DISSERTATION

Fast trajectory planning frameworks for robotic systems

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Doktors der technischen Wissenschaften (Dr. techn.)

unter der Leitung von

Univ.-Prof. Dipl.-Ing. Dr. techn. Andreas KUGI
E376
Institut für Automatisierungs- und Regelungstechnik

eingereicht an der
Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik

von

Minh Nhat VU
Matrikelnummer: 11742814

Wien, 31st May 2023

Studiendekan
ao.-Prof. Dr. Thilo Sauter

Betreuer
Univ.-Prof. Dipl.-Ing. Dr. techn. Andreas Kugi

Tag des Rigorosums
26.05.2023

Prüfungsvorsitzender
Prof. Dr.-Ing. habil. Thomas Meurer
Univ.-Prof. Dipl.-Ing. Dr. techn. Andreas Kugi

Erster Gutachter
Prof. Dr.-Ing. habil. Thomas Meurer

Zweiter Gutachter
ao. Univ.-Prof. Dipl.-Ing. Dr. techn. Markus Vincze

For my loved ones.

Acknowledgement

Approximately 24 years ago, my father presented me with a small robot toy. I was filled with joy and gratitude, knowing that it had cost him a week's salary. In the 1990s, electronic devices in Vietnam were considerably expensive due to the country being in its early stages of development. Unfortunately, the toy stopped working after a few days, and the shop refused to exchange it. Despite my sadness, I found solace in the fact that I had enjoyed playing with it for a while. It was then that I made a personal commitment to learn how to repair and develop robots.

I am profoundly grateful for the opportunity to embark on this journey towards the completion of my PhD thesis in the field of robotics, automation, and control systems. Achieving this goal would not have been possible without the tremendous support, encouragement, and inspiration from numerous individuals who have contributed to my academic and personal growth.

First and foremost, I would like to express my sincere gratitude to my esteemed Doktorvater, Univ. Prof. Dr. Andreas Kugi, for his expert guidance, unwavering support, and encouragement throughout my research. Your mentorship has equipped me with the confidence to overcome the challenges inherent in this academic pursuit and has instilled in me an enduring passion for research. Furthermore, I am grateful to Prof. Andreas for providing me with the opportunity to stay in Vienna, where I had the chance to meet my wife in 2018. Words cannot adequately convey my appreciation for your generosity. I would also like to extend my thanks to the members of my thesis committee, Prof. Dr. Thomas Meurer and ao.Univ. Prof. Dr. Thilo Sauter, for their invaluable insights, thoughtful feedback, and constructive criticism. I would especially like to thank Ms. Maria Ochsenre-

iter for her hard work in the ACIN office, you handle all the paperwork in ACIN smoothly. I don't think we could do the research work in peace without you.

Additionally, I am deeply indebted to my colleagues and collaborators, whose presence and support have enriched my academic experience and provided a nurturing network of like-minded individuals. Your camaraderie, enthusiasm, and creativity have been indispensable in sustaining my passion for research. Thank you, Dipl.-Ing. Thomas, for being my roommate and being my partner in crime, thank you Dr.techn. Christian, for your guidance and motivation, thank you Dipl.-Ing Florian, for the collaboration and discussions in your library and thank you Dipl.-Ing Moine, for my disruption in your office which is the library of ACIN, thank you Dipl.-Ing Michael, for your unconventional help, thank you Dipl.-Ing Gerald and Msc. Lucia, for being the same "monkey"-Zodiac as me, thank Dipl.-Ing Christoph, for sharing the same passion for spicy food with me, and thank Msc. Thies for bringing me cheeses and playing football with me.

To my father, Vu Thai Binh, my mother, Tran Thi Mai Hoa, and my younger brother, Vu Nhat Tan, I offer my heartfelt appreciation for their unwavering love, unconditional support, and constant encouragement. Your unwavering belief in me has been a source of strength and inspiration, enabling me to overcome the challenges of this journey with grace and perseverance.

To my wife, Le Thi Thuy Trang, thank you for entering my life and accompanying me during my PhD journey either in the United States, Austria, or Vietnam. Your presence has been an integral part of my PhD journey, and I am truly grateful for your unconditional love and unwavering support.

Finally, I extend my gratitude to the faculty and staff of the Automation and Control Institute (ACIN), TU Wien, whose unwavering dedication to academic excellence and commitment to fostering a vibrant and supportive academic community have played a crucial role in my success. Your tireless efforts to provide state-of-the-art resources, facilities, and mentorship have been invaluable in enabling me to achieve this significant milestone.

With gratitude and appreciation,

Minh Nhat Vu

Vienna, 4 May 2023.

I would like to write a Vietnamese version of this Acknowledgement as my parents can read it.

Khoảng 24 năm trước đây, ba của tôi tặng cho tôi một con robot đồ chơi. Tôi rất vui mừng và biết ơn, khi biết rằng nó đã tốn cha một tuần lương. Vào thập kỷ 1990, các thiết bị điện tử ở Việt Nam rất đắt đỏ do đất nước đang ở giai đoạn phát triển. Thật không may, sau vài ngày, con đồ chơi bị hỏng và cửa hàng từ chối đổi nó. Mặc dù rất buồn, nhưng tôi tìm được sự an ủi bởi vì tôi cũng đã chơi với nó trong một thời gian ngắn. Lúc đó, tôi đã tự nhủ rằng mình phải trở thành một nhà khoa học để có thể sửa chữa robot.

Tôi rất biết ơn vì đã được trao cơ hội để hoàn thành luận án tiến sĩ trong lĩnh vực robot, tự động hóa và hệ thống điều khiển. Để đạt được mục tiêu này không thể thiếu sự hỗ trợ, động viên và nguồn cảm hứng từ thầy của tôi, gia đình của tôi, và cả bạn bè của tôi.

Đầu tiên và quan trọng nhất, tôi xin bày tỏ lòng biết ơn chân thành đến Doktorvater đáng kính của tôi, GS. TS. Andreas Kugi, vì sự hướng dẫn tận tâm, sự hỗ trợ vững chắc, và sự động viên trong suốt quá trình nghiên cứu của tôi. Sự hướng dẫn của người thầy đã trang bị cho tôi sự tự tin để vượt qua những thách thức trong việc theo đuổi học thuật và đã truyền cảm hứng cho tôi trong việc nghiên cứu. Hơn nữa, tôi biết ơn GS. Andreas đã cho tôi cơ hội để làm việc tại Vienna, nơi tôi đã có cơ hội gặp gỡ vợ tôi vào năm 2018. Lời nói không thể diễn đạt đủ lòng biết ơn của tôi vì sự giúp đỡ vô bờ bến của thầy. Tôi cũng muốn gửi lời cảm ơn đến các thành viên trong hội đồng đánh giá luận án của tôi, GS. TS. Thomas Meurer và ao.Univ. GS. TS. Thilo Sauter, vì những đóng góp, những phản hồi tích cực, và sự tranh luận mang tính xây dựng cao từ họ. Tôi đặc biệt muốn cảm ơn cô Maria, thư ký của thầy Andreas, vì đã làm việc chăm chỉ tại văn phòng ACIN, cô xử lý mọi thủ tục giấy tờ ở ACIN một cách trôi chảy. Tôi không nghĩ rằng chúng tôi có thể thực hiện công việc nghiên cứu một cách yên bình mà không có cô Maria.

Hơn nữa, tôi rất cảm kích đối với đồng nghiệp và các cộng tác viên, sự hiện diện và sự hỗ trợ của họ đã làm phong phú thêm trải nghiệm học thuật của tôi. Sự đoàn kết, sự hăng hái, và sự sáng tạo của các bạn đã không thể thiếu trong việc duy trì đam mê nghiên cứu của tôi.

Với ba, Vũ Thái Bình, mẹ, Trần Thị Mai Hoa, và em trai, Vũ Nhật Tân, "Con xin

bày tỏ lòng biết ơn chân thành vì tình yêu không đổi, sự hỗ trợ vô điều kiện, và sự động viên liên tục của gia đình tôi. Lòng tin của của gia đình đã mang đến nguồn sức mạnh và cảm hứng cho con, giúp con vượt qua những khó khăn trong hành trình này."

Với vợ, Lê Thị Thùy Trang, "Cảm ơn em đã đến trong cuộc đời anh và đi cùng anh trong hành trình tiến sĩ, dù ở Hoa Kỳ, Áo hoặc Việt Nam. Sự hiện diện của em đã là một phần quan trọng trong hành trình của anh và anh thật sự biết ơn tình yêu không điều kiện và sự hỗ trợ vững chắc của em".

Cuối cùng, tôi muốn bày tỏ lòng biết ơn nhân viên của Viện Automation và Control (ACIN). Những nỗ lực không ngừng nghỉ của bạn để cung cấp điều kiện, cơ sở vật chất là không thể thiếu trong việc giúp tôi đạt được mốc quan trọng này.

Với tất cả lòng thành,

Vũ Nhật Minh

Thành phố Viên, 4 May 2023.

Kurzzusammenfassung

In den letzten Jahren ist die Nachfrage nach agilen und reaktionsschnellen Robotersystemen erheblich gestiegen, was schnelle und zuverlässige Bahnplanungsalgorithmen erfordert. Die Bewegungsplanung von Robotern war Gegenstand umfangreicher Forschungsarbeiten, die zu verschiedenen Lösungen geführt haben. Trotz der Vielfalt der verfügbaren Bewegungsplanungsmethoden ist die Berechnung einer kollisionsfreien Bahn in Echtzeit unter systematischer Berücksichtigung von Zustands- und Stellgrößenbeschränkung je nach Anwendungsfall eine Herausforderung. Bei vielen industriellen Bewegungsplanungsaufgaben, wie z.B. Montage- oder Pick-and-Place-Aufgaben, treten wiederkehrende Bewegungen in sehr ähnlichen Szenarien auf, bei denen die Start- und Zielpositionen innerhalb vordefinierter Teilräume bleiben und nur kleine Abweichungen aufweisen. In solchen Fällen ist es nicht notwendig, die gesamte Trajektorie komplett neu zu berechnen, da die sich wiederholende Natur des Prozesses ausgenutzt werden kann. Diese Arbeit widmet sich diesem Thema und entwickelt rechnerisch effiziente Lösungen, die in Echtzeitanwendungen eingesetzt werden können.

Die vorgeschlagenen Konzepte basieren auf physikalischen Modellen und nutzen fortgeschrittene Optimierungsmethoden, um Trajektorien zu generieren, die Hindernisse vermeiden mit der Systemdynamik konsistent sind. Die Konzepte werden durch Simulationen und Experimente auf verschiedenen Plattformen evaluiert, um ihre Effektivität bei der Erzeugung kollisionsfreier und dynamisch realisierbarer Trajektorien in Echtzeit zu demonstrieren. Dies trägt zur Verbesserung der Leistung von Robotersystemen bei und ermöglicht agilere und reaktionsfähigere Systeme, die sich an veränderte Umgebungen und Anforderungen anpassen können.

In dieser Arbeit wird das folgende Szenario betrachtet. Ein 3D-Portalkran im Labormaßstab wird verwendet, um Güter oder Materialien von einem bestimmten Ort zu einem anderen Ort in einer statischen Umgebung mit bekannten Hindernissen zu transportieren. Ziel ist es, eine zeitoptimale Trajektorie von einem gegebenen Startpunkt zu einem gegebenen Zielpunkt zu planen, wobei sowohl Hindernisse als auch dynamische Beschränkungen für Zustandsvariablen und Stellsgrösen berücksichtigt werden. Der Fokus der Arbeit liegt darin eine schnelle Trajektorienplanung in Echtzeit zu realisieren, wenn sich der Start- und/oder Zielpunkt ändert oder bewegt. Zu diesem Zweck werden zwei Konzepte für die Trajektorienplanung vorgeschlagen.

Das erste, in dieser Arbeit vorgeschlagene Konzept basiert auf dem informierten optimalen Rapid Exploring Random Tree-Algorithmus (informierter RRT*). Mit diesem Algorithmus werden Trajektorienbäume erstellt, die für die Neuplanung wiederverwendet werden können, wenn sich der Start- und/oder Zielpunkt ändert. Im Gegensatz zu bestehenden Ansätzen enthält der vorgeschlagene Algorithmus einen lokalen Planer basierend auf einem linear quadratic minimum-time (LQTM) Algorithmus. Um die Effizienz des Algorithmus weiter zu verbessern, wird die Branch-and-Bound-Methode integriert. Dadurch werden Punkte im Baum eliminiert, die nicht dazu beitragen, bessere Lösungen zu finden, was den Speicherbedarf und die Rechenkomplexität reduziert. Simulationsergebnisse anhand eines validierten mathematischen Modells eines 3D-Portalkrans im Labormaßstab belegen die Brauchbarkeit des vorgeschlagenen Ansatzes. Die Ergebnisse zeigen, dass der vorgeschlagene Trajektorienplanungsalgorithmus nahezu zeitoptimale und kollisionsfreie Trajektorien generieren kann, während er gleichzeitig dynamische Beschränkungen und Änderungen der Zielzustände berücksichtigt. Insgesamt bietet das vorgeschlagene Konzept eine neue und innovative Lösung für eine schnelle und effiziente Trajektorienplanung in Umgebungen mit Hindernissen. Durch die Integration des lokalen LQTM Planers und der Branch-and-Bound-Methoden kann der Algorithmus dynamische Eigenschaften systematisch berücksichtigen und die Berechnungseffizienz verbessern, so dass er sich für eine schnelle Neuplanung eignet.

Das erste Trajektorienplanungskonzept bietet eine gute Lösung für Online Neuplanungsaufgaben. Um jedoch ein sich dynamisch bewegendes Ziel zu berücksichtigen, muss der Trajektorienbaum neu berechnet werden, was die Echtzeitfähigkeit einschränkt. Daher wird ein zweites optimierungsbasiertes Konzept für die Trajektorienplanung entwickelt. Dieses besteht aus zwei Hauptkomponenten, nämlich ei-

nem Offline-Trajektorienplaner und einem Online-Trajektorienreplanner. In der Offline-Phase wird eine zeitoptimale, kollisionsfreie und dynamisch konsistente Trajektoriendatenbank erzeugt, indem alle möglichen Trajektorien berechnet werden, die den vordefinierten Unterraum der Startpunkt mit dem Unterraum der Zielpunkte verbinden. Die resultierende Trajektoriendatenbank wird dann für die Online-Nutzung gespeichert. In der Online-Phase nutzt der Trajektorienplaner diese Datenbank, um eine optimale Trajektorie in Echtzeit zu erzeugen. Der Online-Trajektorienplaner basiert auf einem linearen, beschränkten quadratischen Programm. Der Online-Planer berücksichtigt alle Änderungen des Zielzustands und stellt sicher, dass die generierte Trajektorie kollisionsfrei und dynamisch konsistent ist. Um eine genaue Bahnverfolgung zu gewährleisten, wird ein Trajektorienfolgeregler entwickelt, der die dynamischen Beschränkungen des Portalkrans berücksichtigt und Modellungenauigkeiten, Störungen und andere nicht modellierte Effekte kompensiert. Die Simulationen und experimentellen Ergebnisse zeigen die Brauchbarkeit des vorgeschlagenen Konzepts für die Trajektorienplanung und regelung des 3D-Portalkrans. Das Konzept generiert glatte und zeitoptimale Trajektorien, die den dynamischen Einschränkungen des Systems genügen und Hindernissen in Echtzeit ausweichen.

Aufgrund der Allgemeingültigkeit ist das vorgeschlagene optimierungsbasierte Trajektorienplanungskonzept nicht auf den 3D-Portalkran beschränkt und kann auf andere Robotersysteme angewandt werden. Um dies zu demonstrieren, wird der Algorithmus dazu eingesetzt, ein sphärisches Pendel mit Hilfe eines Roboters mit 7 Freiheitsgraden (KUKA LBR iiwa 14 R820) von einer in einem bestimmten Arbeitsbereich beliebigen Ausgangslage in die obere Ruhelage auf zu schwingen. Die mit dem Algorithmus geplante Trajektorie wird mit einem zeitvarianten Riccati-Regler stabilisiert. Die Simulationen und die experimentellen Ergebnisse bestätigen die Funktionsfähigkeit des vorgeschlagenen Konzepts.

Abstract

In recent years, the demand for agile and responsive robotic systems has grown significantly, requiring fast and reliable trajectory planning algorithms. Robot motion planning has been the subject of extensive research, resulting in various solutions. Despite the plethora of available motion planning methods, real-time calculation of a collision-free trajectory that systematically accounts for state and control input constraints remains a challenge depending on the specific application. For many industrial motion planning tasks, such as assembly or pick-and-place tasks, repetitive motions occur in very similar scenarios where the starting and target positions remain within predefined subspaces and only exhibit small changes. In such cases, it is unnecessary to completely re-calculate the entire trajectory, as the repetitive nature of the process can be exploited. This thesis is devoted to this topic and develops computationally efficient solutions that can be employed in real-time applications.

The proposed frameworks are based on first-principles models and utilize advanced optimization techniques to generate trajectories that can avoid obstacles and meet the dynamic constraints of the system. The frameworks are evaluated by simulations and experiments on different platforms to demonstrate their feasibility in generating collision-free and dynamically consistent trajectories in real time. This helps to enhance the performance of robotic systems, enabling more agile and responsive systems that can adapt to changing environments and requirements.

In this thesis, the following scenario is considered. A lab-scale 3D gantry crane is employed to move goods or materials from one dedicated place to another place in a static environment with known obstacles. The goal is to plan a time-optimal

trajectory from a given starting point to a given target point, taking into account both obstacles and dynamic constraints on the state variables and control inputs. The focus of this work is to realize fast real-time trajectory planning when the start and/or target state changes or moves. To this end, two trajectory planning frameworks are proposed.

The first framework is based on the informed optimal rapidly exploring random tree algorithm (informed RRT*). This algorithm is used to build trajectory trees that can be reused for replanning when the start and/or goal states change. In contrast to existing approaches, the proposed algorithm includes a local planner with a linear quadratic minimum-time (LQTM) solver. To enhance the efficiency of the algorithm, a branch-and-bound method is integrated. This helps to eliminate points in the tree that do not contribute to finding better solutions, reducing memory consumption and computational complexity. Simulation results using a validated mathematical model of a lab-scale 3D gantry crane demonstrate the feasibility of the proposed approach. The results show that the proposed trajectory planning algorithm can generate near time-optimal and collision-free trajectories while considering dynamic constraints and changes in target states. Overall, the proposed trajectory planning framework offers a new and innovative solution for fast and efficient trajectory planning in environments with obstacles. The integration of the local LQTM planner and the branch-and-bound method enable the algorithm to systematically consider dynamic properties and improve the computational efficiency, making it suitable for fast replanning.

The first trajectory planning framework shows a good performance for online replanning tasks. However, to account for a dynamically moving target, the trajectory tree has to be recomputed, which limits the real-time capability. Therefore, a second optimization-based framework is developed. The optimization-based trajectory planning framework consists of two main components, i.e., an offline trajectory planner and an online trajectory replanner. In the offline phase, a time-optimal, collision-free, and dynamically feasible trajectory database is generated by computing all possible trajectories that connect starting points in a the predefined subspace to target points in another subspace. The resulting trajectory database is then stored for online use. In the online phase, the trajectory planner utilizes the database to generate an optimal trajectory in real time. The online trajectory planner is based on linear constrained quadratic programming. The online planner takes into account all changes in the target state and ensures that the generated

trajectory is collision free and dynamically feasible. To ensure accurate trajectory tracking, a trajectory tracking controller is developed that takes into account the dynamic constraints of the gantry crane and compensates for model inaccuracies, disturbances, and other non-modeled effects. Simulation and experimental results demonstrate the effectiveness of the proposed trajectory planning and control framework for the 3D gantry crane. The framework can generate smooth and time-optimal trajectories that satisfy the dynamic constraints of the system and avoid obstacles in real time.

Due to its generality, the proposed optimization-based trajectory planning algorithm is not limited to the 3D gantry crane and can be applied to other robotic systems. To demonstrate this, the algorithm is used to swing up a spherical pendulum with a 7-axis robot (KUKA LBR iiwa 14 R820) from an arbitrary position in a starting subspace. The trajectory planned with the proposed framework is stabilized by a discrete time-variant linear quadratic regulators. Simulations and experimental results demonstrate the feasibility of the proposed approach.

Contents

1

Introduction

The focus of this thesis is motivated and presented in this section. Additionally, the objectives and the content of this work are outlined.

1.1 Motivation

In recent decades, the rapid development in robotics and automation has enormously changed the structure of society and the way we live and work. In many manual tasks, such as picking, assembling, gluing, welding, transferring, etc., robots and machines are designed to perform these tasks autonomously, precisely, and quickly. Mostly, their performance is better than that of human workers. As a result, production time and costs can be significantly reduced. This leads to the emergence of new industries and directs our time and efforts to further improve the standards of living of our society.

Industrial robots and collaborative robots, e.g., SCARA robots [1], delta robots [2], serial manipulators [3, 4], gantry systems [5, 6, 7], are integrated with general-purpose hardware architecture combined with application-based software for process control and monitoring. This creates flexibility and helps these robots to be used in many different processes and applications. In industry, these robots are

mostly programmed and developed by experts for specific repetitive tasks in structured environments. Custom programming for a specific industrial application is still very costly, although much research effort has been made to render robot programming easier in the last years, see, e.g., [8, 9, 10, 11]. In mass production, the long-term benefits outweigh these high costs, e.g., in automotive assembly, pick-and-place, lifting and sorting in large warehouses and ports. For low-volume high-mix production, the increasing flexibility and adaptability of cognitive robotic systems, developed in the last years, make it more and more possible that robots also slowly penetrate this market.

In this context, motion planning is an important task that is widely used in various robotic applications. The goal of motion planning is to find a collision-free, dynamically feasible path or trajectory that drives the robot from its starting configuration to a target configuration to fulfill specific tasks. Robot motion planning has been extensively studied in the literature, leading to various solutions which can be essentially classified into sampling-based, e.g., [12, 13] and optimization-based approaches [14, 15]. Despite this plethora of motion-planning methods available, calculating a collision-free trajectory which systematically accounts for state and control input constraints in real time, depending on the application, is still a challenge. Many motion planning tasks in industry, e.g., assembly or pick-and-place tasks, refer to scenarios with nearby repetitive motions where starting and target position are always within predefined subspaces and sometimes are only slightly changing. In this case, there is no need to completely calculate the whole trajectory, but exploit the repetitive properties of the process. This leads to computationally much more efficient solutions which may be used in real-time applications. In this work, novel trajectory planning frameworks are proposed for such scenarios, which include an offline and an online planner capable of reacting to changing targets or changing environmental conditions during process execution. The offline trajectory planner is used to create a trajectory database which serves as a basis for the online trajectory replanner. The online planner can quickly compute locally deformed trajectories that result in suboptimal, but still collision-free and dynamically feasible solutions at moderate computational costs.

1.2 Objectives and approaches presented in this work

The aim of this thesis is to develop a trajectory planning framework that is able to compute collision-free and dynamically feasible trajectories in real time for a certain class of robotic applications. The following scenario is considered: A 3D gantry crane has to move goods or material from one dedicated place (starting space) to another place (target space) in a static environment with known obstacles. The task is to plan a time-optimal trajectory from a given starting point to a given target point, which systematically accounts for both the obstacles and the dynamic constraints on the state variables and control inputs. The focus of this work is on a solution that allows for a fast online planning if the starting and/or target point are changing or moving. Although system constraints and static obstacles increase the complexity of the problem, the proposed trajectory planning framework is able to provide a real-time solution. In this work, two trajectory planning frameworks, one based on sampling-based and the other one on optimization-based motion planning, are presented.

Inspired by the sampling-based trajectory planning approach RRT*, the so called flat-informed RRT* is developed. First, a collision-free and dynamically feasible trajectory tree is built by sampling the system states in an informed set that guarantees finding a better solution. To connect the sampled states in the trajectory tree, the linear quadratic minimum-time (LQMT) approach is employed, which exploits the property of differential flatness of the system. In the second step, when the target state is changed, the proposed flat-informed RRT* quickly generates a feasible trajectory. In contrast to other works in the literature, e.g., [16, 13, 12], the proposed flat-informed RRT* systematically considers the system dynamics constraints and the state and input constraints, which are important safety factors.

The proposed flat-informed RRT* achieves good performance for online replanning tasks ($\leq 100\,\text{ms}$ for the considered lab-scale 3D gantry crane). However, to obtain a better time-optimal solution, the size of the trajectory becomes larger, which limits the real-time capability. Therefore, an optimization-based framework for trajectory planning is developed, which consists of two steps. First, an offline trajectory planner is implemented to compute a time-optimal, collision-free, and

dynamically feasible trajectory database that connects all possible initial states from a starting subspace to all possible states in a target subspace. Second, using the linear constrained quadratic programming (LCQP), the online trajectory replanner utilizes the computed trajectory database to generate a suboptimal trajectory in real time. In comparison with the most successful practical studies known from the literature, e.g., [5, 17], the proposed optimization-based trajectory planning has the ability to generate near time-optimal dynamically feasible trajectories online for scenarios with obstacles and a moving target. The feasibility of the proposed optimization-based approach is shown by extensive Monte-Carlo simulations and experiments. Due to its generality, this proposed optimization-based trajectory planning algorithm can be applied to a wide range of other robotic systems, e.g., the collaborative robot KUKA LBR iiwa 14 R820 [18].

Main parts of this work and extended robotic applications are already published or accepted in the following journal and conference papers:

[19] M. N. Vu, P. Zips, A. Lobe, F. Beck, W. Kemmetmueller, and A. Kugi, "Fast motion planning for a laboratory 3D gantry crane in the presence of obstacles". *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9508-9514, 2020.

[20] M. N. Vu, C. Hartl-Nesic, and A. Kugi, "Fast swing-up trajectory optimization for a spherical pendulum on a 7-dof collaborative robot," in Proceedings of the International Conference on Robotics and Automation (ICRA), pp. 10114–10120, 2021.

[21] M. N. Vu, A. Lobe, F. Beck, T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Fast trajectory planning and control of a lab-scale 3d gantry crane for a moving target in an environment with obstacles," *Control Engineering Practice*, vol. 126, p. 105255, 2022.

[22] M. N. Vu, M. Schwegel, C. Hartl-Nesic, and A. Kugi, "Sampling-based trajectory (re)planning for differentially flat systems: Application to a 3d gantry crane," *IFAC-PapersOnLine*, vol. 55, no. 38, pp. 33–40, 2022

[23] F. Beck, M. N. Vu, C. Hartl-Nesic, and A. Kugi, "Singlularity avoidance with application to online trajectory optimization for serial manipulators," *IFAC-PapersOnLine*, [Accepted for IFAC World Congress], 2023.

[24] M. Zimmermann, M. N. Vu, F. Beck, C. Hartl-Nesic, A. Nguyen, and A. Kugi, "Two step online trajectory planning of a quadcopter in indoor environments with

obstacles," *IFAC-PapersOnLine*, [Accepted for IFAC World Congress], 2023.

[25] M. N. Vu, F. Beck, M. Schwegel, C. Hartl-Nesic, A. Nguyen, and A. Kugi, "Machine learning-based framework for optimally solving the analytical inverse kinematics for redundant manipulators," *Mechatronics*, vol. 91, p. 102970, 2023.

1.3 Outline of the thesis

The remainder of this thesis is structured as follows.

In Section 2 "Mathematical modeling", the basics for the mathematical modeling of the two considered robotic systems used in this work, i.e. a lab-scale 3D gantry crane and the KUKA LBR iiwa 14 R820, are presented. In addition, for the lab-scale 3D gantry crane, the differential flatness property of the corresponding mathematical model is derived.

Section 3 "Sampling-based trajectory planning for a 3D gantry crane" presents a novel sampling-based trajectory planning framework, named flat informed RRT*, which is applied to the lab-scale 3D gantry crane. The focus of this approach is to develop a novel fast motion planning algorithm for differentially flat systems, where intermediate results can be stored and reused for further tasks, such as online replanning. After a review of related sampling-based trajectory planning approaches in the literature, the flat-informed RRT* algorithm is presented in Section 3.2. This allows to create trajectory trees that are reused for fast replanning processes when, for instance, the target state changes. In Section 3.3, simulation studies show the feasibility of the proposed algorithm for different scenarios. This section is largely based on the author's publication [22].

In Section 4 "Optimization-based trajectory planning for a 3D gantry crane", a more flexible optimization-based approach is presented, which outperforms the flat-informed RRT* of Section 3 in terms of computational speed. The focus of this section is on a novel two-step optimization-based trajectory planning algorithm. First, an offline trajectory planner is implemented to compute a time-optimal, collision-free, and dynamically feasible trajectory database that connects all possible initial states of the considered lab-scale 3D gantry crane from a predefined starting subspace to all possible states in a target subspace. Second, an online trajectory replanner is designed which is based on this trajectory database and a linear constrained quadratic programming (LCQP) approach to generate an opti-

mal trajectory in real time. In addition, a model predictive control (MPC)-based trajectory tracking controller is designed for the considered lab-scale 3D gantry crane to compensate for model inaccuracies, disturbances, and other unmodeled effects. Finally, simulation and experimental results are presented to demonstrate the performance of the proposed trajectory planning framework and the trajectory tracking controller. This section is largely based on the author's publications [19] and [21].

Section 5 "Swing-up trajectory optimization for a pendulum on a collaborative robot" presents an extended application of the novel two-step optimization-based trajectory replanning introduced in Section 4. The considered system has nine degrees of freedom (DoF) and consists of a spherical 2-DoF pendulum mounted on a 7-DoF robot. The task is to compute an optimal swing-up trajectory of the spherical 2-DoF pendulum using the 7-DoF KUKA LBR iiwa 14 R820 from an arbitrary initial configuration in a certain admissible range of the complete system. The main focus of this section is to apply the optimization-based trajectory planning presented in Section 4 to the swing-up by systematically incorporating the kinematic and dynamic constraints. After a literature review, offline trajectory optimization is employed to build a database of swing-up trajectories. Then, a fast trajectory replanner based on a constrained quadratic program is developed to compute the swing-up trajectory. Simulations and experiments are demonstrated for the swing up of the spherical pendulum using a discrete time-varying linear quadratic regulator (LQR) as a feedback controller. This section is largely based on the author's publication [20].

