MVC can be used as an initial guess, and a distinction of cases based on the possible properties and shapes of the bounding surface can help to find the maximum. If a fully analytical derivation of the extremal acceleration-velocity values is not possible, a bisection search or similar algorithm can be used together with the derived equations to find the maximum velocity where feasible acceleration and jerk values are still possible[1]. This would be more efficient and easier to make real-time-capable than the line search used by Dong et al. (2007) for the complete optimization problem. Finding this maximum velocity-acceleration pair is needed for the initial scan, where large velocity discontinuities can

[1] If a reformulation or simplification into a function of a single variable is possible, a numerical real-time-capable root finding method can be applied, e. g., the Anderson-Bjoerck-King method in combination with bisection search, see Kröger (2010)

happen. The second scan already produces a continuous velocity profile, which simplifies this optimization.

4.3.4 Trajectory Sampling

When a trajectory in phase space is computed by the TOPP algorithm in terms of $\dot{s}$ and $\ddot{s}$, Eqs. (4.3) and (4.4) can be used to map the final phase space trajectory back to joint space for sampling. It needs to be possible to sample the computed joint space trajectory at a given cycle time to feed a position controller. This is required to apply the parameterization to real robotic systems. Robot controllers usually operate at fixed cycle times, and for every cycle, a new goal position and optionally velocity are expected. This means that the algorithm needs to output the exact path-constrained position and velocity that result from the time-optimal trajectory at these time instants without constraint violations. The selected solution highly depends on the integration discretization approach, see Section 4.3.2.

Although related work often does not discuss this challenge, we identified the following possible options:

- Fit a spline to the discrete trajectory data $\boldsymbol{q}(s, t, \dot{s})$, e. g., using the Eigen library for the velocity profile. Depending on the sparseness of the grid points, this interpolation may be too inaccurate. It is also not ensured that the constraints are respected between grid points. This approach requires a significant portion of the final trajectory to be precomputed for the interpolation to work. Both Barnett and Gosselin (2020) and TOPP-RA (Pham and Pham, 2018) apply this technique to interpolate the offline-generated trajectory.
- Use the ROS-control trajectory following controller and its custom cubic spline interpolation.[2] This is a well-tested interpolation procedure exactly for our use case, and it is already integrated into a robot control framework, but it has the same drawbacks as the first option.
- Instead of a Reflexxes interpolation controller, use Ruckig (Berscheid and Kröger, 2021) to interpolate between grid points. This would be the most "informed" interpolation, including jerk limitation, but it can cause path deviations and overshooting, as our approach is not yet able to support jerk limits. Neither Reflexxes nor Ruckig is able to ensure exact path following according to the original path shape given by the pre-processing.[3]

The previously presented two-pass path scan uses, by default, local equidistant integration for both the forward and backward scans. This approach is simple and ensures that no evaluation between grid points is necessary, as the forward integration values are determined at the same positions as the backward limit curve. The time information is then added by computing the timestamps of every grid point during the forward pass. From the equation of motions (assuming constant acceleration), $\dot{s}_{i+1} = \ddot{s}_{i+1}\,(t_{i+1} - t_i) + \dot{s}_i$ and

[2] Documentation is available at `http://wiki.ros.org/joint_trajectory_controller`.

[3] This approach is used in *MoveIt* (Sucan and Chitta, 2026) where the work of Kunz and Stilman (2012) for path parameterization and Ruckig for jerk-limited smoothing or filtering afterwards is combined. To minimize the mentioned issues, only small or restrictive acceleration limits can be used there.

$s_{i+1} = s_i + v_i t + 0.5\ddot{s}_{i+1} t^2$, follows

$$t_{i+1} = t_i + \frac{2\left(s_{i+1} - s_i\right)}{\dot{s}_{i+1} + \dot{s}_i} \,. \tag{4.37}$$

The above interpolation options could then be readily applied, but the already mentioned drawbacks would persist. However, these interpolation options are common practice and the algorithm would provide a trajectory that can be sampled at a fixed cycle time using one of these options.

We also developed an option that integrates directly with the forward integration pass. This allows for only one integration step for the forward pass in one control cycle. The equidistant forward integration is replaced with time integration using the cycle time as the integration width. The challenge then lies in interpolating the equidistant backward integration between grid points, when the forward integration has accelerated to the backward limit curve and must follow it. First, the corresponding segment from the backward pass needs to be found. Then, linear interpolation is used until the path position and corresponding velocity are found that lie on the linear interpolated velocity curve between the corresponding grid points. The spline derivatives are computed at these positions, and then the joint trajectory data can be determined.

Although initial tests of this routine showed promising results, we were not able to fully resolve oscillating behavior around grid points and resulting interpolation errors at the end of this thesis. It is possible that an implementation error occurred and the correct segments are not reliably identified, or that necessary switches to the following path segment are not always detected. We assume that a simplification of the routine is possible. Currently, we integrate up to the path position and then compare velocities with the backward parameterization. This may need to be repeated until the correct path velocity-position pair is found. It may be possible to directly find the interpolated velocity and then compute the corresponding path position.

4.4 Implementation

The algorithm was implemented in C++ as a free and open-source standalone library with few dependencies[4]. A full parameterization with complete forward and backward path parameterization is available, and the preferred solution with full backward parameterization and sampling with one forward integration step, using time integration as a draft.

The implementation checks for different errors that can arise from an invalid start or goal state: start state velocity too high (above the MVC) and goal state velocity too high (velocity not reachable on path, returns parameterization with the highest possible velocity). The theoretically possible velocity stall error is checked, but it never occurred in our final tests on our data set. We assume that the pre-processing was always able to generate a smooth enough path profile with C^2-continuity for every random input, so that a path acceleration could be identified to keep moving along the path and avoiding an infeasible parameterization problem.

[4]The public repository with source code (including evaluation, visualization, and examples) and documentation is available at https://github.com/KITrobotics/rt-topp.

The implementation needs to be verified to ensure that it produces time-optimal or close to time-optimal solutions. A necessary but not sufficient condition is that at least one degree of freedom is at its extremal acceleration or its maximum velocity. The *verifyTrajectory()* method examines this. The method also makes sure that the limits are not violated. This procedure is further detailed in the next section.

A simple API is provided to set parameters and hand over waypoints. A sampling method for the output is available as well, including status codes. The sampling is also verified, but still produces issues that we were not able to completely resolve as of the end of the thesis. The problems are related to the interpolation of the backward parameterization between grid points. Further investigation is necessary to find the correct limiting velocity with linear interpolation, especially for braking segments. These issues appear to be solvable, and initial tests show that time sampling in the forward integration with the backward pass as an upper limit is generally possible.

The implementation is real-time-capable with no memory allocation during the runtime of the parameterization. Template classes and data structures are used, which by default only allocate memory on the stack. This limits the maximum usable number of waypoints and integration steps, as the stack size is restricted by the operating system. For the experiments needed for this thesis, this prototype implementation was sufficient. However, for a generally usable implementation by practitioners and integration into control frameworks, a more sophisticated extension is needed that can also use the heap with fewer restrictions on the number of waypoints, path length, etc., while still avoiding dynamic memory allocation.

Additionally, the output can be compared with TOPP-RA (Pham and Pham, 2018), i. e., the path velocity and acceleration, and the final joint values. However, TOPP-RA's interpolation procedure between waypoints is different from our custom scheme, which leads to different path representations and thus makes a direct comparison difficult. TOPP-RA has a free and open-source implementation available. Most of TOPP-RA is implemented in Python, so it needs to be compiled to Cython to offer a fairer comparison. The correctness of the formulas for the analytical MVC can be verified by comparing the resulting MVC with the feasible sets calculated via linear programming in TOPP-RA, which are the corresponding techniques for the absolute path velocity limits.

Implementing a MoveIt (Sucan and Chitta, 2026) plugin is also possible based on the RT-TOPP library, similar to the MoveIt kinematics plugin in Section 3.2. The plugin can then be used as a drop-in replacement to the existing path parameterization algorithms in MoveIt, but it is then not used in the hard real-time control framework, rather in the non-real-time online planning stage. This is due to MoveIt's architecture.

Additionally, the time parameterization is sent afterwards as a discrete set of waypoints with timing information to the real-time trajectory following controller, where another algorithm interpolates the trajectory for the position control interface of the robot controller. This means that in this way, proper sampling of the trajectory cannot be ensured, as path parameterization information is lost between the controller and MoveIt. A better solution is to integrate the path parameterization directly into the trajectory following controller and take the waypoints generated by a MoveIt path planner or a custom path planner as input. Then, a real-time joint position controller thread can directly access the path parameterization data via shared memory, and the path parameterization can run in a different hard or soft real-time thread. So, the recommended procedure is to integrate the RT-TOPP

library into the ROS Control (Chitta et al., 2017) controller framework if the trajectory sampling algorithm should be able to access the complete parameterization data and not a discrete and sparse subset that was previously generated. This issue became apparent during the IK evaluation experiments with the custom trajectory generation pipeline used there in Section 3.3.2, where the approach from Kunz and Stilman (2012) was combined with a Reflexxes trajectory interpolation algorithm.

4.5 Experiments

For the iiwa manipulator validation, similar to the kinematics experiments in Section 3.3, the manufacturer's constraints for the KUKA LBR iiwa 7 R800 lightweight robot (KUKA AG, 2026) are used, with the same acceleration limits as in Eqn. (3.13), this time without safety reduction, so the full capabilities are used.

Additionally, generic 7-DoF manipulator constraints are used and modeled after the evaluation by Barnett and Gosselin (2020), also to make the two evaluations comparable. Waypoints are selected in the position limit interval $[-2.5, 2.5]$ radians. A small start and goal velocity is selected as 0.3 rad/s for every joint. Kinematic constraints are selected symmetrically as

$$\begin{aligned} \forall i \in \{1, .., 7\} \\ \dot{q}_{min,i} = -5\,[\mathrm{rad/s}]\,,\ \dot{q}_{max,i} = 5\,[\mathrm{rad/s}] \\ \ddot{q}_{min,i} = -10\,\left[\mathrm{rad/s^2}\right]\,,\ \ddot{q}_{max,i} = 10\,\left[\mathrm{rad/s^2}\right]\ . \end{aligned} \tag{4.38}$$

4.5.1 Maximum Velocity Curve

The formulas for the MVC and their implementation were separately tested and verified, independently of the full parameterization algorithm. For the second-order MVC, it was successfully verified that $\ddot{s}_{min} < \ddot{s}_{max}$ and that the acceleration limits converge to a singular value or are very close to each other. For the first-order MVC, we made sure that at least one DoF is at its velocity limit. For the overall MVC, we ensured that joint velocity and acceleration limits are within bounds and admissible.

We modified the iiwa constraints to include different combinations of asymmetric limits. Both for the iiwa constraints and the asymmetric constraints, we tested every segment of 30.000 random five-waypoint paths. For all of them, valid MVC values with the above conditions verified were found.

Only for the special case of $\exists i:\ q_i'(s) = 0$, we noticed that it is not a requirement for a path velocity on the second-order MVC to be singular. We verified that these velocities are nevertheless upper bounds by checking the resulting joint accelerations. For velocities slightly above the second-order MVC in these cases, $\ddot{s}_{min} < \ddot{s}_{max}$ holds at these $\dot{s}$, but $\ddot{s}_{min}$ is actually not within the joint acceleration limits anymore. Kunz and Stilman (2012) discusses these singularities as continuous and non-differentiable switching points on the limit curve. It is unclear if their claim is correct that the path acceleration needs to be zero at these points, as proof is lacking. We are treating these points in no other way and selecting the minimum or maximum possible acceleration in the parameterization, as discussed in Section 4.3.2.

4.5.2 Runtime Performance and Trajectory Output

For the main evaluation of the entire algorithm, 10 000 paths with five waypoints were generated using a Mersenne Twister pseudo-random number generator. Only the waypoints were selected by the generator within position limits and handed over to the preprocessing of the RT-TOPP algorithm without further modifications. This means that the algorithm had to deal with sparsely selected waypoints, with some waypoints very close to each other, sharp path edges, and further corner cases. This is the largest single test set with randomly generated paths in the related work known to us. In contrast to the evaluation by Barnett and Gosselin (2020), 10 times more paths are evaluated, and output trajectory parameters are not filtered or otherwise post-processed for constraint violation evaluation. The goal here is also to inspect the acceleration profile of the final trajectory if it is clean enough and ready for an extension with jerk constraints in future work.

A full parameterization, with first a backward pass and then a forward pass using local integration, is executed for every path. The pre-processing tries to keep the distance between grid points approximately the same, and a constant path resolution of 0.05 rad is used to generate the grid points. This small resolution is selected to allow sampling at low cycle times below 20 ms later on, and it leads to a comparable number of grid points as in the evaluation of Barnett and Gosselin (2020). Future work needs to explore whether an even smaller path resolution is needed for sampling at 1 ms cycle time to get good sampling results without constraint overstepping.

An example parameterization from the test set in joint space is shown in Figure 4.6. The velocity from the backward pass in green starting from the goal position is staying below or on the MVC. The final parameterized velocity resulting from the forward pass starting at the start position uses the backward pass as an upper limit. Close to the intermediate waypoints (vertical blue lines), several second-order singular velocities are located, which can be further inspected in the path acceleration plot. At positions around 90 rad and 140 rad path progression, a_{min} and a_{max} converge to a single value, and the final path acceleration from the forward pass in blue is able to exactly go through these singular areas without leaving the admissible acceleration tunnel comprised of a_{min} and a_{max}. At the singular path progression of about 250 rad, an even larger jump in the acceleration bounds can be observed; still, the algorithm is able to integrate through these singular areas. It can also be seen that during braking segments (forward velocity decreasing along velocity from backward pass in green), forward (blue) and backward (red) accelerations are converging to the same values, which shows that the algorithm is able to merge the two integration passes very well. While the forward acceleration stays within the bounds, the acceleration from the backward pass is allowed to leave the admissible range if the velocity is clipped to the MVC.

Figure 4.7 shows the joint space results for the same example parameterization. The singularity discussed above, close to the fourth waypoint, can here be seen as related to a sharp path edge, especially for the joints q_2 and q_5 at 250 rad. Joint velocities and acceleration exactly hit the maximum and minimum constraints along the path. The acceleration profiles have jumps at switches between acceleration and deceleration segments, as no jerk constraints are applied. However, the profiles overall look clean without oscillations, and the acceleration and deceleration segments are clearly distinguishable. Only at the singularity at 250 rad, there is an oscillation. This is caused by the algorithm directly integrating through a singularity where jumps in acceleration values can happen,

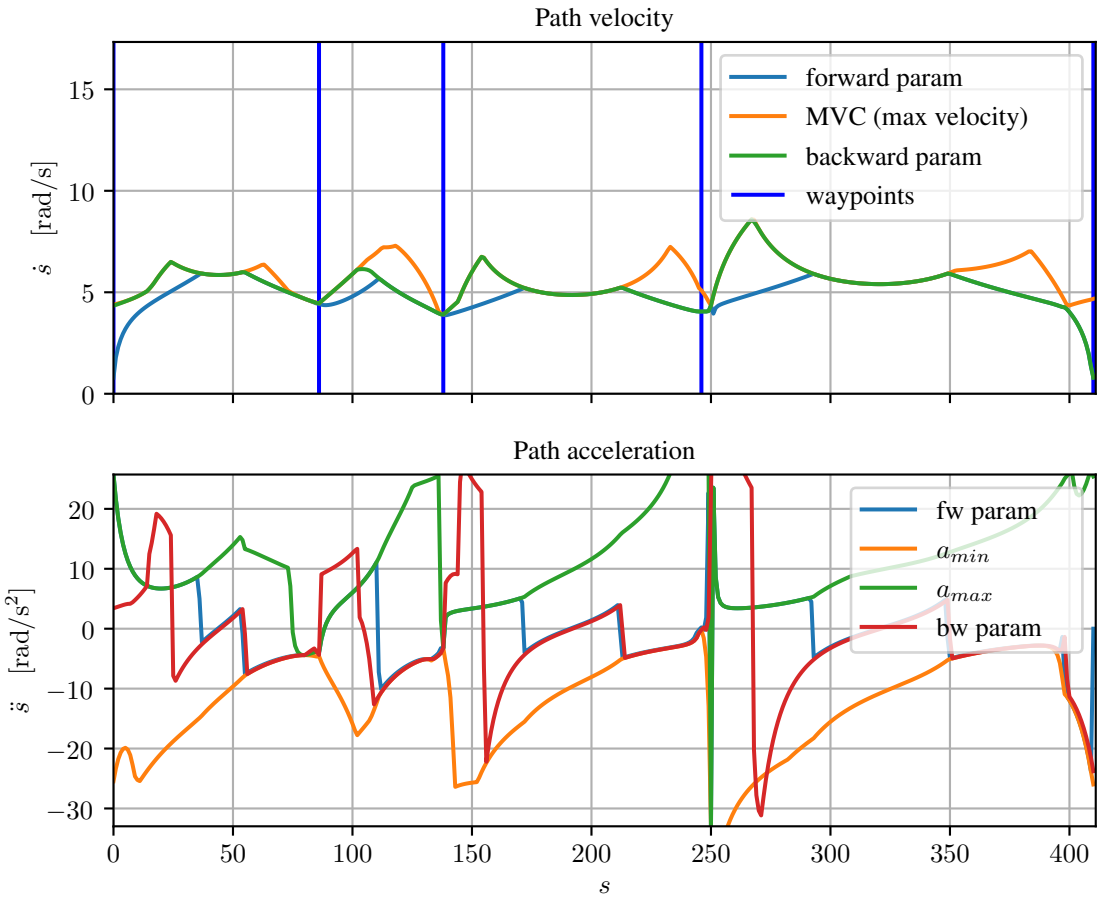


Figure 4.6: Phase space trajectory result for a given pseudo-random path from the five-waypoint test set. Path velocity and path acceleration are shown.

but time-optimality is ensured this way. Still, acceleration bounds are also respected in this segment, and our expectation is that with a future extension to jerk constraints, such segments will be automatically smoothed.

Every produced trajectory was successfully verified to ensure that the necessary conditions in phase space for every grid point are fulfilled for an admissible and time-optimal parameterization. The following conditions were verified:

- The backward parameterization is below or on the MVC.
- The forward parameterization is below or on the backward parameterization.
- The final forward path acceleration is within the path acceleration limits.

The test system was a PC running Ubuntu 18.04 (Intel Core i7-7700K at 4.50GHz, 32GB RAM). The overall results for the random waypoints test set are listed in Table 4.1. Our computation times are roughly 4 times faster than the optimization and interpolation time results Barnett and Gosselin (2020) reported for their algorithm with a similar number of grid points. They used a different CPU (Intel Core i7-8750H at 4.10 GHz) with a slightly smaller single-core frequency. And their algorithm was already 15 to 20 times faster than TOPP-RA. The results show that on average, trajectories with a duration of 4,5 seconds can be computed in less than 340 µs. One grid point correlates to less than 0.73 µs

Parameter	min	max	mean	nominal
Optimization time [µs]	191	485	338.712	-
Trajectory duration [s]	2.8301	5.9704	4.4991	-
grid points N	264	715	482	-
$\max(\lvert\dot{\boldsymbol{q}}_{opt}(s)/\dot{\boldsymbol{q}}_{max}(s)\rvert)$	0.8400	1.0	0.9999	1
$\max(\lvert\ddot{\boldsymbol{q}}_{opt}(s)/\ddot{\boldsymbol{q}}_{max}(s)\rvert)$	1.0	1.0	1.0	1
$\mu\left(\lVert\gamma\rVert_\infty(s)\right)$	0.9819	0.9974	0.9939	1
$\sigma\left(\lVert\gamma\rVert_\infty(s)\right)$	0.0360	0.0787	0.0457	0

Table 4.1: Parameterization results for 10000 pseudo-random paths with five waypoints for a generic seven-DoF manipulator

computation time on average on our platform for a complete forward and backward pass, including the pre-processing, even in the worst case with the maximum optimization time. So, using this worst-case computation time for one grid point, the worst-case computation time for the parameterization is expected to scale linearly with the path length and thus the number of grid points. After execution of the pre-processing, the path length and number of grid points are known, and this benchmark value (computation time per grid point) can be used to calculate the expected worst-case computation time of the parameterization algorithm for a given path in a conservative manner on a real-time system.

In addition to the optimization and trajectory duration times, these metrics are also considered for the joint space:

1. $\max\left(\lvert\dot{\boldsymbol{q}}_{opt}(s)/\dot{\boldsymbol{q}}_{max}(s)\rvert\right)$: normalized maximum joint velocity across all grid points of a final trajectory. The value is in the evaluation 1.0 or less, which means that velocity bounds are respected and never exceeded across all tested paths.
2. $\max\left(\lvert\ddot{\boldsymbol{q}}_{opt}(s)/\ddot{\boldsymbol{q}}_{max}(s)\rvert\right)$: normalized maximum joint acceleration across all grid points of a trajectory. This value is 1.0, so no violation of acceleration bounds across all generated trajectories, and for every trajectory, the acceleration bound is hit at least once for at least one joint.
3. $\boldsymbol{\gamma}(s) = \begin{bmatrix}\lvert\dot{\boldsymbol{q}}_{opt}(s)/\dot{\boldsymbol{q}}_{max}(s)\rvert \\ \lvert\ddot{\boldsymbol{q}}_{opt}(s)/\ddot{\boldsymbol{q}}_{max}(s)\rvert\end{bmatrix}$: stacked normalized joint velocities and accelerations. For a trajectory consisting of N grid points and seven joints, $\boldsymbol{\gamma}(s)$ is a matrix with 14 rows and N columns.
4. $\lVert\boldsymbol{\gamma}\rVert_\infty(s)$: column-wise infinity norm for $\boldsymbol{\gamma}(s)$

A necessary but not sufficient condition for optimality is that each column in $\boldsymbol{\gamma}$ equals 1, which means that at least one joint is at its limit in every grid point, either at the velocity limit or at the acceleration limit. In the optimal case, $\mu\left(\lVert\boldsymbol{\gamma}\rVert_\infty(s)\right) = 1$ and $\sigma\left(\lVert\boldsymbol{\gamma}\rVert_\infty(s)\right) = 0$. Due to small numerical inaccuracies, the values in the evaluation in Table 4.1 do not exactly match the optimal ones, but they are very close and closer to the optimal ones than the Bisection Algorithm and TOPP-RA in the evaluation from Barnett and Gosselin (2020). Trajectory durations are similar to the values reported by Barnett and Gosselin (2020), which gives credit to our claim of time-optimality, as their evaluated algorithms have proofs of time-optimality.

An example of a longer path with a larger number of waypoints is shown in Figure 4.8. The iiwa's maximum joint constraints are used, and for the path with 30 randomly selected

waypoints and 3583 grid points, a time-optimal trajectory with 58.9 seconds length is found in 2534 µs. This still corresponds to 0.71 µs computation time per grid point, almost identical to the evaluation above for the generic manipulator. This shows that this metric can be used to predict the expected computation time. A future extension to the implementation can be to limit the number of grid points during pre-processing to enforce an upper limit for the computation time in one control cycle, as discussed in Section 4.1.2.

We carried out further tests of the parameterization to verify that a valid solution is always found for every input. For a six-DoF robot with the unified generic constraints used for the generic seven-DoF case, one million random five-waypoint paths with zero start and goal velocity were generated, and the algorithm was always able to find an admissible trajectory that fulfills the necessary condition for time-optimality. For the same setup, 43.000 paths were generated, this time with the same small start and goal velocity as in the above evaluation. This test was also successful. Only for four paths, the specified velocity of the start state was too high, which is an expected error if the random path contains a sharp corner right at the beginning of the path. This completes an exhaustive evaluation of the algorithm to validate its stability and general applicability to arbitrary path shapes.

4.6 Summary

This chapter presents the concept of a real-time-capable TOPP algorithm. The state of the art related to offline planning is summarized in Chapter 2.2 with an emphasis on approaches that provide promising starting points for the development of an online and real-time solution. The concept is presented as a combination and extension of several existing approaches in the literature. The pre-processing of waypoints to achieve a continuous path representation is detailed. A strategy is also described to make the integration process along the path real-time capable and to achieve completeness of the algorithm. The math and algorithmic structure were added in the next sections, including support for asymmetric joint limits and handling of special cases like singularities during the trajectory generation. Finally, notes on the implementation and the testing procedure are provided, including extensive experiments to evaluate the algorithm. Using large sets of randomly generated paths, no unexpected failures occurred, kinematic limits were respected at every grid point, and the necessary conditions for optimality were fulfilled.

The experiments show that it is possible to generate in one control cycle of less than $0.5\mu s$ trajectories that are several seconds long, so a large look-ahead and pre-computation of time-optimal solutions for a comfortable portion of the requested path is possible. This ensures that even for partially parameterized paths, a portion of the globally time-optimal solution is found. This means that the next seconds parameterized are identical to the globally optimal solution, as the MVC will already be hit several times, or several path acceleration switches are covered. The computation time depends on the number of grid points. The number of grid points and computation time per grid point can be used to determine an upper bound for the computation time of the algorithm, and independently of the number of waypoints, a fixed path length can then be parameterized in every control cycle. A benchmark similar to our evaluation can be used to determine the worst-case computation time per grid point.

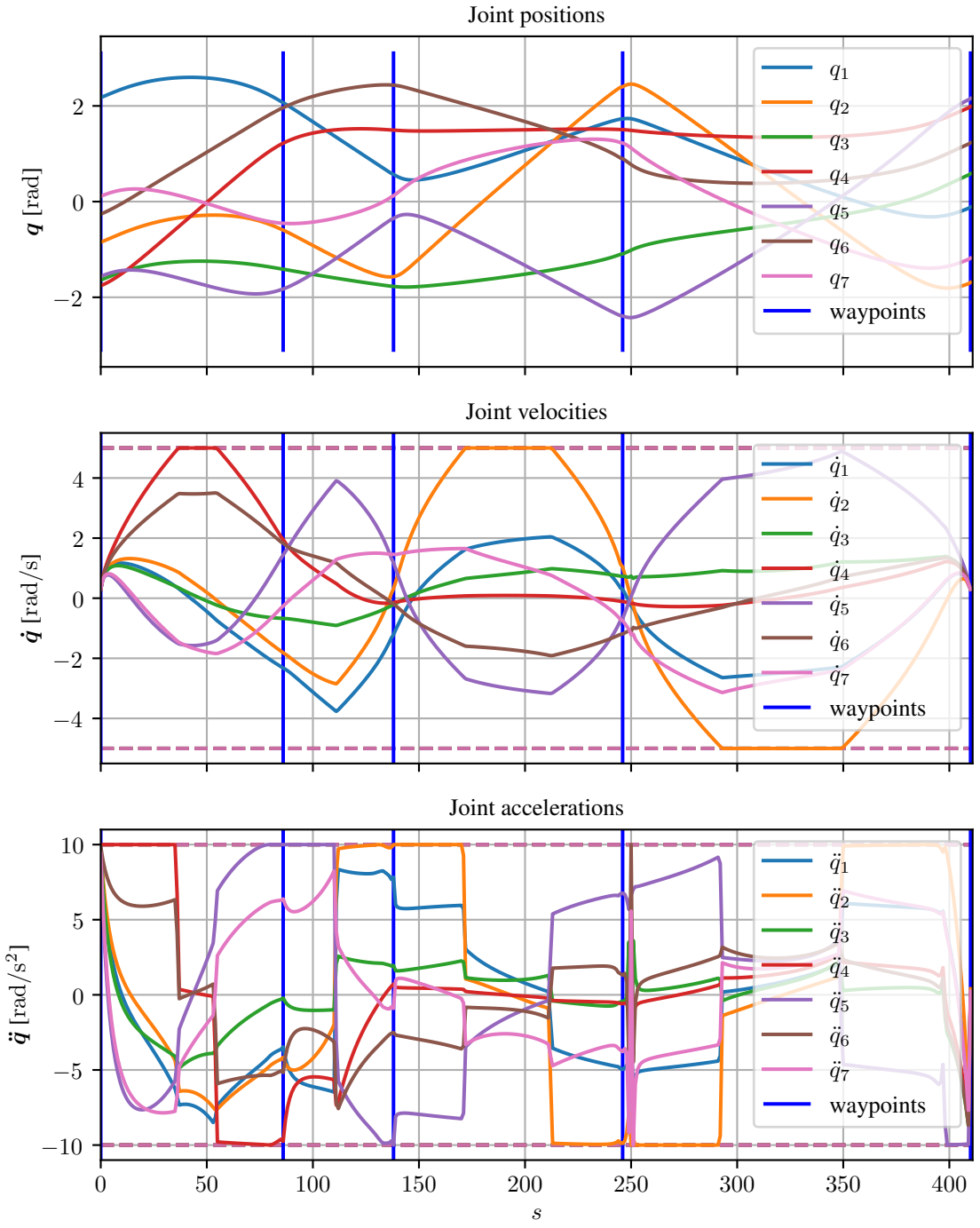


Figure 4.7: Joint space trajectory result for a 7-DoF given pseudo-random path from the five-waypoint test set. The unified generic constraints for joint velocity and acceleration are applied. The vertical blue lines mark the locations of the waypoints along the path.

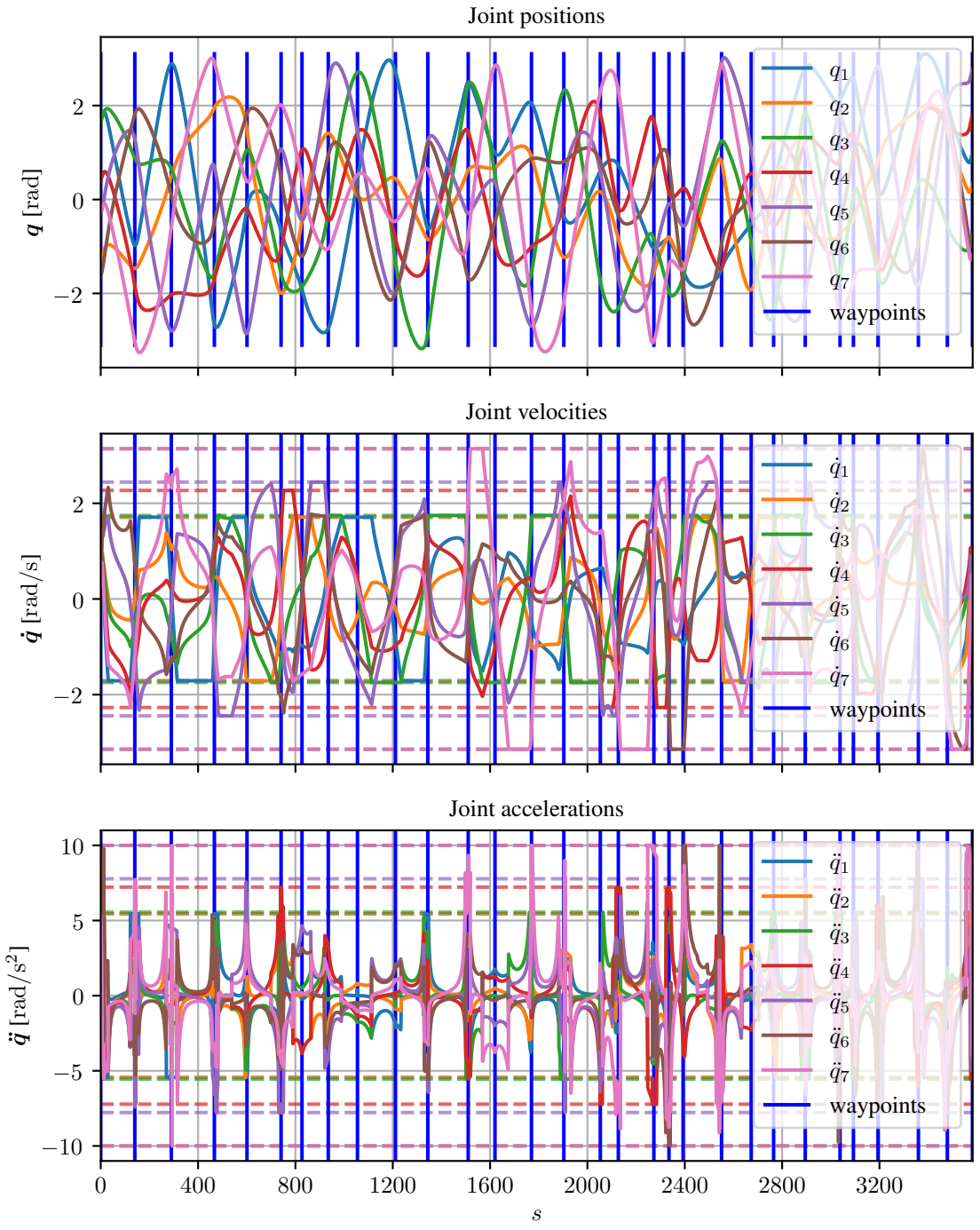


Figure 4.8: Joint space trajectory result for the 7-DoF iiwa path test with 30 waypoints. The manufacturer's maximum constraints for joint velocity and acceleration are applied and are marked with horizontally dashed lines in the color of the respective joint. The vertical blue lines mark the locations of the waypoints along the path.