Finally, the thesis is concluded with a brief summary as well as an outlook on possible future research activities in Section 6.

2

Mathematical modeling

This section presents the mathematical modeling of the considered lab-scale 3D gantry crane and the seven degrees-of-freedom (DoF) collaborative robot KUKA LBR iiwa 14 R820 with a spherical pendulum mounted on the flange.

A gantry crane is a robotic system widely used in industry to transport goods, such as steel coils in the steel industry or containers in ports. In these scenarios, the objects must be moved along a process-oriented trajectory, while avoiding swinging of the payload for safety reasons. In this work, a lab-scale 3D gantry crane is considered which constitutes a mechanical system with five degrees of freedom (DoF). Three of these five DoFs are actuated by DC motors equipped with the incremental encoders. In Section 2.1, the mathematical model of the 3D gantry crane is derived. Assuming that the payload of the 3D gantry crane is a point mass, the position of the payload is a differentially flat output, a property which will be exploited in the trajectory planning.

The collaborative robot KUKA LBR iiwa 14 R820 is a lightweight industrial manipulator with seven rotational DoF. This robot is designed for human collaboration as an “intelligent industrial work assistant” (iiwa). Each rotary joint is equipped with a torque sensor and a harmonic drive with a high gear ratio. In Sec-

tion 2.2, homogeneous transformations are introduced as the basis for deriving the forward kinematics and dynamics of the robot. Moreover, a custom-built spherical pendulum, which is mounted on the flange of the robot for swing-up experiments, will be presented and included in the mathematical model.

2.1 Lab-scale 3D gantry crane

In the following, the dynamic model of the 3D gantry crane is derived using the Euler-Lagrange formalism, see, e.g., [26].

2.1.1 Equations of motion

The CAD model of the lab-scale 3D gantry crane is illustrated in Fig. 2.1. The gantry crane system consists of five degrees of freedom $\mathbf{q}^{\mathrm{T}} = [s_x, s_y, s_z, \alpha, \beta]$, where s_x, s_y denote the position of the trolley on the bridge in x-, y-direction and s_z is the current hoisting cable length in z-direction. The variables α and β refer to the sway angles of the hoisting cables in the zy- and zx-plane, respectively, see Fig. 2.2. Note that the system state $\mathbf{q}$ is measured by five incremental encoders located at the three actuators for s_x, s_y, and s_z, and at the lifting drum for the two sway angles α and β.

By assuming that the two hoisting cables suspending the payload are identical and always under tension, the lab-scale 3D gantry crane can be modeled as a rigid-body system. Here, only the two sway angles α and β are considered as degrees of freedom and the twisting motion of the payload is neglected. The coordinate systems are given in Fig. 2.3 where $\mathcal{W}$ and $\mathcal{B}$ represent the world frame and the body frame, respectively. Since there are offset parameters $\mathbf{s}_{x,0}, \mathbf{s}_{y,0}$, and $\mathbf{s}_{z,max}$ caused by the construction of the gantry crane, the displacement vector from the world frame $\mathcal{W}$ to the trolley frame $\mathcal{C}$ reads as

$$(\mathbf{d}_{\mathcal{W}}^{\mathcal{C}})^{\mathrm{T}} = [s_{x,0} + s_x, s_{y,0} + s_y, s_{z,max}] \ . \tag{2.1}$$

Using the Euler angle parameterization, the rotation matrix for transforming from the trolley frame $\mathcal{C}$ to the body frame $\mathcal{B}$, located at the center of mass (CoM) of the payload, by a rotation of the angle β around the y-axis and a rotation of the

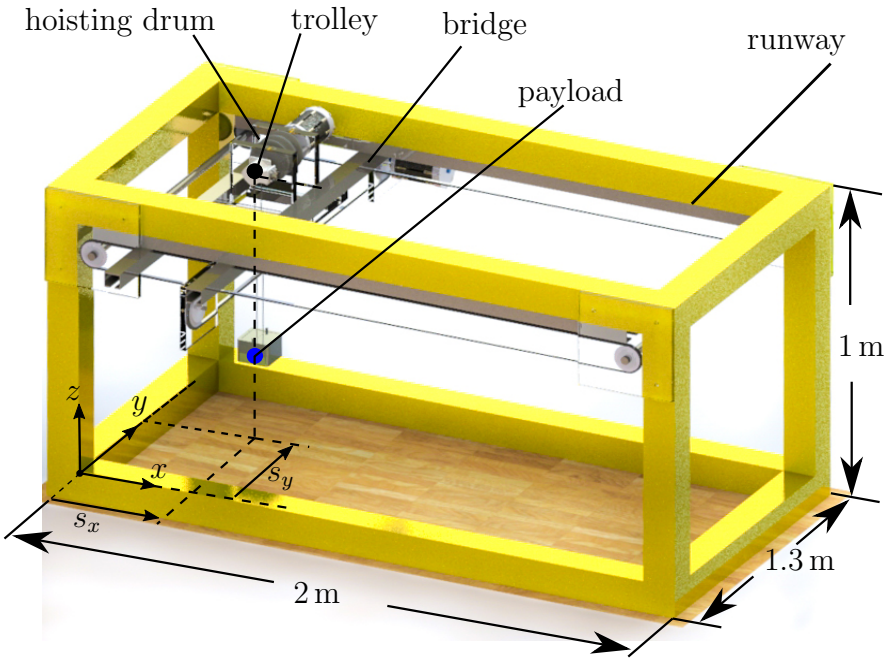


Figure 2.1: Schematic of the lab-scale 3D gantry crane for $\alpha = \beta = 0$.

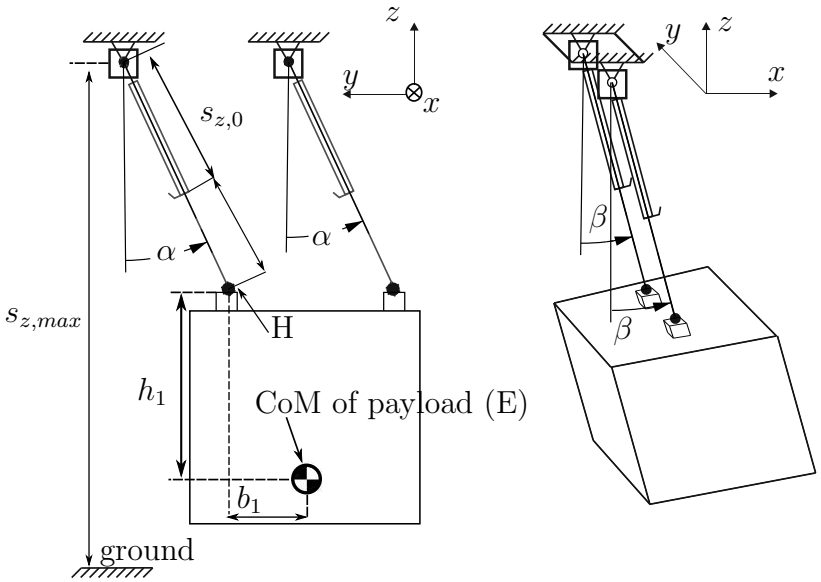


Figure 2.2: The payload with the corresponding hoisting cable angles α and β.

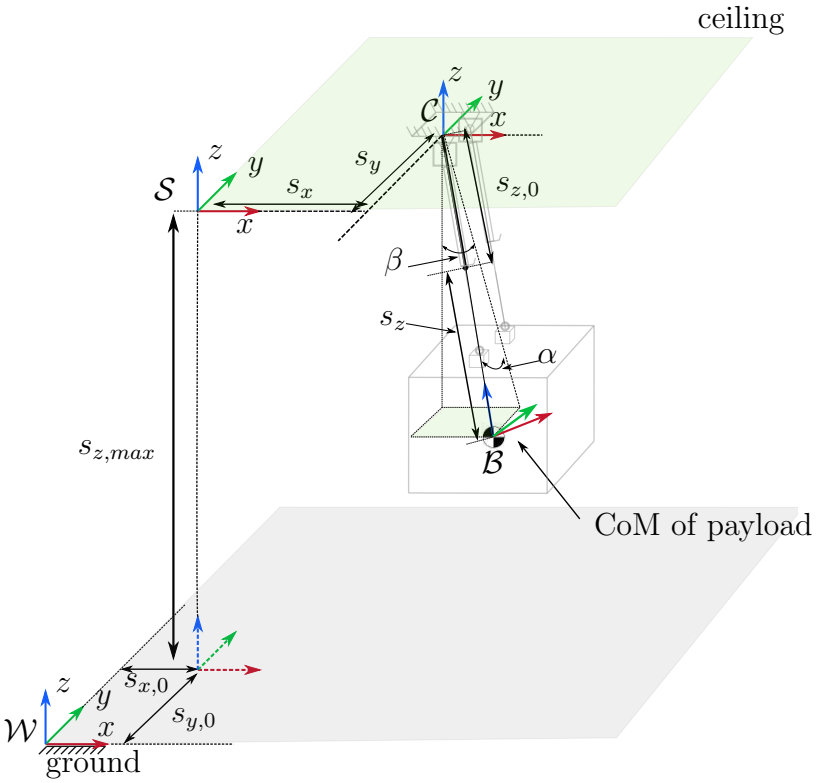


Figure 2.3: Coordinate systems of the 3D gantry crane.

angle α around the x-axis results in

$$\mathbf{R}_{\mathcal{C}}^{\mathcal{B}} = \mathbf{R}_{\beta,y}\mathbf{R}_{\alpha,x} \ , \tag{2.2}$$

where

$$\mathbf{R}_{\beta,y} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}, \ \mathbf{R}_{\alpha,x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \ . \tag{2.3}$$

The position of the CoM in the trolley coordinate frame $\mathcal{C}$, denoted as $\mathbf{r}_{\mathrm{CoM}}^{\mathcal{T}}$, comprises the position of $\mathbf{r}_{\mathcal{C}}^{\mathrm{H}}$ and the vector from the point H to the CoM $\mathbf{r}_{\mathcal{C}}^{\mathrm{H-CoM}}$, see Fig. 2.2. Thus, $\mathbf{r}_{\mathcal{C}}^{\mathrm{CoM}}$ is expressed in the form

$$\mathbf{r}_{\mathcal{C}}^{\mathrm{CoM}} = \mathbf{r}_{\mathcal{C}}^{\mathrm{H}} + \mathbf{r}_{\mathcal{C}}^{\mathrm{H-CoM}} \ , \tag{2.4}$$

with

$$\mathbf{r}_{\mathcal{C}}^{\mathrm{H}} = \mathbf{R}_{\mathcal{C}}^{\mathcal{B}} \begin{bmatrix} 0 \\ 0 \\ -s_{z,0} + s_z \end{bmatrix}, \ \mathbf{r}_{\mathcal{C}}^{\mathrm{H-CoM}} = \mathbf{R}_{\beta,y} \begin{bmatrix} 0 \\ -b_1 \\ -h_1 \end{bmatrix} \ . \tag{2.5}$$

Combining (2.1) and (2.5), the CoM of the payload in the world frame $\mathcal{W}$ reads as

$$\mathbf{r}_{\mathcal{W}}^{\mathrm{CoM}} = \mathbf{d}_{\mathcal{W}}^{\mathcal{C}} + \mathbf{r}_{\mathcal{C}}^{\mathrm{CoM}} = \begin{bmatrix} s_x + s_{x,0} + \sin(\beta)\cos(\alpha)(s_z - s_{z,0}) - \sin(\beta)h_1 \\ s_y + s_{y,0} - \sin(\alpha)(s_z - s_{z,0}) - b_1 \\ s_{z,max} + \cos(\beta)\cos(\alpha)(s_z - s_{z,0}) - \cos(\beta)h_1 \end{bmatrix} . \tag{2.6}$$

In the following, for a compact notation, the position of the CoM of the payload $\mathbf{r}_{CoM}^{\mathcal{W}}$ in the world coordinate frame is denoted by $\mathbf{r}$. The translational kinetic energy of the payload is computed as

$$T_{t,z} = \frac{1}{2} m_z \dot{\mathbf{r}}^{\mathrm{T}} \dot{\mathbf{r}} \ , \tag{2.7}$$

with the mass m_z of the payload. The translational parts of the kinetic energy caused by the movement of the bridge and the trolley in x-direction, $T_{t,x}$, and the

movement of the trolley along y-direction, $T_{t,y}$, are given in the form

$$T_{t,x} = \frac{1}{2}(m_x + m_y)\dot{s}_x^2 \ , \ T_{t,y} = \frac{1}{2}m_y\dot{s}_y^2 \ , \tag{2.8}$$

where m_x is the mass of the bride and m_y denotes the mass of the trolley and the lifting drum. The rotational parts of the kinetic energies caused by the two unactuated pendulum angles α and β read as

$$T_{r,\alpha} = \frac{1}{2}I_\alpha\dot{\alpha}^2 \ , \ T_{r,\beta} = \frac{1}{2}I_\beta\dot{\beta}^2 \ , \tag{2.9}$$

where I_α and I_β are the corresponding mass moments of inertia. The kinetic energies of the motor drives for the motion in x-, y-, and z-direction take the form

$$T_{r,\text{driver}} = \sum_i \frac{I_i\dot{s}_i^2}{R_i^2} \ , \tag{2.10}$$

where I_i is the moment of inertia of the motor associated to the axis $i \in \{x, y, z\}$ and R_i is the radius of the associated sprockets of the drives. Combining (2.7), (2.8), (2.9), and (2.10), the total kinetic energy of the 3D gantry crane reads as

$$T = \frac{1}{2}m_z\dot{\mathbf{r}}^\mathrm{T}\dot{\mathbf{r}} + \frac{1}{2}(m_x + m_y)\dot{s}_x^2 + \frac{1}{2}m_y\dot{s}_y^2 + \frac{1}{2}I_\alpha\dot{\alpha}^2 + \frac{1}{2}I_\beta\dot{\beta}^2 + \sum_i \frac{I_i\dot{s}_i^2}{R_i^2} \ . \tag{2.11}$$

The potential energy of the gantry crane with respect to the generalized coordinates $\mathbf{q}$ is calculated in the form

$$V = m_z g(s_{z,max} + \cos(\alpha)\cos(\beta)(s_z - s_{z,0}) - \cos(\beta)h_1), \tag{2.12}$$

where g is the gravitational acceleration. Applying the Lagrange formalism to the Lagrange function $L = T - V$, with the driving forces $\mathbf{u}^\mathrm{T} = [u_1, u_2, u_3]$ of the motors, yields the mathematical model written in state-space form

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mathbf{u}) = \begin{bmatrix} \dot{\mathbf{q}} \\ \mathbf{M}^{-1}(\mathbf{q})\left(\begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix} - \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} - \mathbf{g}(\mathbf{q}) \right) \end{bmatrix}, \tag{2.13}$$

with $\mathbf{z}^\mathrm{T} = [\mathbf{q}^\mathrm{T}, \dot{\mathbf{q}}^\mathrm{T}]$. The matrix $\mathbf{M}(\mathbf{q})$ denotes the symmetric and positive definite

mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ includes Coriolis and centrifugal terms, and $\mathbf{g}(\mathbf{q})$ are the forces associated with the potential energy. With the expressions

$$m_X = m_x + m_y + m_z + \frac{I_x}{R_x^2}\ ,\ m_Y = m_y + m_z + \frac{I_y}{R_y^2}\ ,\ m_Z = m_z + \frac{I_z}{R_z^2}\ ,$$

the entries of the system matrices

$$\mathbf{M}(\mathbf{q}) = \begin{bmatrix} M_{11} & \cdots & M_{15} \\ \vdots & \ddots & \vdots \\ M_{51} & \cdots & M_{55} \end{bmatrix}, \quad \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} C_{11} & \cdots & C_{15} \\ \vdots & \ddots & \vdots \\ C_{51} & \cdots & C_{55} \end{bmatrix}\ ,\ \mathbf{g}(\mathbf{q}) = \begin{bmatrix} g_1 \\ \vdots \\ g_5 \end{bmatrix}$$

follow as

$$\begin{aligned}
M_{11} &= m_X \\
M_{13} &= \cos(\alpha)\sin(\beta)m_z \\
M_{14} &= -\sin(\alpha)\sin(\beta)(s_z - s_{z,0}) \\
M_{15} &= m_z\cos(\beta)(\cos(\alpha)(s_z - s_{z,0}) - h_1) \\
M_{22} &= m_Y \\
M_{23} &= -m_z\sin(\alpha) \\
M_{24} &= -m_z(s_z - s_{z,0})\cos(\alpha) \\
M_{31} &= m_z\cos(\alpha)\sin(\beta) \\
M_{32} &= -m_z\sin(\alpha) \\
M_{33} &= m_Z \\
M_{41} &= m_z(s_{z,0} - s_z)\sin(\alpha)\sin(\beta) \\
M_{42} &= m_z(s_{z,0} - s_z)\cos(\alpha) \\
M_{44} &= m_z(s_z - s_{z,0})^2 + I_\alpha \\
M_{51} &= m_z\cos(\beta)(s_z\cos(\alpha) - h_1) \\
M_{55} &= -m_z(\cos(\alpha)(s_{z,0} - s_z) + h_1)^2 + I_\beta \\
M_{12} &= M_{21} = M_{25} = M_{34} = M_{35} = 0 \\
M_{43} &= M_{45} = M_{52} = M_{53} = M_{54} = 0,
\end{aligned}$$

$$C_{13} = -m_z(\cos(\alpha)\cos(\beta)\dot{\beta} - \sin(\alpha)\sin(\beta)\dot{\alpha})$$

$$
\begin{aligned}
C_{14} &= -m_z(s_z - s_{z,0})(\sin(\alpha)(\sin(\beta)\dot{s}_z + \dot{\beta}\cos(\beta) + \sin(\beta)\cos(\alpha)\dot{\alpha}) \\
&\quad + \sin(\beta)\cos(\alpha)\dot{\alpha}s_z) \\
C_{15} &= m_z((-\cos(\alpha)\sin(\beta)\dot{\beta} - \sin(\alpha)\cos(\beta)\dot{\alpha})(s_z - s_{z,0}) \\
&\quad + \cos(\alpha)\cos(\beta)\dot{s}_z + \dot{\beta}\sin(\beta)h_1) \\
C_{23} &= -\cos(\alpha)m_z\dot{\alpha} \\
C_{24} &= m_z(-\cos(\alpha)\dot{s}_z + \sin(\alpha)\dot{\alpha}(s_z - s_{z,0}) \\
C_{34} &= -m_z\dot{\alpha}(s_z - s_{z,0}) \\
C_{35} &= \dot{\beta}\cos(\alpha)m_z(-\cos(\alpha)(s_z - s_{z,0}) + h_1) \\
C_{43} &= m_z(s_z - s_{z,0})\dot{\alpha} \\
C_{44} &= m_z(s_z - s_{z,0})\dot{s}_z \\
C_{45} &= m_z\dot{\beta}\sin(\alpha)(\cos(\alpha)(s_z - s_{z,0})^2 - h_1(s_z - s_{z,0})) \\
C_{53} &= m_z\dot{\beta}\cos(\alpha)(\cos(\alpha)(s_z - s_{z,0}) - h_1) \\
C_{54} &= -m_z\dot{\beta}\sin(\alpha)(\cos(\alpha)(s_z - s_{z,0})^2 - h_1(s_z - s_{z,0})) \\
C_{55} &= m_z(-\cos(\alpha)\sin(\alpha)\dot{\alpha}(s_z - s_{z,0})^2 \\
&\quad + (-\cos(\alpha)^2\dot{s}_z + \sin(\alpha)\dot{\alpha}h_1)(s_z - s_{z,0}) - \cos(\alpha)h_1\dot{s}_z) \\
C_{11} &= C_{12} = C_{21} = C_{22} = C_{25} = 0 \\
C_{31} &= C_{32} = C_{33} = C_{41} = C_{42} = C_{51} = C_{52} = 0,
\end{aligned}
$$

$$
\begin{aligned}
g_1 &= g_2 = 0 \\
g_3 &= m_z\cos(\alpha)\cos(\beta)g \\
g_4 &= -m_z g\sin(\alpha)\cos(\beta)(s_z - s_{z,0}) \\
g_5 &= m_z g(-\cos(\alpha)\sin(\beta)(s_z - s_{z,0}) + \sin(\beta)h_1).
\end{aligned}
$$

The parameters of the lab-scale 3D gantry crane are obtained from [27] and can be found in Appendix A.1.

2.1.2 Differential flatness

For the definition of a flat system, see, e.g., [28], a vector

$$\mathbf{r} = \mathbf{\Phi}(\mathbf{z}, \mathbf{u}, \dot{\mathbf{u}}, ..., \mathbf{u}^{(n_u)}), \tag{2.14}$$

with n_u denoting the highest order of the derivative of $\mathbf{u}$, is a flat output of a system

$$\dot{\mathbf{z}} = \mathbf{f}(\mathbf{z}, \mathbf{u}) \tag{2.15}$$

if there exist two mapping functions $\mathbf{H_z}$, $\mathbf{H_u}$ and the index n_f such that the following equations hold

$$\mathbf{z} = \mathbf{H_z}(\mathbf{r}, \dot{\mathbf{r}}, ..., \mathbf{r}^{(n_f-1)}), \tag{2.16a}$$
$$\mathbf{u} = \mathbf{H_u}(\mathbf{r}, \dot{\mathbf{r}}, ..., \mathbf{r}^{(n_f)}) \ . \tag{2.16b}$$

Under the assumption that the payload is a point mass, the gantry crane is a well-known flat system and the CoM of the payload in (2.6)

$$\mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} = \begin{bmatrix} s_x + s_{x,0} + \sin(\beta)\cos(\alpha)(s_z - s_{z,0}) - \sin(\beta)h_1 \\ s_y + s_{y,0} - \sin(\alpha)(s_z - s_{z,0}) - b_1 \\ s_{z,max} + \cos(\beta)\cos(\alpha)(s_z - s_{z,0}) - \cos(\beta)h_1 \end{bmatrix} , \tag{2.17}$$

is a flat output, see, e.g., [29], [28].

The hoisting drum system generates the axial force along the rope $(\mathbf{F}^s_{\mathcal{B}})^{\mathrm{T}} = [0, 0, f_s]$ expressed in the body frame $\mathcal{B}$, illustrated in Fig. 2.4. Using Newton's law, the equations of motion of the CoM read as

$$m_z \ddot{\mathbf{r}} = \mathbf{F}^s_{\mathcal{W}} + m_z \mathbf{F}^g_{\mathcal{W}} \ , \tag{2.18}$$

where $\mathbf{F}^s_{\mathcal{W}} = \mathbf{R}^{\mathcal{B}}_{\mathcal{W}} \mathbf{F}^s_{\mathcal{B}}$, and $(\mathbf{F}^g_{\mathcal{W}})^{\mathrm{T}} = [0, 0, -g]$. Since only translations are required for the transformation of the world frame $\mathcal{W}$ into the trolley frame $\mathcal{C}$, $\mathbf{R}^{\mathcal{B}}_{\mathcal{W}} = \mathbf{R}^{\mathcal{B}}_{\mathcal{C}}$. Substituting (2.2) into (2.18), we obtain the formulation

$$m_z \ddot{r}_x = f_s \cos(\alpha)\sin(\beta) \tag{2.19a}$$
$$m_z \ddot{r}_y = -f_s \sin(\alpha) \tag{2.19b}$$
$$m_z \ddot{r}_z = f_s \cos(\alpha)\cos(\beta) - m_z g \ . \tag{2.19c}$$

From (2.19), the two sway angles α and β are computed as

$$\alpha = \arctan\left(\frac{-\ddot{r}_y}{\sqrt{\ddot{r}_x^2 + (\ddot{r}_z + g)^2}}\right) \tag{2.20a}$$

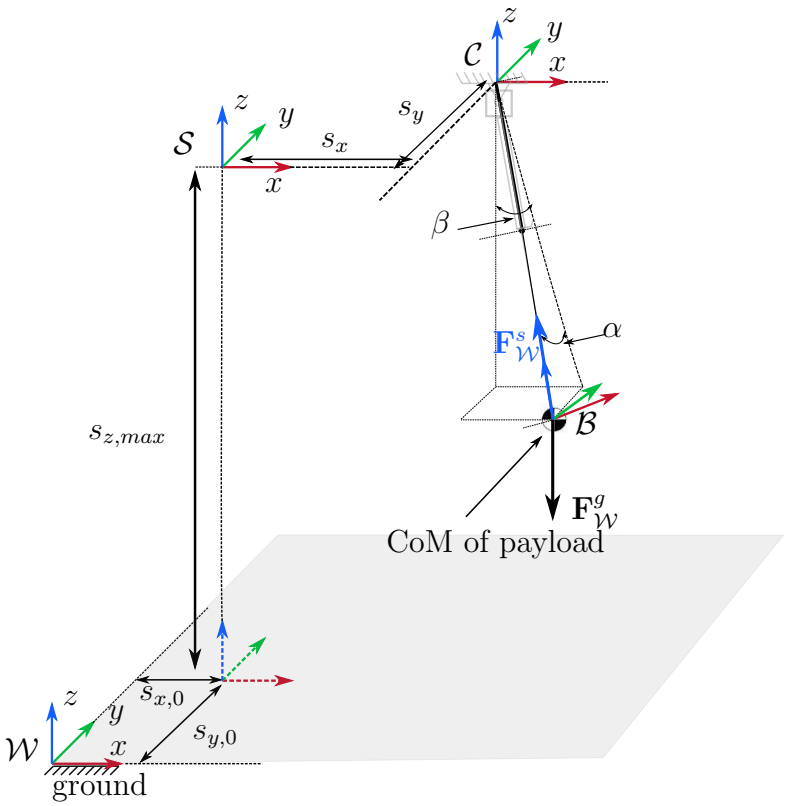


Figure 2.4: Free-body diagram of the 3D gantry crane assuming the payload to be a point mass.

$$\beta = \arctan\left(\frac{\ddot{r}_x}{\ddot{r}_z + g}\right). \tag{2.20b}$$

Using (2.17) and (2.20), the three system states s_x, s_y, and s_z are parameterized by the flat output $\mathbf{r}$ in the form

$$s_x = \left(r_x - s_{x,0}\right) + \left(s_{z,max} - r_z\right)\frac{\ddot{r}_x}{\ddot{r}_z + g} \tag{2.21a}$$

$$s_y = r_y + b_1 - s_{y,0} + \left(s_{z,max} - r_z\right)\frac{\ddot{r}_y}{\ddot{r}_z + g} - h_1\frac{\ddot{r}_y}{\sqrt{\left(\ddot{r}_z + g\right)^2 + \ddot{r}_x^2}} \tag{2.21b}$$

$$s_z = s_{z,0} + \left(r_z - s_{z,max}\right)\frac{\sqrt{\left(\ddot{r}_z + g\right)^2 + \ddot{r}_x^2 + \ddot{r}_y^2}}{\ddot{r}_z + g} + h_1\sqrt{\frac{\left(\ddot{r}_z + g\right)^2 + \ddot{r}_x^2 + \ddot{r}_y^2}{\left(\ddot{r}_z + g\right)^2 + \ddot{r}_x^2}}\ . \tag{2.21c}$$

By combining the two equations (2.21) and (2.20) with their first derivatives, analytical expressions for the state parameterization

$$\mathbf{z} = \mathbf{H}_\mathbf{z}(\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}, \mathbf{r}^{(3)})\ , \tag{2.22}$$

and the control input parameterization

$$\mathbf{u} = \mathbf{H}_\mathbf{u}(\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}, \mathbf{r}^{(3)}, \mathbf{r}^{(4)})\ , \tag{2.23}$$

according to (2.16) are obtained.

2.2 Collaborative 7-axis robot with a spherical pendulum

The considered system consists of the 7-axis KUKA LBR iiwa 14 R820 robot and a custom-built spherical pendulum mounted on the flange of the robot, see Fig. 2.5. The spherical pendulum has two crossing axes as depicted in Fig. 2.6. This allows to move the tip of the pendulum on a sphere. Each axis of the spherical pendulum is equipped with a magnetic encoder for measuring the angle and angular velocity. As shown in Fig. 2.6, EtherCAT terminals are employed to read out the two encoders attached to the mounting housing.

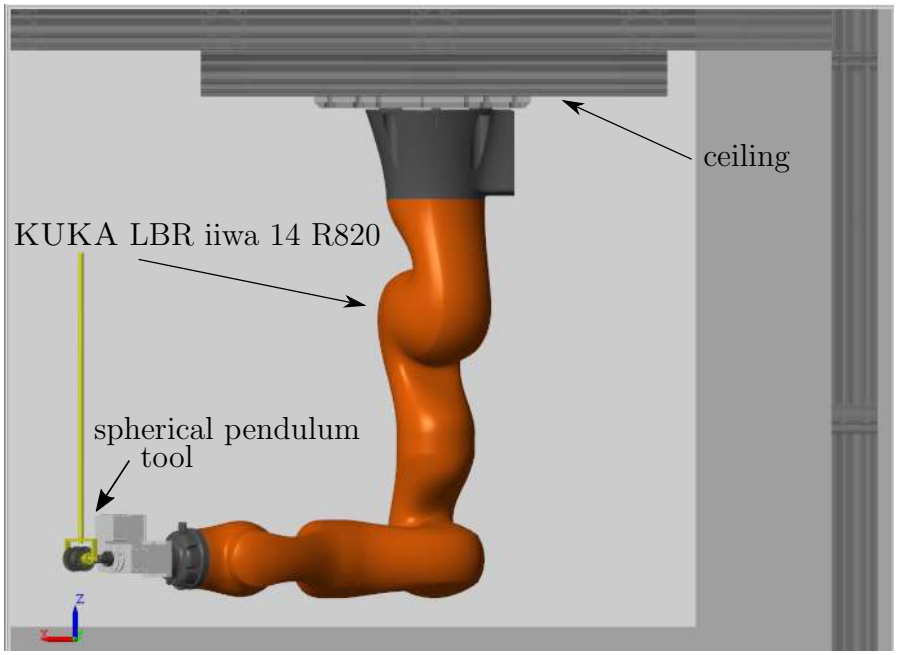


Figure 2.5: The considered 7-axis robot with a spherical pendulum.