CHAPTER 5

Application: A Remotely Accessible Robotics Lab

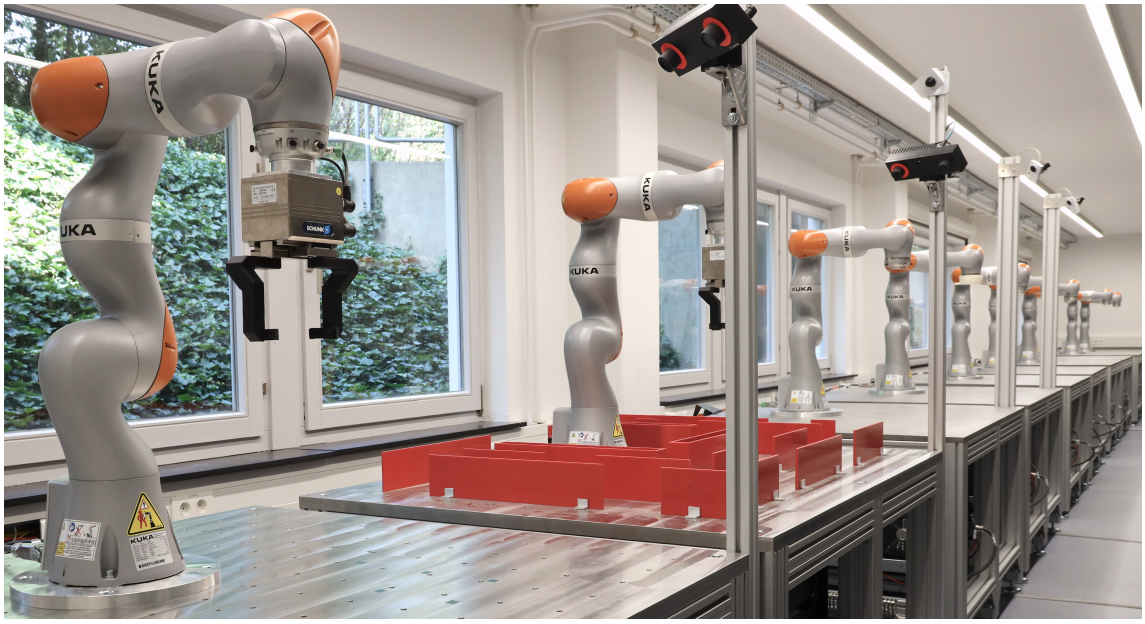


Figure 5.1: The newly designed lab room with all ten robot work cells. The Robot Learning Lab logo is shown above the image.

The Robot Learning Lab (RLL), as shown in Figure 5.1, is a remote-access robotics testbed explicitly designed to make industrial lightweight robotic manipulators more readily available to students and researchers. It is the third contribution of this thesis and also serves as an application of the first two algorithmic contributions within its reactive and safe motion planning framework. These first two real-time motion planning contributions to this thesis (see Chapters 3 and 4) further improve and facilitate safe remote access capabilities to such a lab. The developed free and open-source framework and architecture enable the lab to provide parallel access over the Internet to robotic work cells. It has

a publicly available website and web interface at rll.ipr.iar.kit.edu. Users can create an account and start using the lab right away via an online editor or by submitting demo projects.

The ten robot work cells in the lab are flexible, allowing for various experimental setups for educational purposes or research challenges. The distinguishing feature of the Robot Learning Lab, compared to other remotely accessible testbeds, is its emphasis on enabling the development of applications with complexity for research or educational purposes, while maximizing scalability through automation and minimal invasive safety/security measures. Other testbeds lack flexibility or the ability to write code in general-purpose programming languages to qualify as development platforms, or they cannot support a large number of users and multiple robotic work cells, since experiments are not executed in a fully automated manner and high robot availability in the lab is not ensured (see Section 2.3). In this chapter, we discuss how the Robot Learning Lab is structured and, in particular, how it constitutes an *easy-to-use*, *flexible*, *safe*, *secure*, and *automated* remote-access development platform.

Publications Related to this Chapter
Parts of the work presented in this chapter have been published in:

- **Wolfgang Wiedmeyer**, Michael Mende, Dennis Hartmann, Rainer Bischoff, Christoph Ledermann, and Torsten Kröger. Robotics education and research at scale: A remotely accessible robotics development platform. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3679–3685. IEEE, 2019.

The previously published paper (Wiedmeyer et al., 2019) introduced an initial lab setup with two robot work cells and a rudimentary web interface draft, as it existed in the summer of 2018. In this chapter, the developments of the following years are added, including additional use cases. This covers the addition of eight more robot work cells, and the lab moved to a new space specifically designed for it. A public-facing web interface with multiple options to interact with the lab, user management with free registration open to the public, multiple extensions and improvements to the automated submission processing and motion planning, and further safety and security enhancements were added. The additions also result in significant improvements to ensure low maintenance during regular operation and increase usability and accessibility. Some of the additions, such as ready-to-use development environments and usability improvements to the programming interfaces, have been mainly implemented in the supervised student thesis by Weinreuter (2020).

The existing primary application is in education and public outreach, but the design also factors in requirements for a powerful development platform targeting researchers. Several options for remote robot programming in Python, C++, or block-based are available, including a web-based development environment and easy submission of programs to the lab with one click or one shell command. The processing pipeline is intended to be used with several robot work cells to run submissions for different experimental setups in parallel.

In Section 5.1, we present the design considerations that guided the development of the Robot Learning Lab, and elaborate on the hardware setup in Section 5.2, and the software

architecture of the lab in Section 5.3. Finally, usage of the testbed is reported in Section 5.4.

5.1 General Design Considerations

As a robotics development facility that can be accessed remotely, the primary goal of the Robot Learning Lab is to broaden access to state-of-the-art industrial robots for both education and research. To achieve this, it must satisfy several high-level design requirements focused on accessibility, scalability, automation, and the safe and secure execution of source code:

- Provide enough flexibility to host research projects and challenging robotics competitions, while also being able to teach students the foundations of robotics with a low barrier to entry.
- Enable fully automated execution of code submissions with an efficient job processing pipeline that can handle a large number of users and allows for parallel execution of jobs in multiple robot work cells, and that requires minimal maintenance and monitoring by a human operator.
- Integrate safety and security mechanisms to prevent damage and misuse in the Robot Learning Lab by ensuring safe robot operation, including collision avoidance and isolation of user-submitted code.
- Allow users to develop applications using a simulation that closely replicates the lab environment, and ensure these applications can run in the lab without requiring any further modifications.
- Enable interaction with and data retrieval from the lab through a website and web API that can be easy-to-use for novices and supports different client types (e. g., web, mobile, command-line).
- Make it easy to design further projects with their own experimental setups in the robot cell and custom software interfaces.

Each project has its own experimental setup in a robot work cell and the needed custom user-facing software interfaces. Users can submit their own developed software applications for a specific project that is hosted in the lab. Fulfilling the above requirements allows beginners to write simple programs in a web-based development environment with a simulation of the robot cell and trigger a run on the robot in the lab with one click, like submitting a print job. More experienced users can use the lab with several robot work cells similar to a continuous integration platform (Duvall et al., 2007) or a hardware-in-the-loop robotics testbed (Afzal et al., 2020). The code is automatically executed in a preconfigured environment inside one of the lab's robot cells. Additionally, users do not need to worry about any possible damage to the lab setup if their code misbehaves, because sufficient safety measures are put in place by the lab framework and individual project designs. From the perspective of the lab operators, a low maintenance burden can be ensured and security risks mitigated. It is then sustainable for a small team of operators to maintain several robot work cells and provide continuous public access to a large group of users, without the need for constant monitoring.

A remotely accessible robotics development platform can provide multiple benefits in terms of public outreach, education in robotics, and the broader robotics research community. When lab users are not required to be on-site and can be anywhere with internet access, a far greater audience can be reached. Preconfigured demos are set up in the lab that showcase the platform's capabilities. These can then be started from anywhere, e. g., lecture halls or conferences/trade shows, and live-streamed without the need to have a full demonstrator set up on location. High-school students or generally interested members of the public can solve introductory programming exercises or simply play with the robots. The robots can serve as a visualization tool and motivational factor to teach programming and algorithms. The experience of remotely working with robots and seeing how your code helps the robot to interact with its environment can even inspire some to pursue a career in robotics.

In addition, it should be possible to host competitions for the robotics community using the lab. The processing pipeline automatically evaluates and rates submissions, while participants can inspect results in the web app. It should also be possible to use the lab as a benchmark platform with a clearly defined experimental setup and evaluation routine. As simulation environments still do not sufficiently reflect reality, e. g., for manipulation tasks (Collins et al., 2019), hardware experiments are a must to assess the performance of newly developed methods. Researchers can test their algorithms and compare them directly with other approaches, which were also tested in the lab.

5.2 Hardware Setup

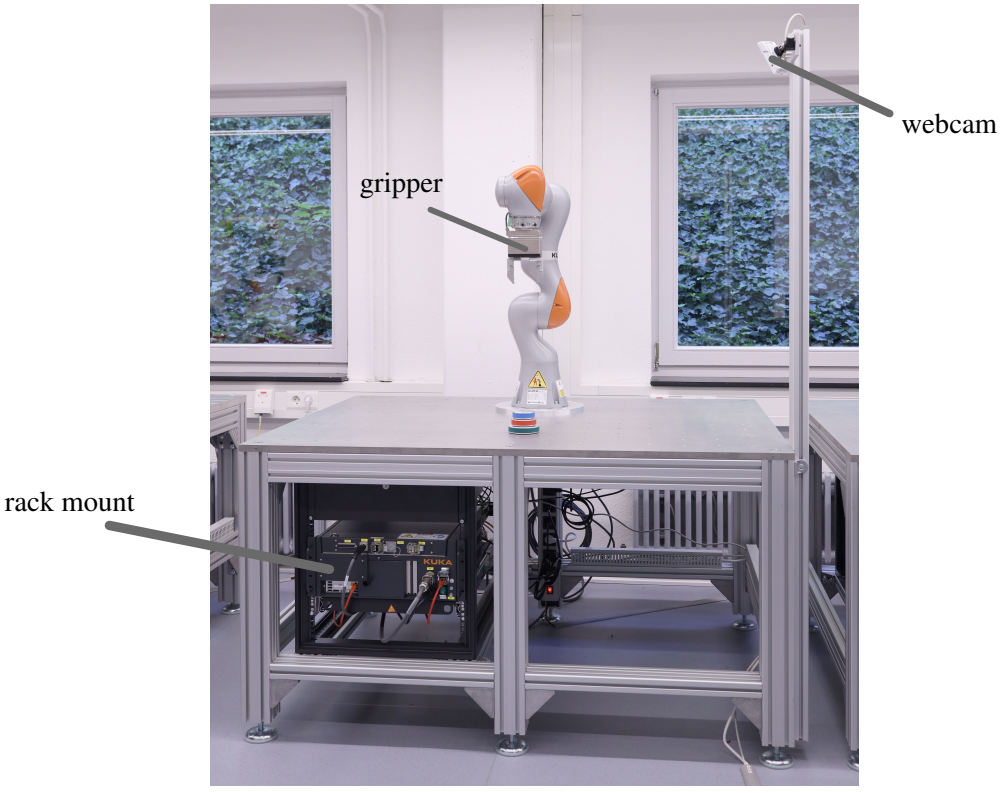


Figure 5.2: A single robot work cell in the lab. The rack mount below the table can be seen, and a webcam is mounted to a profile at the side of the table.

The lab consists of ten robot work cells with the exact same basic hardware setup, as shown in Figures 5.1 and 5.2. Each setup is outfitted with a Kuka LBR iiwa 7 R800 lightweight robotic arm. The iiwa is a seven-joint redundant industrial manipulator that features integrated force/torque sensors in every joint (KUKA AG, 2026). A SCHUNK EGL 90-CN gripper (SCHUNK SE & Co. KG, 2026) is attached to the robot's end-effector in two work cells. In addition, a Weiss CRG 200 (Weiss Robotics GmbH & Co. KG, 2026) gripper with more fine-grained grasp sensing was also added for one robot. The robot is mounted on a table with a plate made of stainless steel. The steel plate has a surface area of $120\text{x}160\,\text{cm}$ and features a grid of threaded holes spaced at $100\,\text{mm}$ intervals, enabling secure mounting of various experimental setups. Using an adapter plate and dowel pins, the robot can be positioned at different locations on the table according to this grid, ensuring precise placement. Due to its weight, the steel plate effectively damps vibrations at the robot base, allowing the robot to perform highly dynamic motions with high accuracy. The grippers have custom 3D-printed gripper fingers made from plastic to minimize damage to the gripper and table in the case that software safety precautions fail. The fingers act as predetermined breaking points because they do not pose a significant loss in case of damage, as they can be quickly reprinted.

A vision system, such as a stereo camera, can be mounted on an aluminum profile on the side of the table. It enables visual monitoring of the work cell, and the camera's point cloud data can be utilized in experiments. Mounted at the top of the profile is a network camera that is used to record or live-stream the experiments. In addition to mounting directly on the steel plate, additional hardware extensions can be easily added by mounting to the support profiles of the table.

The KUKA robot controller cabinet is mounted in a rack below the table and can be pulled out for maintenance like a rack server. Each table has its own network switch to which the robot controller and other sensors in the work cell can be connected. The table switch is connected through glass fiber to a central switch in a server room. Several physical servers are hosted in the server room for the software infrastructure, which is described in the next section.

5.3 Software Infrastructure and Design

The primary software components responsible for running the Robot Learning Lab can be divided into three main categories: applications for the processing of submissions and user interaction with the lab (regarding submission and data retrieval), components related to safety and security, and components that facilitate application development and project design.

5.3.1 Submission Processing and Interaction with the Lab

The Robot Learning Lab can process a submission entirely automatically. When a user creates a new submission via the web interface, this action initiates the scheduling of a single run of the submitted code. In the following, such a submission is referred to as a "job". Every job is tracked in the system by a unique job ID. Jobs need to be submitted for a specific "project" and the submitted code, which corresponds to a job, should provide

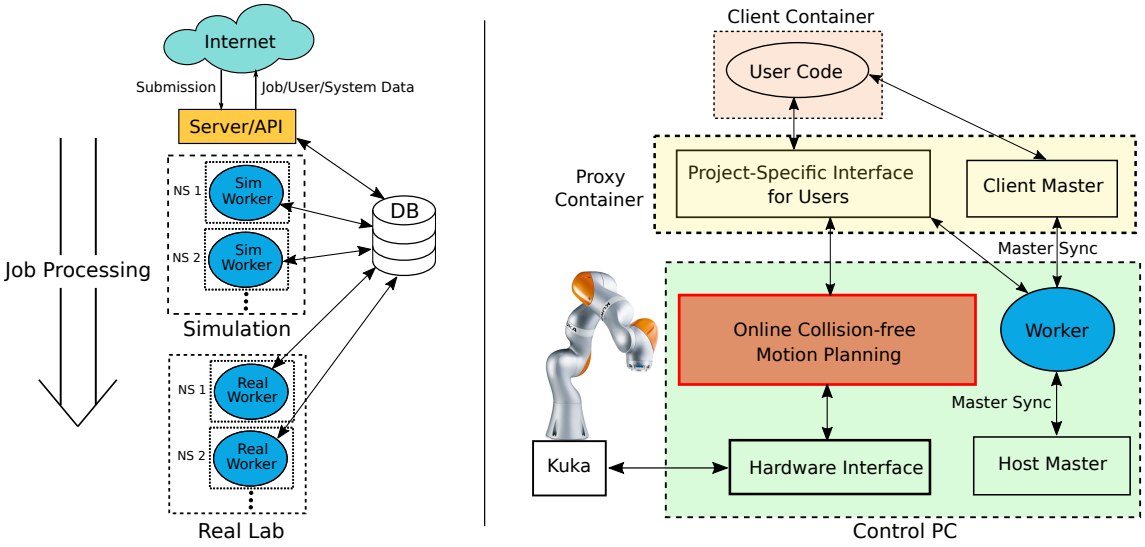


Figure 5.3: Overview of the job processing pipeline and the user code execution. The left figure shows the job processing pipeline that includes the server with the website and web API, and the workers for simulation and the real robots. The right figure illustrates how user code is executed by the worker in an isolated environment with online checks for collision-free execution on the robot and synchronization of network resources between the lab's host master and client master. The figure is adapted from Wiedmeyer et al. (2019).

solutions to the tasks that are defined within a project. One possible task is to implement an inverse kinematics solution for the robot, or to recognize the poses of objects that the robot is supposed to grasp. A project is defined by a certain experimental setup in a robot work cell and by its own software user interfaces that are supplied by a project software package. The system is aware of which projects the robot work cells are currently configured for and processes the jobs accordingly. The jobs are processed by "workers" which will be further detailed in the following (see Figure 5.3 for an overview).

Outside communication with the Robot Learning Lab is possible through a RESTful web API and website, based on the Tornado framework[1]. Due to its non-blocking network I/O features, the server can handle a large number of open connections and allows users to submit experiments and retrieve data in parallel. See Figure 5.4 for an overview of the website. Keycloak[2] is leveraged for the user registration, authentication, and management. It is possible to get the current status of a job and data for the job after it has finished in the web interface. Jobs are queried by their ID, which is provided to the user when the job is successfully submitted. For finished jobs, log files, a video recording of the job, and optionally additional data collected during the job run are provided on the website.

The status of the job is reflected by several status codes like "downloading code", "building", "finished". When a job is submitted and waiting for execution, the position in the queue is displayed on the "My Jobs" page. If the job is actively running in the lab, a link is provided to the live video stream from the network camera in the corresponding robot work cell where the job is being carried out. A separate statistics page provides an overview of the lab usage. Users can get familiar with the job processing by submitting demo implementations on the "Demos" page. Their own code can be submitted by users

[1] http://www.tornadoweb.org/en/stable/
[2] https://www.keycloak.org/

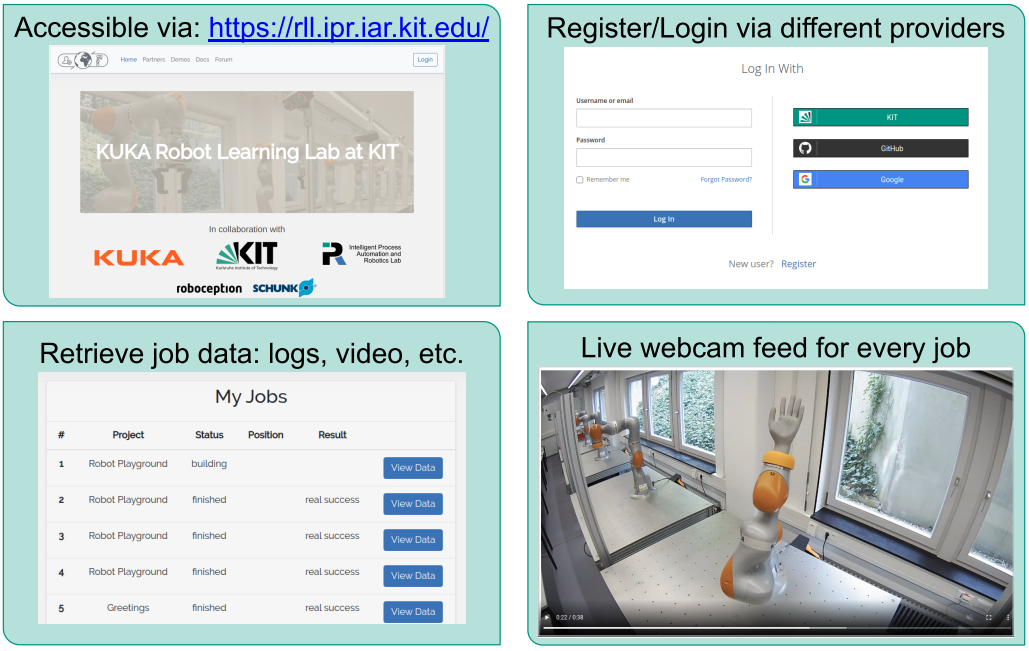


Figure 5.4: Website overview for registration and job processing. The homepage provides initial information about the lab and links to documentation. Users can register by creating an account or using a single sign-on provider like the KIT Shibboleth for students and employees. An overview page shows users all submitted jobs and their status, and allows them to view and download job data. When the job is currently running in the lab, users can view it with a live webcam feed.

using one of the available development environments (see Section 5.3.3), either by clicking on a button or by executing a shell command. A compressed archive of the user's code is then created in the background and submitted to the API. If users want to upload from outside the RLL web development environment, they can download and locally use an API access token for their account, which authenticates them during submission. If a submitted job is not executed quickly and has to wait in the queue, users receive an email notification when the job starts to be processed.

A job is first executed in simulation. The simulation environment is the same one that is provided to users (see Section 5.3.3). The primary aim of the simulation run is to verify that the user code can be executed by the pipeline. Section 5.3.2 provides an overview of the environment in which the user code runs. The simulation check provides users with early feedback if they made any mistakes that could prevent their code from running in the lab, and it ensures that the usage time of the robots is not wasted by erroneous user code. Possible safety or security issues with the code can also be detected at this stage. Depending on the project configuration, the simulation check can also be used to prevent user code from running on the robots if it is not able to reach certain goals (see Section 5.4.2 for an example). The user management allows assigning a "trusted" flag to a user, and jobs from users with this flag bypass the simulation run and are directly executed in the lab.

Simulation runs are controlled by sim workers, and code executions for the robot work cells in the lab are handled by real workers (see Figure 5.3). The expressions "sim" and "real" are typically employed to distinguish simulated work cells from the actual robot

work cells in the lab. The workers handle downloading the code, compiling the user code, and running it. They also retrieve and store job-related data, including metadata such as job result codes (e. g., "build failed", "sim success", "launching project failed"), and they initiate a reset of the robot work cell to its initial state after each job execution. The server and workers use a database[3] that stores the job metadata and necessary user data. The server provides an internal API for the workers to which the workers connect and retrieve the necessary data for a job, including the archive of the user's code. The workers are launched with configuration flags, including which project they should process jobs for, or if they should process jobs for simulation or a specific work cell in the lab. They ask the internal API for jobs that were submitted for this specific project, and select the oldest submission as the next job to be executed. As only a network connection to the internal server API is required, the server and the workers can be distributed among several hosts.

The Robot Learning Lab leverages the ROS robotics middleware (Quigley et al., 2009). Every worker instance is isolated in its own ROS namespace, together with the user code execution environment and necessary ROS network resources and parameters. This allows multiple sim workers for parallel simulation checks and several real workers for any number of robot work cells to operate simultaneously and handle jobs in parallel, using a single ROS master for the whole lab.

The worker and server components in the job processing pipeline are hardware-agnostic and are compatible with any type of robotic platform. Alongside this thesis, their source code is provided as free and open-source software in the hope that it will help others to build their own remotely accessible robotics testbed.[4]

5.3.2 Safety and Security

The system is able to detect failures that cannot be automatically corrected by itself. Such "internal errors" are recorded during job execution when operations that are expected to always succeed fail, or when the system encounters an unexpected or undefined environment state (in simulation or on the real system) from which it cannot recover to the required state. Internal errors are also raised if resetting the environment after job execution does not succeed. While the system and project design aim to reduce the chance of such internal errors, users may still be able to provoke them. If a worker detects an internal error, it immediately stops its job execution environment, including the user code, notifies the operator, and terminates itself so that no further jobs are executed in the environment. This enables an operator to review the error and guarantees that the system does not continue running in an unintended state, which could result in hardware damage.

For collision-free planning of robot trajectories (the red box in Figure 5.3), the motion planning framework *MoveIt* (Sucan and Chitta, 2026) is used. MoveIt is able to construct a collision model for the robot cell from the environment description that is defined for every project. We use a custom collision model for the robot consisting only of spheres to speed up the path planning. The real-time inverse kinematics, the first contribution of this thesis in Section 3, is used here to transfer commanded Cartesian poses or paths to joint space for trajectory generation. The inverse kinematics ensures reproducibility of

[3] MongoDB is used for the database: https://www.mongodb.com/

[4] The pipeline's core source code repository: https://github.com/KITrobotics/rll_stack
Further needed repositories are available at https://git.ipr.iar.kit.edu/rll

paths without drift effects of the arm angle, smooth adaptation of the arm angle, and that joint position constraints are always respected. The project-specific interface for users contains motion planning abstractions with further safety checks to verify commanded paths and poses from users, and has its own modular state machine, which ensures that the user's code can only request robot motions when the system is in an error-free running state and no concurrent planning requests are allowed. This software component is further described in Section 5.3.3. For trajectory generation, we use the TOTG algorithm introduced by Kunz and Stilman (2012).

The control cabinet for the iiwa robot provides the KUKA Fast Robot Interface (FRI), which allows for real-time control of the robot via a network interface. We implemented a real-time ROS hardware interface for position control with 1 kHz (and optional hybrid position-torque control) using the FRI API in order to use FRI together with MoveIt as part of our motion planning interface (see Figure 5.3). Reflexxes (Kröger, 2011) is used in a custom real-time ROS trajectory following controller to interpolate the commanded trajectories with added jerk limits. It is ensured that only velocity-, acceleration-, and jerk-limited trajectories within safe constraints are sent to the KUKA FRI interface, to avoid the cabinet needing to trigger a safety stop, e. g., due to an inadmissible position control command causing large jerk. Various additional safety measures for the robot workspace are configured on the KUKA control cabinet, but these are intended as a last resort, as they always trigger a safety stop, which then would need manual intervention by a lab operator to start the job execution again. We try to catch as many safety violations as possible in our framework so the job processing can reset the cell and resume operation on its own automatically. An external real-time system controller for the control cabinet was also implemented. The worker can leverage this controller to check if the robot is running without error, and when the worker is paused, it can deactivate the drives and activate the hardware brakes of the robot, thus pausing the robot. It can also pause the robot this way when an internal error occurs.

For security reasons, the user code is compiled and run inside a Docker[5] container without root privileges. Docker containers are more lightweight than virtual machines because they use and share the host operating system's kernel directly. The system resources for containers, such as the number of CPU cores, RAM, and disk space, are restricted. For every project, an image with all the needed software packages and configurations can be created, and the worker sets up a proxy and client container based on the images and additional configuration options. The client container runs the downloaded user code, and the proxy container provides the project-specific interface for the user code and runs the separate ROS master for the client. The worker places the downloaded user code in the client container and extracts log files from both the proxy and client containers. When a submission was first successfully built and checked in simulation by a "sim" worker, the Docker container with the submission's code is saved, and the image is stored in an internal RLL Docker registry. A "real" worker can then pull the image from the registry and run it directly without needing to compile the project code again. All workers also pull base images for the individual projects for the proxy container from the registry to ensure all job execution environments have the same clean state.

The proxy container is primarily used to block user code from directly accessing both the host network and the lab network. If users were able to call arbitrary ROS network

[5] https://www.docker.com/

resources[6] or even issue commands straight to the robot, they might circumvent all safety mechanisms and inflict severe damage on the robot's work cells. For each task, the worker launches a fresh client container and connects it to the same Docker network used by the proxy container. The client container is restricted to communicating only with the proxy container, whereas the proxy container can additionally access the lab network. The project-specific interface inside the proxy container offers all the ROS network resources the user needs for the project (see Section 5.3.3). The robot control interfaces (e. g., FRI) are additionally isolated in a dedicated virtual LAN with a firewall to which only select control PCs have access.

The separate ROS master within the proxy container enables the user code to look up the IP addresses and ports of the interface nodes, as well as to register its own services, topics, actions, and parameters. The interface inside the proxy container relies on the host's ROS master, allowing ROS nodes running on the host to connect to it. The worker must mirror the registrations that the interface performs on the host master to the client master so that the user code can access them. The interface provides resources that should only be accessible from the host, e. g., the actions for job execution and environment reset. These resources are not synchronized to the client master, but a malicious user can still try to call these resources by using the same IP address and ports of registered resources on the client master. By using a simple authentication scheme, we are preventing the client code from gaining access to these resources. The registrations on the client master by the client code are selectively synchronized by the worker to the host master, so that the interface can communicate with the client-implemented code (see Figure 5.3). The worker reads the registrations that need to be synchronized from project-specific configuration files.

The worker uses a custom master synchronization mechanism instead of existing multi-master solutions, because those are designed for discovering and synchronizing ROS systems over unreliable networks and do not offer the flexibility to synchronize specific registrations on an on-demand basis. Multiple ROS security extensions have been proposed (White et al., 2016; Dieber et al., 2016, 2017), and ROS 2.0 includes similar security features (DiLuoffo et al., 2018). However, the aim of these security enhancements is to allow for fine-grained access control and transport encryption, which would require a significant configuration overhead for every project, and users would not be able to add their own ROS network resources. The user code isolation in its own Docker container and via a proxy container separates the user code from the ROS system of the lab and still allows for the relay of any ROS host resource to the client container. Even if user code succeeds in crashing the interface or the client master within the proxy container, the safety of the system is still ensured.

Lab operators with administration privileges in the user management get access to an admin console in the web interface where they can see the workers currently running and their state ("online", "paused", "internal error"). They can pause job execution for a worker or shut down a worker. For "real" workers, a safety stop can be triggered from the admin console. A WebSocket connection with low latency is relayed by the server to the worker and triggers an interruption of the real-time control connection to the robot, which in turn triggers a safety stop on the robot controller. The admin console allows a lab operator in the lab to manage job execution on a touch screen, or even remotely, without having to have several physical safety stop buttons at hand for several robots. The

[6] See the ROS wiki for an introduction to the peer-to-peer network concepts of ROS: http://wiki.ros.org/ROS/Concepts

workers need to send heartbeats to a worker monitor as part of the server, and if a worker does not send a signal, the admin console reports the worker as “crashed”.

A suite of unit and integration tests, and a continuous integration pipeline for the lab’s code projects, ensure that the safety and security measures work properly and that regressions are caught during the development work.

5.3.3 Application Development and Project Design

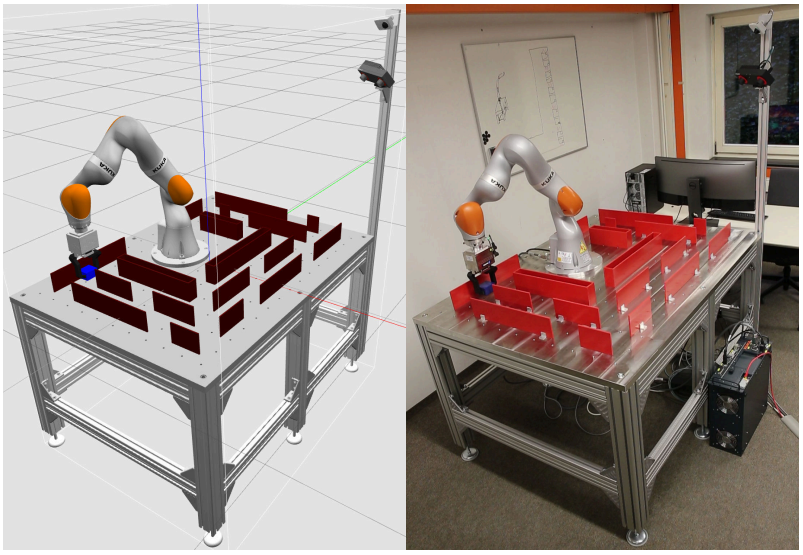


Figure 5.5: The simulation and the real robot cell side-by-side. The figure is taken from Wiedmeyer et al. (2019).

Users can write their own applications for a specific project that is hosted in the Robot Learning Lab. A page with documentation on how to use the lab and the different available options to develop applications is available[7]. Tutorial videos and video showcases of the lab are also available[8]. In addition to letting users participate in projects, we also want to make it straightforward to design new projects for the Robot Learning Lab. A project design can have its own experimental setup in a robot work cell, and custom user-facing interfaces can be implemented.

Applications can be written in C/C++, Python, and Bash. A web-based development environment additionally allows the use of a simple block-based programming interface (see Section 5.3.3). All software packages that users need to develop applications for the Robot Learning Lab are available as free and open-source software. The complete set of ROS packages is available to users, and additional libraries can be added to the Docker container images if needed. Gazebo[9] is used for the simulator, and an exact model of the robot work cell can be used for development (see Figure 5.5). Thanks to Gazebo’s ROS integration, the hardware interface of the physical robot (see Section 5.3.2) can be substituted with a control layer dedicated to the simulation. In this way, identical user-facing interfaces can be provided to control the simulated and real robot.

[7] The documentation can be found at https://rll-doc.ipr.iar.kit.edu/

[8] The videos are available on the lab’s YouTube channel at https://www.youtube.com/@kukarobotlearninglabatkit

[9] http://gazebosim.org/

In addition to a video, users can retrieve the build log, the log file that their code generated during execution, and the part of the log file from the interface that corresponds to their job run, both for the simulation and the real run. These files usually suffice for debugging and evaluation.

The software architecture of each project needs to consist of two components: user-facing interface nodes and the user's own code. Although users can run both on the same PC they use for development, the interface nodes are meant to be running in the proxy container, and the user code runs in the client container in the Robot Learning Lab (see Section 5.3.2). User code is written by the users of the Robot Learning Lab and makes use of ROS resources that are exposed by the interface nodes. These interface nodes can either introduce new functionality themselves or simply forward existing resources, such as sensor data.

For each project, the user-facing interface is expected to provide routines for starting and stopping a job, as well as for resetting the environment to its initial state. Resetting the environment should not only move the robot back to its home position, but also ensure that all manipulable objects are placed back in their original poses. The interface should additionally include some means of controlling the robot. To ease the development of interfaces and facilitate the project designs for the Robot Learning Lab, we created an SDK that contains the simulation environment and libraries for interface development[10]. The libraries provide abstractions for MoveIt and ROS functionality and directly work with the simulation environment. For controlling the real robot, a slightly modified library with driver abstractions and improved trajectory generation, but with the same API, is used, so that no changes are required to the project-specific interface code. The move interface library included in the SDK defines a set of services that allow users to request target poses in joint space or Cartesian space, including randomly generated joint space targets. For target poses in Cartesian space, linear or point-to-point movements can be requested, and the arm angle (see Figure 2.1) can be controlled. A service for picking up or placing a graspable object is also available.

In addition to the move interface library, a client library for users called Move Client is also included. It abstracts ROS functionality so that users do not need to know anything about the communication in the background or setting up a ROS node, and can send a command to move the robot with a single line of code. During job execution, the library also handles the start of the program, where the interface notifies the client that it can start sending commands. Example code and documentation of available commands are provided to make it easy to start programming. A project called Robot Playground is available, where users can get familiar with the Move Client and generate movements in an empty workspace.

Development Environments

Users can select a suitable development environment for themselves among three options, depending on their level of expertise: manual installation, virtual machine, and web editor. The most advanced users, such as researchers or experienced university students, may already have a Linux- or even ROS-based system at hand and want to use the RLL packages with their familiar software development workflow. Using the documentation, they

[10]The Robot Learning Lab SDK is available at https://github.com/KITrobotics/rll_sdk

can set up an RLL workspace with a few shell commands and are ready to submit a Hello World example program to the lab. This option allows users the most freedom to design their solution inside a ROS package using Python or C/C++, which is then uploaded as an archive to the lab using a shell command.