2.2.1 Forward kinematics

The forward kinematics (FK) determines the pose of the end effector from joint angles. This is done by using the homogeneous transformation

$$\mathbf{H}_{\mathcal{X}}^{\mathcal{Y}} = \begin{bmatrix} \mathbf{R}_{\mathcal{X}}^{\mathcal{Y}} & \mathbf{d}_{\mathcal{X}}^{\mathcal{Y}} \\ \mathbf{0} & 1 \end{bmatrix} , \tag{2.24}$$

where $\mathbf{R}_{\mathcal{X}}^{\mathcal{Y}}$ and $\mathbf{d}_{\mathcal{X}}^{\mathcal{Y}}$ are the rotation matrix and the distance vector that rotate and translate the system from the coordinate system $\mathcal{X}$ to the coordinate system $\mathcal{Y}$, respectively. In this work, a pure rotation by the angle φ around the local axis $i \in \{x, y, z\}$ is denoted by $\mathbf{H}_{\mathrm{R}i,\varphi}$ and a pure translation in the direction of the local axis i by a length d is denoted by $\mathbf{H}_{\mathrm{T}i,d}$. Moreover, the transformation from a coordinate system $\mathcal{Z}$ to a coordinate system $\mathcal{X}$ by an intermediate coordinate system $\mathcal{Y}$ takes the form

$$\mathbf{H}_{\mathcal{X}}^{\mathcal{Z}} = \mathbf{H}_{\mathcal{X}}^{\mathcal{Y}} \mathbf{H}_{\mathcal{Y}}^{\mathcal{Z}} . \tag{2.25}$$

Forward kinematics of serial manipulators is derived using a set of successive ho-

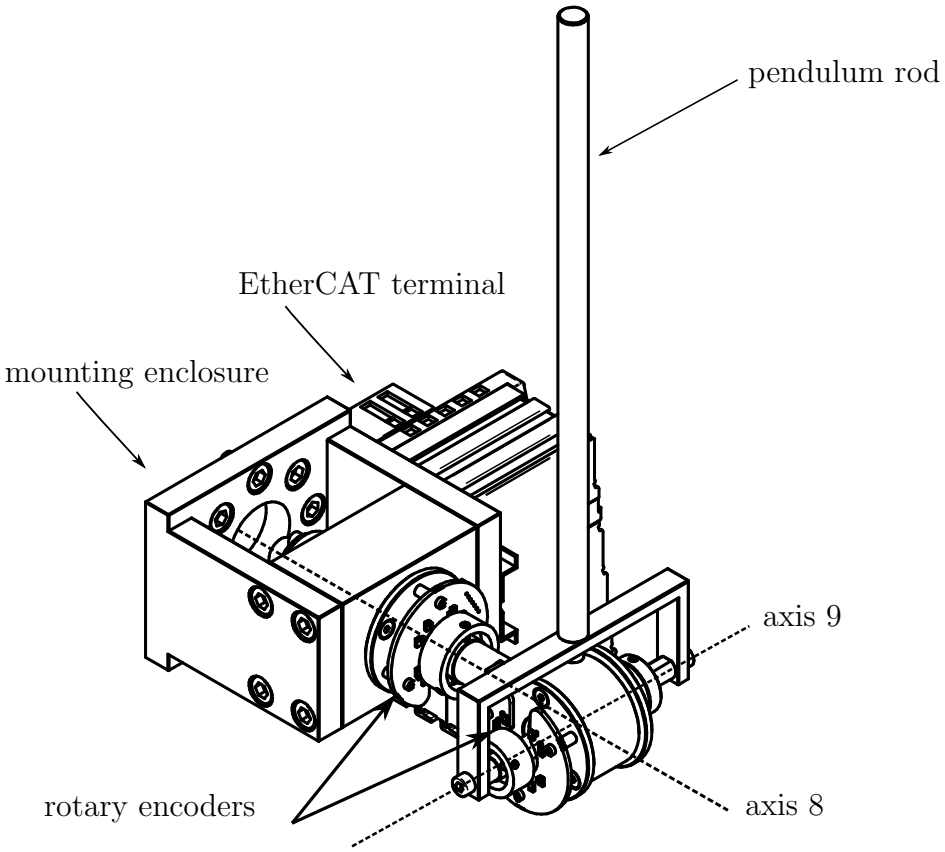


Figure 2.6: The spherical inverted pendulum [30].

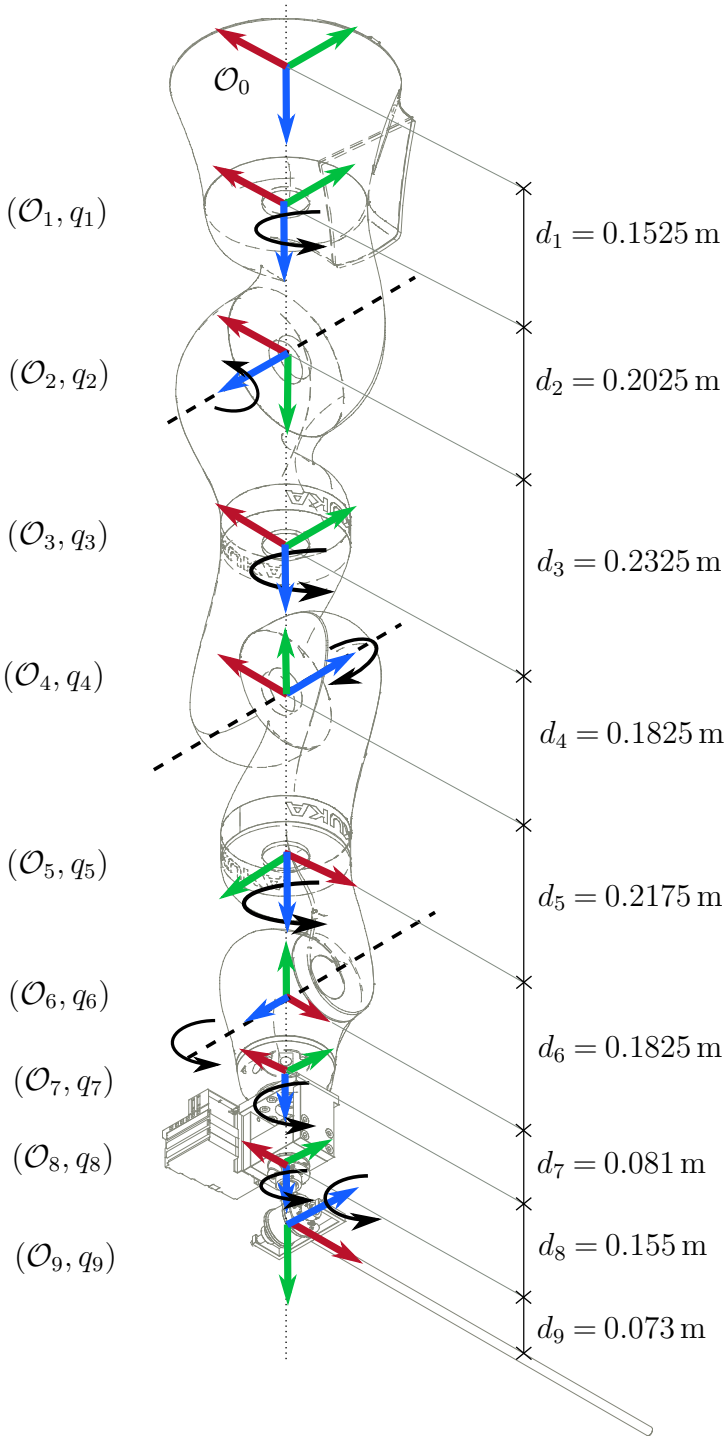


Figure 2.7: Schematic drawing of the ceiling-mounted robot KUKA LBR iiwa in the hanging position and its corresponding coordinate frames $\mathcal{O}_i$, $i = 1, \ldots, 9$. The x-, y-, and z-axis of each coordinate frame are drawn as red, green, and blue arrows, respectively.

Table 2.1: Coordinate transformation of the system

Frame $\mathcal{O}_n$	**Frame $\mathcal{O}_m$**	**Transformation matrix $\mathbf{H}_{\mathcal{O}_n}^{\mathcal{O}_m}$**
$\mathcal{O}_0$	$\mathcal{O}_1$	$\mathbf{H}_{\mathrm{T}z,d_1}\mathbf{H}_{\mathrm{R}z,q_1}$
$\mathcal{O}_1$	$\mathcal{O}_2$	$\mathbf{H}_{\mathrm{T}z,d_2}\mathbf{H}_{\mathrm{R}x,-\pi/2}\mathbf{H}_{\mathrm{R}z,q_2}$
$\mathcal{O}_2$	$\mathcal{O}_3$	$\mathbf{H}_{\mathrm{T}y,d_3}\mathbf{H}_{\mathrm{R}x,\pi/2}\mathbf{H}_{\mathrm{R}z,q_3}$
$\mathcal{O}_3$	$\mathcal{O}_4$	$\mathbf{H}_{\mathrm{T}z,d_4}\mathbf{H}_{\mathrm{R}x,\pi/2}\mathbf{H}_{\mathrm{R}z,q_4}$
$\mathcal{O}_4$	$\mathcal{O}_5$	$\mathbf{H}_{\mathrm{T}y,d_5}\mathbf{H}_{\mathrm{R}z,\pi}\mathbf{H}_{\mathrm{R}x,\pi/2}\mathbf{H}_{\mathrm{R}z,q_5}$
$\mathcal{O}_5$	$\mathcal{O}_6$	$\mathbf{H}_{\mathrm{T}z,d_6}\mathbf{H}_{\mathrm{R}x,\pi/2}\mathbf{H}_{\mathrm{R}z,q_6}$
$\mathcal{O}_6$	$\mathcal{O}_7$	$\mathbf{H}_{\mathrm{T}y,d_7}\mathbf{H}_{\mathrm{R}z,\pi}\mathbf{H}_{\mathrm{R}x,\pi/2}\mathbf{H}_{\mathrm{R}z,q_7}$
$\mathcal{O}_7$	$\mathcal{O}_8$	$\mathbf{H}_{\mathrm{T}y,d_8}\mathbf{H}_{\mathrm{R}z,q_8}$
$\mathcal{O}_8$	$\mathcal{O}_9$	$\mathbf{H}_{\mathrm{T}y,d_9}\mathbf{H}_{\mathrm{R}x,\pi/2}\mathbf{H}_{\mathrm{R}z,\pi}\mathbf{H}_{\mathrm{R}z,q_9}$

mogeneous transformations $\mathbf{H}_{\mathcal{O}_{i-1}}^{\mathcal{O}_i}$. Note that the axis z_i of the coordinate frame $\mathcal{O}_i$ is chosen to be aligned with the corresponding axis of the joint i, see Fig. 2.7. The complete mechanical system of the 7-axis robot and the spherical pendulum consists of nine DoF. The nine coordinate frames $\mathcal{O}_i$, $i = 1, \ldots, 9$, of the complete system are depicted in Fig. 2.7, where the red, green, and blue arrows represent the x-, y-, and z-axis, respectively. The spherical pendulum has two intersecting axes, i.e., the z-axis of the frame $\mathcal{O}_8$ and the z-axis of the frame $\mathcal{O}_9$, which can be rotated freely. The system is modeled as a rigid-body system in the state-space form using the generalized coordinates $\mathbf{q}^{\mathrm{T}} = [q_1, q_2, \ldots, q_9]$, see Fig. 2.7, which are the rotation angles q_i around the z-axes (blue arrows) of the coordinate frames $\mathcal{O}_i$, $i = 1, \ldots, 9$. The kinematic relation between two consecutive coordinate frames $\mathcal{O}_{i-1}$ and $\mathcal{O}_i$ is given in Table 2.1. The homogeneous transformation of joint i^{th}, $i = 1, ..., 9$ w.r.t. the world coordinate frame $\mathcal{O}_0$ can be computed as

$$\mathbf{H}_{\mathcal{O}_0}^{\mathcal{O}_i} = \mathbf{H}_{\mathcal{O}_0}^{\mathcal{O}_1}\mathbf{H}_{\mathcal{O}_1}^{\mathcal{O}_2}...\mathbf{H}_{\mathcal{O}_{i-1}}^{\mathcal{O}_i} = \begin{bmatrix} \mathbf{R}_i & \mathbf{d}_i \\ \mathbf{0} & 1 \end{bmatrix} . \tag{2.26}$$

Thus, the forward kinematics of the i^{th} joint is given by

$$\mathbf{y}_i = \begin{bmatrix} \mathbf{y}_{i,t} \\ \mathbf{y}_{i,r} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_i \\ \mathbf{\Phi}(\mathbf{R}_i) \end{bmatrix}, \; i = 1, ..., 9, \tag{2.27}$$

where $\Phi(\mathbf{R}_i)$ is the minimal representation of the orientation which can be computed from the rotation matrix $\mathbf{R}_i$. In the following, ZYX-Euler angle representa-

tion

$$\mathbf{\Phi} = \begin{bmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \end{bmatrix} = \mathbf{\Phi}_{ZYX}(\mathbf{R}_i) \tag{2.28}$$

is used, where $\mathbf{R}_i$ represents a sequence of three pure rotations in the form

$$\mathbf{R}_i(\mathbf{\Phi}) = \mathbf{R}_{z,\Phi_1}\mathbf{R}_{y,\Phi_2}\mathbf{R}_{x,\Phi_3}. \tag{2.29}$$

2.2.2 Dynamics

Let $\mathbf{y}_{i,t,c}^{\mathcal{O}_0}$ denotes the vector from the origin of the world frame $\mathcal{O}_0$ to the center of mass of joint $i = 1, ..., 9$, expressed in the coordinate frame $\mathcal{O}_0$. Then, the translational velocity is $\mathbf{v}_{i,c}$ of the i-th link is given by

$$\mathbf{v}_{i,c} = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y}_{i,t,c}^{\mathcal{O}_0},\ i = 1, ..., 9\ , \tag{2.30}$$

and the angular velocity $\boldsymbol{\omega}_i^{\mathrm{T}} = [\omega_{i,x}, \omega_{i,y}, \omega_{i,z}]$ results from the skew-symmetric operator

$$\mathbf{S}(\boldsymbol{\omega}_i) = \frac{\mathrm{d}}{\mathrm{d}t}\mathbf{R}_i(\mathbf{R}_i^{\mathrm{T}}) = \begin{bmatrix} 0 & -\omega_{i,z} & \omega_{i,y} \\ \omega_{i,z} & 0 & -\omega_{i,x} \\ -\omega_{i,y} & \omega_{i,x} & 0 \end{bmatrix}\ ,\ i = 1, ..., 9\ . \tag{2.31}$$

Thus, from (2.30) and (2.31) we get the relation between the velocities $\dot{\mathbf{q}}$ in the joint space and $\mathbf{v}_{i,c}^{\mathcal{O}_0}$ and $\boldsymbol{\omega}_i$ in the task space in the form

$$\begin{bmatrix} \mathbf{v}_{i,c} \\ \boldsymbol{\omega}_i \end{bmatrix} = \mathbf{J}_i^{\mathcal{O}_0}(\mathbf{q})\dot{\mathbf{q}}\ ,\ i = 1, ..., 9, \tag{2.32}$$

with the geometric Jacobian

$$\mathbf{J}_i^{\mathcal{O}_0}(\mathbf{q}) = \begin{bmatrix} \mathbf{J}_{i,v}^{\mathcal{O}_0} \\ \mathbf{J}_{i,\omega}^{\mathcal{O}_0} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial}{\partial \mathbf{q}}\mathbf{y}_{i,t,c}^{\mathcal{O}_0} \\ \dfrac{\partial}{\partial \dot{\mathbf{q}}}\boldsymbol{\omega}_i \end{bmatrix}\ ,\ i = 1, ..., 9\ . \tag{2.33}$$

The Lagrange formalism applied to the Lagrange function $L = T - V$, with the kinetic energy

$$T = \sum_{i=1}^{9} \frac{1}{2} m_i \left(\mathbf{v}_{i,c}\right)^{\mathrm{T}} \mathbf{v}_{i,c} + \frac{1}{2} \left(\boldsymbol{\omega}_i\right)^{\mathrm{T}} \mathbf{R}_i \mathbf{I}_i \mathbf{R}_i^{\mathrm{T}} \boldsymbol{\omega}_i, \tag{2.34}$$

where m_i denotes the mass and I_i the inertia matrix of the i-th link, and the potential energy

$$V = \sum_{i=1}^{9} \begin{bmatrix} 0 & 0 & -g \end{bmatrix} \mathbf{y}_{i,t,c}^{\mathcal{O}_0} m_i \,, \tag{2.35}$$

leads to the equations of motion written in state-space form

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \begin{bmatrix} \boldsymbol{\tau}_A & 0 & 0 \end{bmatrix}^{\mathrm{T}} \,. \tag{2.36}$$

Here $\mathbf{M}(\mathbf{q})$ denotes the symmetric and positive definite mass matrix, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ is the Coriolis matrix, $\mathbf{g}(\mathbf{q})$ is the force vector related to the potential energy, and $\boldsymbol{\tau}_A \in \mathbb{R}^{7\times 1}$ are the motor torque inputs of the robot. Friction is neglected in this model. The parameters of the presented 7-axis robot with the spherical pendulum are obtained from [30] and can be found in Appendix A.2.

3

Sampling-based trajectory planning for a 3D gantry crane

In this section, a sampling-based trajectory planning algorithm for a lab-scale 3D gantry crane is presented, which takes into account the system state and input constraints in an environment with static obstacles. The mathematical model including the equations of motion and the differential flatness property of the 3D gantry crane are presented in Section 2.1.

The focus of this section is on the development of a fast trajectory planning algorithm, where intermediate results can be stored and reused for further tasks, e.g. replanning. The proposed approach is based on the informed optimal rapidly-exploring random tree (informed RRT*) algorithm, which is used to build trajectory trees. The proposed motion planning algorithm takes advantage of the differential flatness property and incorporates a local planner with a linear quadratic minimum-time (LQTM) approach. Therefore, dynamic properties such as trajectory smoothness are directly considered in the proposed algorithm. Moreover, the proposed algorithm is able to eliminate points in the trajectory tree that do not contribute to finding better solutions by using the branch-and-bound method to perform the pruning process in the trajectory tree. This helps to reduce mem-

ory consumption and computational complexity. Simulation results for a validated mathematical model of a lab-scale 3D gantry crane show the feasibility of the proposed approach.

First, Section 3.1 reviews the existing literature. Section 3.2 presents the proposed trajectory replanning algorithm, which consists of the informed RRT* algorithm and the LQMT local planner. Simulation results are presented in Section 3.3 and Section 3.4, respectively. Large parts of this section are identical to the author's publication [22].

3.1 Introduction

Motion planning is an important task in robotics that aims to calculate collision-free and dynamically feasible paths connecting an initial configuration to a target configuration. In motion planning tasks for unloading containers on ships, trucks, and in warehouses, repeated movements from and to adjacent positions have to be planned. In these applications, the motion planning tasks have to be repeatedly solved for nearly similar starting and target configurations. This leads to a waste of computational resources if every motion planning job is computed from scratch.

There are a number of different concepts in the literature, which can be essentially divided into sampling-based approaches and optimization-based approaches. This section focuses on the sampling-based approaches. There are a number of different concepts for sampling-based motion planning algorithms in the literature, which can be essentially divided into two groups. In the first group, the motion planning problem is commonly solved by discretizing the continuous state space into grids, i.e., graph-based search [31]. Graph-based search methods, also known as deterministic motion planning methods, see, e.g., A* [32], D* [33], are resolution-complete methods that guarantee optimal resolution. The search procedure is mainly guided by heuristic minimization of a cost function from the current sampled state to the target state. There are variant methods in this group such as the Life Long Planning algorithm [34], Replanning D* [35, 36], and the Anytime algorithm ARA*, see, [37], in which their solution can be computed and refined in a reasonable computation time depending on the chosen grid resolution. However, the computational costs of graph-based search are increasing with the used resolution of the discretization. This circumstance is well known as the "curse of dimensionality,"

see, [33]. Despite this disadvantage, graph-based searches were successfully applied to several types of planning tasks, e.g., manipulation planning, see, [38], [39], and kinodynamic planning, see, [40].

Different from the first group, methods in the second group randomly sample the state space, i.e., sampling-based search [41]. This method, also known as probabilistic motion planning, builds the motion tree by randomly sampling the system state. Sampling-based methods are globally probabilistically optimal, see, e.g., [42, 41], in which the probability of finding the global optimal solution approaches one when the number of iterations goes to infinity. The probabilistic roadmap (PRM) [43], i.e., a variant of the sampling-based methods, randomly samples collision-free states in the working space. These sampled states are connected to the respective neighboring states using a local motion planner to set up the motion planning graph. The main advantage of the PRM, especially useful for high-dimensional configuration space problems, is that only collision-free states are collected in the data tree. This leads to fewer states which need to be evaluated in the search space and hence to an improvement of the computation speed.

Another variant of the sampling-based methods, i.e., the optimal rapidly-exploring random tree (RRT*) algorithm, finds the path from the initial state to each state in the planning tree by incrementally rewiring the tree of sampled states, see, e.g., [41]. Rewiring helps to reconstruct the tree by not only adding new states to the tree, but also considering them as surrogate nodes for existing states in the data tree. Further improvements of RRT* have been proposed in the literature, such as RRT*-SMART, see, e.g., [44], which uses additional heuristics to speed up the convergence rate. Additionally, Karaman et al. developed and adapted the RRT* for online motion planning. Therein, the robot moves along the initial path while the algorithm is still refining the part that the robot has not yet reached. Recently, Gammel et al. [45] introduces the informed RRT*. Instead of sampling the system state in the entire workspace, this algorithm randomly samples the system state in the subspace created by the first solution. Note that there is no difference between informed RRT* and RRT* until the first solution is found. When a first collision-free path is found, only feasible samples from the subset of states, i.e., the informed set, are taken into account by the informed RRT* algorithm. This helps to narrow the search space and increase the probability of obtaining a better solution in a given time. Later, Strub et al. [46] introduced the advanced batch informed RRT* (ABIT*) in order to unify sampling-based and graph-based search

without sacrificing the advantages of either method. This algorithm discretizes the continuous search space with an edge-implicit Random Geometric Graph (RGG), see [41]. This can improve the computation time by applying the informed RRT* in parallel for each space region created by the RGG method. Since a local planner is applied iteratively when a new state is sampled, the overall performance of a sampling-based approach depends on the computational speed of that local planner. Typically, a local planner computes the cost for moving the system between two sampled states. To account for the dynamic constraints of the system, a two-point boundary value problem (TP-BVP) solver, see, e.g., [47, 48], is often utilized as local planner. Webb et al. [49] employed the linear quadratic minimum-time (LQMT) analytical solution [50] as local planner for RRT*. Since the LQMT solver provides an analytical solution for the near-time optimal trajectory connecting any two system states, the computation time of this local planner is faster than numerical solvers, see, e.g., [51, 52]. However, the analytical solution of the LQMT is only available for linear systems.

In this section, a trajectory planning algorithm based on the informed optimal rapidly-exploring random trees (RRT*) [45] is introduced, which is capable of fast replanning if target configurations are changed. Due to the high dimension of the 3D gantry crane system, i.e., 5 degrees of freedom (DoF) corresponding to a 10-dimensional system state, the sampling-based motion planning approach is favorable for this system compared to the grid-based motion planning approach.

The main objective of this section is to develop a fast sampling-based trajectory planning algorithm that drives the payload of the lab-scale 3D gantry crane from a given starting state to a given target state by avoiding obstacles and respecting the dynamic constraints of the system states at the same time. To do so, the informed RRT* algorithm is extended to meet the requirements of the repeated motion (re)planning from and to adjacent configurations.

The sampling-based trajectory planning proposed in this section, referred to as flat-informed RRT*, includes four modifications compared to RRT* [12].

- First, the informed subset of randomly selected states is controlled using a heuristic function to prevent the selection of new states that do not contribute to a better solution.
- Second, the branch-and-bound method is used to prune the parts of the

trajectory tree that do not provide a better solution compared to the current cost. This helps to reduce both memory requirements and computation time.

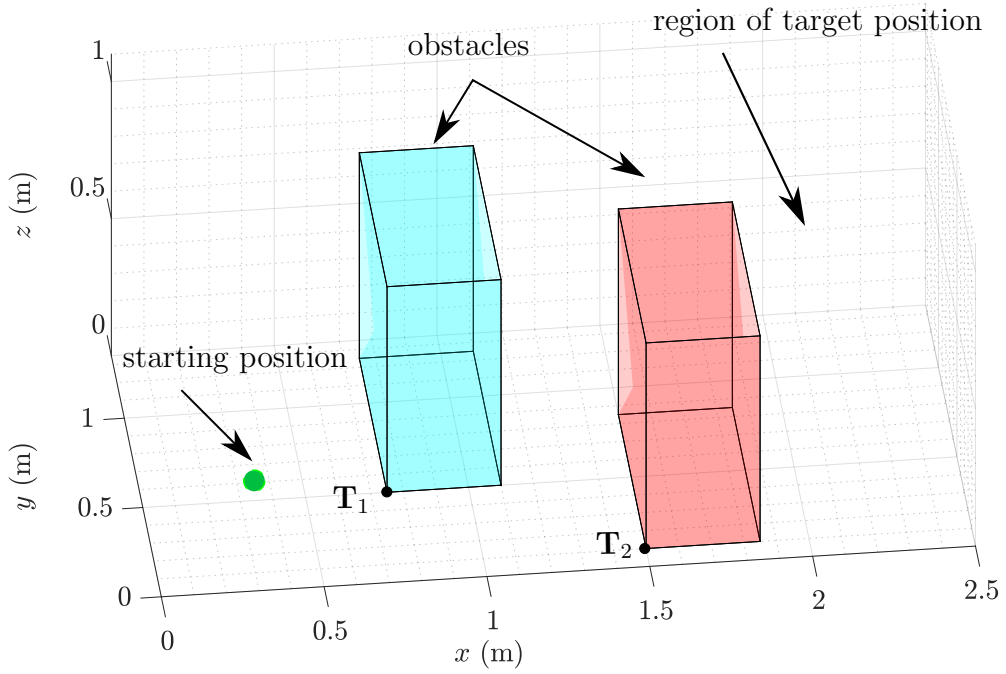
- Third, taking advantage of parallel data processing, multiple trees are generated and concatenated to enrich the motion trajectory tree. When such a collision-free and dynamically feasible trajectory tree is available, the proposed algorithm quickly generates a feasible trajectory when the position of the target state changes.
- In order to achieve a fast computation time, the local planner uses the analytical solution of LQMT [50]. Since the dynamic system of the 3D gantry crane is nonlinear, the differential flatness property of the 3D gantry crane is exploited, see, e.g., [53], [29], [54], where all system states and inputs can be parameterized by the flat output and its time derivatives.

3.2 Flat-informed RRT* trajectory planning

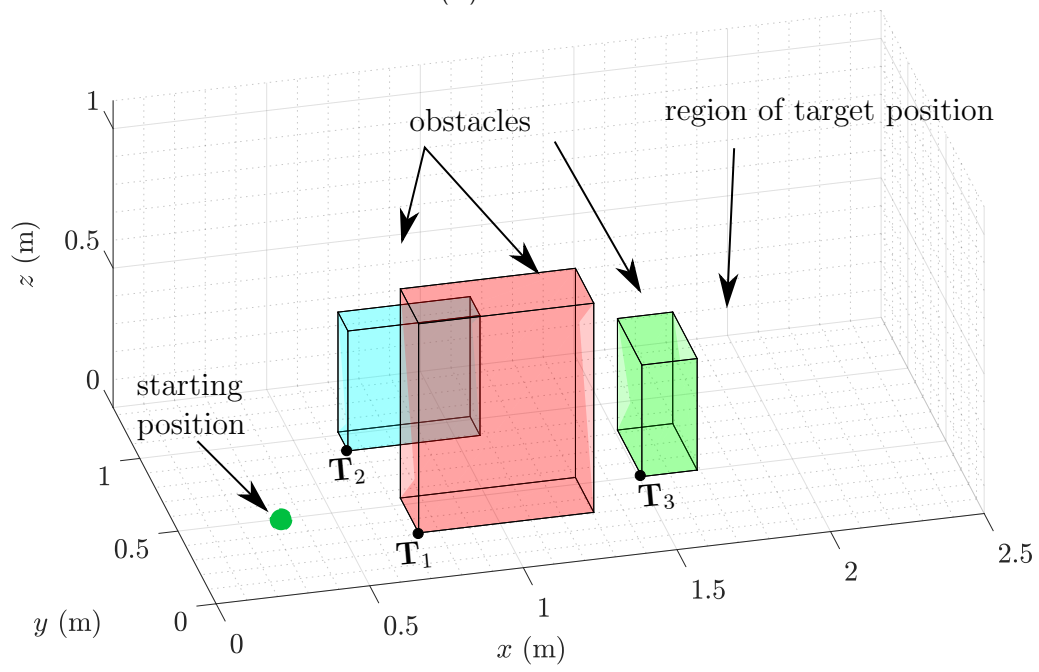
First, an overview of the trajectory planning scenario is given. Then, the flat-informed RRT* trajectory planning algorithm with the LQMT local planner [50] is presented. To this end, the mathematical modeling of the 3D gantry crane, presented in Section 2.1, is utilized.

3.2.1 Overview of the trajectory planning scenario

In this section, the following scenario is considered. The lab-scale 3D gantry crane in Fig. 2.1 is used to move goods or material from an initial state to a target state in a static environment with known obstacles. The equations of motion of the gantry crane in (2.13) and the differential flatness property in (2.22)-(2.23) are utilized. The task is to plan a dynamically feasible and obstacle-free trajectory. Moreover, the proposed algorithm allows a very fast replanning when the target state changes. In Fig. 3.1, the two considered workspaces with different obstacle structures are shown. In both scenarios, the region of possible of target states is behind the obstacles in x-direction, while the initial state is $\mathbf{z}_S^{\mathrm{T}} = [\mathbf{q}_S, \mathbf{0}]$.