The second option is a preconfigured virtual machine that users can download and run using VirtualBox on their own machine. It has a GNU/Linux desktop environment with a similar look and feel to popular desktop environments. It has an RLL workspace and code editor configured, and users do not need to be familiar with using the terminal, as desktop shortcuts are available to build, run the code in simulation, and submit it to the lab. They are also free to extend and configure the virtual machine as they like and use it similarly to the first option.

The third and easiest option to get started is the fully web-based RLL Web Editor hosted at the lab, which does not require any installation step by the user and provides a simple interface. It is integrated into the RLL website and available for logged-in users. It runs in a modern web browser using WebGL and has no special requirements for the user's machine. It can even run on a tablet. Figure 5.6 shows the two programming interfaces, Python and Block-based (Google LLC, 2026), where users can edit and save code in a single file. The Block-based editor allows users to get started without having to learn textual programming. It works by connecting interlocking blocks, where each block represents a statement or expression. The blocks are translated into Python code in the background. An overview page allows the creation, start, and saving of multiple editor sessions for different projects and coding interfaces.

Docker containers with the user session and simulation environment are hosted on a dedicated machine in the lab and are automatically started and stopped based on the current demand by an asynchronous control and user management component. WebSocket connections are relayed from the containers via the server to the user's browser. A similar security scheme, such as for the user containers for code execution in the lab on a real robot (see Section 5.3.2), is used to otherwise isolate the containers from the lab's network and restrict resources. The browser-based visualization also does not require privileged resources and access to graphics cards for the containers, as would be the case if remote access to desktop environments were provided. This design choice thus improves the security. Gazebo's browser-based visualization, Ros3DJS (Toris et al., 2015) and Rosbridge (Crick et al., 2017) are leveraged to stream the visualization of the simulation to the browser interface, while the computing power for the actual simulation is provided in the hosted infrastructure at the lab. A custom WebSocket controller was implemented to retrieve logging output and other information about the user's code execution in the container and handle the interaction with the web editor.

5.4 Usage of the Remote Robotics Testbed

With activities in the following three categories (conference demos, student challenges, and general long-term operation), this section presents the key usage capabilities of the Robot Learning Lab: safe and secure remote access for application development, along with automated, continuous operation. The primary intended use is for public outreach and educational purposes.

5.4.1 Conference Demos

The Robot Learning Lab was presented at the Udacity Intersect 2018 conference in Mountain View, California, USA. At a booth there, visitors were able to choose between two pre-programmed projects on a website and trigger remote execution of the selected project in the Robot Learning Lab. The version of the website used at the time is shown in Figure 5.7. The website relied on the Robot Learning Lab API (see Section 5.3.1) as its backend and offered conference participants status updates on their submitted jobs, a live video feed of the job execution, and access to the project's code log file. The job processing pipeline handled the distribution of project submissions across the two robot work cells.

One of the two project options enables the robot to perform a waving motion to greet visitors via the webcam video stream (Greetings project). The other one makes the robot play the Tower of Hanoi game with three disks[11]. Both projects use the default robot move interface from the Robot Learning Lab SDK (see Section 5.3.3). This initial use case of the lab proved that the job processing pipeline works robustly and the jobs are successfully distributed among the robots. The demos were later improved using the experience and feedback from the conference and are also available in the current version of the web interface. However, the usage rate at the conference was low and both robot work cells had the same experimental setup.

Later, the lab was presented at the "Bunte Nacht der Digitalisierung" 2019 in Karlsruhe, an event showcasing projects in the digital sphere. Visitors at the lab's booth in the city were able to remotely run several more demos than at the Udacity conference: the robot planning and executing a path for a grasp object through a maze (see Section 5.4.2), and the robot executing randomly generated point-to-point movements in two separate work cells. Up to six robot work cells could be used in parallel, and visitors were encouraged to try editing example code for the Playground Project to move the robot and test their modifications directly by submitting to the lab.

5.4.2 Student Challenges

This section highlights three use cases involving robotics software engineer students from an online university and secondary school students. A competition for students of the Udacity Robotics Software Engineer Nanodegree program was carried out in the summer of 2018. In this competition, students were tasked with developing a program that moves a grasped object through a maze to a target pose in the shortest possible time. Using a custom user interface, they could command 2D poses for the robot's end-effector and verify whether a straight-line path between two end-effector poses would be free of collisions. Navigation around corners in the maze is not possible without rotating the end-effector, which requires students to solve a 3D path planning problem. The students were free to implement any path planning algorithm, but most of the students chose a solution based on A* or RRT.

[11] The source code for both projects is available in the RLL public code instance at https://git.ipr.iar.kit.edu/rll/

The path planning project was designed using the Robot Learning Lab SDK[12], and the Robot Learning Lab API was integrated into the Udacity online student workspace. During a one-month period, students could submit their code and access the data of their jobs.[13]. For each distinct pair of start and goal poses, the leaderboards in the online workspace ordered the submissions based on how long the planning algorithms took to move the grasped object from the start to the goal.

The students gave general positive feedback, but some struggled with the complexity of the task. The experience from the challenge was used to improve the path planning project, create documentation, and make more tutorial videos. In the course of the public launch of the lab, the project was also made publicly available and registered users can write and submit their implementations, also using the RLL Webeditor.

At the beginning of 2020, the RLL virtual machine was used in two sessions of a programming class at the local St. Dominikus all-girls high school. The students had started to learn Python and solved different small Python exercises to move the robot, e. g., to draw symbols in the air. The Robot Playground project with the simplified Move Client was used as a base to design the exercises. The students were successfully able to use the available shortcuts in the virtual machine to test their code in simulation and submit it to the lab. Some struggled with configuring simple joint motions, as visualization of the robot's kinematics and workspace was lacking, and following error messages in the terminal also posed difficulties. The feedback was used to improve the instructions and also helped in the creation of the Webeditor, where such usability difficulties were addressed. The Webeditor was tested in several sessions of a local programming club for students from different schools, ranging from fifth to tenth graders. Both the block-based and Python programming interfaces were used. The students already had experience with a visual programming language and were able to solve exercises with some help from a local instructor. Server-side performance issues arose with up to 20 parallel Webeditor sessions. As a consequence, the Webeditor infrastructure with its Docker container setup was moved to a dedicated server machine with sufficient computing power and memory.

5.4.3 Long-term Operation

To demonstrate the ability to continuously operate the Robot Learning Lab for many users, we operated the lab for 274 minutes in the fall of 2018[14]. Throughout the experiment, we ran scripts that submitted a selection of solutions for the path planning challenge (see Section 5.4.2) and the Tower of Hanoi and Greetings projects (see Section 5.4.1) to the Robot Learning Lab under different usernames. The Tower of Hanoi and Greetings projects were executed in one robot cell, while various path planning project solutions were executed simultaneously in the other robot cell, where the maze was installed. The scripts generated more submissions than the lab could process immediately, so submissions had to be queued up. In this way, the experiment simulated a high usage rate of the lab with different work cell configurations.

[12] The source code of the path planning project can be retrieved at https://github.com/KITrobotics/rll_path_planning_project

[13] A video showing a selection of six job executions, which were submitted by six different participants, can be found at https://youtu.be/ExMFy833H8g.

[14] A time-lapse video of the experiment can be viewed at https://youtu.be/ZTAxb92EkbQ.

The lab officially opened to the public in the fall of 2019 and has been accessible to all types of users since then. Statistics up to the summer of 2025 are presented below. The stats page for logged-in users on the website provides current usage statistics. Over 230 users had registered and used the lab at least once. A large portion are KIT students and employees. The lab was initially promoted on social media, and many users registered from different backgrounds when they found the lab online. We only collected minimal personal data for privacy reasons, so no more detailed user statistics can be provided here.

Table 5.1: Number of submitted and finished jobs with the robots in the lab and their types, per project and in total. The numbers only show the use of the lab after its public launch, not previous experiments like the collaboration with Udacity. The exclusive demo projects "Tower of Hanoi" and "Greetings" had no regular submissions.

Project	total	regular submissions	demo submissions
Tower of Hanoi	253	0	253
Robot Playground	1118	630	488
Greetings	358	0	358
Path Planning	254	14	240
Getting Started	11	11	0
In total	1994	655	1339

Table 5.2: Final job result codes and their distribution among the projects in the lab, per project and in total. All jobs with the intermediate result code "sim success" were afterwards run on a real robot.

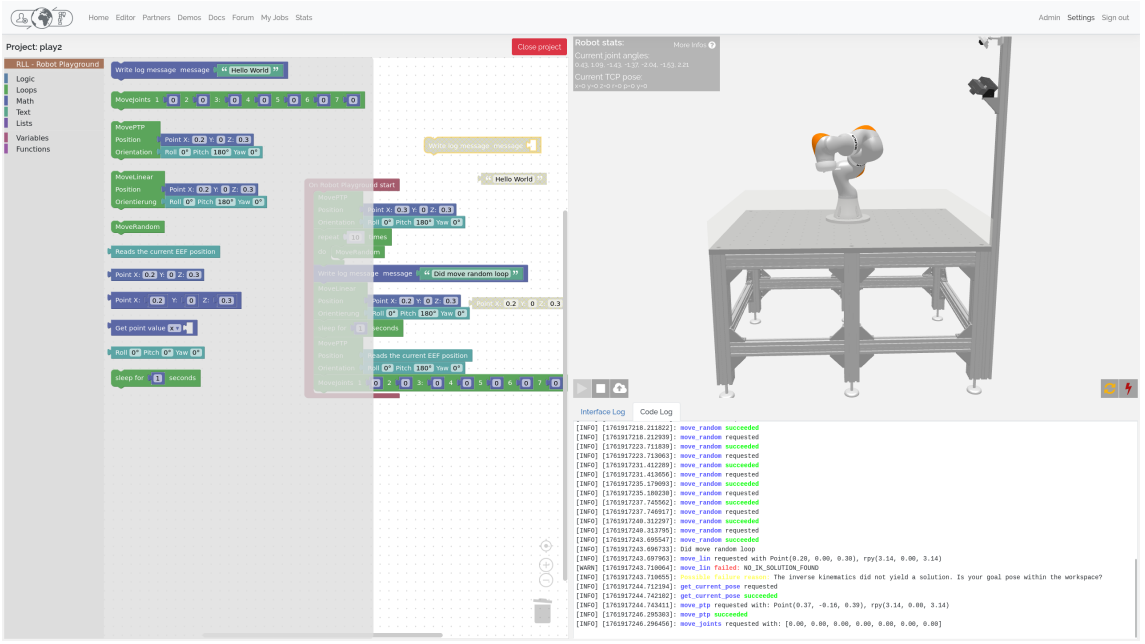
Project	total	real success	real failure	sim failure	real internal error	building project failed	total real
Tower of Hanoi	253	237	8	0	8	0	253
Robot Playground	1118	973	22	66	7	47	1002
Greetings	358	341	11	0	3	3	355
Path Planning	254	216	24	1	10	3	250
Getting Started	11	11	0	0	0	0	11
In total	1994	1778	65	67	28	53	1871

Table 5.1 shows the number of jobs submitted per project. Almost 2000 submissions were made in total. Two-thirds are demo submissions where users clicked on one of the available demo projects. Most of the regular submissions were for the Robot Playground project, where users could move the robot freely in an empty workspace using the Move Client interface. The Getting Started project is an initially evaluated project where students have to solve different tasks around motion planning and kinematics. The Path Planning project has attracted little interest so far from users in submitting their own implementation. Table 5.2 provides insights into the results of the submissions. 93% of the submitted jobs were able to pass all necessary checks and were run on a robot in the lab, 95% of them with a successful job result.

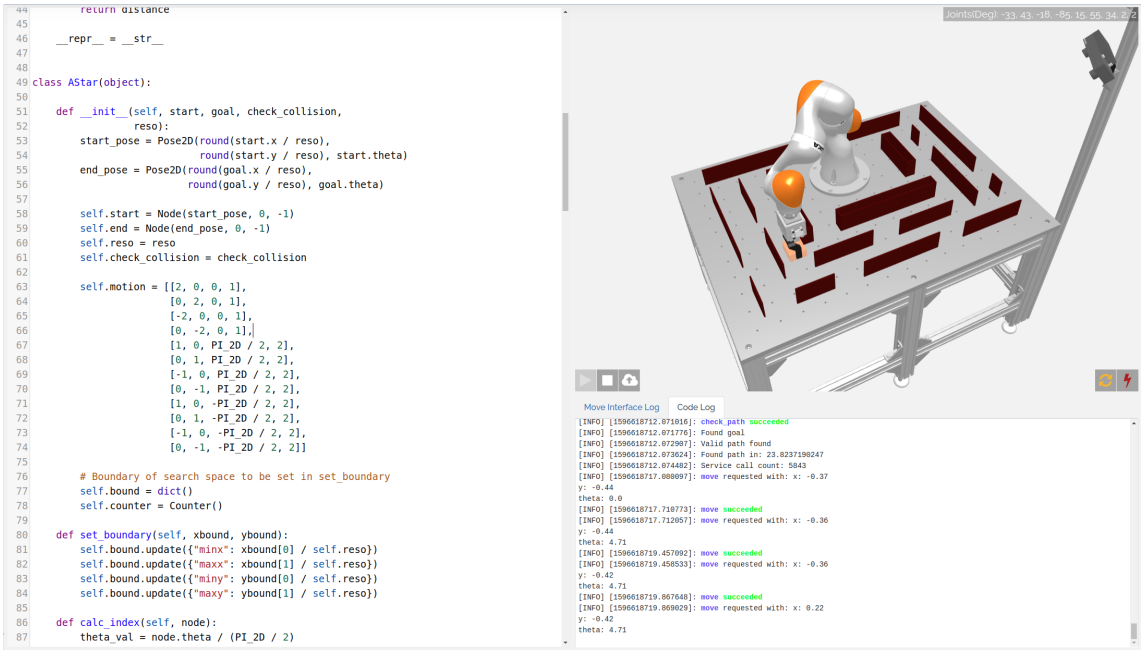
Only 1.4% of the submitted jobs had the final result “internal error”. However, the total number of lab internal errors is slightly higher. Some jobs with such a result were reset and run again during regular lab operation when the error was immediately obvious, e. g., when a faulty initial state was present in a work cell during initialization of a job execution session and the first executed job for a project failed with an internal error. The “sim internal error” did not occur in the final results for all projects, but it is also possible that a few jobs were reset after such an error when the simulation obviously misbehaved and could simply be restarted. The “reading project archive failed” error only occurred once for the Robot Playground Project and was likely due to a temporary storage error and not a faulty submitted archive. The results show that the lab can be reliably operated over several years and that users can make successful submissions.

5.5 Summary

In this chapter, we have detailed the development of a remotely accessible robotics development platform for education and research – the *KUKA Robot Learning Lab at KIT*. In addition to presenting the hardware and software elements needed to support remote access to up to ten experimental industrial robot work cells, the Robot Learning Lab addresses the five key concerns of usability, flexibility, safety, security, and scalability. Unlike other remotely accessible testbeds, the Robot Learning Lab features an automated job processing pipeline that can execute experiments in parallel for a large number of users. Complex project implementations can be submitted in general-purpose programming languages, while safe and secure execution in the lab is ensured so that the hardware is protected from damage, and the lab system cannot be compromised by malicious users. Motion planning capabilities are detailed, which can be enhanced by the first two contributions to this thesis. To demonstrate the flexibility and reliability of this testbed, we have shown several use case examples with external users, conducted a long-term operation experiment, and operated the lab open to the public for several years.



(a) Block-based



```
        return distance

    __repr__ = __str__


class AStar(object):

    def __init__(self, start, goal, check_collision,
                 reso):
        start_pose = Pose2D(round(start.x / reso),
                            round(start.y / reso), start.theta)
        end_pose = Pose2D(round(goal.x / reso),
                          round(goal.y / reso), goal.theta)

        self.start = Node(start_pose, 0, -1)
        self.end = Node(end_pose, 0, -1)
        self.reso = reso
        self.check_collision = check_collision

        self.motion = [[2, 0, 0, 1],
                       [0, 2, 0, 1],
                       [-2, 0, 0, 1],
                       [0, -2, 0, 1],
                       [1, 0, PI_2D / 2, 2],
                       [0, 1, PI_2D / 2, 2],
                       [-1, 0, PI_2D / 2, 2],
                       [0, -1, PI_2D / 2, 2],
                       [1, 0, -PI_2D / 2, 2],
                       [0, 1, -PI_2D / 2, 2],
                       [-1, 0, -PI_2D / 2, 2],
                       [0, -1, -PI_2D / 2, 2]]

        # Boundary of search space to be set in set_boundary
        self.bound = dict()
        self.counter = Counter()

    def set_boundary(self, xbound, ybound):
        self.bound.update({"minx": xbound[0] / self.reso})
        self.bound.update({"maxx": xbound[1] / self.reso})
        self.bound.update({"miny": ybound[0] / self.reso})
        self.bound.update({"maxy": ybound[1] / self.reso})

    def calc_index(self, node):
        theta_val = node.theta / (PI_2D / 2)
```

(b) Python

Figure 5.6: The Webeditor available as a browser application with support for Python and block-based programming, and simulation-based testing. On the left is the code editor for which, on session creation, the Python or block-based interface can be selected. On the right top is the visualization of the simulation environment with control buttons for code start/stop, submission to the lab, and simulation restart. The log of the user's code and the project interface is shown on the bottom right.

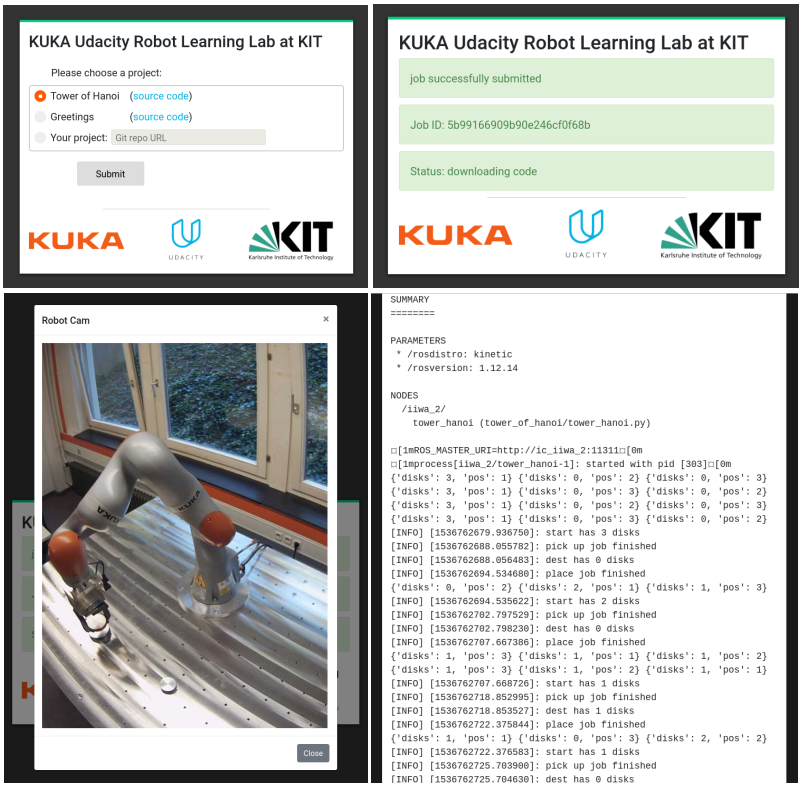


Figure 5.7: Webpage for the Udacity Intersect conference. Participants of the conference were able to select one out of two demo projects and view status information, a video live stream, and the user log file of their job. The webpage was an initial draft for the conference, and the functionality was later included in the current version of the web interface. The figure is taken from Wiedmeyer et al. (2019).

CHAPTER 6

Discussion and Outlook

This chapter discusses the presented work and gives an outlook on future work. Section 6.1 outlines the contributions of this thesis. Section 6.2 examines the limitations of the proposed methods and details the starting points for future research and extensions of the presented work. And Section 6.3 highlights further use cases and potential applications of the first two contributions of the thesis to motion planning.

6.1 Contributions of this Thesis

This thesis focused on developing real-time-capable solutions to common motion planning problems for redundant robot manipulators. Algorithms for two subproblems, Inverse Kinematics (IK) and trajectory generation for path following, were developed. As an application of these solutions and as a contribution to the field of remote robotics, a novel remotely accessible robotics development platform for industrial manipulators (the Robot Learning Lab) was introduced. The need for the first two contributions originally arose during the development of the motion planning framework for the Robot Learning Lab.

Real-time motion planning requires motions to be generated deterministically within strict computation time limits and without unexpected failure. The algorithms are only allowed to fail in known cases where no solution exists. Then safe and reactive robotic systems are possible that can plan motions online, e. g., based on sensor input. To realize such a motion planning pipeline for redundant manipulators, open challenges were identified in two stages: mapping Cartesian paths to joint-space paths and generating time-optimal trajectories to follow them. The literature review revealed that methods suitable only for offline planning could address these challenges and there were no suitable solutions for real-time planning.

In the case of IK and non-offset seven-DoF manipulators, closed-form solutions are most suitable for real-time applications, as deterministic results can be obtained directly at the position level and high-frequency cycle times can be achieved without loss of precision. Compared to classical differential solvers, several more useful features are available. The redundancy space can be parameterized by an arm angle, and joint position limits can be projected onto it to identify its boundaries. Singularities and joint position limits can

be avoided, the global configuration manifold can be controlled, and the complete redundancy space can be determined in a given pose. However, existing redundancy resolution schemes in these closed-form approaches pose difficulties for real-time use, especially when following a Cartesian path rather than point-to-point movements. Most redundancy resolution methods cannot continuously adapt the arm angle while avoiding jumps during path following. Some approaches require optimizing several tuning parameters, which cannot be done online, and the reachability or smoothness of the movement cannot be directly optimized. A multi-objective optimization for redundancy resolution was designed to address these issues, allowing for a closed-form solution. The distance from the joint position limits, joint velocities, and accelerations are optimized directly. Velocities and accelerations were previously optimized only by differential approaches. The evaluation shows significantly faster motion times with increased smoothness of the end-effector motion. The redundancy space is visualized to investigate the behavior of the arm angle compared to the redundancy resolution of the most advanced state-of-the-art method.

The above IK contribution converts Cartesian paths into joint-space paths. Typically, time-optimal trajectories are generated offline from these paths, considering kinematic and, optionally, dynamic constraints, a process called time-optimal path parameterization (TOPP). Robot joint position controllers can then execute these synchronized time-optimal velocity profiles. Offline TOPP methods have steadily been improved to increase computational efficiency, but they are not designed for real-time applications. During real-time planning, the robot may not be at rest at the start pose, or a non-zero goal velocity is requested. Computation times need to be predictable, and a concept is required to enforce a maximum computation time barrier. The algorithm must be complete and may fail only for physically infeasible combinations of input paths and constraints, while always respecting kinematic constraints. The second contribution aims to address these requirements. Other real-time trajectory generators only support point-to-point movements or short path segments with specific properties. Our contribution generalizes to arbitrary path shapes and supports paths consisting of waypoints. Asymmetric velocity and acceleration limits are supported, mainly due to the derivation of a closed-form Maximum Velocity Curve (MVC). Switching points for path acceleration and singularities are treated directly during a path scan or integration pass, and only two passes along the path are needed without additional time integration steps. The MVC and the complete trajectory generation algorithm are evaluated using large sets of pseudo-randomly generated paths. Constraints are respected, and necessary conditions for time-optimality fulfilled. In our test system, the algorithm was able to generate trajectories with a length of several seconds for a seven-DoF manipulator in less than $0.5\mu s$, so a large look-ahead of several seconds is possible in one control cycle.

The two contributions described above can be applied to several robotic use cases, which are discussed in more detail in Section 6.3. The target application in this thesis is the motion planning pipeline for a remotely accessible robotics lab, which must be fast, reactive, and flexible while ensuring the safety of the lab hardware. The design and architecture of such a remote lab constitute the third contribution of this thesis. A robotics testbed, which is remotely accessible over the Internet for application development and testing, is helpful in education and research, according to the literature study conducted in this thesis. It increases access to robots for target groups that usually cannot work with them or create their own setups. However, no existing lab offers public access to several modern professional-grade industrial manipulators.

Additionally, methods are lacking to automate, safeguard, and secure such a testbed with expensive hardware, and to provide easy-to-use programming or submission interfaces. A novel job-processing pipeline for the submitted source code is introduced, including safe and secure motion-planning interfaces in containerized environments and simulation-based pretesting. The architecture features a hardware setup and software components that can be adapted and extended to create projects in robot cells for which users can submit their developed solutions. Web-based programming and simulation interfaces are provided, or users can submit their packaged software. Block-based, Python, and C/C++ are supported. General public use of the lab is reported, while specific experiments or events targeting various target groups are also presented to showcase the setup's capabilities.

In summary, the research carried out in this thesis resulted in the following three contributions to the fields of real-time motion planning and remote robotics:

- **Contribution 1: Real-time multi-objective redundancy resolution for seven-DoF manipulators**
  This contribution introduced a multi-objective optimization approach to redundancy resolution as part of a fully closed-form IK algorithm for seven-DoF manipulators. The extensive evaluation of the closed-form solution shows that it is suitable for real-time planning, and it only requires a single tuning parameter. The behavior of the arm angle in the redundancy space is also investigated. In addition to maximizing the distance to the joint limits, the joint velocities and accelerations are optimized to allow smooth following of Cartesian paths. Reachability is ensured, while motion times are reduced simultaneously compared to the most advanced state-of-the-art approach.

- **Contribution 2: Real-time time-optimal path parameterization**
  Joint-space paths generated from the first contribution or other path-planning sources must be parameterized to create trajectories that joint controllers can execute. In this contribution, a real-time-capable time-optimal trajectory generation algorithm was developed for paths consisting of waypoints, suitable for redundant manipulators. By applying a custom pre-processing to form a continuous path profile and a closed-form solution for the maximum velocity curve, a two-pass scan algorithm was developed based on offline planning methods. The robustness and predictability of maximum computation times were verified using large sets of pseudo-random paths.

- **Contribution 3: Safe, secure, easy-to-use, and scalable remote access to industrial manipulators**
  The goal of this contribution was to make industrial robots remotely accessible for software development within teaching or research activities. The first two contributions are needed to enable a safe and reactive motion-planning framework for such a remote lab, and the lab serves as an application for the algorithms. As part of this contribution, a scalable and parallel job-processing pipeline was developed for the submitted source code, which has a low maintenance burden. In addition, easy-to-use programming interfaces are provided that target different levels of programming experience. Various use cases were presented and lab usage was reported.

6.2 Limitations and Future Work

In the following, the limitations of the thesis's contributions are discussed separately for each contribution, and possible future avenues for further research are detailed. An emphasis is placed on potential remedies for identified limitations.

Kinematics

In the current work, the multi-objective optimization computes a local optimum for the arm angle for every pose. Thus, the optimization cannot adapt the arm angle based on the future shape of the null-space along the requested path. To address the issue of running into abruptly appearing null-space borders, information about the shape of the upcoming null-space can be integrated into the local resolution as heuristics to guide the arm angle. This information can be obtained by extrapolating along the current path or by evaluating the null-space at the goal pose, if it is known. Reachability could be significantly improved with this extension, and the motion's smoothness could be further increased. To assist with such an extension, a global optimization of the arm angle along the entire path using an off-the-shelf solver can serve as a baseline. The initial work in this direction has already been done. Such a global optimization itself would not be real-time-capable. However, it would give insight into the quality of the existing solution based on local optimization, and the look-ahead heuristics can be evaluated and compared.

Similarly, the robot's workspace is currently restricted by the singularity-avoidance scheme in the null-space. Kinematic singularities where the redundant DoF is lost are treated in the same way as joint position limits. Compared to the six-DoF structure, where the arm angle is kept fixed, the IK cannot directly transition to other global configurations because it would need to pass through a singularity. A six-DoF robot with otherwise the same kinematic structure and standard IK is then, in some scenarios, able to follow paths while our IK fails. However, the algorithm can be modified to allow such transitions through non-critical singularities where a smooth transition is possible. Only shortly, the redundancy is lost, but a decision algorithm is needed that makes an informed decision whether it is better to stay in the current null-space and avoid the singularity, or if it is beneficial to transition through the singularity into a different global configuration with a different null-space shape. The above-described heuristics can be used here as the decision algorithm, using the computed null-space properties in the global configurations where a transition would be possible. We already investigated how the reachable workspace can change with such additions, and the results indicate that significant improvements are possible.

The IK solution targets 7-DoF manipulators without joint offsets. The redundancy resolution can be ported to other manipulators with a single degree of redundancy that have offsets in the joints. This is feasible as long as a closed-form IK solution exists and the redundancy can be parameterized using a null-space angle. A projection of joint angle limits and singularities to the null-space is also required. The multi-objective optimization can be extended to include additional objectives, for which closed-form solutions with an arm angle have already been developed. To improve the physical interaction of the robot with its environments, optimization of rigidity (Busson et al., 2017) and maximization of force capabilities (Busson and Béarée, 2018) would be helpful additions. Collision avoidance can also be integrated by adding further blocked intervals in the null-space where obstacles intersect with the arm angle circle.

Path Parameterization

The RT-TOPP algorithm does not support jerk limits, which are necessary to properly use the algorithm with an industrial robot and its real-time control interface. This is further explained in Section 4.3.3, and in the same section, we also discuss how the algorithm can be extended to third-order constraints such as jerk, including open challenges. Currently, the algorithm only parameterizes joint-space paths, and no support for Cartesian constraints, e. g., the absolute translational end-effector velocity, is available. Such functionality can be added by interpolating the corresponding Cartesian path in parallel with a separate spline interpolation and imposing the constraints. A proper interpolation scheme using unit quaternions on SO(3) is needed to limit rotational constraints. Such an approach results in an approximation between waypoints, as no exact mapping between Cartesian and joint space is given. Alternatively, forward kinematics can be used at every grid point for a precise but also more time-consuming solution. Other spline representations like B-splines can be tested for the pre-processing to also impose constraints at the position level in-between waypoints, like joint limits or Cartesian workspace borders, or to shape the path using heuristics to enable faster traversal or to accommodate higher start state velocities.

Asymmetric joint constraints are supported, which permit specifying constraints computed from a dynamics model and torque limits. However, these constraints are applied to the full path. It needs to be investigated how a robot dynamics model can be used along the path while retaining real-time capability at the same time. A possible approach can be to update the constraints only using the dynamics model at a certain path resolution or at discontinuities of the computed velocity profile during the backward scan, and interpolate the values in between. Efficient dynamics solvers like Pinocchio (Carpentier et al., 2025) can be tested. The existing analytical MVC may not be directly usable, as it treats velocity and acceleration independently of each other. However, acceleration bounds depend on the current velocity in robot dynamics, so the maximum velocity needs to be searched for, for which the dynamics model still returns valid acceleration ranges. The velocity from the previous state can be used as an approximation, but this may be too inaccurate or only a conservative approach for the backward scan, where the velocity profile is discontinuous with jumps to smaller velocities.

The evaluation lacks a direct comparison with a similar TOPP approach. The evaluation is designed similarly to Barnett and Gosselin (2020) to allow an approximate comparison with their method and TOPP-RA (Pham and Pham, 2018). It needs to be ensured that all algorithms used have very similar paths after their respective pre-processing. Feeding a TOPP algorithm that has available proofs of time-optimality, like TOPP-RA, with a very similar set of grid points from the interpolated paths makes it possible to inspect and compare the time-optimality and other properties like constraint violations more closely. Our algorithm adopted several techniques from other methods for which time-optimality or completeness is proven, but a complete proof that the time-optimal solution is always returned, and a proper proof of completeness of RT-TOPP is missing. However, the evaluation shows that the necessary conditions are met, and it appears that existing proofs from methods we built upon can be transferred.

The implementation of RT-TOPP is only a prototype of the core algorithm for the evaluation, and it is incomplete. A complete parameterization is always executed, and the partial parameterization across multiple control cycles with fallback from the concept in Section

4.1.2 is missing. This needs to be implemented and tested. Further limitations specific to the current implementation and suggested remedies are described in Section 4.4. In particular, the remaining issues regarding the sampling procedure need to be addressed to prepare for use with a real-time control interface.

Remote Robotics Lab

This thesis presents the current state of the Robot Learning Lab. It is possible to continue the operation of the lab, and further projects can be hosted for students and researchers. The lab can be included in the curriculum at KIT and other universities as a standalone hands-on student laboratory internship or an extension to robotics lectures with exercises. The implementation of more exercises to teach kinematics and other basic robotics knowledge has already started. Having such exercises stable in the same way as existing projects, such as the path planning challenge, would make them ready for unsupervised regular use with automatic reset in the lab.

For roboticists, it could serve as a benchmark platform for research and robotics competitions. A new focus area for the projects could also be the application of data-driven methods, as the lab is suitable for large-scale parallel data collection and the automated training of neural networks for robot control, manipulation, etc. For this, it is also possible to extend the server infrastructure so that the Docker containers get access to GPU-accelerated hardware. This would allow projects that use machine learning frameworks to use these resources.

Special attention needs to be devoted to the move interface for controlling the robots in the lab, as it cannot generate highly dynamic motions with real-time constraints and within short control cycles. Although the first contribution, the closed-form IK, is fully integrated into the motion planning framework of the lab, the second contribution of this thesis, the RT-TOPP algorithm, is not yet integrated. If a jerk-limited extension of this algorithm succeeds in the future, it can be used for online and real-time trajectory generation for robots in the lab. Currently, Kunz and Stilman (2012) is used in combination with Reflexxes (Kröger, 2011) as a custom solution, which is jerk-limited, but the combination of the two results in a non-time-optimal solution, and the path following is not highly accurate, as Reflexxes does not respect the shape of the path between grid points. The time parameterization can only use small acceleration limits for Reflexxes to be able to add jerk limits in the following as a smoothing.