(a) Scenario 1



(b) Scenario 2

Figure 3.1: Visualization of the 3D gantry crane workspace in two simulation scenarios with different obstacle structures.

3.2.2 Local planner

The RRT* algorithm typically selects a random state and adds it to a set of states, also called a tree $\mathcal{T}$. It then tries to connect this random state to other states in the set using a local planner. In the simple case, such as holonomic planning, the local planner often performs only a straight-line connection between two system states. Since dynamic constraints of the 3D gantry crane have to be considered, the local planner is generally used to connect any two system states $\mathbf{z}_A$ and $\mathbf{z}_B$ subject to the nonlinear equations of motion in (2.13). Iterative solvers, e.g. [51], are mostly employed as local planners. The computation time of iterative solvers often constitutes a bottleneck, since the local planner is repeatedly processed to build the trajectory tree.

To overcome this bottleneck, the differential flatness property of the 3D gantry crane can be exploited. Since the system states $\mathbf{z}$ and the control inputs $\mathbf{u}$ are parameterized by flat outputs $\mathbf{r}(t)$ and their time derivatives according to (2.22) and (2.23), the proposed flat-informed sampling-based trajectory planning algorithm directly samples the flat outputs $\mathbf{x}_l^{\mathrm{T}} = [\mathbf{r}, \dot{\mathbf{r}}, \ddot{\mathbf{r}}, \mathbf{r}^{(3)}]$. The linear time-invariant system of the local planner reads as

$$\dot{\mathbf{x}}_l = \mathbf{A}\mathbf{x}_l + \mathbf{B}\mathbf{u}_l \ , \tag{3.1}$$

with the system matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{I}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \mathbf{I}_3 \end{bmatrix} .$$

Additionally, $\mathbf{u}_l = \mathbf{r}^{(4)}$ is the input and $\mathbf{I}_n$ is the identity matrix of size n.

The local planner is used to compute the optimal cost $c^*(\mathbf{x}_{l,0}, \mathbf{x}_{l,1})$ to drive the system (3.1) from an initial state $\mathbf{x}_l(t_0) = \mathbf{x}_{l,0}$ to a target state $\mathbf{x}_l(t_1) = \mathbf{x}_{l,1}$. This optimal cost $c^*(\mathbf{x}_{l,0}, \mathbf{x}_{l,1})$ can be analytically obtained by solving the LQMT [50]

problem in the form

$$\min_{\mathbf{x}_l,\mathbf{u}_l,\Delta t} c = \Delta t + \frac{1}{2}\int_{t_0}^{t_1} \mathbf{u}_l^{\mathrm{T}}\mathbf{R}\mathbf{u}_l \mathrm{d}t \tag{3.2a}$$

$$\text{s.t.} \quad \dot{\mathbf{x}}_l = \mathbf{A}\mathbf{x}_l + \mathbf{B}\mathbf{u}_l, \quad t_0 \leq t \leq t_1 \tag{3.2b}$$

$$\mathbf{x}_l(t_0) = \mathbf{x}_{l,0} \ , \ \mathbf{x}_l(t_1) = \mathbf{x}_{l,1} \ , \tag{3.2c}$$

where $\Delta t = t_1 - t_0$ is the travel time, (3.2b) corresponds to (3.1), and $\mathbf{R}$ is a user-defined positive definite matrix that weighs the trade-off between the trajectory smoothness and the travel time Δt. The Hamiltonian $\mathbb{H}$ of the optimization problem (3.2) takes the form

$$\mathbb{H} = 1 + \frac{1}{2}\mathbf{u}_l^{\mathrm{T}}\mathbf{R}\mathbf{u}_l + \boldsymbol{\lambda}_l^{\mathrm{T}}(\mathbf{A}\mathbf{x}_l + \mathbf{B}\mathbf{u}_l) \, , \tag{3.3}$$

with the adjoint state $\boldsymbol{\lambda}_l(t)$ is the slack variable. The necessary first-order optimality conditions for the optimality are given by

$$\dot{\mathbf{x}}_l^* = \left(\frac{\partial \mathbb{H}}{\partial \boldsymbol{\lambda}_l}\right)^{\mathrm{T}} = \mathbf{A}\mathbf{x}_l^* + \mathbf{B}\mathbf{u}_l^* \ , \ \mathbf{x}_l^*(t_0) = \mathbf{x}_{l,0}^* \ , \tag{3.4a}$$

$$\dot{\boldsymbol{\lambda}}_l^* = -\left(\frac{\partial \mathbb{H}}{\partial \mathbf{x}_l}\right)^{\mathrm{T}} = -\mathbf{A}^{\mathrm{T}}\boldsymbol{\lambda}_l^* \ , \ \boldsymbol{\lambda}_l^*(t_1) = \boldsymbol{\lambda}_{l,1}^* \ , \tag{3.4b}$$

$$\mathbf{0} = \left(\frac{\partial \mathbb{H}}{\partial \mathbf{u}_l}\right)^{\mathrm{T}} = \mathbf{R}\mathbf{u}_l^* + \mathbf{B}^{\mathrm{T}}\boldsymbol{\lambda}_l^* \ . \tag{3.4c}$$

Using (3.4b) and (3.4c), the optimal control input $\mathbf{u}_l^*$ and the optimal adjoint state $\boldsymbol{\lambda}_l^*$ as

$$\mathbf{u}_l^* = -\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}}\boldsymbol{\lambda}_l^*(t) \tag{3.5a}$$

$$\boldsymbol{\lambda}_l^*(t) = \exp\left(\mathbf{A}^{\mathrm{T}}(t_1 - t)\right)\boldsymbol{\lambda}_{l,1}^* \ . \tag{3.5b}$$

Substituting (3.5a) and (3.5b) into (3.4a), the optimal state trajectory $\mathbf{x}_l^*(t)$ is calculated in the form

$$\mathbf{x}_l^*(t) = \exp\left(\mathbf{A}(t - t_0)\right)\mathbf{x}_{l,0} - \mathbf{G}(t_0, t)\boldsymbol{\lambda}_{l,1}^*, \tag{3.6}$$

with

$$\mathbf{G}(t_0, t) = \int_{t_0}^{t} \exp\left(\mathbf{A}(t-\tau)\right)\mathbf{B}\mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}} \exp\left(\mathbf{A}^{\mathrm{T}}(t_1-\tau)\right)\mathrm{d}\tau \tag{3.7}$$

as the continuous reachability Gramian [50]. Evaluating (3.6) at $t = t_1$, we get $\boldsymbol{\lambda}_{l,1}^*$ in the form

$$\boldsymbol{\lambda}_{l,1}^* = -\mathbf{G}^{-1}(t_0, t_1)\mathbf{d}_{\Delta t}, \tag{3.8}$$

with $\mathbf{d}_{\Delta t} = \mathbf{x}_{l,1} - \exp\left(\mathbf{A}\Delta t\right)\mathbf{x}_{l,0}$. Combining (3.4c), (3.5b), and (3.8), the optimal input $\mathbf{u}_l^*$ is obtained. Therefore,

$$\mathbf{u}_l^*(t) = \mathbf{R}^{-1}\mathbf{B}^{\mathrm{T}} \exp\left(\mathbf{A}^{\mathrm{T}}(t_1-t)\right)\mathbf{G}^{-1}(t_0, t_1)\mathbf{d}_{\Delta t} \ . \tag{3.9}$$

Substituting the optimal input $\mathbf{u}_l^*$, given in (3.9), into (3.2a), the cost function (3.2a) is expressed as

$$c(\mathbf{x}_{l,0}, \mathbf{x}_{l,1}) = \Delta t + \frac{1}{2}\mathbf{d}_{\Delta t}^{\mathrm{T}}\mathbf{G}^{-1}(t_0, t_1)\mathbf{d}_{\Delta t} \ . \tag{3.10}$$

Since the initial time t_0 is known, the function $c(\mathbf{x}_{l,0}, \mathbf{x}_{l,1})$ in (3.10) depends only on the final time t_1. Note that the analytical expression of the reachability Gramian (3.7) can be computed since the system matrix $\mathbf{A}$ is nilpotent, see (3.1). Thereby, $c(\mathbf{x}_{l,0}, \mathbf{x}_{l,1})$ is analytically expressed as a polynomial of the final time t_1. Thus, the optimal value t_1^* can be determined by finding the roots of the first order derivative of (3.10) with respect to t_1. Subsequently, the corresponding optimal state $\mathbf{x}_l^*$ and the control input $\mathbf{u}_l^*$ are computed using (3.5a), (3.6), and (3.8). To check the feasibility of the optimal flat ouput trajectory $(\mathbf{x}_l^*, \mathbf{u}_l^*)$ with respect to the system state limits and the control input limits, (2.22) and (2.23) are employed to compute the corresponding system state $\mathbf{z}^*$ and the control input $\mathbf{u}^*$.

Finally, the LQMT local planner returns the optimal cost $c^*(\mathbf{x}_{l,0}, \mathbf{x}_{l,1})$ and a feasibility flag c_{flag} which indicates if the computed trajectory $(\mathbf{z}^*, \mathbf{u}^*)$ is collision free and dynamically feasible.

3.2.3 Flat-Informed RRT* trajectory planning algorithm

In this subsection, the flat-informed RRT*, taking into account the advantages of the LQMT local planner and the informed RRT*, is presented in detail and is summarized in Alg. 1. The starting position and the target position are denoted

as $\mathbf{r}_S$ and $\mathbf{r}_T$, respectively, which can be obtained from the initial state $\mathbf{z}_S$ and the target state $\mathbf{z}_T$ by (2.6), see also (2.13). Note that the starting state and the target state of the flat system (3.2) are $\mathbf{x}_{l,S}^{\mathrm{T}} = [\mathbf{r}_S, \mathbf{0}, \mathbf{0}, \mathbf{0}]$ and $\mathbf{x}_{l,T}^{\mathrm{T}} = [\mathbf{r}_T, \mathbf{0}, \mathbf{0}, \mathbf{0}]$, respectively.

The proposed algorithm builds a tree $\mathcal{T} = \{\mathcal{V}, \mathcal{P}, \mathcal{C}, \mathcal{L}\}$ consisting of the set of states $\mathcal{V}$, the set of parent states $\mathcal{P}$, the set of child states $\mathcal{C}$, and the set of binary masks $\mathcal{L}$. The cost between any two states in the tree $\mathcal{T}$ is computed using the local planner LQMT. The set $\mathcal{V}$ contains all states of the tree. A state $\tilde{\mathbf{x}}_{l,parent} = \mathcal{P}(\tilde{\mathbf{x}}_l)$ is considered as a parent state of $\tilde{\mathbf{x}}_l$ if the cost $c^*(\tilde{\mathbf{x}}_{l,parent}, \tilde{\mathbf{x}}_l)$ is the smallest compared to costs $c^*(\mathbf{x}_l, \tilde{\mathbf{x}}_l)$ from other states $\mathbf{x}_l \in \mathcal{V}$ to $\tilde{\mathbf{x}}_l$. Similarly, a state $\tilde{\mathbf{x}}_{l,child} = \mathcal{C}(\tilde{\mathbf{x}}_l)$ is considered as a child node of $\tilde{\mathbf{x}}_l$ if the cost $c^*(\tilde{\mathbf{x}}_l, \tilde{\mathbf{x}}_{l,child})$ is the smallest compared to costs $c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_l)$ from $\tilde{\mathbf{x}}_l$ to other states $\mathbf{x}_l \in \mathcal{V}$. The set $\mathcal{L}$ is the mask set containing mask values of all states $\mathbf{x}_l \in \mathcal{V}$ in the form

$$\mathcal{L}(\mathbf{x}_l) = \begin{cases} 0 & \text{if } c^*(\mathbf{x}_{l,S}, \mathbf{x}_l) > J^* \\ 1 & \text{otherwise} \end{cases}, \tag{3.11}$$

where

$$J^* = \sum_{\mathbf{x}_l = \mathbf{x}_{l,S}}^{\mathcal{P}(\mathbf{x}_{l,T})} c^*(\mathbf{x}_l, \mathcal{C}(\mathbf{x}_l)) \ , \tag{3.12}$$

is the total optimal cost of the collision-free trajectory from $\mathbf{x}_l = \mathbf{x}_{l,S}$ to a sequence of child nodes $\mathcal{C}(\mathbf{x}_l)$ that leads to the target state $\mathbf{x}_{l,T}$. This total cost is initialized to $J^* = \infty$ at the beginning of the algorithm, since a collision-free trajectory from the initial state $\mathbf{x}_{l,S}$ to the target state $\mathbf{x}_{l,T}$ has not yet been found. Once the optimal total cost J^* is updated, the mask set $\mathcal{L}$ is used to prune the trajectory tree $\mathcal{T}$, i.e., the states whose corresponding masks are equal to zero are not considered in the subsequent loop of the algorithm.

In lines 1-3, the algorithm is initialized by inserting the starting state $\mathbf{x}_{l,S}$ into the trajectory tree $\mathcal{T}$. The stopping criterion is chosen to be the total allowed computing time. In the following, important steps of the Alg. 1 are briefly explained.

First, the proposed algorithm randomly samples the flat output $\tilde{\mathbf{x}}_l \in \mathbb{X}_{free}$ in each iteration (line 6 in Alg. 1) with $\mathbb{X}_{free} = \mathbb{P}_{free} \times [\underline{\mathbf{z}}^{\mathrm{T}}, \overline{\mathbf{z}}^{\mathrm{T}}] \times [\underline{\mathbf{u}}^{\mathrm{T}}, \overline{\mathbf{u}}^{\mathrm{T}}]$. Here, $\underline{\mathbf{z}}$, $\overline{\mathbf{z}}$, $\underline{\mathbf{u}}$, $\overline{\mathbf{u}}$ are the lower and upper bounds of the system state and control input in (2.13). Additionally, $\mathbb{P}_{free}$ denotes the free space not occupied by the obstacles. In line 9

of Alg. 1, the algorithm searches for the parent state $\tilde{\mathbf{x}}_{l,parent}$ of the sampled state $\tilde{\mathbf{x}}_l$ by utilizing the best-first search algorithm, see, e.g., [55] and Alg. 2. In line 2 of the best-first search algorithm in Alg. 2, the standard C++ PopQueue function is utilized to take the first item of the set $\mathcal{Q}$ and to effectively reduce the size of this set by removing this item. Note that the set $\mathcal{Q}$ is first initialized by the starting state $\mathbf{x}_{l,S}$ in line 7 of Alg. 1. Next, in line 4 of Alg. 2, the LocalPlanner, presented in Section 3.2.2, is employed to compute the optimal cost c_{temp} and the feasibility flag c_{flag} from a state in $\mathcal{Q}$ to the sampled state $\tilde{\mathbf{x}}_l$. This process is repeated until the parent state $\tilde{\mathbf{x}}_{l,parent}$ of the sampled state $\tilde{\mathbf{x}}_l$ is chosen.

Next, in line 10 of Alg. 1, the feasibility condition $\mathcal{X}_{\hat{f}_c}(\tilde{\mathbf{x}}_l)$ is verified if the sampled state $\tilde{\mathbf{x}}_l$ provides a better solution than the current total cost J^*. The set $\mathcal{X}_{\hat{f}_c}$, also named "informed set", reads as

$$\mathcal{X}_{\hat{f}_c} = \left\{ \hat{f}_c(\tilde{\mathbf{x}}_l) < J^* \right\} , \tag{3.13}$$

where

$$\hat{f}_c(\tilde{\mathbf{x}}_l) = \hat{c}(\mathbf{x}_{l,S}, \tilde{\mathbf{x}}_l) + c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_{l,T}) , \tag{3.14}$$

and

$$\hat{c}(\mathbf{x}_{l,S}, \tilde{\mathbf{x}}_l) = \sum_{\mathbf{x}_l = \mathbf{x}_{l,S}}^{\mathcal{P}(\tilde{\mathbf{x}}_l)} c^*(\mathbf{x}_l, \mathcal{C}(\mathbf{x}_l)) \tag{3.15}$$

is the optimal cost incurred by moving the system from the starting state $\mathbf{x}_{l,S}$ through all corresponding child states to the sampled state $\tilde{\mathbf{x}}_l$. Once the condition of the informed set $\mathcal{X}_{\hat{f}_c}$ is satisfied for $\tilde{\mathbf{x}}_l$, in line 12 of Alg. 1, the sampled state $\tilde{\mathbf{x}}_l$ and its corresponding cost $\hat{c}(\mathbf{x}_{l,S}, \tilde{\mathbf{x}}_l)$ and $c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_{l,T})$ (3.15) are appended to the set $\mathcal{V}$.

In lines 14-19 of the Alg. 1, the rewiring process is performed after inserting the sampled state $\tilde{\mathbf{x}}_l$. This process is essential because the newly added state $\tilde{\mathbf{x}}_l$ may be a parent state of some states in $\mathcal{V}$. Only states $\mathbf{x}_l \in \mathcal{V}$ that have the non-zero corresponding mask value $\mathcal{L}(\mathbf{x}_l)$, see (3.11), are considered for the rewiring process. Here, the newly added state $\tilde{\mathbf{x}}_l$ is considered as the parent state of $\mathbf{x}_l$, if the condition $\mathbf{c}^*(\mathbf{x}_{l,parent}, \mathbf{x}_l) > \mathbf{c}^*(\tilde{\mathbf{x}}_l, \mathbf{x}_l)$ holds, see line 16 in Alg. 1. Then, the parent and child sets are updated in lines 17-18 of Alg. 1.

In lines 21-22 in Alg. 1, the proposed algorithm is checked to ensure that the

sampled state $\tilde{\mathbf{x}}_l$ can be connected to the target state $\mathbf{x}_T$ via a collision-free and dynamically feasible trajectory. If so, in line 23 of Alg. 1, the total cost

$$J^* = \hat{c}(\mathbf{x}_{l,S}, \tilde{\mathbf{x}}_l) + c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_{l,T}) \tag{3.16}$$

is updated where $\hat{c}(\mathbf{x}_{l,S}, \tilde{\mathbf{x}}_l)$ can be obtained via (3.15). Subsequently, the mask values for each state $\mathbf{x}_l \in \mathcal{V}$ are also updated in line 29 of Alg. 1. Note that this set of masks $\mathcal{L}$ helps to prune the tree $\mathcal{T}$. The advantage of this procedure is twofold. First, this procedure helps to reduce the number of states to be checked during the BFS process in Alg. 2. Second, any state with a mask value of zero can be eliminated since it does not contribute to finding a better solution. To enrich the trajectory tree $\mathcal{T}$, the proposed algorithm is processed in parallel to generate different trajectory tree stacks. Then, these trees are concatenated to form a single trajectory tree $\mathcal{T}$.

Note that the proposed algorithm can be viewed as a subclass of "anytime" algorithms that quickly finds a feasible (but not necessarily optimal) trajectory and gradually improves the solution over time towards global optimality, see, e.g., [12]. This "anytime" property can be achieved thanks to the structure of the trajectory tree $\mathcal{T}$, which contains extensive information about the relationship between the considered robot system and the environment. Alg. 3 presents the replanning algorithm that takes into account the change of the target state $\mathbf{x}_{l,T}$. When the target state is changed from $\mathbf{x}_{l,T}$ to $\mathbf{x}_{l,T_{new}}$, the refinement process updates the associated costs of all states in the set $\mathcal{V}$, see line 4 of Alg. 3. Then, similar to (3.16), the optimal total cost is recomputed if a state $\mathbf{x}_l \in \mathcal{V}$ can be connected collision-free to the new target state $\mathbf{x}_{l,T_{new}}$, see line 7 of Alg. 3. After the optimal cost J^* is determined, the mask set $\mathcal{L}$ is updated (in lines 13-15 of Alg. 3). If the trajectory tree $\mathcal{T}$ is dense enough and the deviation of the new target state from the original target state is small, the proposed algorithm can quickly find a feasible solution and gradually improve the solution.

3.3 Simulation results

The simulation was performed using MATLAB R2021b on a desktop computer with 3.4 GHz Intel Core i7 and 32GB RAM. Simulations were performed with the two scenarios depicted in Fig. 3.1. Without loss of generality, all obstacles are

Algorithm 1: Flat-informed RRT* trajectory planning

Input: $(\mathbf{x}_{l,S}, \mathbf{x}_{l,T}) \in \mathbb{X}_{free}$
Output: $\mathcal{T} = \{\mathcal{V}, \mathcal{P}, \mathcal{C}, \mathcal{L}\}$

$\mathcal{V} \leftarrow \{\mathbf{x}_{l,S}, 0, c^*(\mathbf{x}_{l,S}, \mathbf{x}_{l,T})\}$
$\mathcal{P} \leftarrow \varnothing, \quad \mathcal{C} \leftarrow \varnothing, \quad \mathcal{L}(\mathbf{x}_{l,S}) \leftarrow 1$
$J^* \leftarrow \infty$;
while NotStopCriteria **do**
  while NotFeasibleSample **do**
    $\tilde{\mathbf{x}}_l \leftarrow \mathsf{GetRandomSample}(\mathbb{X}_{\mathsf{free}})$
    $\mathcal{Q} \leftarrow \mathbf{x}_{l,S}$
    #perform the best-first search (BFS) in Alg. 2
    $\tilde{\mathbf{x}}_{l,parent} \leftarrow \mathrm{BFS}(\mathcal{Q}, \tilde{\mathbf{x}}_l)$
    $\mathsf{NotFeasibleSample} = \hat{f}_c(\tilde{\mathbf{x}}_l) \neq 1$
  end
  $\mathcal{V} \leftarrow \{\tilde{\mathbf{x}}_l, \hat{c}(\mathbf{x}_{l,S}, \tilde{\mathbf{x}}_l), c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_{l,T})\}$
  $\mathcal{P}(\tilde{\mathbf{x}}_l) \leftarrow \tilde{\mathbf{x}}_{l,parent}$, $\mathcal{C}(\tilde{\mathbf{x}}_{l,parent}) \leftarrow \tilde{\mathbf{x}}_l$
  #perform the rewiring process
  for $\mathbf{x}_l \in \mathcal{V}$ and $\mathcal{L}(\mathbf{x}_l) = 1$ **do**
    if $c^*(\mathbf{x}_{l,parent}, \mathbf{x}_l) > c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_l)$ **then**
      $\mathcal{P}(\mathbf{x}_l) \leftarrow \tilde{\mathbf{x}}_l$, $\mathcal{C}(\tilde{\mathbf{x}}_l) \leftarrow \mathbf{x}_l$
      $\mathcal{C}(\mathbf{x}_{l,parent}) \leftarrow \mathcal{C}(\mathbf{x}_{l,parent}) \setminus \mathbf{x}_l$
    end
  end
  $c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_{l,T}), c_{flag} \leftarrow \mathsf{LocalPlanner}(\tilde{\mathbf{x}}_l, \mathbf{x}_T)$
  if $c_{flag} = 1$ **then**
    $J^*_{temp} \leftarrow \hat{c}(\mathbf{x}_{l,S}, \tilde{\mathbf{x}}_l) + c^*(\tilde{\mathbf{x}}_l, \mathbf{x}_{l,T})$
    if $J^*_{temp} < J^*$ **then**
      $J^* \leftarrow J^*_{temp}$
      #pruning process
      for $\mathbf{x}_l \in \mathcal{V}$ **do**
        if $c^*(\mathbf{x}_{l,S}, \mathbf{x}_l) > J^*$ **then**
          $\mathcal{L}(\mathbf{x}_l) \leftarrow 0$
        end
      end
    end
  end
end

Algorithm 2: Best-first search (BFS) algorithm

Input: $\mathcal{Q}, \tilde{\mathbf{x}}_l$
Output: $\tilde{\mathbf{x}}_{l,parent}$

$c_{min} = \infty$
while isNotEmpty($\mathcal{Q}$) **do**
  $\mathbf{x}_{l,min} \leftarrow$ PopQueue($\mathcal{Q}$)
  $c_{temp}, c_{flag} \leftarrow$ LocalPlanner($\mathbf{x}_{l,min}, \tilde{\mathbf{x}}_l$)
  if $c_{flag} = 1 \wedge c_{temp} < c_{min}$ **then**
    $\tilde{\mathbf{x}}_{l,parent} \leftarrow \mathbf{x}_{l,min}$
    $c_{min} \leftarrow c_{temp}$
    $\mathcal{Q} \leftarrow \mathcal{C}(\mathbf{x}_{l,min})$
  else
  end
  $\mathcal{Q} \leftarrow \mathcal{C}(\mathbf{x}_{l,min})$
end

Algorithm 3: Update tree $\mathcal{T}$ w.r.t. a new target state

Input: $\mathcal{T}, \mathbf{x}_{l,t_{new}} \in \mathbb{X}_{free}$
Output: $\mathcal{T}$

$J^* \leftarrow \infty$
#recompute the trajectory cost J^*
for $\mathbf{x}_l \in \mathcal{V}$ **do**
  $c^*(\mathbf{x}_l, \mathbf{x}_{l,T_{new}}), c_{flag} \leftarrow$ LocalPlanner($\mathbf{x}_l, \mathbf{x}_{l,T_{new}}$)
  if $c_{flag} = 1$ **then**
    if $c_{temp} + c^*(\mathbf{x}_S, \mathbf{x}_l) < J^*$ **then**
      $J^* \leftarrow c_{temp} + \hat{c}(\mathbf{x}_S, \mathbf{x}_l)$
    end
  end
end
#pruning process
for $\mathbf{x}_l \in \mathcal{V}$ **do**
  if $c^*(\mathbf{x}_{l,S}, \mathbf{x}_l) > J^*$ **then**
    $\mathcal{L}(\mathbf{x}_l) \leftarrow 0$
  end
end

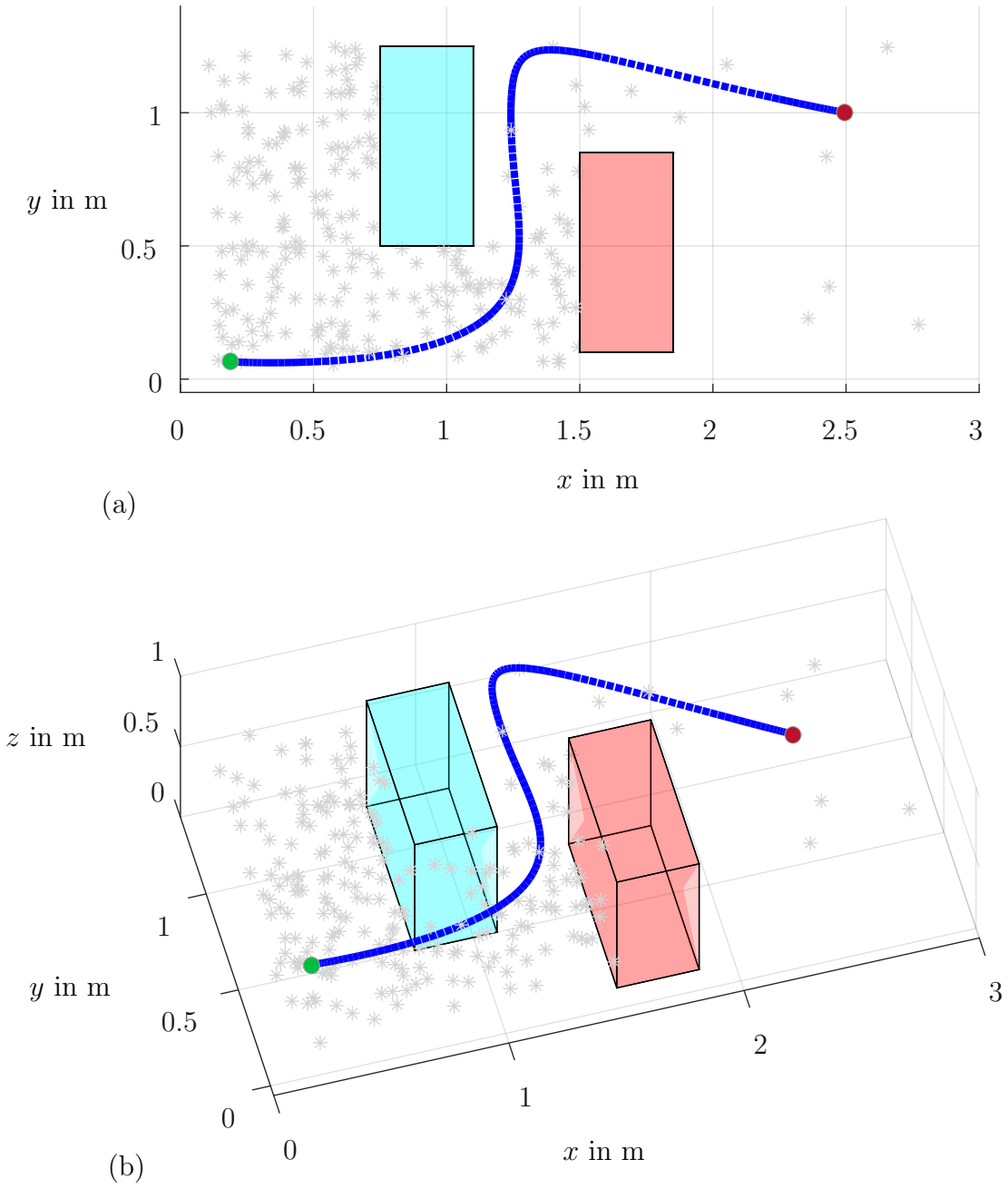


Figure 3.2: Scenario 1: Collision-free path from a starting state (green dot) to a target state (red dot) resulting as a solution of Algorithm 1. The grey asterisks are the flat output states in the trajectory tree. (a) Path in the xy-plane. (b) Path in 3D space.

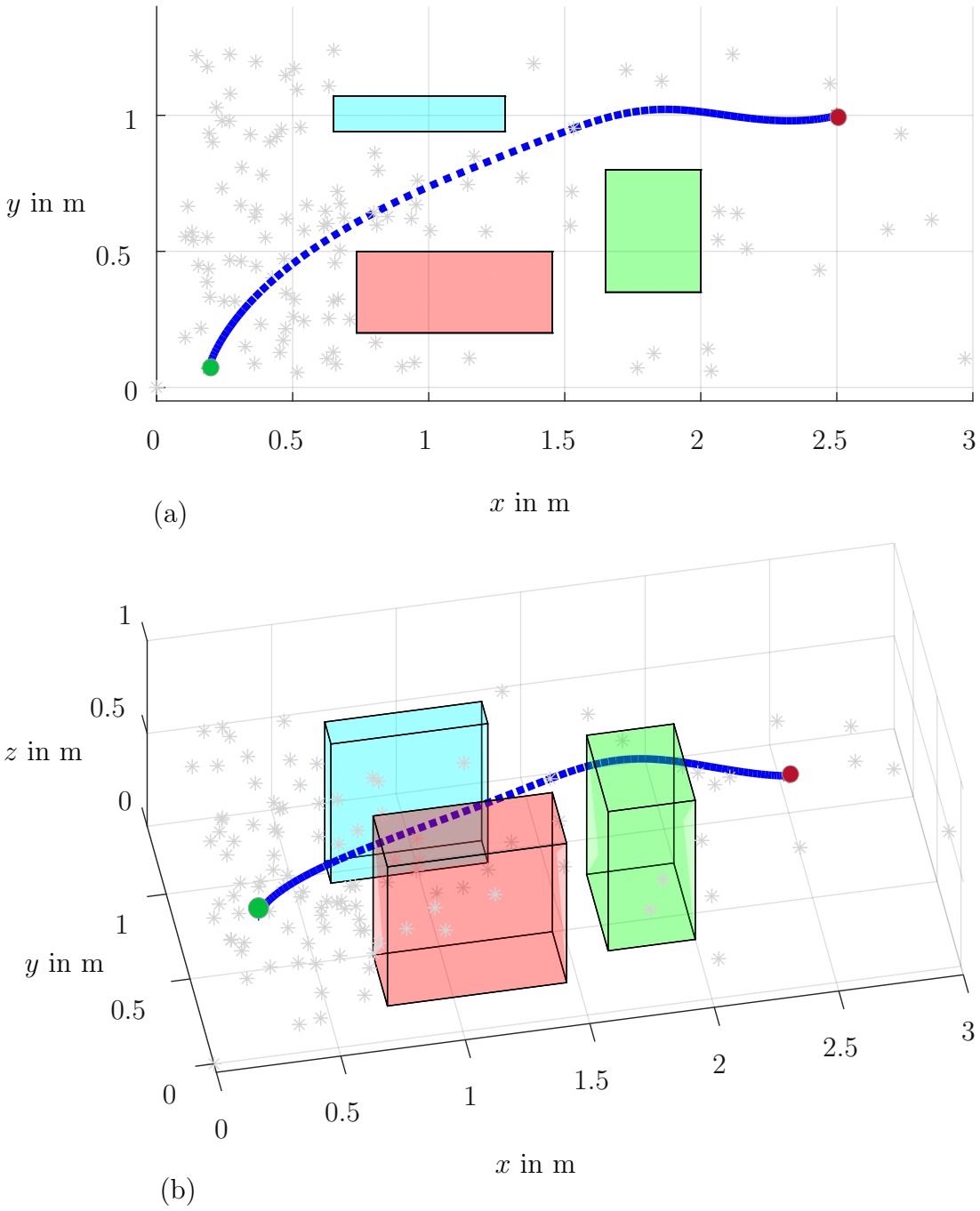


Figure 3.3: Scenario 2: Collision-free path from a starting state (green dot) to a target state (red dot) resulting as a solution of Algorithm 1. The grey asterisks are the flat output states in the trajectory tree. (a) Path in the xy-plane. (b) Path in 3D space.

Table 3.1: Obstacle parameters.

(a) Scenario 1 in Fig. 3.1(a)

Obstale number	Cartesian position $\mathbf{T}_i^{\mathrm{T}}$	Dimension $\mathbf{p}_i^{\mathrm{T}}$
1	$[1.5, 0.1, 0]$	$[0.35, 0.75, 0.75]$
2	$[0.75, 0.5, 0]$	$[0.35, 0.75, 0.75]$

(b) Scenario 2 in Fig. 3.1(b)

Obstale number	Cartesian position $\mathbf{T}_i^{\mathrm{T}}$	Dimension $\mathbf{p}_i^{\mathrm{T}}$
1	$[0.735, 0.20, 0]$	$[0.72, 0.3, 0.75]$
2	$[0.65, 0.94, 0]$	$[0.63, 0.13, 0.75]$
3	$[1.65, 0.35, 0]$	$[0.35, 0.45, 0.75]$

on the ground and aligned with the x-, y-, and z-axis of the world coordinate system. The obstacles' positions and dimensions are introduced in Tab. 3.1, where $\mathbf{p}_i^{\mathrm{T}} = [w_i, h_i, d_i]$ contains the width w_i, the height h_i, and the depth d_i of the i^{th} obstacle at the location $\mathbf{T}_i^{\mathrm{T}} = [T_{i,x}, T_{i,y}, T_{i,z}]$, see Fig. 3.1.

The starting and target positions of the 3D gantry crane payload in both scenarios are $\mathbf{r}_S = [0.19, 0.065, 0.7]^{\mathrm{T}}$ and $\mathbf{r}_T = [2.5, 1, 0.2]^{\mathrm{T}}$, respectively.

The proposed flat-informed sampling-based trajectory planning takes random samples of the flat initial system state $\tilde{\mathbf{x}}_l$ and builds the tree $\mathcal{T}$ according to Alg. 1. To connect a randomly sampled system state $\tilde{\mathbf{x}}_l$ with a point in the trajectory tree $\mathcal{T}$, the local planner LQMT from Subsection 3.2.2 was employed where the user-defined weighting matrix $\mathbf{R} = 10^{-1}\mathbf{I}_3$ with the identity matrix $\mathbf{I}_3$, was chosen.

In both scenarios, the total allowed planning time t_{plan} of 30 s was used as the termination criterion, i.e. it is the maximum time allowed to build the tree $\mathcal{T}$. To calculate the distance from each point in the workspace to the obstacles, the two scenario maps are discretized into equidistant 3D voxel grids of 0.01 m in each dimension and the value 1 is assigned to all voxels occupied by an obstacle. Successively, the fast Euclidean distance transform (EDT), see [56], was used to check whether the trajectory of the payload is obstacle-free or not.

Figs. 3.2 and 3.3 show collision-free paths from a payload start position (green dot) to a payload target position (red dot) in the two scenarios 1 and 2, respectively. In the first scenario, the optimal travel time Δt^* is 14.94 s and the optimal total

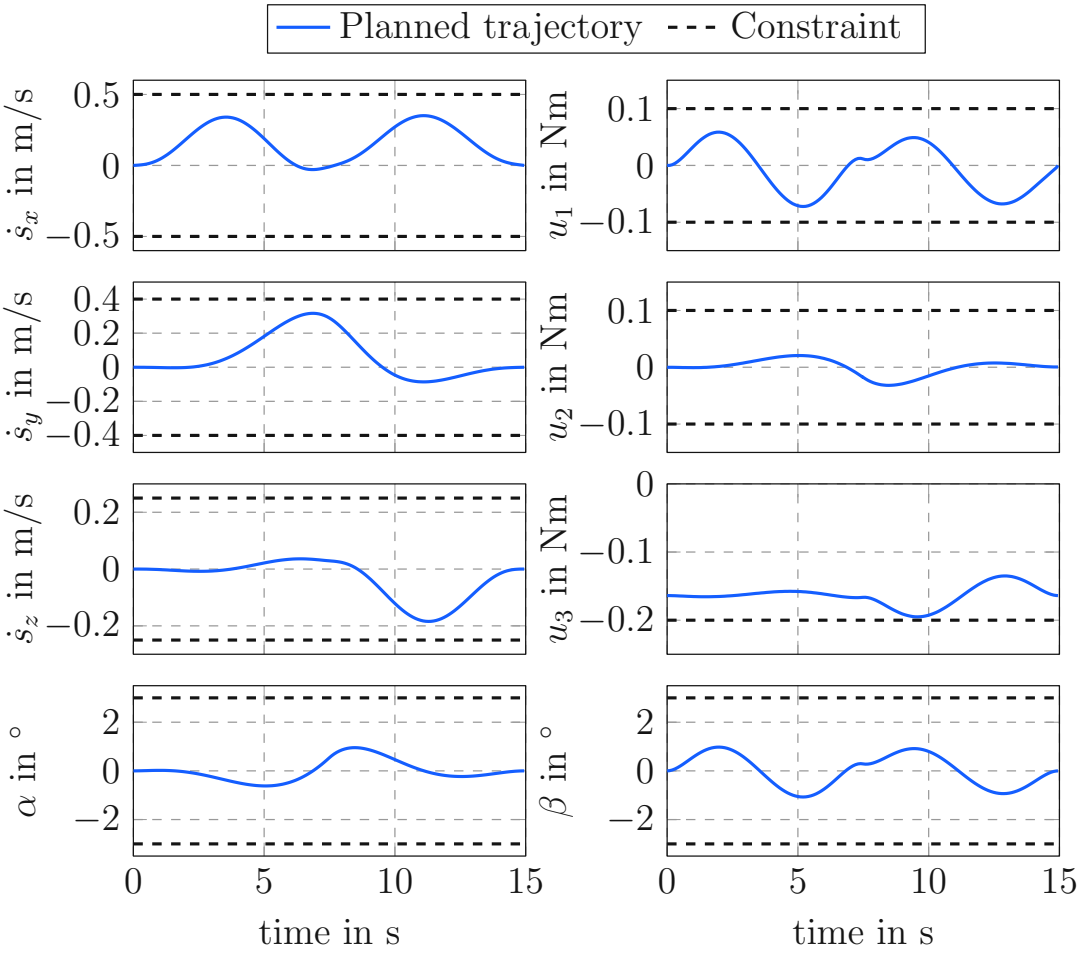


Figure 3.4: Scenario 1: Time evolution of the states and control inputs of the system (2.13).

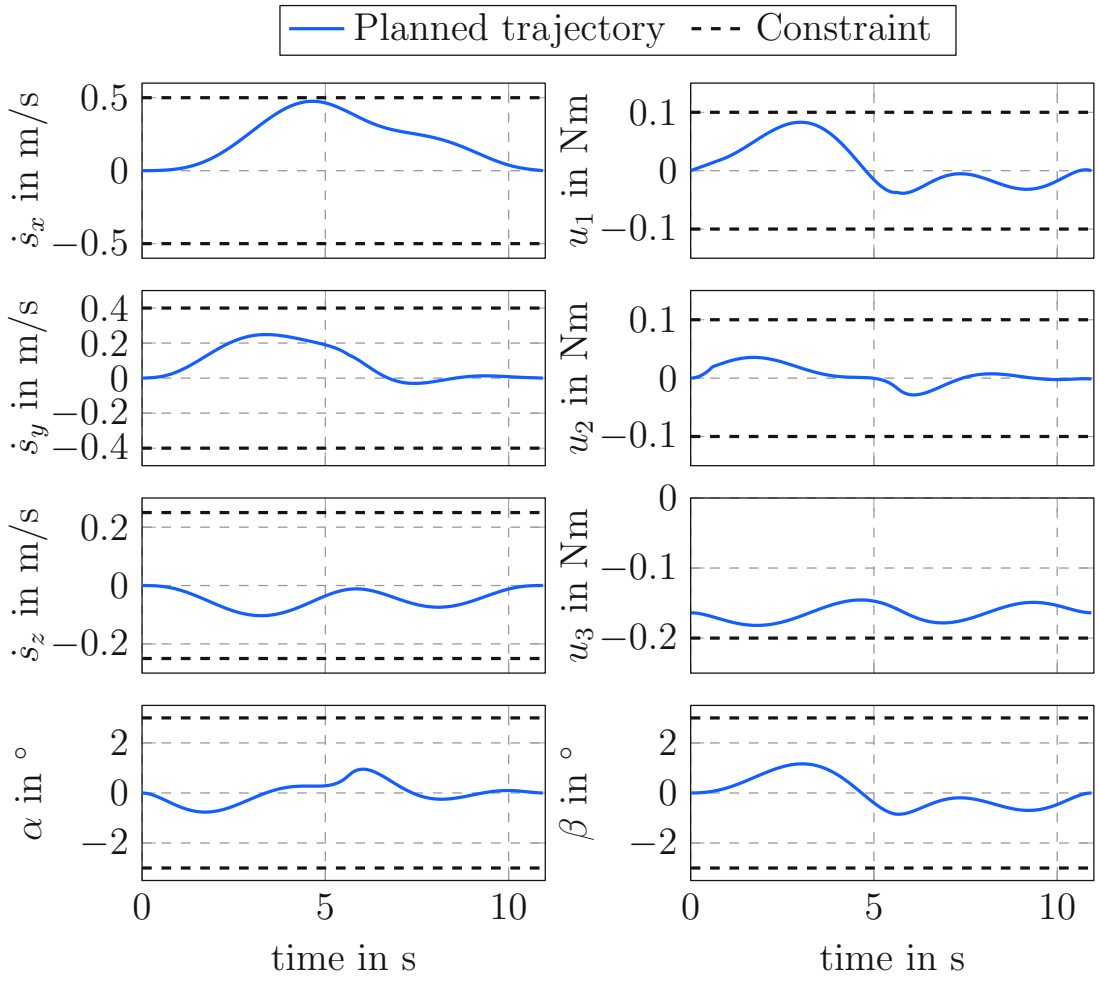


Figure 3.5: Scenario 2: Time evolution of the states and control inputs of the system (2.13).

cost J^* is 15.32. Although there are more obstacles in the second scenario, the optimal travel time is shorter than in the first scenario with $\Delta t^* = 11.36\,\mathrm{s}$ and the optimal total cost J^* is 12.5. Since the informed property (3.13) and the pruning process (3.11) are employed in Alg. 1, random states which are behind the target region are discarded since these states do not contribute to find a better trajectory. Therefore, the density of the trajectory tree gradually decreases towards the target region.

Fig. 3.4 and 3.5 illustrate the time evolution of the corresponding states $\dot{s}_x$, $\dot{s}_y$, $\dot{s}_z$, α and β, together with the three control inputs u_1, u_2 and u_3 of the 3D gantry crane system (2.13). Note that the state and input constraints according to (3.13), shown as black dashed lines, are well respected.

Table 3.2: Performance for different maximum allowed planning times t_{plan} for the considered scenarios.

Scenario 1				
t_{plan}	30 s	50 s	100 s	200 s
Δt^*	14.94	12.54	12.84	11.24
J^*	15.32	13.96	13.79	12.05
$\lvert\mathcal{T}\rvert$	1102	1372	1979	2955
Scenario 2				
Δt^*	11.36	10.94	9.42	8.97
J^*	12.5	11.83	10.23	9.75
$\lvert\mathcal{T}\rvert$	874	998	1049	942

Table 3.2 shows the performance of the proposed algorithm for different maximum allowed planning times t_{plan} for both scenarios. Note that in this Monte Carlo simulations, the proposed algorithm is processed in parallel on 8 CPU cores. After the maximum allowed time t_{plan} elapses, all trees generated by the different CPU cores are merged into a single tree. Overall, Tab. 3.2 shows that the travel time and the total cost J^* decrease when the proposed algorithm is given a longer planning time t_{plan}. The size of the trajectory tree is denoted by $|\mathcal{T}|$. In the classical RRT* algorithm, increasing the planning time leads to a more complex data tree. However, the proposed algorithm uses the informed set (3.13) and performs the pruning procedure (3.11), which reduces the complexity of the trajectory tree. For example, in Table 3.2 of scenario 2, although the planning time t_{plan} is increased from 100 s to 200 s, the size of the trajectory tree $|\mathcal{T}|$ is reduced from 1049 nodes

to 942 nodes. This also helps to increase the computational speed compared to the classical RRT* algorithm.

In both scenarios, the proposed algorithm not only provides a smooth flat output trajectory that can be used to parameterize the system state using (2.22) and (2.23), but also makes available the trajectory tree $\mathcal{T}$ that can be reused in the replanning process in Alg. 3 when the target state is changed. If a new target state is obtained, the trajectory tree $\mathcal{T}$ is updated by recalculating the costs of the trajectory points in the set $\mathcal{V}$ with respect to the new target state. In case the current trajectory tree can generate a collision-free and dynamically feasible trajectory to the new target state, the remainder of the tree $\mathcal{T}$ remains unchanged. Otherwise, Alg. 1 is processed again with the current trajectory tree as the initial tree. This property is presented in the following simulations.

Figs. 3.6 and 3.7 show the replanned trajectories from the initial payload position (green dot) to random target payload positions (red asterisks) in a certain region of possible target positions for the two scenarios. The new target position $\mathbf{r}_{T_{new}}$ is randomly chosen according to the current target state $\mathbf{r}_T = [r_{T,x}, r_{T,y}, r_{T,z}]^{\mathrm{T}}$ in the form

$$\mathbf{r}_{T_{new}} = \begin{bmatrix} r_{T_{new},x} \\ r_{T_{new},y} \\ r_{T_{new},z} \end{bmatrix} = \begin{bmatrix} r_{T,x} - 0.3 + 0.6 \cdot \mathrm{rand}() \\ r_{T,y} - 0.3 + 0.6 \cdot \mathrm{rand}() \\ r_{T,z} - 0.3 + 0.6 \cdot \mathrm{rand}() \end{bmatrix} , \tag{3.17}$$

where the function rand() returns a uniform distributed random number in the range of $[0, 1]$.

The blue trajectories are the initial obstacle-free trajectories and the green trajectories are the replanned trajectories to the new target payload positions. In this simulation, the trajectory tree does not need to be recalculated because the computed tree $\mathcal{T}$ is dense and all generated paths are collision-free and smooth in both scenarios. However, even small deviations can lead to drastically different paths, due to the dynamic constraints of the crane and the presence of obstacles. As is shown in this example, the necessary information to deal with these changes in the global optimization is captured by the tree. The average computation time of the rescheduling process in scenarios 1 and 2 is 70 ms and 50 ms, respectively.

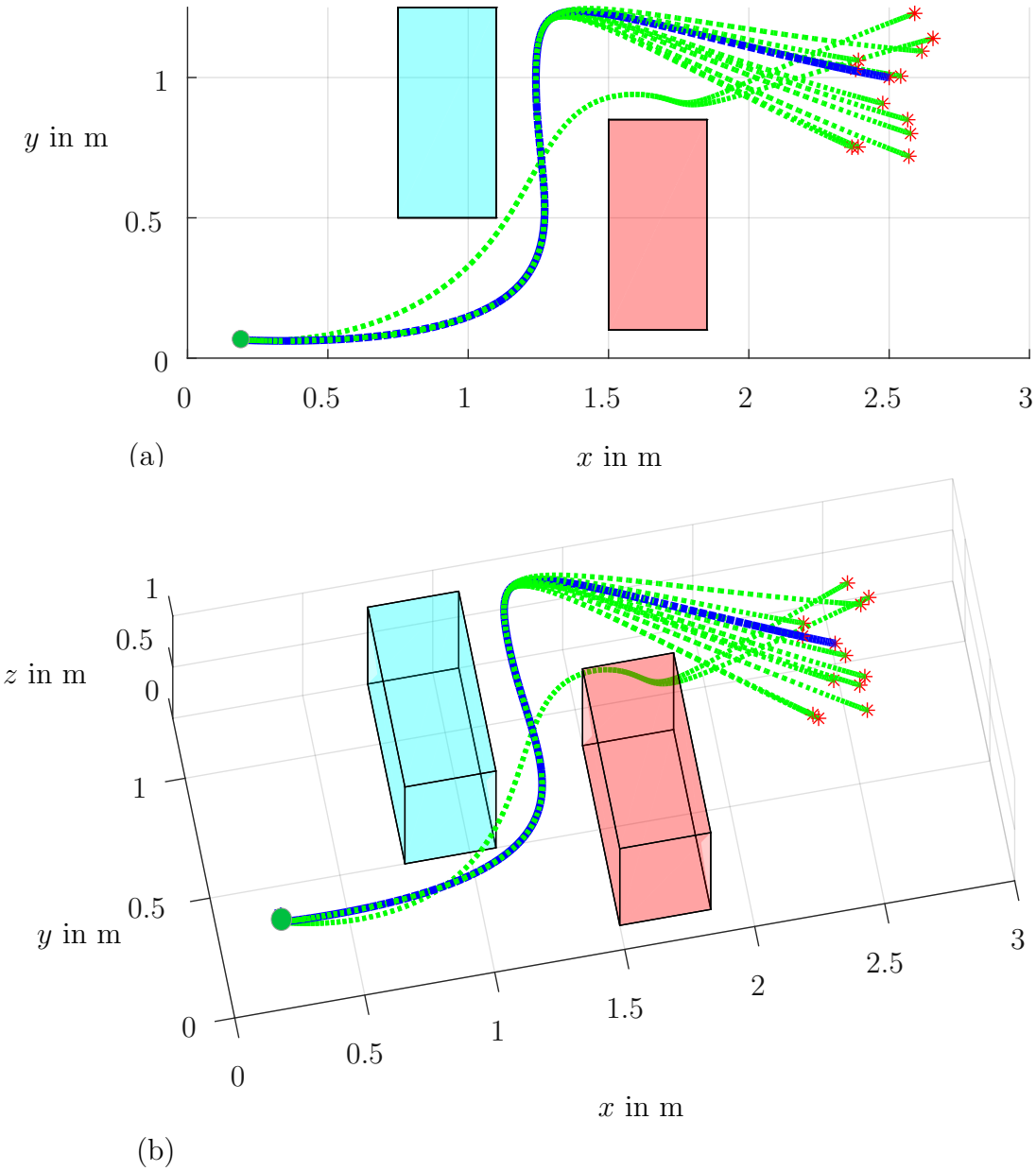


Figure 3.6: Scenario 1: Collision-free paths (green lines) for random target payload positions in a certain region of possible target positions. The blue path is the initial collision-free path of the tree. Red asterisks denote the random target payload positions. (a) Paths in the xy-plane. (b) Paths in 3D space.

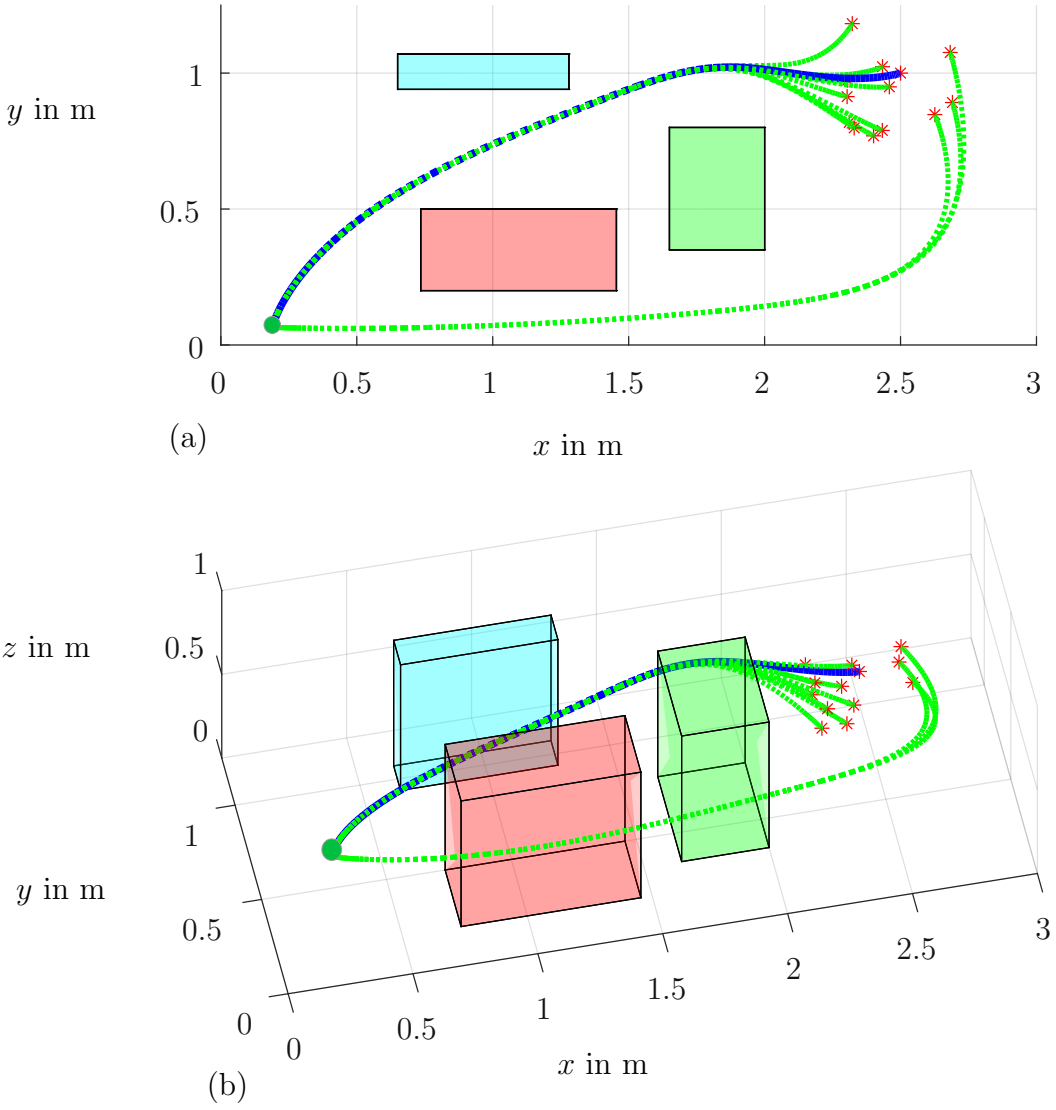


Figure 3.7: Scenario 2: Collision-free paths (green lines) for random target payload positions in a certain region of possible target positions. The blue path is the initial collision-free path of the tree. Red asterisks denote the random target payload positions. (a) Paths in the xy-plane. (b) Paths in 3D space.

3.4 Conclusions

This section presents a sampling-based flat-informed RRT* trajectory planning for a 3D gantry crane in an environment with static obstacles. The proposed algorithm considers an informed set that helps to sample a state if that state leads to an improvement in the overall trajectory cost. Moreover, the informed set indirectly helps to reduce the computation time of the proposed trajectory planning algorithm. A branch-and-bound technique is utilized to prune the trajectory tree to reduce the computational complexity. Since a local planner is a core part of the sampling-based method, the analytical solution of a linear quadratic minimum-time problem was utilized. Moreover, the structure of the trajectory tree can be quickly updated when the target state changes. This is useful in many practical scenarios with repetitive tasks, such as picking up and dropping goods at a factory or port, where the trajectory tree can be reused instead of solving the entire trajectory planning problem. The proposed trajectory planning method is in particular well suited for systems which feature the differential flatness property, such as gantry cranes and unmanned aerial vehicles.

The proposed flat-informed RRT* can quickly generate replanned trajectories when the targets are static and the allowable time for building the trajectory tree and memory consumption are not major issues. However, in a scenario with moving targets, the informed RRT* is not guaranteed to generate a re-planned trajectory in real time. This because the entire trajectory tree may not be dense enough to produce a feasible trajectory and leads to the need for regenerating the entire trajectory tree, which takes more time. Therefore, the following section proposes an optimization-based trajectory planning method whose real-time capability can be demonstrated both in simulation and experiment for the moving target scenario.

4

Optimization-based trajectory planning for a 3D gantry crane

This section presents an optimization-based trajectory planning algorithm for the lab-scale 3D gantry crane in an environment with *static obstacles* and a *dynamically moving target*. In the first step, an offline optimization-based trajectory planner is implemented to compute a time-optimal, collision-free and dynamically feasible trajectory database that connects all possible initial states from a predefined starting subspace to the target states in a target subspace in the workspace of the gantry crane. Second, the online trajectory replanning makes use of this trajectory database to generate an optimal trajectory in real time that accounts for changes in the target state. The proposed algorithm systematically takes into account dynamic constraints and control input limits. To realize the proposed trajectory planning algorithm in an experimental setup, a cascaded trajectory tracking controller is developed which compensates for model inaccuracies, disturbances, and other non-modeled effects. Both simulation and experimental results are performed to demonstrate the feasibility of the proposed trajectory planning algorithm and the control concept. Large parts of this section are identical to the author's publication [21].

4.1 Introduction

There are different concepts for solving motion planning problems in the literature which can be essentially divided into two major groups, i.e., sampling-based approaches and optimization-based approaches. In the previous section, the flat-informed RRT* was introduced, which is a sampling-based approach and allows for fast replanning to a new target state. However, if the target is dynamically changing, further post-processing and higher memory consumption are required, which prevents the execution in real time.

Optimization-based methods are particularly useful for finding locally optimal trajectories and systematically respecting the dynamic constraints of the system, see, e.g., [57, 58]. Covariant Hamiltonian Optimization for Motion Planning (CHOMP), developed by [59], is one of the most successful algorithms. The following key features are considered by CHOMP. First, the trajectory costs are formulated to be invariant w.r.t. the time parametrization of the discretized trajectory. Second, the precalculated signed distance field (SDF), which relies on the Euclidean distance transform (EDT) [56], provides the distance and gradient from each discretized trajectory point to obstacle surfaces. The covariant Hamiltonian optimization solver is utilized to find the locally optimal trajectory. CHOMP has proven its effectiveness in several applications including the Little-Dog quadruped, PR2 robot, and the HERB mobile manipulation platform, see, e.g., [59]. More recently, inspired by CHOMP, [60] introduced the TrajOpt optimization open-source software package. There are two different points between CHOMP and TrajOpt, e.g., the approach for collision detection and the numerical optimization scheme. The obstacle avoidance relies on the convex-convex collision checking approach, which takes two shapes (e.g. the robot's link and the obstacle) and computes the minimal translation between them by the Gilbert-Johnson-Keerthi (GJK) algorithm, see, [61]. Additionally, Sequential Quadratic Programming (SQP) is employed in the Trajopt package as the numerical optimization. Here, the advantage of optimization-based trajectory planning over flatness-based methods becomes noticeable in terms of obstacle avoidance, sway suppression, and compliance with dynamic constraints, state and control input limits. However, the computational time of optimization-based trajectory planning is still too long for a real-time implementation on a standard electronic control unit, see, e.g., [62, 63].