Visual monitoring was available through Roboception rc_visard 160 stereo cameras (Roboception GmbH, 2026) mounted below the webcams. Initial work on a framework was done in the supervised thesis by Wirth (2019) to support the detection and pose estimation of known objects in the workspace, and to reference the actual state with the desired state for every movable object using a state machine. Integrating such functionality into the SDK and job processing would allow projects to easily leverage this functionality. This would allow automated supervision of the integrity of an experimental setup, and, during reset after job execution, the robot can restore the original starting setup for a given project. Making sensor data, such as that from vision systems or joint torques, available to lab users would allow for more advanced projects and further increase the benefits of developing and automating testing with real hardware in the loop. Initial work was already done to record such data during a job run and provide it to users afterwards. Building

upon this, a more powerful motion planning interface with safe grasp planning would be possible, and more motion primitives could be used. Hybrid position and torque control with contact could be another focus area, as the iiwa robots already support such control capabilities with their torque sensing. A drawing project on a whiteboard was already initially developed and tested as a first possible application. Direct safe force/torque control capabilities can then be added to the lab's move interface. When hosting grasp and manipulation benchmarks in the lab, standardized task boards can be used, which already have built-in connectivity to collect data and sensory feedback to check if a task was completed successfully. The *Dr.J* task board (So et al., 2024) can be used as a reference, and data can be sent to the internal RLL web API and processed in a custom way per project.

The software systems were intensively developed until mid-2019, and then largely only maintenance was applied to keep the systems available. The existing software infrastructure is largely based on ROS-1, which reached its end-of-life and is no longer officially supported. Components need to be ported to ROS-2 and more current Ubuntu/Debian operating systems. Security fixes and enhancements can then be added during the process.

6.3 Real-Time Kinematics and Trajectory Generation: Further Applications

The redundancy resolution for IK and the trajectory generation RT-TOPP, the first two presented contributions of this thesis in Chapters 3 and 4, propose real-time-capable algorithms for fundamental motion planning problems in the context of robotic manipulators. The third contribution, the remote robotics lab, is used as a target application, but the algorithms can be applied to various other use cases. This thesis focused on the initial development and testing of these methods, but the evaluation in different practical applications and showcases is left for future work. As a starting point, this section provides a short overview of further applications that can benefit or be enabled by these contributions to the core of a typical industrial robot motion planning pipeline, mainly due to the added real-time capability.

Robotics applications in industry require reliability and safety with as little downtime as possible, and processes must be as fast as possible. As motions are usually only planned offline and recurring motions are optimized without sensing and reaction capabilities, new processes and setups become possible with online and real-time planning. More flexibility can be introduced to the manufacturing process when motions can be adapted in real-time. A potential field is visual serving, where a Cartesian approach path to objects is planned with intermediate waypoints by a vision system. The IK and RT-TOPP can generate a joint trajectory to, e. g., pick up objects from a moving conveyor belt or grasp objects on a warehouse shelf and place them in containers. Long custom Cartesian paths with sharp corners need to be processed in burr removing operations, where a custom trajectory is needed for workpieces whose poses may not be precisely fixed. Flexibility can be introduced in assembly operations, where a cyclic motion may be interrupted by sensor input due to changes in the work environment, and a sudden Cartesian path change needs to be processed. Setups where multiple robots interact or coexist and potentially collide can be automated without complex, centralized, coordinated motion planning. These applications require the combination of IK and RT-TOPP with a suitable real-time (Cartesian)

path planner. Other similar industrial application areas include inspection, proximity sensor servoing, spray painting, and gluing operations, where smooth motions are required.

The IK can be applied to large-scale collision-free path planning where many Cartesian paths can be sampled fast, in parallel, and optimized for fast traversal. Human-robot collaboration scenarios with the generation of collision avoidance motions or, in general, collision avoidance in unstructured environments with moving obstacles can be enabled with the combination of real-time sensing, path planning capabilities, and the developed algorithms. Research in collision reaction after impact typically focuses on torque control methods in which the robot behaves compliantly and with less inertia to reduce impact forces. Such safety measures are especially needed in physical human-robot interaction. Real-time motion planning may yield improvements in this area, and it can be used to investigate collision reaction strategies where the robot moves as fast as possible away from the collision zone to a safe pose. A possible approach would be combining the developed algorithms with a Cartesian real-time point-to-point trajectory generator (Huber and Wollherr, 2019b), which can impose safe kinematic workspace constraints, to support Cartesian and joint space constraints. The SE(3) trajectory can be transferred to joint space by the IK and parameterized by RT-TOPP, which additionally imposes the converted path velocity and acceleration limits computed by the Cartesian trajectory generator. The RT-TOPP algorithm is also suitable for kinematic structures with more than seven revolute joints. It can be evaluated with higher DoF systems, like humanoid robots' arms and upper torso.

In conclusion, the discussion of this thesis highlighted the three key contributions, limitations, and potential future work to address these limitations. Finally, further possible practical use cases and applications for the two developed algorithms were presented, in addition to the third contribution as an application, the remotely accessible robotics testbed. The contribution to remote robotics aims to provide a design and architecture to make industrial manipulators accessible over the public Internet for research and education as a safe, secure, automated, easy-to-use, and flexible platform. The contributions to real-time motion planning have the goal of helping robots to dynamically sense, reason, and act quickly, safely, and reliably. Specifically, open challenges were addressed in the lower-level reasoning stage, where a mapping from the Cartesian task space to the actuated joint space is needed, and trajectories are generated from planned paths with multiple waypoints to allow fast execution by joint controllers.

Bibliography

Adjih, C., Baccelli, E., Fleury, E., Harter, G., Mitton, N., Noel, T., Pissard-Gibollet, R., Saint-Marcel, F., Schreiner, G., Vandaele, J., and Watteyne, T. (2015). FIT IoT-LAB: A large scale open experimental IoT testbed. In *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pages 459–464. 27

Afzal, A., Le Goues, C., Hilton, M., and Timperley, C. S. (2020). A study on challenges of testing robotic systems. In *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)*, pages 96–107. IEEE. 75

Al Khudir, K. and De Luca, A. (2018). Faster motion on cartesian paths exploiting robot redundancy at the acceleration level. *IEEE Robotics and Automation Letters*, 3(4):3553–3560. 12

Ames, B., Morgan, J., and Konidaris, G. (2022). Ikflow: Generating diverse inverse kinematics solutions. *IEEE Robotics and Automation Letters*, 7(3):7177–7184. 8

Asfour, T. and Dillmann, R. (2003). Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem. In *2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 2, pages 1407–1412. IEEE. 9

Balestrino, A., Caiti, A., and Crisostomi, E. (2009). From remote experiments to web-based learning objects: An advanced telelaboratory for robotics and control systems. *IEEE Transactions on Industrial Electronics*, 56(12):4817–4825. 25

Barnett, E. and Gosselin, C. (2020). A Bisection Algorithm for Time-Optimal Trajectory Planning Along Fully Specified Paths. *IEEE Transactions on Robotics*, pages 1–15. 18, 19, 45, 46, 57, 61, 62, 65, 66, 67, 68, 97

Bauer, S., Wüthrich, M., Widmaier, F., Buchholz, A., Stark, S., Goyal, A., Steinbrenner, T., Akpo, J., Joshi, S., Berenz, V., Agrawal, V., Funk, N., Urain De Jesus, J., Peters, J., Watson, J., Chen, C., Srinivasan, K., Zhang, J., Zhang, J., Walter, M., Madan, R., Yoneda, T., Yarats, D., Allshire, A., Gordon, E., Bhattacharjee, T., Srinivasa, S., Garg, A., Maeda, T., Sikchi, H., Wang, J., Yao, Q., Yang, S., McCarthy, R., Sanchez, F., Wang, Q., Bulens, D., McGuinness, K., O'Connor, N., Stephen, R., and Schölkopf, B. (2022). Real robot challenge: A robotics competition in the cloud. In Kiela, D., Ciccone, M., and Caputo, B., editors, *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, pages 190–204. PMLR. 25

Berscheid, L. and Kröger, T. (2021). Jerk-limited real-time trajectory generation with arbitrary target states. *Robotics: Science and Systems XVII*. 15, 19, 21, 62

Bobrow, J., Dubowsky, S., and Gibson, J. (1985). Time-Optimal Control of Robotic Manipulators Along Specified Paths. *The International Journal of Robotics Research*, 4(3):3–17. 16

Busson, D. and Béarée, R. (2018). A pragmatic approach to exploiting full force capacity for serial redundant manipulators. *IEEE Robotics and Automation Letters*, 3(2):888–894. 96

Busson, D., Bearee, R., and Olabi, A. (2017). Task-oriented rigidity optimization for 7 DOF redundant manipulators. *IFAC-PapersOnLine*, 50(1):14588 – 14593. 96

Carpentier, J., Valenza, F., Mansard, N., et al. (2015–2025). Pinocchio: fast forward and inverse dynamics for poly-articulated systems. https://stack-of-tasks.github.io/pinocchio. Accessed July 29, 2026. 97

Casalino, A., Zanchettin, A. M., and Rocco, P. (2016). Online planning of optimal trajectories on assigned paths with dynamic constraints for robot manipulators. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 979–985. IEEE. 21

Casañ, G. A., Cervera, E., Moughlbay, A. A., Alemany, J., and Martinet, P. (2015). ROS-based online robot programming for remote education and training. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6101–6106. 25

Casini, M., Chinello, F., Prattichizzo, D., and Vicino, A. (2008). RACT: a remote lab for robotics experiments. *IFAC Proceedings Volumes*, 41(2):8153 – 8158. 17th IFAC World Congress. 25

Chitta, S., Marder-Eppstein, E., Meeussen, W., Pradeep, V., Rodríguez Tsouroukdissian, A., Bohren, J., Coleman, D., Magyar, B., Raiola, G., Lüdtke, M., and Fernández Perdomo, E. (2017). ros_control: A generic and simple control framework for ROS. *The Journal of Open Source Software*. 65

Collins, J., Howard, D., and Leitner, J. (2019). Quantifying the reality gap in robotic manipulation tasks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6706–6712. IEEE. 23, 76

Consolini, L., Locatelli, M., Minari, A., Nagy, A., and Vajk, I. (2019). Optimal Time-Complexity Speed Planning for Robot Manipulators. *IEEE Transactions on Robotics*, 35(3):790–797. 19

Corke, P., Greener, E., and Philip, R. (2016). An innovative educational change: Massive open online courses in robotics and robotic vision. *IEEE Robotics Automation Magazine*, 23(2):81–89. 24

Crick, C., Jay, G., Osentoski, S., Pitzer, B., and Jenkins, O. C. (2017). Rosbridge: ROS for non-ROS users. In *Robotics Research: The 15th International Symposium ISRR*, pages 493–504. Springer. 85

Dahm, P. and Joublin, F. (1997). Closed form solution for the inverse kinematics of a redundant robot arm. Technical report, Institut für Neuroinformatik Ruhr-Universität, Bochum. 9

De Luca, A. and Ferrajoli, L. (2008). Exploiting robot redundancy in collision detection and reaction. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3299–3305. IEEE. 8

Debrouwere, F., Van Loock, W., Pipeleers, G., Dinh, Q. T., Diehl, M., De Schutter, J., and Swevers, J. (2013). Time-optimal path following for robots with trajectory jerk constraints using sequential convex programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 1916–1921. IEEE. 19

DeMarinis, N., Tellex, S., Kemerlis, V. P., Konidaris, G., and Fonseca, R. (2019). Scanning the internet for ROS: A view of security in robotics research. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8514–8521. IEEE. 26

Diankov, R. (2010). *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute. 8, 44

Dieber, B., Breiling, B., Taurer, S., Kacianka, S., Rass, S., and Schartner, P. (2017). Security for the Robot Operating System. *Robotics and Autonomous Systems*, 98:192 – 203. 82

Dieber, B., Kacianka, S., Rass, S., and Schartner, P. (2016). Application-level security for ROS-based applications. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4477–4482. 82

DiLuoffo, V., Michalson, W. R., and Sunar, B. (2018). Robot Operating System 2: The need for a holistic security approach to robotic architectures. *International Journal of Advanced Robotic Systems*, 15(3). 82

do Nascimento, L. P. (2021). Third-order real-time trajectory generation on SE(3) for robot manipulators. Master's thesis, Karlsruhe Institute of Technology. 46

Dong, J., Ferreira, P. M., and Stori, J. A. (2007). Feed-rate optimization with jerk constraints for generating minimum-time trajectories. *International Journal of Machine Tools and Manufacture*, 47(12-13):1941–1955. 18, 61

Dong, J. and Stori, J. A. (2006). A generalized time-optimal bidirectional scan algorithm for constrained feed-rate optimization. *Journal of dynamic systems, measurement, and control*, 128(2):379–390. 18, 50, 61

Dou, R., Yu, S., Li, W., Chen, P., Xia, P., Zhai, F., Yokoi, H., and Jiang, Y. (2022). Inverse kinematics for a 7-dof humanoid robotic arm with joint limit and end pose coupling. *Mechanism and Machine Theory*, 169:104637. 12, 14

Duvall, P. M., Matyas, S., and Glover, A. (2007). *Continuous integration: improving software quality and reducing risk*. Pearson Education. 75

Elias, A. J. and Wen, J. T. (2024). Redundancy parameterization and inverse kinematics of 7-dof revolute manipulators. *Mechanism and Machine Theory*, 204:105824. 10

Faria, C., Ferreira, F., Erlhagen, W., Monteiro, S., and Bicho, E. (2018). Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance. *Mechanism and Machine Theory*, 121:317–334. 11, 12, 14, 15, 30, 32, 35, 36, 37, 38, 39

Faroni, M., Beschi, M., and Pedrocchi, N. (2020). Inverse kinematics of redundant manipulators with dynamic bounds on joint movements. *IEEE Robotics and Automation Letters*, 5(4):6435–6442. 8

Faroni, M., Beschi, M., Visioli, A., and Pedrocchi, N. (2021). A real-time trajectory planning method for enhanced path-tracking performance of serial manipulators. *Mechanism and Machine Theory*, 156:104152. 21

Flacco, F., De Luca, A., and Khatib, O. (2015). Control of redundant robots under hard joint constraints: Saturation in the null space. *IEEE Transactions on Robotics*, 31(3):637–654. 7

Flacco, F. and Luca, A. D. (2015). Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties. *Robotics and Autonomous Systems*, 70:191 – 201. 12

Flacco, F., Luca, A. D., and Khatib, O. (2012). Motion control of redundant robots under joint constraints: Saturation in the null space. In *2012 IEEE International Conference on Robotics and Automation*. 7

Gao, J., Guo, H., Cao, Z., Huang, P., and Zhou, G. (2023). Real is better than perfect: Sim-to-real robotic system in secondary school education. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4480–4487. IEEE. 24, 25, 26

Glass, K., Colbaugh, R., Lim, D., and Seraji, H. (1995). Real-time collision avoidance for redundant manipulators. *IEEE transactions on robotics and automation*, 11(3):448–457. 7

Goldstain, O., Ben-Gal, I., and Bukchin, Y. (2007). Remote learning for the manipulation and control of robotic cells. *European Journal of Engineering Education*, 32(4):481–494. 25

Google LLC (2026). Blockly. https://docs.blockly.com/. Accessed: July 29, 2026. 85

Guigue, A., Ahmadi, M., Langlois, R., and Hayes, M. J. (2010). Pareto optimality and multiobjective trajectory planning for a 7-DOF redundant manipulator. *IEEE Transactions on Robotics*, 26(6):1094–1099. 12

Gürtler, N., Widmaier, F., Sancaktar, C., Blaes, S., Kolev, P., Bauer, S., Wüthrich, M., Wulfmeier, M., Riedmiller, M., Allshire, A., Wang, Q., McCarthy, R., Kim, H., Baek, J., Kwon, W., Qian, S., Toshimitsu, Y., Michelis, M. Y., Kazemipour, A., Raayatsanati, A., Zheng, H., Cangan, B. G., Schölkopf, B., and Martius, G. (2022). Real robot challenge 2022: Learning dexterous manipulation from offline data in the real world. In Ciccone, M., Stolovitzky, G., and Albrecht, J., editors, *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pages 133–150. PMLR. 24

Haddadin, S., Johannsmeier, L., Schmid, J., Ende, T., Parusel, S., Haddadin, S., Schappler, M., Lilge, T., and Becker, M. (2019). roboterfabrik: A pilot to link and unify german robotics education to match industrial and societal demands. In Lepuschitz, W., Merdan, M., Koppensteiner, G., Balogh, R., and Obdržálek, D., editors, *Robotics in Education*, pages 3–17, Cham. Springer International Publishing. 24

He, S., Hu, C., Lin, S., Zhu, Y., and Tomizuka, M. (2022). Real-time time-optimal continuous multi-axis trajectory planning using the trajectory index coordination method. *ISA transactions*, 131:639–649. 22

Huang, H., Liu, H., Xia, C., Mei, H., Gao, X., and Liang, B. (2023). Sampling-based time-optimal path parameterization with jerk constraints for robotic manipulation. *Robotics and Autonomous Systems*, 170:104530. 20

Huber, G. and Wollherr, D. (2019a). Efficient closed-form task space manipulability for a 7-DOF serial robot. *Robotics*, 8(4):98. 10, 34

Huber, G. and Wollherr, D. (2019b). An Online Trajectory Generator on SE(3) for Human-Robot Collaboration. *Robotica*, pages 1–22. 21, 100

International Organisation of Standardisation (2016). *ISO-TS 15066: Robots and robotic devices: Collaborative robots*. ISO. 21

Jara, C. A., Candelas, F. A., Puente, S. T., and Torres, F. (2011). Hands-on experiences of undergraduate students in automatics and robotics using a virtual and remote laboratory. *Computers & Education*, 57(4):2451 – 2461. 25