In scenarios with repetitive tasks, the optimization-based trajectory planning can be solved in a computationally more efficient way. For example, if the starting and/or target states are only changed slightly, only the deviations from the previous trajectory need to be computed instead of running the full optimization. For this reason, trajectory replanning algorithms are developed using dynamic motion primitives (DMPs) or Gaussian Mixture Models (GMMs), see, e.g., [64, 65]. However, the computation of GMMs and DMPs becomes inefficient in a high-dimensional state space. For the considered lab-scale 3D gantry crane, the dimension of the state space is 10. Moreover, GMM- and DMP-based approaches typically neglect state constraints and control input limitations. Therefore, these concepts are not suitable for the considered application.

In this section, the following scenario is considered. A 3D gantry crane has to move goods or material from one dedicated place (starting space $\mathcal{X}_S$) to another place (target space $\mathcal{X}_T$) in a static environment with known obstacles. The task is to plan a time-optimal trajectory from a given starting point to a given target point, which systematically accounts for both the obstacles and the dynamic constraints on the state variables and control inputs. The focus is to compute a solution that allows a very fast (re)planning if the starting and/or target point are changing. Additionally, the proposed online trajectory planner should be also able to handle moving targets.

The proposed two-step trajectory planning algorithm is briefly introduced as follows. First, an offline optimization-based trajectory planning is utilized to build a time-optimal and dynamically feasible trajectory database that connects points in the starting space $\mathcal{X}_S$ with points in the target space $\mathcal{X}_T$. Then, a fast search algorithm is employed to quickly look up the relevant offline trajectory in the database which is then used for the online trajectory optimization in the second step. This online trajectory replanner uses a quadratic programming solver to generate a trajectory in real time when the target state is moving.

Together with the trajectory planning, the trajectory tracking controller also plays an important role for the 3D gantry crane system. In the literature, the mathematical model of a gantry crane is often decomposed into the slow pendulum subsystem and the fast subsystem which contains the dynamics of the trolley and the hoist drum, see, e.g., [29, 27]. Thus, the trajectory tracking controller typically relies on a cascaded structure, in which the outer loop of the cascaded trajectory tracking

controller keeps the unactuated angles of the hoist cable around the desired trajectory. The inner control loop provides tracking of the desired payload trajectory, see, e.g., [29, 27, 66]. In these works, the system state and control input constraints are not directly considered. Thus, there is room for improving the cascaded controller design by using a control scheme that systematically accounts for these constraints, i.e., a model predictive control. This feature of the controller, proposed in Section 4.3, is particularly important for the system to navigate around obstacles and for suppressing sway of the payload.

In recent literature for automated cranes, e.g., [67, 68], the trajectories for a 3D gantry crane are calculated offline and tested in simulations. In contrast, the proposed algorithm in this work generates collision-free trajectories online, and is validated by experiments. Moreover, in comparison with the most successful practical studies, e.g., [5, 17], the proposed combined method has the ability to generate near time-optimal dynamically feasible trajectories online for scenarios with obstacles and a moving target. In addition, both the fast trajectory planning and the trajectory tracking controller systematically take into account the system constraints and control input limits, which is also experimentally validated in this section.

This section is organized as follows. In Section 4.2, the novel two-stage trajectory planning algorithm is presented. Section 4.3 introduces the details of the cascaded control concept including a model predictive controller (MPC). Simulations and experiments are presented in Section 4.4. Finally, the last section gives some conclusions.

4.2 Two-step trajectory planning

In this section, the fast trajectory planning algorithm is presented, which consists of an offline trajectory optimization to build up a trajectory database, a fast search algorithm to search within this trajectory database, and an online trajectory replanner. Note that the mathematical modeling of the 3D gantry crane is presented in Section 2.1.

4.2.1 Offline trajectory optimization

The general task of a gantry crane is to transport the payload from a starting (initial) state $\mathbf{z}_S$ to a target state $\mathbf{z}_T$ in a minimum time t_F, while respecting the constraints on the state variables and control inputs and avoiding collisions with obstacles. Essentially, the approaches for solving trajectory optimization problems known from the literature can be classified into direct and indirect methods. A core feature of direct methods is formulate the trajectory optimization problem as a nonlinear program. Thus, the trajectory of the 3D gantry crane is discretized in time with $N+1$ grid points, the so called collocation points. By using trapezoidal direct collocation, see, e.g., [69], [70], the system dynamics (2.13) serve as nonlinear constraints (4.1b) and the nonlinear optimization problem for offline trajectory planning is formulated as

$$\min_{\boldsymbol{\xi}} \quad t_F + \frac{1}{2}h \sum_{k=1}^{N-1} \sum_{i=1}^{m} \varphi_i(\mathbf{q}_k) \tag{4.1a}$$

$$\text{s.t.} \quad \mathbf{z}_{k+1} - \mathbf{z}_k = \frac{1}{2}h(\mathbf{f}_k + \mathbf{f}_{k+1}) \tag{4.1b}$$

$$\mathbf{z}_0 = \mathbf{z}_S, \quad \mathbf{z}_N = \mathbf{z}_T \tag{4.1c}$$

$$\underline{\mathbf{z}} \leq \mathbf{z}_k \leq \overline{\mathbf{z}}, \; k = 0, ..., N \tag{4.1d}$$

$$\underline{\mathbf{u}} \leq \mathbf{u}_k \leq \overline{\mathbf{u}}, \; k = 0, ..., N, \tag{4.1e}$$

where (4.1a) is the objective function with the final time t_F and the time step $h = t_F/N$. Moreover, (4.1c) refers to the desired starting and target state, and (4.1d), (4.1e) reflect the state and input constraints. Additionally, $\underline{\mathbf{z}}$, $\overline{\mathbf{z}}$, $\underline{\mathbf{u}}$, and $\overline{\mathbf{u}}$ denote the lower and upper bounds of the state variables $\mathbf{z}_k^{\mathrm{T}} = [\mathbf{q}_k^{\mathrm{T}}, \dot{\mathbf{q}}_k^{\mathrm{T}}]$ and control input $\mathbf{u}_k$, $k = 0, ..., N$. Note that the sway of the payload can be kept small by adjusting the admissible range for α_k and β_k in (4.1d). The index k in (4.1) refers to the discrete-time step $t = kh$. Henceforth, $\boldsymbol{\xi}$ contains all optimization variables

$$\boldsymbol{\xi}^{\mathrm{T}} = [t_F, \mathbf{z}_0^{\mathrm{T}}, ..., \mathbf{z}_N^{\mathrm{T}}, \mathbf{u}_0^{\mathrm{T}}, ..., \mathbf{u}_N^{\mathrm{T}}] \in \mathbb{R}^{1+13(N+1)}. \tag{4.2}$$

The expression $\varphi_i(\mathbf{q}_k)$ in (4.1a) is an artificial potential function, which is formulated in a convex and smooth way and is evaluated at the collocation points $\mathbf{q}_k$, $k = 0, ..., N-1$, for avoiding collisions with obstacles $i = 1, ..., m$, in the operating range of the 3D gantry crane. The detailed formulation of the artificial potential

function is given in Section 4.2.2.

The interior point method (IPM) solver from the open source package Interior Point OPTimize (IPOPT) [71] is used to solve the optimization problem (4.1). Automatic differentiation (AD) is applied to compute the analytical gradients and the Hessian functions of the objective function (4.1a) and the constraint function (4.1b).

4.2.2 Collision avoidance

In this section, the artificial potential functions φ_i, $i = 1, ..., m$, are introduced. As illustrated in Fig. 4.1, obstacles are considered to be bounded by boxes with the parameter vector $\mathbf{p}_i = [w_i, h_i, d_i]^{\mathrm{T}}$ containing the width w_i, the height h_i, and the depth d_i of the box along the x-, y-, and z-axis in the box frame $\mathcal{O}_i$ of the i-th box, $i = 1, ..., m$. Furthermore, the location of the obstacles is known with the translation vector $\mathbf{T}_i$ and the rotation matrix $\mathbf{R}_i$ w.r.t. the world frame $\mathcal{W}$. Recalling (2.17), the position of the center of mass (CoM) of the payload described in the world frame $\mathcal{W}$ is denoted by $\mathbf{r}$ and is associated with a trajectory point $\mathbf{q}^{\mathrm{T}} = [s_x, s_y, s_z, \alpha, \beta]$ in the form

$$\mathbf{r}(\mathbf{q}) = \begin{bmatrix} s_x + \sin(\beta)\cos(\alpha)s_z - \sin(\beta)h_1 \\ s_y - \sin(\alpha)s_z - b_1 \\ \cos(\beta)\cos(\alpha)s_z - \cos(\beta)h_1 \end{bmatrix}, \tag{4.3}$$

with the parameters h_1 and b_1 depicted in Fig. 2.2. This CoM position can be expressed in the i-th box frame $\mathcal{O}_i$ as

$$\mathbf{r}_{\mathcal{O}_i} = \mathbf{R}_i^{\mathrm{T}} \Big(\mathbf{r} - \mathbf{T}_i \Big) . \tag{4.4}$$

A point $\mathbf{q}$ on the trajectory is considered as obstacle-free if and only if the condition

$$\bar{S}_i(\mathbf{q}) = \min(\Delta p_{i,j})_{j=1,2,3} < 0 \tag{4.5}$$

is satisfied, where $\Delta p_{i,j}(\mathbf{q})$ is the j-th component of the vector

$$\mathbf{\Delta p}_i = \Big(\mathbf{r}_{\mathcal{O}_i} \Big) \circ \Big(\mathbf{p}_i - \mathbf{r}_{\mathcal{O}_i} \Big). \tag{4.6}$$

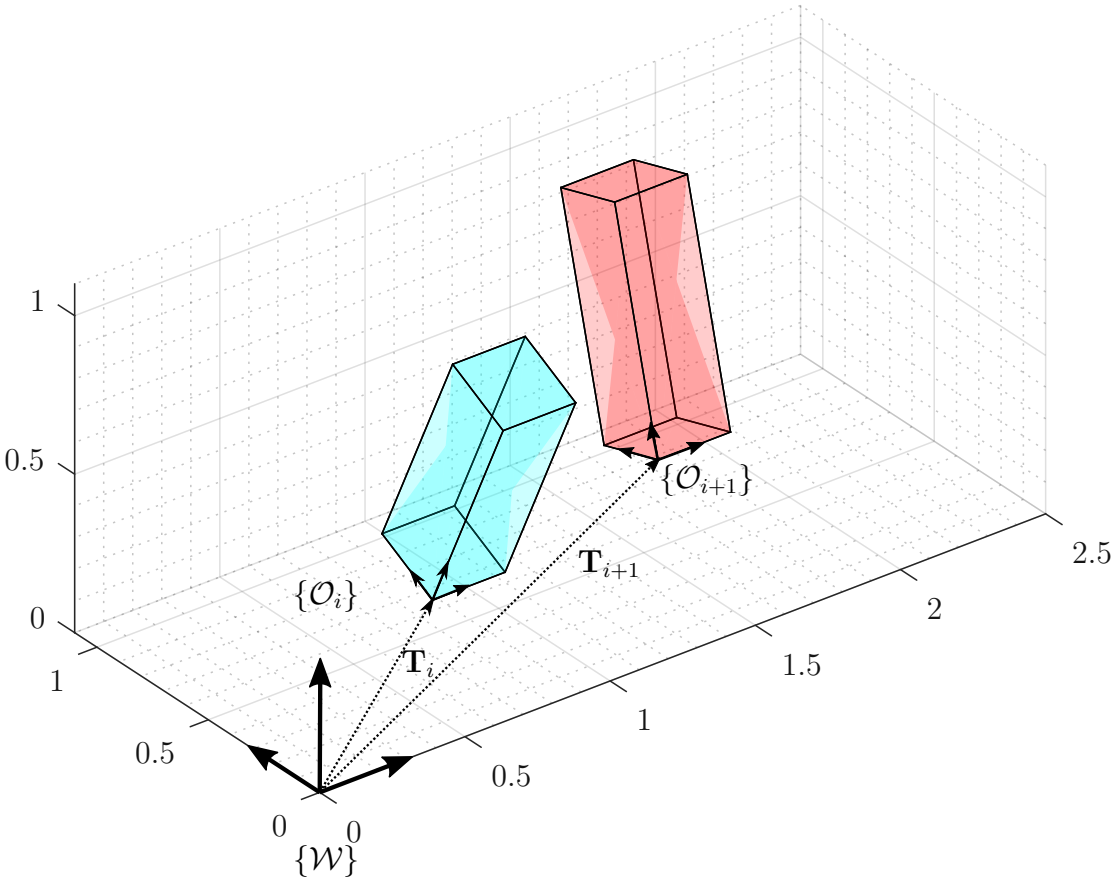


Figure 4.1: Illustration of the obstacles in the working space and their coordinate frames.

The operator $\circ$ in (4.6) refers to the element-wise product. The artificial potential function is defined in the form

$$\bar{\varphi}_i(\mathbf{q}) = \max(\gamma_i \bar{S}_i(\mathbf{q}), 0), \tag{4.7}$$

where $\gamma_i > 0, i = 1, ..., m$, is a user-defined scaling parameter. In order to render the potential function (4.7) with (4.5) sufficiently smooth, the LogSumExp function is employed, see, e.g., [72, 73], resulting in

$$S_i(\mathbf{q}) = \frac{1}{\eta_1} \log \left(\sum_{j=1}^{3} \mathrm{e}^{\eta_1 \Delta p_{i,j}} \right) \tag{4.8a}$$

$$\varphi_i(\mathbf{q}) = \frac{1}{\eta_2} \log\left(1 + \mathrm{e}^{\eta_2 \gamma_i S_i(\mathbf{q})}\right), \tag{4.8b}$$

with the so-called softness coefficients $\eta_1 < 0$ and $\eta_2 > 0$. Additionally, the analytic gradient of the potential function $\varphi_i(\mathbf{q})$ reads as

$$\frac{\partial \varphi_i(\mathbf{q})}{\partial \mathbf{q}} = \gamma_i \frac{\mathrm{e}^{\gamma_i \eta_2 S_i(\mathbf{q})}}{1 + \mathrm{e}^{\gamma_i \eta_2 S_i(\mathbf{q})}} \frac{\partial S_i(\mathbf{q})}{\partial \mathbf{q}} , \tag{4.9}$$

with

$$\frac{\partial S_i(\mathbf{q})}{\partial \mathbf{q}} = \sum_{j=1}^{3} \frac{\mathrm{e}^{\eta_1 \Delta p_{i,j}(\mathbf{q})}}{\sum_{j=1}^{3} \mathrm{e}^{\eta_1 \Delta p_{i,j}(\mathbf{q})}} \frac{\partial \Delta p_{i,j}(\mathbf{q})}{\partial \mathbf{q}} . \tag{4.10}$$

In order to create a safety margin around the obstacles, the margin $\boldsymbol{\delta} = [\delta_x, \delta_y, \delta_z]^{\mathrm{T}}$ is added in the form $\mathbf{p}_i = [w_i + \delta_x/2, h_i + \delta_y/2, d_i + \delta_z/2]^{\mathrm{T}}$ and $\mathbf{T}_i$ is replaced by $\mathbf{T}_i - \boldsymbol{\delta}/2$. Finally, the artificial potential functions φ_i of all obstacles $i = 1, ..., m$ are combined by simply forming the sum $\sum_{i=1}^{m} \varphi_i$. Note that the proposed obstacle avoidance does not directly consider the collision with the ropes. However, in the optimization problem (4.1), the two sway angles α and β are restricted to a small range of $\pm$ 0.05 rad ($\approx \pm 3°$). This together with the safety margin reduce the risk of rope collisions with the obstacles.

4.2.3 Trajectory database

Although the offline trajectory planner in Section 4.2.1 is able to compute the optimal trajectory very quickly, with an average time of 50 ms for one trajectory with the desired convergence tolerance of 10^{-8}, this computation time is still not sufficient for the considered real-time application. A gantry crane often performs repetitive tasks, which means that many similar trajectories have to be tracked during a work shift. Therefore, it suggests itself to reuse the previous trajectories in the form of an offline trajectory database. Since (4.1) only leads to locally optimal solutions and the obstacle potential functions φ_i, $i = 1, ..., m$, are represented as soft constraints in the objective function (4.1a), the solution of (4.1) might be trapped in a local minimum and may violate the obstacle constraint. In this case, (4.1a) is recomputed with a different random initial guess. In this way, we ensure that all trajectories in the offline database are collision-free and dynamically feasible. Note that the average computation time of 50 ms for one trajectory in

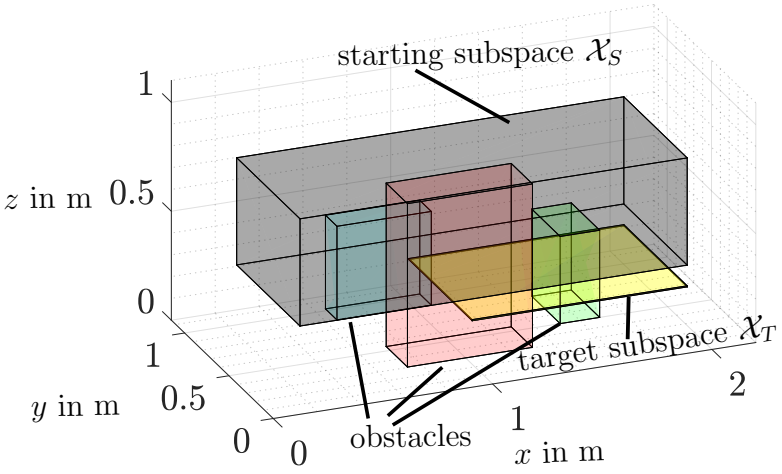


Figure 4.2: Illustration of the two subspaces $\mathcal{X}_S$ and $\mathcal{X}_T$.

the offline database consisting of 10^4 optimal trajectories includes the violation checking and recomputation.

Without loss of generality, the starting subspace $\mathcal{X}_S$ is assumed to cover the entire workspace, while the target subspace $\mathcal{X}_T$ covers the workspace of a moving target, in our case a moving truck on the ground which is represented by a plane parallel to the bottom of $\mathcal{X}_S$, as shown in Fig. 4.2. For each subspace, a grid with equal spacing is chosen. Then, the offline trajectory planner is used to plan offline trajectories from each of the n_S grid points of the starting subspace $\mathcal{X}_S$ (starting state $\mathbf{z}_S^{\mathrm{T}} = [\mathbf{q}_S^{\mathrm{T}}, \mathbf{0}]$ in (4.1c)) to each of the n_T grid points in the target subspace $\mathcal{X}_T$ (target state $\mathbf{z}_T^{\mathrm{T}} = [\mathbf{q}_T^{\mathrm{T}}, \mathbf{0}]$ in (4.1c)), in total $n_S n_T$ trajectories. The forward kinematics (4.3) computes the position of the center of mass $\mathbf{r}(\mathbf{q})$ of the payload based on the five degrees of freedom $\mathbf{q}$. In addition, each trajectory in the database is represented by two labels containing the position $\mathbf{r}_S = \mathbf{r}(\mathbf{q}_S)$ in the Cartesian space of the starting state $\mathbf{z}_S$ and the position $\mathbf{r}_T = \mathbf{r}(\mathbf{q}_T)$ in the Cartesian space of the target state $\mathbf{z}_T$.

In order to efficiently search for the nearest trajectory, an offline database is constructed using the two labels for each trajectory according to the k-d tree algorithm [74], which is a well-known space partitioning data structure for partitioning and organizing points. The search complexity of this algorithm for N labels in the database is $\mathcal{O}(\log N)$ compared to $\mathcal{O}(N)$ for an unprocessed database, see, e.g., [75].

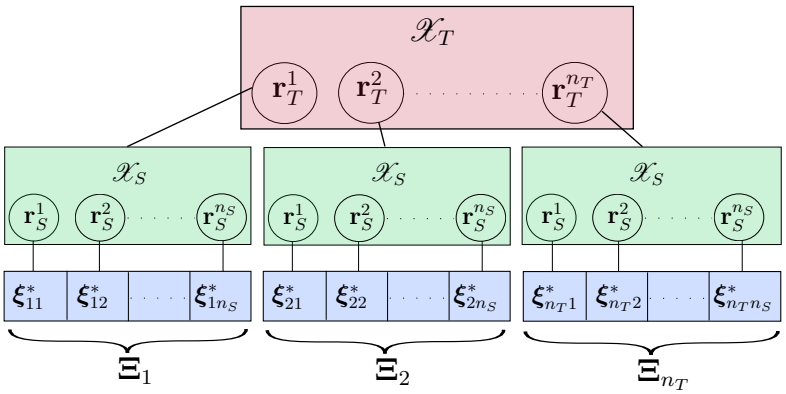


Figure 4.3: Offline database structure.

The offline database structure is shown in Fig. 4.3, where the labels in $\mathscr{X}_T$ are denoted by

$$\mathscr{X}_T = \{\mathbf{r}_T^1, \mathbf{r}_T^2, \ldots, \mathbf{r}_T^{n_T}\}$$

and the labels in $\mathscr{X}_S$ by

$$\mathscr{X}_S = \{\mathbf{r}_S^1, \mathbf{r}_S^2, \ldots, \mathbf{r}_S^{n_S}\}.$$

The third layer of the offline database contains the time-optimal offline trajectories $\boldsymbol{\Xi}_i = [\boldsymbol{\xi}_{ij}^*] \in \mathbb{R}^{[1+13(N+1)]\times n_S}$ with $i = 1, \ldots, n_T$ and $j = 1, \ldots, n_S$ connecting the Cartesian positions $\mathbf{r}_S^i$ and $\mathbf{r}_T^j$, see (4.1) and (4.2). The details of the search algorithm are presented in the next subsection.

4.2.4 Online trajectory replanner

The online trajectory replanner computes the optimal trajectory according to the following procedure. After receiving the command to start the motion from the starting state $\tilde{\mathbf{z}}_S$ to the target state $\tilde{\mathbf{z}}_T$, the online trajectory replanner first searches the trajectory database to find the closest trajectory. Later in this subsection, we will thoroughly explain what we exactly mean with the term closest. For this, we will specify the metrics used for measuring distances. Then, linear constrained quadratic programming is applied to minimize the deviation from the offline trajectory while satisfying the dynamic constraints of the 3D gantry crane. To achieve fast trajectory replanning in real time, a computationally efficient algorithm for the search task is summarized in Alg. 4 and presented next.

In the first step of Alg. 4, the target position and the target state are predicted for

Algorithm 4: Retrieving offline trajectory in database

```
1  Function FastRetrievingOfflineTrajectory(r_T^curr, v_T^curr, z_S^curr):
2      r̂_T, ẑ_T ← TargetStatePrediction(r_T^curr, v_T^curr, T_s,2)
3      r̂_S, ẑ_S ← StartingStatePrediction(z_S^curr, T_s,2)
4      i ← knnsearch(X_T, r̂_T)
5      if IsTargetStationary(r̂_T, r̂_T^prev) then
6          if IsTheFirstSearch then
7              j ← knnsearch(X_S, r̂_S)
8              ξ* = ξ*_ij
9          end
10     else
11         (ξ*_ij, l) = knnsearch(Ξ_i, ẑ_S)
12         ξ* = TrajectoryRefinement(ξ*_ij, l)
13     end
14     return ξ*
```

one sampling time $T_{s,2}$ of the trajectory replanner denoted by $\hat{\mathbf{r}}_T$ and $\hat{\mathbf{z}}_T$, based on the current target position $\mathbf{r}_T^{\text{curr}}$ and velocity $\mathbf{v}_T^{\text{curr}}$. For this prediction, it is assumed that the target velocity $\mathbf{v}_T^{\text{curr}}$ remains constant for the sampling time $T_{s,2}$. Similarly, in line 3 of Alg. 4, the predicted starting position $\hat{\mathbf{r}}_S$ and the predicted state $\hat{\mathbf{z}}_S$ are estimated using the current starting state $\mathbf{z}_S^{\text{curr}}$ measured by encoders. Once the predicted target state $\hat{\mathbf{r}}_T$ is obtained, the search algorithm k nearest neighbor (k-nn) [76] is employed to find the index i of the offline target state $\mathbf{r}_T^i, i = 1, ..., n_T$, in $\mathcal{X}_T$ that is closest in the sense of the smallest Euclidean distance $||\hat{\mathbf{r}}_T - \mathbf{r}_T^i||_2$ to the predicted target position $\hat{\mathbf{r}}_T$, see line 4 of Algorithm 4. By comparing the predicted state $\hat{\mathbf{r}}_T$ with the previous predicted state $\hat{\mathbf{r}}_T^{\text{prev}}$, the search algorithm can detect whether the target is moving or stationary. At this point, the search algorithm distinguishes between two cases:

- If the target state is stationary and the trajectory replanning algorithm was not executed before, the k-nn search is applied to find the index j of the closest starting position to $\hat{\mathbf{r}}_S$, see line 7 in Algorithm 4 and Fig. 4.3. Using the two indexes i, j of the label in $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively, the closest trajectory $\boldsymbol{\xi}_{ij}^*$ is directly taken from the database.
- If the target moves, the k-nn search must be executed in the trajectory set $\boldsymbol{\Xi}_i$ (line 11 of Algorithm 4). In this case, the predicted starting point is

normally not a stationary point, but $\hat{\mathbf{z}}_S^{\mathrm{T}} = [\hat{\mathbf{q}}_S^{\mathrm{T}}, \dot{\hat{\mathbf{q}}}_S^{\mathrm{T}}]$, with $\dot{\hat{\mathbf{q}}}_S \neq \mathbf{0}$. Thus, the closest starting state in the trajectory database is obtained from the trajectory set $\boldsymbol{\Xi}_i$ by choosing those trajectory $\boldsymbol{\xi}_{ij}^*, j = 1, ..., n_S$, which has a collocation point $(\mathbf{z}_{ij,l}^*)^{\mathrm{T}} = [(\mathbf{q}_{ij,l}^*)^{\mathrm{T}}, (\dot{\mathbf{q}}_{ij,l}^*)^{\mathrm{T}}]$, see (4.2), that is closest to $\hat{\mathbf{z}}_S$ in the weighted Euclidean distance metric

$$||\mathbf{q}_{ij,l}^* - \hat{\mathbf{q}}_S||_2 + ||\mathrm{diag}(\rho_n, n = 1, ..., 5)(\dot{\mathbf{q}}_{ij,l}^* - \dot{\hat{\mathbf{q}}}_S)||_2,$$

with the user-defined weighting parameter $\rho_n > 0, n = 1, ..., 5$. Since the range of the system state $\underline{\mathbf{q}} \leq \mathbf{q} \leq \overline{\mathbf{q}}$ and of the system state velocity $\underline{\dot{\mathbf{q}}} \leq \mathbf{q} \leq \overline{\dot{\mathbf{q}}}$ are different, ρ_n is chosen to normalize the velocity error with respect to the state error in the form $\rho_n = \dfrac{\overline{q_n} - \underline{q_n}}{\overline{\dot{q}_n} - \underline{\dot{q}_n}}, n = 1, ..., 5$. Note that the sway angles α, β and the angular velocities $\dot{\alpha}$, $\dot{\beta}$ are also considered in this case, since the algorithm considers the whole state $\hat{\mathbf{z}}_S$. There are two return variables j and l of the k-nn search in line 11 of Algorithm 4, where j is the index of the retrieved trajectory $\boldsymbol{\xi}_{ij}^*$ in $\boldsymbol{\Xi}_i$ and l is the index of the closest state $\mathbf{z}_l^*$ of this offline trajectory $\boldsymbol{\xi}_{ij}^*$. At this point, the closest trajectory from the offline database reads as

$$(\boldsymbol{\xi}^*)^{\mathrm{T}} = [\frac{l}{N} t_F^*, (\mathbf{z}_l^*)^{\mathrm{T}}, \ldots, (\mathbf{z}_N^*)^{\mathrm{T}}, (\mathbf{u}_l^*)^{\mathrm{T}}, \ldots, (\mathbf{u}_N^*)^{\mathrm{T}}]. \tag{4.11}$$

Note that only a segment of the whole offline trajectory $\boldsymbol{\xi}_{ij}^*$ is used in (4.11) and therefore the number of grid points of $(\boldsymbol{\xi}^*)$ is reduced to $N - l + 1$. Since the number of $(N + 1)$ grid points is fixed during the computation of the online trajectory replanner, an interpolation scheme has to be employed in line 12 of Algorithm 4. Between two adjacent collocation points k and $k + 1$, with $k = l, ..., N-1$, the input $\mathbf{u}^*(t)$ and the state $\mathbf{z}^*(t)$ for $t = [kh^*, (k+1)h^*]$, are interpolated as linear and quadratic splines, respectively, i.e.,

$$\mathbf{z}^*(t) \approx \mathbf{z}_k^* + (t - kh^*)\mathbf{f}_k^* + \frac{(t - kh^*)^2}{2h^*}(\mathbf{f}_{k+1}^* - \mathbf{f}_k^*), \tag{4.12a}$$

$$\mathbf{u}^*(t) \approx \mathbf{u}_k^* + \frac{t - kh^*}{h^*}(\mathbf{u}_{k+1}^* - \mathbf{u}_k^*), \tag{4.12b}$$

for $k = l, ..., N - 1$, with $h^* = t_F^*/N$.

For brevity, the same notation as in (4.2) is used after the refinement of the closest

trajectory (4.11). To this end, the trajectory is expressed as

$$(\boldsymbol{\xi}^*)^{\mathrm{T}} = [t_F^*, (\mathbf{z}_0^*)^{\mathrm{T}}, \ldots, (\mathbf{z}_N^*)^{\mathrm{T}}, (\mathbf{u}_0^*)^{\mathrm{T}}, \ldots, (\mathbf{u}_N^*)^{\mathrm{T}}]. \tag{4.13}$$

Recalling the system dynamics condition (4.1b) for the closest offline trajectory, the relation

$$\mathbf{z}_{k+1}^* = \mathbf{z}_k^* + \frac{t_F^*}{2N}(\mathbf{f}_k^* + \mathbf{f}_{k+1}^*) \tag{4.14}$$

holds.

In the following, the online trajectory replanner is explained in detail. Assuming that the number of grid points in the starting and target subspace is sufficiently dense, only small deviations

$$\delta\boldsymbol{\xi}^{\mathrm{T}} = [\delta t_F, (\delta\mathbf{z}_0)^{\mathrm{T}}, ..., (\delta\mathbf{z}_N)^{\mathrm{T}}, (\delta\mathbf{u}_0)^{\mathrm{T}}, ..., (\delta\mathbf{u}_N)^{\mathrm{T}}]$$

need to be taken into account to compute the new trajectory connecting the predicted starting state $\hat{\mathbf{z}}_S$ with the predicted target state $\hat{\mathbf{z}}_T$. The first-order linearization of the discrete-time system dynamics (4.1b) w.r.t. the closest database trajectory $\boldsymbol{\xi}^*$ reads as

$$\begin{aligned}\mathbf{z}_{k+1}^* + \delta\mathbf{z}_{k+1} = \mathbf{z}_k^* + \delta\mathbf{z}_k + \frac{t_F^* + \delta t_F}{2N}\Bigg(&\mathbf{f}_k^* + \boldsymbol{\Gamma}_k^{\mathbf{z}}\delta\mathbf{z}_k + \boldsymbol{\Gamma}_k^{\mathbf{u}}\delta\mathbf{u}_k \\ &+ \mathbf{f}_{k+1}^* + \boldsymbol{\Gamma}_{k+1}^{\mathbf{z}}\delta\mathbf{z}_{k+1} + \boldsymbol{\Gamma}_{k+1}^{\mathbf{u}}\delta\mathbf{u}_{k+1}\Bigg),\end{aligned} \tag{4.15}$$

with $\delta\mathbf{z}_k = \mathbf{z}_k - \mathbf{z}_k^*$, $\delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^*$, $\delta t_F = t_F - t_F^*$, and

$$\boldsymbol{\Gamma}_k^{\mathbf{z}} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{z}}\right|_{\mathbf{z}_k^*,\mathbf{u}_k^*}, \quad \boldsymbol{\Gamma}_k^{\mathbf{u}} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{\mathbf{z}_k^*,\mathbf{u}_k^*}$$

for $k = 0, ..., N-1$. Note that the state deviations at $k = 0$ and $k = N$ are fixed by

$$\delta\mathbf{z}_0 = \hat{\mathbf{z}}_S - \mathbf{z}_0^* \tag{4.16a}$$

$$\delta\mathbf{z}_N = \hat{\mathbf{z}}_T - \mathbf{z}_N^* \tag{4.16b}$$

due to the predicted starting and target state $\hat{\mathbf{z}}_S$ and $\hat{\mathbf{z}}_T$. Inserting (4.14) into

(4.15) and neglecting the terms containing a product of deviation variables, the constraint function reduces to

$$\delta\mathbf{z}_{k+1} = \delta\mathbf{z}_k + \frac{t_F^*}{2N}\left(\boldsymbol{\Gamma}_k^{\mathbf{z}}\delta\mathbf{z}_k + \boldsymbol{\Gamma}_k^{\mathbf{u}}\delta\mathbf{u}_k + \boldsymbol{\Gamma}_{k+1}^{\mathbf{z}}\delta\mathbf{z}_{k+1} + \boldsymbol{\Gamma}_{k+1}^{\mathbf{u}}\delta\mathbf{u}_{k+1}\right) + \frac{\delta t_F}{t_F^*}(\mathbf{z}_{k+1}^* - \mathbf{z}_k^*). \tag{4.17}$$

In a more compact form, (4.17) is rewritten as

$$\mathbf{C}_{k+1}\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k, \tag{4.18}$$

where

$$\mathbf{C}_{k+1} = \begin{bmatrix} \mathbf{I} - \frac{h^*}{2}\boldsymbol{\Gamma}_{k+1}^{\mathbf{z}} & -\frac{h^*}{2}\boldsymbol{\Gamma}_{k+1}^{\mathbf{u}} & 0 \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix},$$

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{I} + \frac{h^*}{2}\boldsymbol{\Gamma}_k^{\mathbf{z}} & \frac{h^*}{2}\boldsymbol{\Gamma}_k^{\mathbf{u}} & \frac{\mathbf{z}_{k+1}^* - \mathbf{z}_k^*}{t_F^*} \\ \mathbf{0} & \mathbf{0} & 1 \end{bmatrix},$$

$$\mathbf{x}_k = \begin{bmatrix} \delta\mathbf{z}_k \\ \delta\mathbf{u}_k \\ \delta t_{F,k} \end{bmatrix},$$

and $h^* = \frac{t_F^*}{N}$. For simplicity, only one variable for the final time δt_F in (4.15) was introduced instead of $\delta t_{F,k}, k = 0, \ldots, N-1$, which is why $\delta t_{F,k+1} = \delta t_{F,k}$. The deviation vector $\boldsymbol{\xi}_k$ is obtained as the solution of a linear constrained quadratic program (LCQP) of the form

$$\min_{\mathbf{x}_k} \quad \frac{1}{2}\sum_{k=1}^{N-1}\mathbf{x}_k^{\mathrm{T}}\mathbf{Q}_k\mathbf{x}_k \tag{4.19a}$$

$$\text{s.t.} \quad \mathbf{C}_{k+1}\mathbf{x}_{k+1} = \mathbf{A}_k\mathbf{x}_k, \qquad k = 0, ..., N-1 \tag{4.19b}$$

$$\underline{\mathbf{x}_k} \leq \mathbf{x}_k \leq \overline{\mathbf{x}_k}, \qquad k = 0, ..., N\ , \tag{4.19c}$$

with (4.16) and the positive definite weighting matrix

$$\mathbf{Q}_k = \mathrm{diag}(\mathbf{Q}_{\mathbf{z}_k}, \mathbf{Q}_{\mathbf{u}_k}, Q_{t_F}). \tag{4.20}$$

With the choice of $Q_{t_F} > 0$ and the positive definite submatrices $\mathbf{Q}_{\mathbf{z}_k}$ and $\mathbf{Q}_{\mathbf{u}_k}$, the deviation of the online trajectory from the selected database trajectory (4.13) can be specifically weighted in the objective function (4.19a) w.r.t. the travel time t_F, the state $\mathbf{z}_k$, and the control input $\mathbf{u}_k$, respectively. The inequality condition (4.19c) corresponds to (4.1d) and (4.1e), where

$$\begin{aligned}\underline{\mathbf{x}_k}^{\mathrm{T}} &= [\underline{\mathbf{z}_k}^{\mathrm{T}} - (\mathbf{z}_k^*)^{\mathrm{T}}, \underline{\mathbf{u}_k}^{\mathrm{T}} - (\mathbf{u}_k^*)^{\mathrm{T}}, \underline{\delta t_F}],\\ \overline{\mathbf{x}_k}^{\mathrm{T}} &= [\overline{\mathbf{z}_k}^{\mathrm{T}} - (\mathbf{z}_k^*)^{\mathrm{T}}, \overline{\mathbf{u}_k}^{\mathrm{T}} - (\mathbf{u}_k^*)^{\mathrm{T}}, \overline{\delta t_F}],\end{aligned}$$

for $k = 1, \ldots, N-1$, and $\overline{\delta t_F}$ and $\underline{\delta t_F}$ is a sufficiently large upper and lower bound for δt_F, respectively. Note that the equality constraints in (4.16) must be taken into account by taking the same values for the lower and upper bounds of the corresponding variables in the form

$$\begin{aligned}\underline{\mathbf{x}_0}^{\mathrm{T}} &= [\delta\mathbf{z}_0^{\mathrm{T}}, \delta\mathbf{u}_0^{\mathrm{T}}, \underline{\delta t_F}], \;\; \overline{\mathbf{x}_0}^{\mathrm{T}} = [\delta\mathbf{z}_0^{\mathrm{T}}, \delta\mathbf{u}_0^{\mathrm{T}}, \overline{\delta t_F}] \;,\\ \underline{\mathbf{x}_N}^{\mathrm{T}} &= [\delta\mathbf{z}_N^{\mathrm{T}}, \delta\mathbf{u}_N^{\mathrm{T}}, \underline{\delta t_F}], \;\; \overline{\mathbf{x}_N}^{\mathrm{T}} = [\delta\mathbf{z}_N^{\mathrm{T}}, \delta\mathbf{u}_N^{\mathrm{T}}, \overline{\delta t_F}] \;.\end{aligned}$$

It is important to keep $\delta\mathbf{q}_k$ of $\delta\mathbf{z}_k^{\mathrm{T}} = [\delta\mathbf{q}_k, \delta\dot{\mathbf{q}}_k]$ close to zero if the corresponding collocation points $\mathbf{q}_k^*$ on the selected database trajectory are already close to one of the obstacles. Therefore, the submatrix $\mathbf{Q}_{\mathbf{q}_k}$ of the weighting matrix $\mathbf{Q}_{\mathbf{z}_k} = \mathrm{diag}(\mathbf{Q}_{\mathbf{q}_k}, \mathbf{Q}_{\dot{\mathbf{q}}_k})$ in (4.20) is adjusted depending on the distance of the corresponding payload position $\mathbf{r}(\mathbf{q}_k^*)$ to an obstacle. Note that instead of calculating the exact distance, the artificial potential functions $\varphi_i(\mathbf{q})$, $i = 1, ..., m$, according to (4.8), see also (4.1), serve as a basis to indirectly consider the obstacles in (4.19a). In particular, the Hessian of the artificial potential function is used to adjust the weighting matrix $\mathbf{Q}_{\mathbf{q}_k}$ in the form

$$\mathbf{Q}_{\mathbf{q}_k} = \mathbf{Q}_{\mathbf{q}} + \lambda \frac{\partial^2 \left(\sum_{i=1}^{m} \varphi_{i,k}(\mathbf{q}_k) \right)}{\partial \mathbf{q}_k^2} \bigg|_{\mathbf{q}_k^*}, \tag{4.21}$$

with the constant matrix $\mathbf{Q}_{\mathbf{q}} > 0$ and the tuning parameter $\lambda > 0$. Clearly, the closer a collocation point $\mathbf{q}_k^*$ of the database trajectory is to an obstacle, the larger is the corresponding entry in the weighting matrix $\mathbf{Q}_{\mathbf{q}_k}$. This in turn makes the deviation of the online trajectory, which is a solution of (4.19), from the database trajectory in terms of $\delta\mathbf{q}_k$ at the considered point $\mathbf{q}_k^*$ small. This adaption of

the weighting matrix $\mathbf{Q}_{\mathbf{q}_k}$, see (4.21), is introduced to ensure that also the online replanned trajectory is collision-free. Finally, the optimal trajectory of the online trajectory replanner $\boldsymbol{\xi}^{*,onl}$ reads as

$$\boldsymbol{\xi}^{*,onl} = \boldsymbol{\xi}^{*} + \delta\boldsymbol{\xi}^{*}, \tag{4.22}$$

where $\delta\boldsymbol{\xi}^{*}$ results from the solution of (4.19) in the form

$$(\delta\boldsymbol{\xi}^{*})^{\mathrm{T}} = [\delta t_F^{*}, (\delta\mathbf{z}_0^{*})^{\mathrm{T}}, ..., (\delta\mathbf{z}_N^{*})^{\mathrm{T}}, (\delta\mathbf{u}_0^{*})^{\mathrm{T}}, ..., (\delta\mathbf{u}_N^{*})^{\mathrm{T}}].$$

Since the LCQP (4.19) only leads to locally optimal solutions, it may get stuck in a local minimum that violates the obstacle constraint. Thus, if the solution of (4.19) gets stuck in a local minimum, (4.19) is reprocessed with a different nearest offline trajectory resulting from the solution of the k-nearest neighbor search in Alg. 4. This heuristic modification is used in the experiment as a safety measure to protect the laboratory equipment. In Tab. 4.1, these heuristic measures are not included in the Monte Carlo simulations to show the performance of the proposed algorithm without any heuristic modifications.

4.3 Trajectory tracking controller design

The control structure of the 3D gantry crane consists of the online trajectory replanner, the trajectory tracking controller, and a friction compensation. The overall control structure is depicted in Fig. 4.4. To stabilize the payload around a given trajectory, a combination of a feedforward and a feedback controller, also denoted as trajectory tracking controller, is introduced. The control output of the online trajectory replanner is used as feedforward part, while the feedback controller has a cascaded structure consisting of the outer model predictive control (MPC) and the inner velocity controller. The details of the friction compensation are discussed in [27] and the velocity controller is a decoupled standard PI controller. In this section, we focus on the model predictive controller.

As a first step, in order to reduce the complexity of the system dynamics and further increase the computational speed, the state $\mathbf{q}$ is divided into the state of the actuated subsystem $\mathbf{q}_A = [s_x, s_y, s_z]^{\mathrm{T}}$ and the unactuated subsystem $\mathbf{q}_U = [\alpha, \beta]^{\mathrm{T}}$.

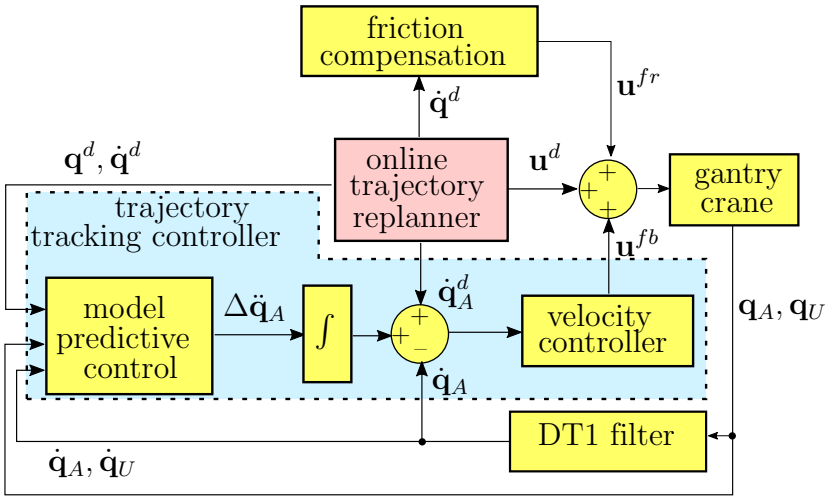


Figure 4.4: Block diagram of the control structure. The yellow blocks are processed with the fast sampling time $T_{s,1}$ while the red block is computed with the slower sampling time $T_{s,2}$, see also Fig. 4.9.

Therefore, the system dynamics (2.13) is expressed as

$$\begin{bmatrix} \mathbf{M}_A & \mathbf{M}_{AU} \\ \mathbf{M}_{AU}^{\mathrm{T}} & \mathbf{M}_U \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_A \\ \ddot{\mathbf{q}}_U \end{bmatrix} + \begin{bmatrix} \mathbf{C}_A & \mathbf{C}_{AU} \\ \mathbf{C}_{UA} & \mathbf{C}_U \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \end{bmatrix} + \begin{bmatrix} \mathbf{g}_A \\ \mathbf{g}_U \end{bmatrix} = \begin{bmatrix} \mathbf{u} \\ \mathbf{0} \end{bmatrix}. \tag{4.23}$$

The acceleration of the unactuated angles $\ddot{\mathbf{q}}_U$ and the control input $\mathbf{u}$ are calculated from (4.23), which yields

$$\ddot{\mathbf{q}}_U = \mathbf{M}_U^{-1}(-\mathbf{M}_{AU}^{\mathrm{T}}\ddot{\mathbf{q}}_A - \mathbf{C}_{UA}\dot{\mathbf{q}}_A - \mathbf{C}_U\dot{\mathbf{q}}_U - \mathbf{g}_U), \tag{4.24a}$$

$$\mathbf{u} = \mathbf{M}_A\ddot{\mathbf{q}}_A + \mathbf{M}_{AU}\ddot{\mathbf{q}}_U + \mathbf{C}_A\dot{\mathbf{q}}_A + \mathbf{C}_{AU}\dot{\mathbf{q}}_U + \mathbf{g}_A. \tag{4.24b}$$

It is worth noting that the acceleration of the actuated state $\ddot{\mathbf{q}}_A$ acts directly on the unactuated subsystem. Thus, considering $\mathbf{u}_A = \ddot{\mathbf{q}}_A$ as a new control input to the system, the state-space representation of (2.13) reads as

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_U \\ \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \\ \mathbf{u}_A \\ \mathbf{M}_U^{-1}(-\mathbf{M}_{AU}^{\mathrm{T}}\mathbf{u}_A - \mathbf{C}_{UA}\dot{\mathbf{q}}_A - \mathbf{C}_U\dot{\mathbf{q}}_U - \mathbf{g}_U) \end{bmatrix}. \tag{4.25}$$

In a more compact form, (4.25) is rewritten as

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{z} = \tilde{\mathbf{f}}(\mathbf{z}, \mathbf{u}_A). \tag{4.26}$$

The system (4.26) is linearized around the desired trajectory $(\mathbf{z}^d, \mathbf{u}_A^d) = (\mathbf{z}^* + \delta\mathbf{z}, \ddot{\mathbf{q}}_A^d)$, with

$$(\mathbf{z}^d)^{\mathrm{T}} = [(\mathbf{q}_A^d)^{\mathrm{T}}, (\mathbf{q}_U^d)^{\mathrm{T}}, (\dot{\mathbf{q}}_A^d)^{\mathrm{T}}, (\dot{\mathbf{q}}_U^d)^{\mathrm{T}}] \ ,$$

obtained from the solution $\boldsymbol{\xi}^{*,onl}$ (4.22) of the online trajectory replanner (4.18). By approximating the system dynamics (4.25) around $(\mathbf{z}^d, \mathbf{u}_A^d)$, the discrete time-varying system dynamics is expressed as

$$\Delta\mathbf{z}_{k+1} = \boldsymbol{\Phi}_k \Delta\mathbf{z}_k + \boldsymbol{\Omega}_k \Delta\mathbf{u}_{A,k} \ , \tag{4.27}$$

with

$$\begin{aligned} \Delta\mathbf{z}_k &= \mathbf{z}_k - \mathbf{z}_k^d & \Delta\mathbf{u}_{Ak} &= \mathbf{u}_{A,k} - \mathbf{u}_{A,k}^d \\ \boldsymbol{\Phi}_k &= \mathbf{I}_{10} + T_{s,1}\mathbf{A}_k & \boldsymbol{\Omega}_k &= T_{s,1}\mathbf{B}_k \\ \mathbf{A}_k &= \left.\frac{\partial\tilde{\mathbf{f}}}{\partial\mathbf{z}}\right|_{\mathbf{z}_k^d, \mathbf{u}_{A,k}^d} & \mathbf{B}_k &= \left.\frac{\partial\tilde{\mathbf{f}}}{\partial\mathbf{u}_A}\right|_{\mathbf{z}_k^d, \mathbf{u}_{A,k}^d} \end{aligned} \ , \tag{4.28}$$

and the sampling time $T_{s,1}$. The model predictive control is applied at the time instant t_k to control the error system (4.27) by solving the following optimization problem w.r.t. $\Delta\mathbf{u}_A^{\mathrm{T}} = [\Delta\mathbf{u}_{A,k}^{\mathrm{T}}, \ldots, \Delta\mathbf{u}_{A,k+N_m}^{\mathrm{T}}]$ over a finite horizon of $N_m + 1$ steps

$$\begin{aligned} \min_{\Delta\mathbf{u}_A} \quad & \sum_{j=0}^{N_m} (\Delta\mathbf{z}_{k+j}^{\mathrm{T}} \mathbf{Q}_m \Delta\mathbf{z}_{k+j} + \Delta\mathbf{u}_{A,k+j}^{\mathrm{T}} \mathbf{R}_m \Delta\mathbf{u}_{A,k+j}) \\ & + \Delta\mathbf{z}_{k+N_m+1}^{\mathrm{T}} \mathbf{Q}_f \Delta\mathbf{z}_{k+N_m+1} \end{aligned} \tag{4.29a}$$

$$\text{s.t.} \quad \Delta\mathbf{z}_{k+j+1} = \boldsymbol{\Phi}_k \Delta\mathbf{z}_{k+j} + \boldsymbol{\Omega}_k \Delta\mathbf{u}_{A,k+j} \ , \tag{4.29b}$$

$$\underline{\Delta\mathbf{z}} \leq \Delta\mathbf{z}_{k+j} \leq \overline{\Delta\mathbf{z}}, \ j = 0, ..., N_m, \tag{4.29c}$$

where $\mathbf{Q}_m$, $\mathbf{R}_m$, and $\mathbf{Q}_f$ are positive definite weighting matrices and $\underline{\Delta\mathbf{z}} = \underline{\mathbf{z}} - \mathbf{z}^d$ and $\overline{\Delta\mathbf{z}} = \overline{\mathbf{z}} - \mathbf{z}^d$ is the lower and upper bound of the state variables, respectively. The first element of the solution $\Delta\mathbf{u}_A$ from (4.29), i.e. $\Delta\mathbf{u}_{A,k}$ is then used as control input for the inner velocity controller. Since the model predictive control is able to

specifically constrain the full state $\mathbf{z}_k$, including the sway angles α_k and β_k, the use of model predictive control is advantageous for the performance of the closed-loop system.

4.4 Simulation and experimental results

In this section, simulation and experimental results are presented for two different scenarios. In the first scenario, the 3D gantry crane follows the trajectory from the online replanner connecting two points, with the states $\mathbf{z}_S$ and $\mathbf{z}_T$ in $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively, and the target is not changing during the motion of the crane. The second scenario concerns the case when the target can move freely within $\mathcal{X}_T$. Therefore, the online replanner must actively update the trajectory according to the new position of the moving target and its current speed. To prove the efficiency of the proposed combined method of fast trajectory optimization and trajectory tracking control, it is assumed that the map containing all obstacles is known, i.e. the obstacles are static. This assumption is justified for many practical scenarios.

4.4.1 Simulation results

The offline trajectory optimization and the offline database are both developed in MATLAB/SIMULINK 2020b on a computer with 3.8 GHz Intel Core i7 and 32 GB RAM. The open-source package Interior Point OPTimize (IPOPT) was employed to solve the nonlinear optimization problem (4.1), see, e.g., [71]. In IPOPT, the multifrontal linear solver (MA57) was used to increase the computational speed. The analytical gradients of the cost function and the constraint functions are computed using CasADi, see, e.g., [77]. In addition, the numerical Hessian is evaluated using the BFGS approximation method, see, e.g., [78].

Each time-optimal obstacle-free trajectory in the offline database is discretized with 26 grid points, resulting in 339 optimization variables. Since the solution of the direct collocation optimization is interpolated between two neighboring grid points, the accuracy of the resulting trajectory depends on the number of grid points. However, there is a trade-off between the number of grid points and the computational speed. Because the workspace is approximately a cuboid of size $2\,\mathrm{m} \times 1\,\mathrm{m} \times 0.55\,\mathrm{m}$, the average travel time from the initial location to the most distant target location is approximately 9 s. Thus, the solution accuracy of direct

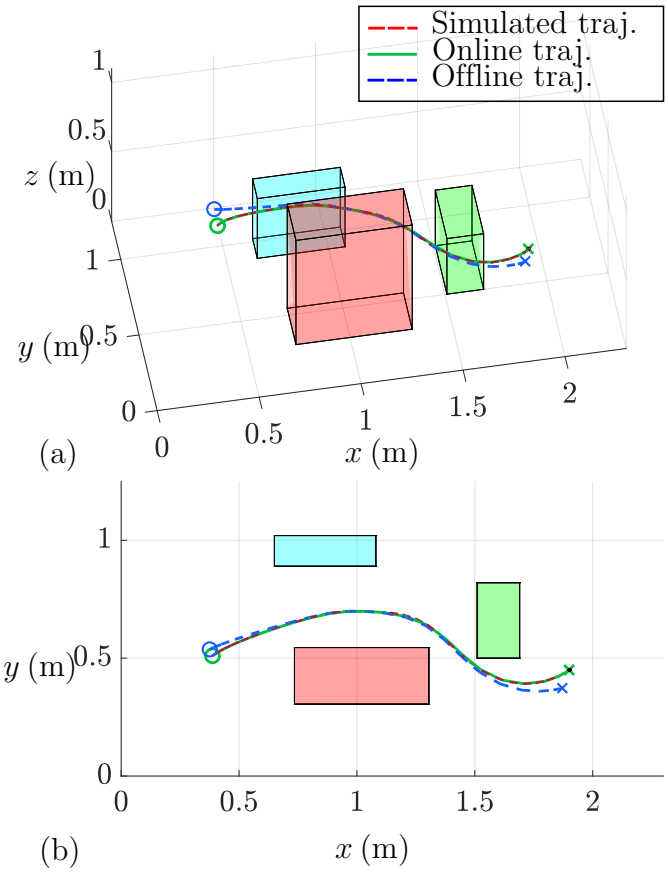


Figure 4.5: Collision-free trajectories in the stationary target scenario: closest trajectory from the offline database, online replanned trajectory, and simulated trajectory using the trajectory tracking controller. The circle and the cross symbol constitute the starting and the target point, respectively. (a) Trajectories in 3D space, (b) Trajectories in the xy-plane.

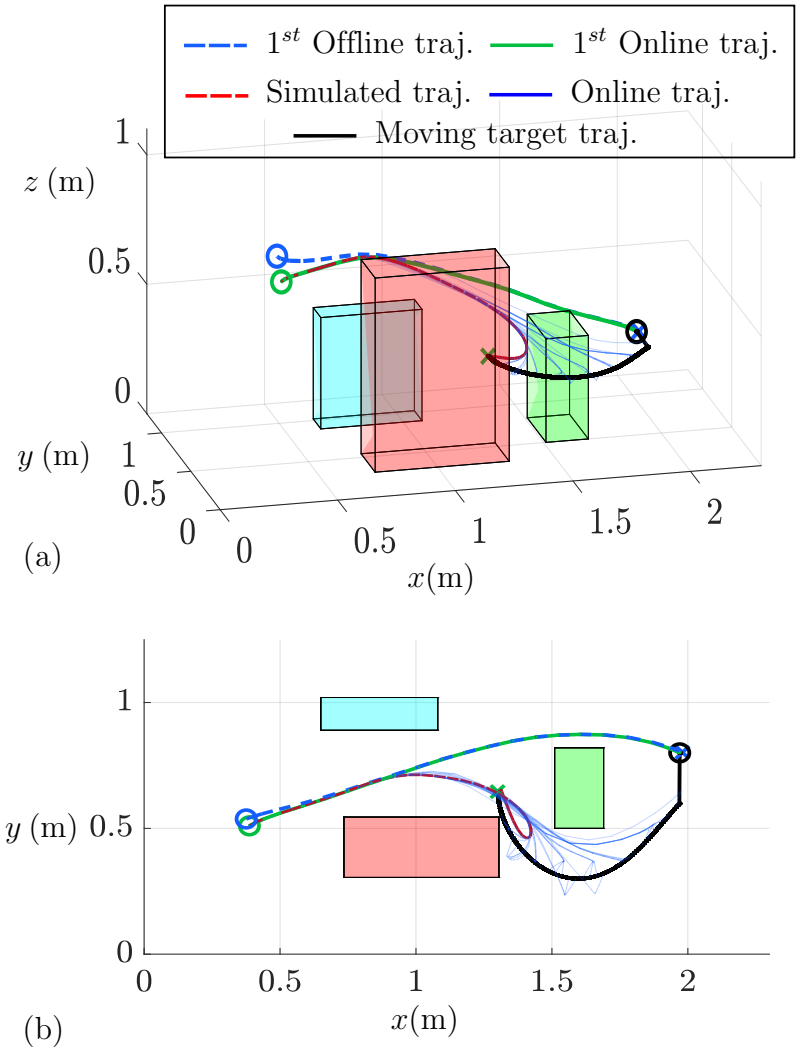


Figure 4.6: The collision-free offline path and the online paths resulting from the online replanner for a moving target. (a) Trajectories in 3D space, (b) Trajectories in the xy-plane.