Ji, C., Zhang, Z., Cheng, G., Kong, M., and Li, R. (2023). A convex optimization method to time-optimal trajectory planning with jerk constraint for industrial robotic manipulators. *IEEE Transactions on Automation Science and Engineering*, 21(4):7629–7646. 19

Khatib, M., Al Khudir, K., and De Luca, A. (2020). Task priority matrix at the acceleration level: Collision avoidance under relaxed constraints. *IEEE Robotics and Automation Letters*, 5(3):4970–4977. 8

Kiemel, J. C. and Kröger, T. (2024). Jerk-limited traversal of one-dimensional paths and its application to multi-dimensional path tracking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7142–7148. IEEE. 19

Kim, J. and Croft, E. A. (2019). Online near time-optimal trajectory planning for industrial robots. *Robotics and Computer-Integrated Manufacturing*, 58:158–171. 21

Kröger, T. (2010). *On-Line Trajectory Generation in Robotic Systems: Basic Concepts for Instantaneous Reactions to Unforeseen (Sensor) Events*, volume 58. Springer. 61

Kröger, T. (2011). Opening the door to new sensor-based robot applications - the Reflexxes Motion Libraries. In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4. IEEE. 15, 21, 39, 49, 81, 98

Krūmiņš, D., Schumann, S., Vunder, V., Põlluäär, R., Laht, K., Raudmäe, R., Aabloo, A., and Kruusamäe, K. (2024). Open remote web lab for learning robotics and ROS with physical and simulated robots in an authentic developer environment. *IEEE Transactions on Learning Technologies*. 25, 26

Kuhlemann, I., Schweikard, A., Jauer, P., and Ernst, F. (2016). Robust inverse kinematics by configuration control for redundant manipulators with seven DoF. In *2016 2nd International Conference on Control, Automation and Robotics (ICCAR)*, pages 49–55. IEEE. 10, 11

KUKA AG (2026). KUKA LBR iiwa. https://www.kuka.com/en-de/products/robot-systems/industrial-robots/lbr-iiwa. Accessed: July 29, 2026. 35, 65, 77

Kulich, M., Chudoba, J., Kosnar, K., Krajnik, T., Faigl, J., and Preucil, L. (2013). SyRoTek–distance teaching of mobile robotics. *IEEE Transactions on Education*, 56(1):18–23. 25

Kumar, A., Jose, J., Jain, A., Kulkarni, S., and Arya, K. (2024). Scalable and low-cost remote lab platforms: Teaching industrial robotics using open-source tools and understanding its social implications. In *International Conference on Social Robotics*, pages 202–215. Springer. 24

Kunz, T. (2015). *Time-optimal sampling-based motion planning for manipulators with acceleration limits*. PhD thesis, Georgia Institute of Technology. 17

Kunz, T. and Stilman, M. (2012). Time-optimal trajectory generation for path following with bounded acceleration and velocity. *Robotics: Science and Systems VIII*. 17, 20, 36, 45, 50, 51, 52, 53, 62, 65, 81, 98

Kunze, M. (2016). *On-the-Fly Workspace Visualization for Redundant Manipulators*. PhD thesis, Fakultät für Informatik, Karlsruher Institut für Technologie (KIT). 11, 12, 113

Lange, F. and Albu-Schäffer, A. (2015). Path-accurate online trajectory generation for jerk-limited industrial robots. *IEEE Robotics and Automation Letters*, 1(1):82–89. 21

Lee, K.-k. and Buss, M. (2006). Redundancy resolution with multiple criteria. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 598–603. IEEE. 12

Li, Q., Xia, Y., Wang, X., Xin, P., Chen, W., and Xiong, C. (2022). Muscle-effort-minimization-inspired kinematic redundancy resolution for replicating natural posture of human arm. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 30:2341–2351. 10

Li, S., Wang, Z., Zhang, Q., and Han, F. (2018). Solving inverse kinematics model for 7-DoF robot arms based on space vector. In *2018 International Conference on Control and Robots (ICCR)*, pages 1–5. IEEE. 10

Lin, J., Rickert, M., and Knoll, A. (2021). Parameterizable and jerk-limited trajectories with blending for robot motion planning and spherical cartesian waypoints. In *Proceedings of the IEEE International Conference on Robotics and Automation*. 20

Lin, S., Hu, C., He, S., Zhao, W., Wang, Z., and Zhu, Y. (2023). Real-time local greedy search for multiaxis globally time-optimal trajectory. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(2):960–971. 22

Liu, Z., Liu, W., Qin, Y., Xiang, F., Gou, M., Xin, S., Roa, M. A., Calli, B., Su, H., Sun, Y., et al. (2021). Ocrtoc: A cloud-based competition and benchmark for robotic grasping and manipulation. *IEEE Robotics and Automation Letters*, 7(1):486–493. 24

Ma, J.-w., Gao, S., Yan, H.-t., Lv, Q., and Hu, G.-q. (2021). A new approach to time-optimal trajectory planning with torque and jerk limits for robot. *Robotics and Autonomous Systems*, 140:103744. 19

Mattmüller, J. and Gisler, D. (2009). Calculating a near time-optimal jerk-constrained trajectory along a specified smooth path. *The International Journal of Advanced Manufacturing Technology*, 45(9-10):1007. 18

Moradi, H. and Lee, S. (2005). Joint limit analysis and elbow movement minimization for redundant manipulators using closed form method. In *International Conference on Intelligent Computing*, pages 423–432. Springer. 30

Morimoto, T., Martinez, M. O., Davis, R., Blikstein, P., and Okamura, A. M. (2020). Teaching with hapkit: Enabling online haptics courses with hands-on laboratories. *IEEE Robotics & Automation Magazine*. 24

Nagy, Á. and Vajk, I. (2019). Sequential time-optimal path-tracking algorithm for robots. *IEEE Transactions on Robotics*, 35(5):1253–1259. 19

Nguyen, H. and Pham, Q.-C. (2016). Time-optimal path parameterization of rigid-body motions: Applications to spacecraft reorientation. *Journal of Guidance, Control, and Dynamics*, 39(7):1667–1671. 18

Oh, J., Bae, H., and Oh, J.-H. (2017). Analytic inverse kinematics considering the joint constraints and self-collision for redundant 7DOF manipulator. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 123–128. IEEE. 10

Ostermeier, D., Külz, J., and Althoff, M. (2025). Automatic geometric decomposition for analytical inverse kinematics. *IEEE Robotics and Automation Letters*, 10(10):9964–9971. 8

Palleschi, A., Garabini, M., Caporale, D., and Pallottino, L. (2019). Time-Optimal Path Tracking for Jerk Controlled Robots. *IEEE Robotics and Automation Letters*, 4(4):3932–3939. 19, 54

Petrovi, P. and Balogh, R. (2012). Deployment of remotely-accessible robotics laboratory. *International Journal of Online Engineering (iJOE)*, 8(S2):31–35. 25

Pham, H. and Pham, Q.-C. (2017). On the Structure of the Time-Optimal Path Parameterization Problem with Third-Order Constraints. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 679–686. arXiv: 1609.05307. 18, 61

Pham, H. and Pham, Q.-C. (2018). A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis. *IEEE Transactions on Robotics*, 34(3):645–659. 19, 20, 62, 64, 97

Pham, Q.-C. (2014). A General, Fast, and Robust Implementation of the Time-Optimal Path Parameterization Algorithm. *IEEE Transactions on Robotics*, 30(6):1533–1540. 18, 57

Pham, T. H. (2019). *Time-optimal motion planning and control*. PhD thesis, Nanyang Technological University, School of Mechanical and Aerospace Engineering. 17

Pickem, D., Glotfelter, P., Wang, L., Mote, M., Ames, A., Feron, E., and Egerstedt, M. (2017). The Robotarium: A remotely accessible swarm robotics research testbed. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1699–1706. 25, 26

Pitzer, B., Osentoski, S., Jay, G., Crick, C., and Jenkins, O. C. (2012). PR2 Remote Lab: An environment for remote development and experimentation. In *2012 IEEE International Conference on Robotics and Automation*, pages 3200–3205. 25

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., and Ng, A. (2009). ROS: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan. 80

Roboception GmbH (2026). Roboception rc_visard. https://roboception.com/3d-stereo-vision/rc-visard-3d-stereo-sensor/. Accessed: July 29, 2026. 98

Šafarič, R., Truntič, M., Hercog, D., and Pačnik, G. (2005). Control and robotics remote laboratory for engineering education. *International Journal of Online Engineering (iJOE)*, 1(1). 25

Schumann, S., Krūmiņš, D., Vunder, V., Aabloo, A., Siiman, L. A., and Kruusamäe, K. (2023). A beginner-level MOOC on ROS robotics leveraging a remote web lab for programming physical robots. In *International Conference on Robotics in Education (RiE)*, pages 285–297. Springer. 25

SCHUNK SE & Co. KG (2026). SCHUNK EGL. https://schunk.com/de_en/gripping-systems/series/egl/. Accessed: July 29, 2026. 77

Shao, J., Zhang, H., Zhu, S., and Song, W. (2024). Online trajectory generation with local replanning for 7-dof serial manipulator in unforeseen dynamic environments. *IEEE Robotics and Automation Letters*. 9

Shen, P., Zhang, X., and Fang, Y. (2018). Complete and time-optimal path-constrained trajectory planning with torque and velocity constraints: Theory and applications. *IEEE/ASME Transactions on Mechatronics*, 23(2):735–746. 49

Shimizu, M., Kakuya, H., Yoon, W., Kitagaki, K., and Kosuge, K. (2008). Analytical inverse kinematic computation for 7-DOF redundant manipulators with joint limits and its application to redundancy resolution. *IEEE Transactions on Robotics*, 24(5):1131–1142. 10, 11

Shin, K. and McKay, N. (1985). Minimum-time control of robotic manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541. 16

Siciliano, B., Khatib, O., and Kröger, T. (2008). *Springer handbook of robotics*, volume 200. Springer. 8

So, P., Sarabakha, A., Wu, F., Culha, U., Abu-Dakka, F. J., and Haddadin, S. (2024). Digital robot judge: Building a task-centric performance database of real-world manipulation with electronic task boards. *IEEE Robotics & Automation Magazine*, 31(4):32–44. 99

Spasojevic, I., Murali, V., and Karaman, S. (2019). Asymptotic optimality of a time optimal path parametrization algorithm. *IEEE Control Systems Letters*, 3(4):835–840. 19

Sucan, I. A. and Chitta, S. (2026). MoveIt! https://moveit.ai. Accessed: July 29, 2026. 18, 34, 62, 64, 80

Sundaralingam, B., Hari, S. K. S., Fishman, A., Garrett, C., Van Wyk, K., Blukis, V., Millane, A., Oleynikova, H., Handa, A., Ramos, F., Ratliff, N., and Fox, D. (2023a). Curobo: Parallelized collision-free robot motion generation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8112–8119. 20

Sundaralingam, B., Hari, S. K. S., Fishman, A., Garrett, C., Wyk, K. V., Blukis, V., Millane, A., Oleynikova, H., Handa, A., Ramos, F., Ratliff, N., and Fox, D. (2023b). curobo: Parallelized collision-free minimum-jerk robot motion generation. Technical report, Nvidia. 20

Tani, J., Daniele, A. F., Bernasconi, G., Camus, A., Petrov, A., Courchesne, A., Mehta, B., Suri, R., Zaluska, T., Walter, M. R., et al. (2020). Integrated benchmarking and design for reproducible and accessible evaluation of robotic agents. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6229–6236. IEEE. 25, 26

Tarkiainen, M. and Shiller, Z. (1993). Time optimal motions of manipulators with actuator dynamics. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 725–730 vol.2. 18

Tellez, R. (2016). A thousand robots for each student: Using cloud robot simulations to teach robotics. In *Robotics in Education: Research and Practices for Robotics in STEM Education*, pages 143–155. Springer. 27

Tian, X., Xu, Q., and Zhan, Q. (2021). An analytical inverse kinematics solution with joint limits avoidance of 7-dof anthropomorphic manipulators without offset. *Journal of the Franklin Institute*, 358(2):1252–1272. 12

Tondu, B. (2006). A closed-form inverse kinematic modelling of a 7R anthropomorphic upper limb based on a joint parametrization. In *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pages 390–397. IEEE. 10

Toris, R., Kammerl, J., Lu, D. V., Lee, J., Jenkins, O. C., Osentoski, S., Wills, M., and Chernova, S. (2015). Robot web tools: Efficient messaging for cloud robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE. 85

Trevelyan, J. (2004). Lessons learned from 10 years experience with remote laboratories. In *International Conference on Engineering Education and Research*. 25

Vaish, R., Gaikwad, S. N. S., Kovacs, G., Veit, A., Krishna, R., Arrieta Ibarra, I., Simoiu, C., Wilber, M., Belongie, S., Goel, S., et al. (2017). Crowd research: Open and scalable university laboratories. In *Proceedings of the 30th annual ACM symposium on user interface software and technology*, pages 829–843. 23

Verscheure, D., Demeulenaere, B., Swevers, J., De Schutter, J., and Diehl, M. (2009). Time-Optimal Path Tracking for Robots: A Convex Optimization Approach. *IEEE Transactions on Automatic Control*, 54(10):2318–2327. 19, 54, 56

Vu, M. N., Beck, F., Schwegel, M., Hartl-Nesic, C., Nguyen, A., and Kugi, A. (2023). Machine learning-based framework for optimally solving the analytical inverse kinematics for redundant manipulators. *Mechatronics*, 91:102970. 12

Wang, M., Xiao, J., Liu, S., and Liu, H. (2021). A third-order constrained approximate time-optimal feedrate planning algorithm. *IEEE Transactions on Robotics*, 38(4):2295–2307. 18

Wang, R., Xie, Y., Chen, X., and Li, Y. (2023). Path-constrained time-optimal motion planning for robot manipulators with third-order constraints. *IEEE/ASME Transactions on Mechatronics*, 28(6):3005–3016. 20

Wang, Y. and Artemiadis, P. (2013). Closed-form inverse kinematic solution for anthropomorphic motion in redundant robot arms. *Adv Robot Autom*, 2(110):2. 10

Weinreuter, M. (2020). A framework for online education using a remotely accessible robotics lab. Master's thesis, Karlsruhe Institute of Technology. 74

Weiss Robotics GmbH & Co. KG (2026). Weiss CRG 200. https://weiss-robotics.com/de/servo-electric/crg-series/. Accessed: July 29, 2026. 77

White, R., Christensen, H. I., and Quigley, M. (2016). SROS: securing ROS over the wire, in the graph, and through the kernel. *CoRR*. 82

Wiedmeyer, W., Altoé, P., Auberle, J., Ledermann, C., and Kröger, T. (2021). A real-time-capable closed-form multi-objective redundancy resolution scheme for seven-dof serial manipulators. *IEEE Robotics and Automation Letters*, 6(2):431–438. 3, 8, 36, 38, 41, 42, 113, 115

Wiedmeyer, W., Mende, M., Hartmann, D., Bischoff, R., Ledermann, C., and Kröger, T. (2019). Robotics education and research at scale: A remotely accessible robotics development platform. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3679–3685. IEEE. 4, 35, 74, 78, 83, 91, 113

Wilson, S., Glotfelter, P., Mayya, S., Notomista, G., Emam, Y., Cai, X., and Egerstedt, M. (2021). The robotarium: Automation of a remotely accessible, multi-robot testbed. *IEEE Robotics and Automation Letters*, 6(2):2922–2929. 26

Wilson, S., Glotfelter, P., Wang, L., Mayya, S., Notomista, G., Mote, M., and Egerstedt, M. (2020). The robotarium: Globally impactful opportunities, challenges, and lessons learned in remote-access, distributed control of multirobot systems. *IEEE Control Systems Magazine*, 40(1):26–44. 26

Wirth, R. (2019). Visual monitoring of a robot cell for experiment automation. Master's thesis, Karlsruhe Institute of Technology. 98

Wittmann, J. and Rixen, D. J. (2022). Time-optimization of trajectories using zero-clamped cubic splines and their analytical gradients. *IEEE Robotics and Automation Letters*, 7(2):4528–4534. 20

Xu, W., Yan, L., Mu, Z., and Wang, Z. (2016). Dual arm-angle parameterisation and its applications for analytical inverse kinematics of redundant manipulators. *Robotica*, 34(12):2669–2688. 10

Zanchettin, A. M. and Lacevic, B. (2022). Safe and minimum-time path-following problem for collaborative industrial robots. *Journal of Manufacturing Systems*, 65:686–693. 21

Zeng, Z., Chen, Z., Shu, G., and Chen, Q. (2018). Optimization of analytical inverse kinematic solution for redundant manipulators using ga-pso algorithm. In *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pages 446–451. IEEE. 12

Zhang, S., Sun, D., and Liao, Q. (2024). A receding horizon online trajectory generation method for time-efficient path tracking with dynamic factors. *IEEE/ASME transactions on mechatronics*. 21

Zhou, G., Dean, V., Srirama, M. K., Rajeswaran, A., Pari, J., Hatch, K., Jain, A., Yu, T., Abbeel, P., Pinto, L., et al. (2023). Train offline, test online: A real robot learning benchmark. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9197–9203. IEEE. 24

List of Figures

List of Tables