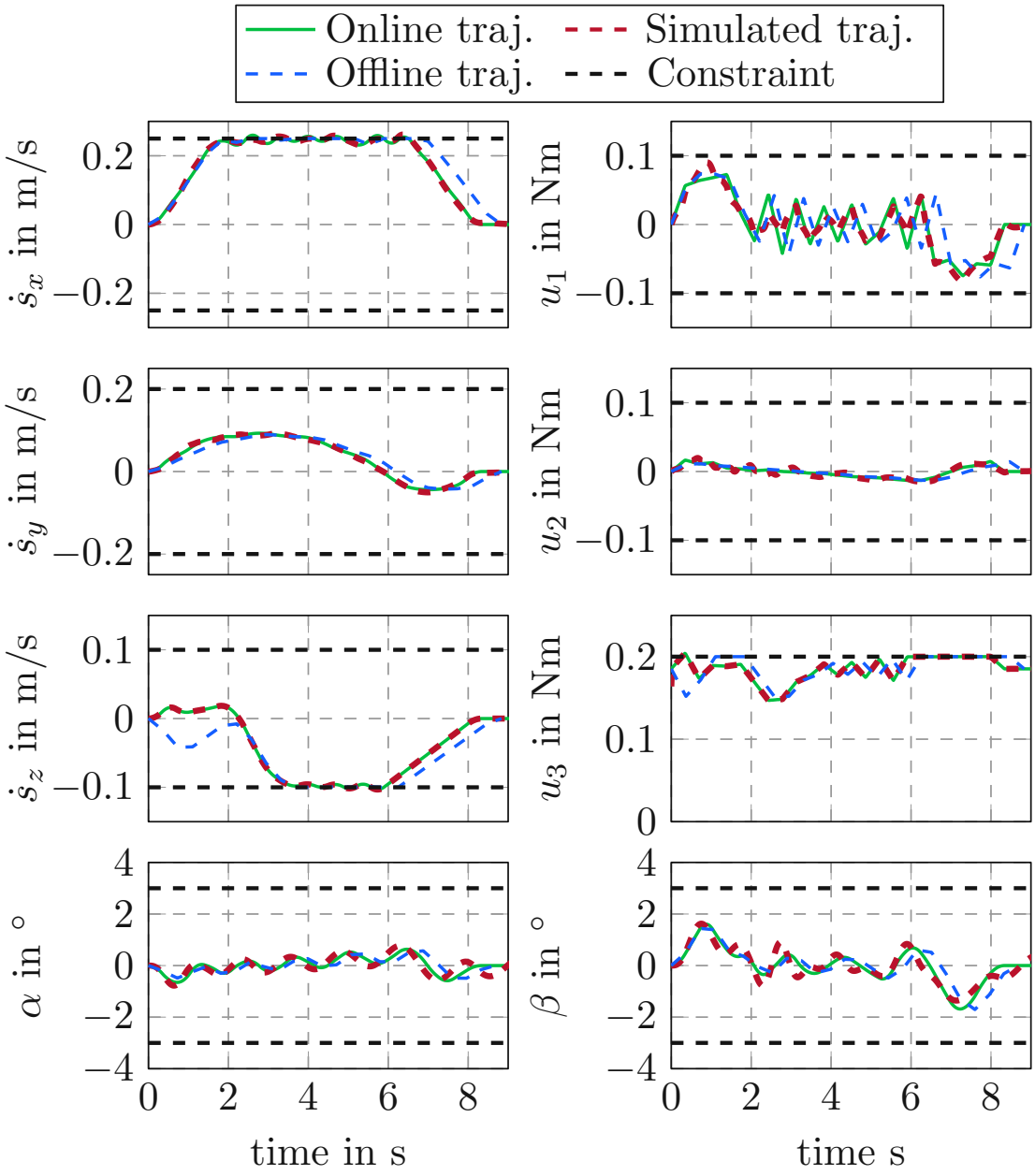


Figure 4.7: Time evolution of the states and control inputs for the stationary target scenario in Fig. 4.5.

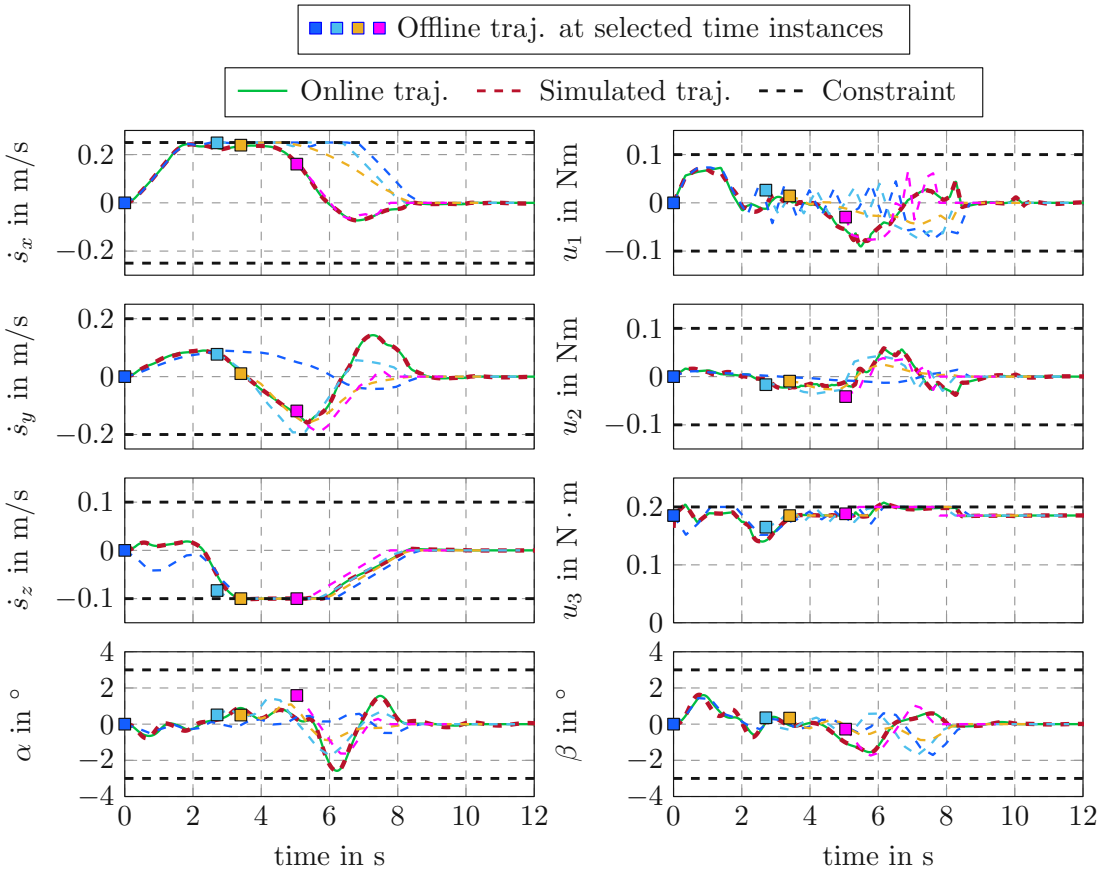


Figure 4.8: Simulations of the states and control inputs for the moving target scenario in Fig. 4.6. The four colored squares and the corresponding dashed lines illustrate the offline trajectories which are utilized by the online replanner at selected time instances.

collocation optimization with 26 grid points turns out to be sufficient for this application. Furthermore, considering the memory requirements for the offline database in the dSPACE MicroLabBox of the real experiment, up to 12.000 trajectories can be stored in the flash memory with 339 variables of type double per trajectory. For a large scene, such as a factory or a port, the number of required grid points, but also the required accuracy, scales accordingly. However, the computational power can be increased depending on the requirements.

In addition, the sparsity of the matrices is exploited to reduce memory consumption. The starting and target points lie in the subspaces $\mathcal{X}_S$ (cuboid of size $1.8\,\mathrm{m} \times 0.8\,\mathrm{m} \times 0.55\,\mathrm{m}$) and $\mathcal{X}_T$ (plane of size $1\,\mathrm{m} \times 0.8\,\mathrm{m}$), shown as gray box and yellow plane in Fig. 4.2, respectively. Based on the offline trajectory planning algorithm, a database of collision-free trajectories is calculated connecting each grid point in $\mathcal{X}_S$ with each grid point in $\mathcal{X}_T$. Even for coarse grids in the offline database, the online trajectory replanner shows a high success rate, i.e. provides feasible trajectories, in the Monte Carlo simulation, see [19]. In this work, a fixed number of grid points $n_S = 12 \times 8 \times 3$ and $n_T = 10 \times 6$ was chosen for $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively. This results in a total of 11.520 near time-optimal collision-free offline trajectories in the database after removing the invalid trajectories having its starting or target state inside an obstacle. Note that the average computation time for a single trajectory in the database is approximately 50 ms. The user-defined weighting parameters $\mathrm{diag}(\rho_n, n = 1, ..., 5) = \mathrm{diag}(4, 2.5, 3.5, 2, 2)$ are used in the k-nearest neighbor search in Alg. 4.

Qualitative results

To illustrate the overall concept consisting of the offline trajectory database, the online trajectory replanner and the underlying trajectory tracking controller, two example cases are shown in Figs. 4.5 and 4.6 :

- Fig. 4.5 shows the results of the stationary target scenario where the offline trajectory (dashed blue line) is deformed to the online trajectory (dashed green line) according to a given pair of starting and target positions (green circle and cross symbols). The dashed red path illustrates the simulated trajectory of the trajectory tracking controller, which is perfectly tracked with respect to the trajectory generated by the online trajectory replanner. The time evolution of the corresponding states $\dot{s}_x$, $\dot{s}_y$, $\dot{s}_z$, α and β as well

as the three control inputs u_1, u_2 and u_3 are depicted in Fig. 4.7. The green lines refer to the online trajectories, which deviate from the offline trajectory (blue dashed line). The simulated trajectory (red dashed line) shows a good tracking performance of the system. Also, the state and input constraints according to (4.19c), depicted as black dashed lines, are well respected.

- Fig. 4.6 shows the performance of the online replanner in the case of a moving target. The first solution of the online trajectory replanner (i.e. the green path) leads the 3D gantry crane to the left side of the green obstacle. Later, the online replanner generates completely different trajectories moving around the right side of the green obstacle (i.e. multiple blue lines). Analogous to Fig. 4.7, the simulation results of the moving target scenario are depicted in Fig. 4.8. The four colored square symbols and the corresponding colored dashed lines represent the offline trajectories that are taken from the offline database. Note that the blue dashed line is the first closest trajectory, while the other three colored dashed lines illustrate offline trajectories in the database at selected time instances, which are chosen by the online replanner during the movement of the 3D gantry crane.

It should be noted that small violations of the constraints are possible between two adjacent collocation points. The simulation results clearly show that the proposed concept is able to replan the offline trajectory while adhering to the constraints also in the case of a moving target. It can be nicely seen that the 3D gantry crane is driven in one of the limits for most of the time, which shows the near time-optimal behavior and that the admissible range of the system is fully exploited.

Quantitative results

Monte Carlo simulations were performed for both scenarios to investigate the versatility and robustness of the proposed approach. In the Monte Carlo simulation of the stationary target scenario, 10^4 trajectories were planned and simulated using pairs of randomly uniformly distributed starting and target states from the two subspaces $\mathcal{X}_S$ and $\mathcal{X}_T$, respectively. In the moving target scenario, the following procedure was repeated 10^4 times. First, a collision-free trajectory for the moving target was generated, where the target moves from a random location A to a random location B within the target subspace $\mathcal{X}_T$. Second, a random starting state was chosen in the starting subspace $\mathcal{X}_S$. Third, the scenario including the

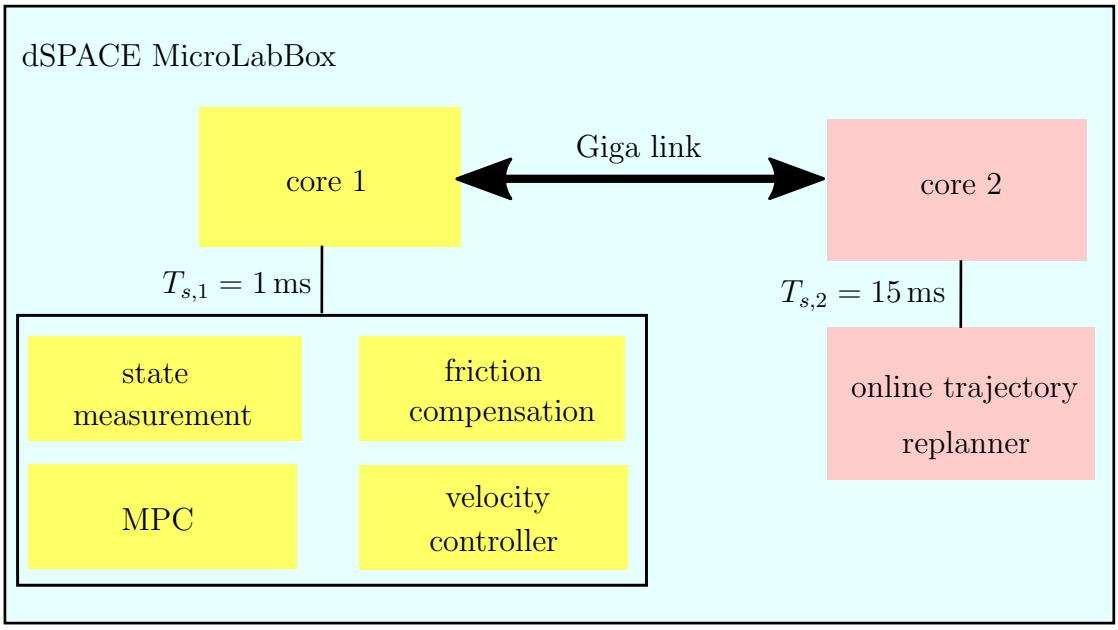
Table 4.1: Monte Carlo simulations with 10^4 test cases for two scenarios.

	Stationary target	Moving target
Number of simulation fails	0	6
Number of failed collision checks	230	251
Success rate	97.70%	97.43%
Average computing time	2 ms	2.5 ms

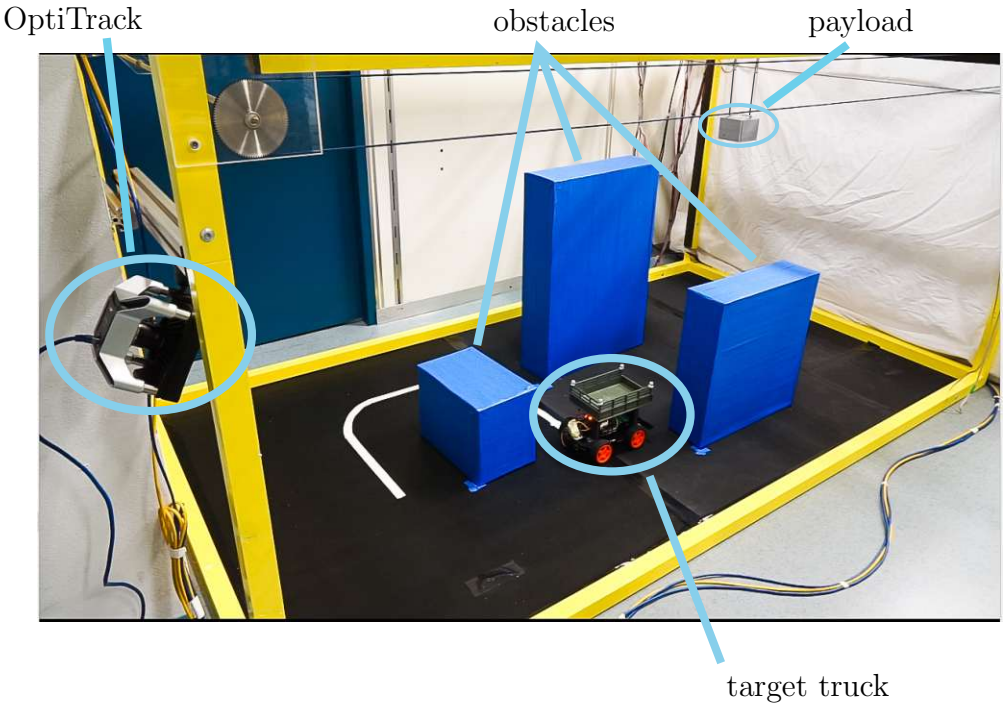
online trajectory replanner and the trajectory tracking controller was simulated. All simulations were run in SIMULINK rapid accelerator mode. After each simulation, two result flags, i.e. the simulation fail flag and the collision check flag, were collected. The simulation fail flag corresponds to an unexpected numerical error during the simulation, i.e., an infeasible trajectory. The collision check flag indicates that the online trajectory collides with obstacles. The statistics of the flags are shown in Table 4.1. The average computation time of the online replanner is 2 ms for the stationary and 2.5 ms for the moving target scenario, respectively. Since the online replanner must actively find the closest trajectory in the third layer of the database for moving targets, a longer computation time was expected. It should be noted that not all trajectories are collision-free, since the obstacles for the online trajectory replanner are only approximated by the Hessian of the artificial potential functions. This is reflected in the success rate in Table 4.1. The simulation only fails in the moving target scenario when the online replanner is unable to generate a feasible trajectory corresponding to the moving target.

4.4.2 Experimental results

The experimental setup shown in Fig. 4.9 consists of the 3D lab-scale gantry crane equipped with five incremental encoders and a dSPACE MicroLabBox. The online trajectory replanner and controller were implemented in MATLAB/SIMULINK and compiled and deployed on the dual-core real-time processor of the dSPACE MicroLabBox. In addition, a six-camera OptiTrack system is connected to dSPACE via Ethernet and is used to estimate the position of a remote-controlled moving truck (target) in the workspace. In the experiment of the moving target scenario, a truck following the colored path at a random speed is used as the moving target.



(a)



(b)

Figure 4.9: System overview of the lab-scale 3D laboratory gantry crane. (a) Software architecture. (b) Experimental setup.

Without loss of generality, the position of the truck is measured using the OptiTrack system. For a real application in a larger or more complex environment, there are many ways to determine the position of a moving truck, e.g., using GPS or vision-based object tracking. The proposed online trajectory replanner and the trajectory tracking controller are implemented on a dSPACE MicroLabBox real-time system with the sampling times of $T_{s,1} = 1\,\text{ms}$ and $T_{s,2} = 15\,\text{ms}$ on the cores 1 and 2, respectively. To solve the LCQP (4.19) in the online trajectory replanner with box constraints, the CVXgen package, see, e.g., [79], is used to generate optimized C code. Note that the time step $h = (t_F^* + \delta t_F)/N$ of the online trajectory is much larger than the sampling time $T_{s,1}$ of the controller. Therefore, the desired trajectory is interpolated before it is fed to the controller. Similar to the previous subsection, experimental results are demonstrated for a stationary and a moving target scenario shown in Figs. 4.10 and 4.11, respectively.

- Fig. 4.10 depicts the experimental results of the stationary target scenario containing the measurements of the system states $\dot{s}_x$, $\dot{s}_y$, $\dot{s}_z$, α and β and the control inputs u_1, u_2 and u_3. The measured signals and the desired trajectories are depicted as dashed red lines and solid green lines, respectively. All measured signals satisfy the state and input constraints (black dashed lines) given by (4.19c). In addition, the online trajectories (solid green lines) of the system states slightly deviate from the closest offline trajectories (dashed blue lines). Note that the travel time of the online trajectory differs from the offline trajectory due to δt_F in the LCQP (4.19).
- In a similar way as in Fig. 4.10, the experimental results of the moving target scenario are shown in Fig. 4.11. The four colored square symbols and the corresponding dashed lines represent the offline trajectories obtained by the search algorithm from the offline database. Note that online trajectory generation is indeed sufficient in satisfying the system state and input constraints at the collocation points. However, since an interpolation scheme is applied between two adjacent collocation points of the desired trajectory, small violations of the system state constraints may occur between the collocation points. Snapshots of the gantry crane and target truck motions are presented in Fig. 4.12. At the turning point (green circle in Fig. 4.12), the gantry crane has not come to a stop, but follows the motion of the moving truck smoothly. This point is approximately at the time $t = 7\,\text{s}$ where the system state velocities $\dot{s}_x, \dot{s}_y$ and $\dot{s}_z$ cross the zero line, see Fig. 4.11.

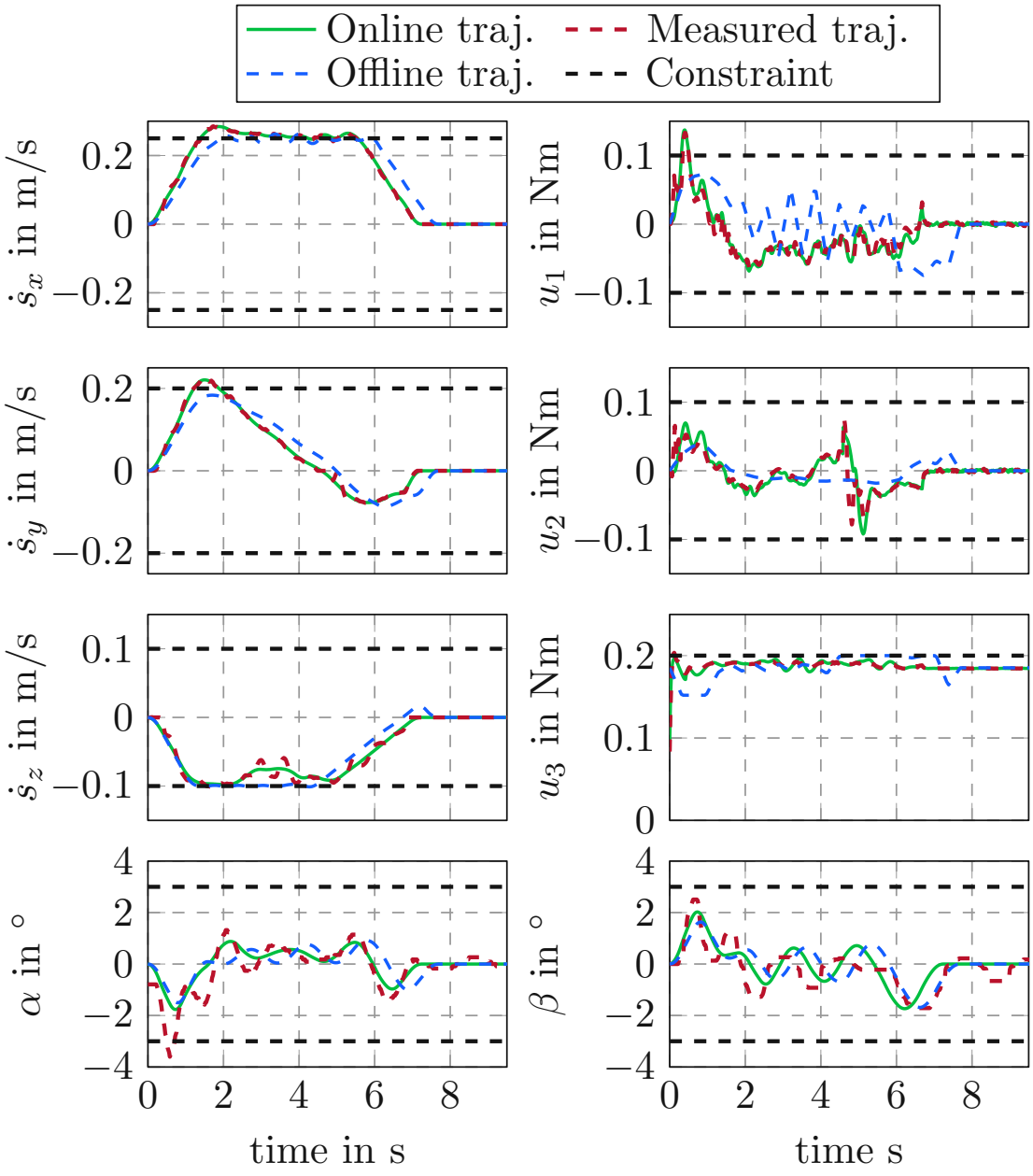


Figure 4.10: Measurements of the system state and control inputs in the stationary target scenario.

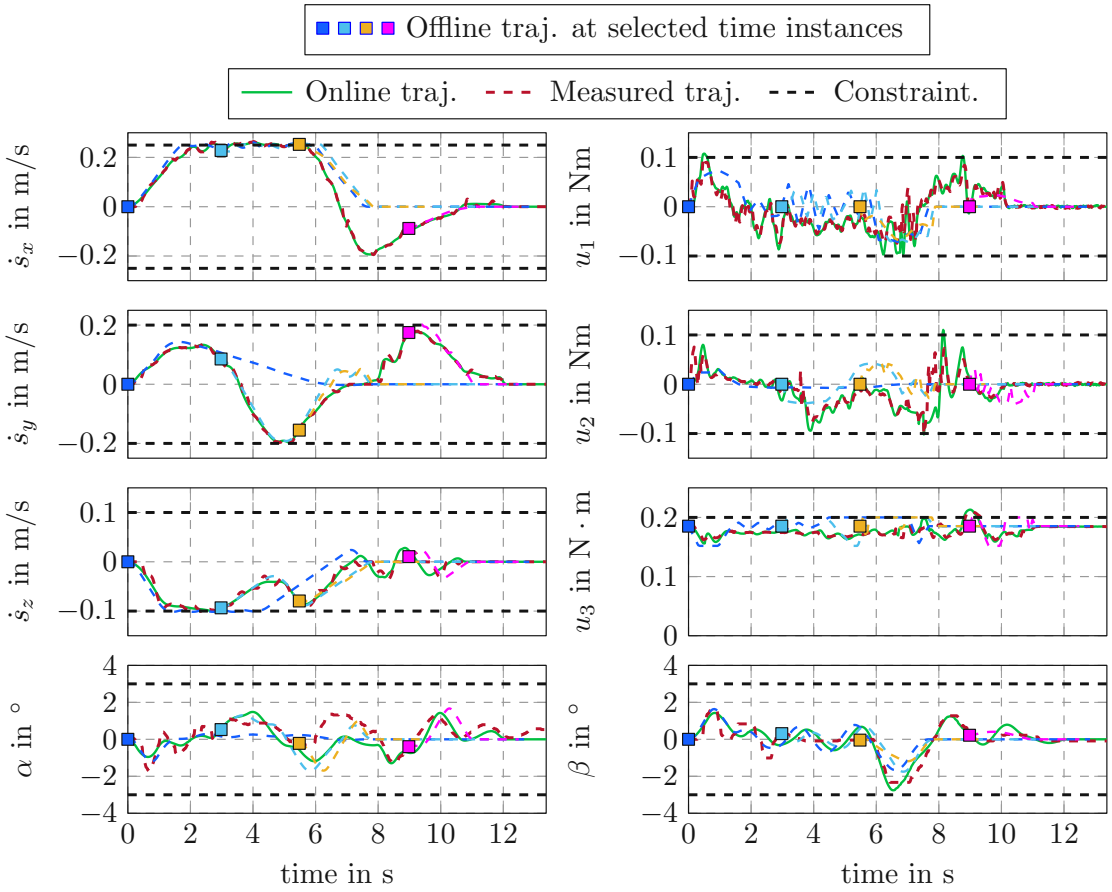


Figure 4.11: Measurements of states and control inputs over travel time for the moving target scenario in Fig. 4.12. The four colored squares and the corresponding dashed lines illustrate the offline trajectories taken by the online replanner at selected time instances.

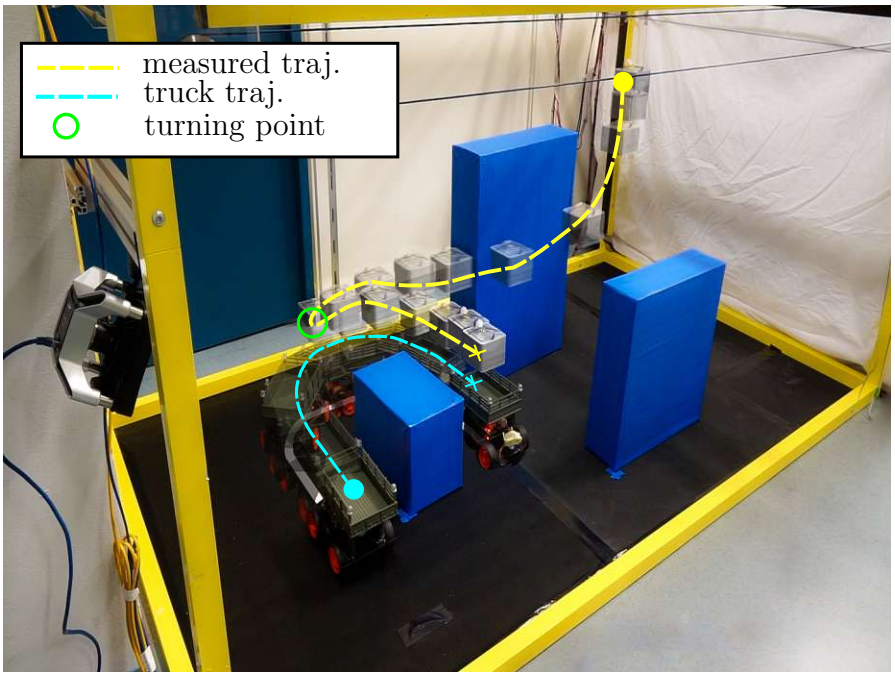


Figure 4.12: Illustration of the experimental execution of the online trajectory replanning for a moving target. The collision-free online path is shown as dashed yellow line. The cyan dashed line is the path of the moving truck. The dot and cross symbols represent the starting and target positions, respectively.

Note that in Fig. 4.10 and Fig. 4.11, although the gantry crane is traveling up to its full speed, i.e., the velocities $\dot{s}_x$, $\dot{s}_y$, and $\dot{s}_z$ reach the respective limits, the sway angles α and β stay within a small range of $[-0.05, 0.05]$ rad ($\approx \pm 3°$).

Overall, the trajectory tracking controller exhibits a good tracking performance, while the deviations of the pendulum angles α and β from the desired trajectory are clearly visible in both scenarios. These deviations can be attributed to model uncertainties and backlash caused by the measurement mechanism of the 3D lab-scale gantry crane. Moreover, a video of the discussed scenarios shown in Fig. 4.10 and Fig. 4.11 is available at `https://www.acin.tuwien.ac.at/en/65ce/`

4.5 Conclusions

In this section, the lab-scale experiment of a 3D gantry crane is considered that performs the task to pick up a payload at an arbitrary position and then deposit it

on a moving truck in a workspace with static obstacles. The time from picking up the payload to its deposition should be kept as small as possible and at the same time collisions with obstacles must be avoided and the system state and control input constraints must be respected. For this, a novel two-step trajectory planning algorithm, consisting of an offline trajectory optimization and an online trajectory replanner, in combination with an MPC (model predictive control)-based trajectory tracking controller was developed. The offline trajectory optimization is used to generate a collision-free and dynamically feasible trajectory database. The computation time for an offline trajectory in the database is about 50 ms on average on a standard PC, which is too long for real-time planning in the moving-truck scenario. Thus, there is a need to develop a real-time online trajectory planner to fulfill the task requirement. The idea of the online trajectory planner is to minimize the deviation from a suitable reference trajectory, which is selected according to a specifically designed strategy from the database, by solving a linear constrained quadratic program. This is computationally very efficient because it yields a feasible trajectory within an average computation time of 2.5 ms on a standard PC and is thus 25 times faster than the offline trajectory optimization. The online trajectory then serves as a reference input for an MPC-based tracking controller. Here an MPC was employed to be able to systematically account for state and control input constraints. Simulation studies and experimental results demonstrate the feasibility of the proposed approach.

This work is a proof of concept in the form of a laboratory experiment for real-time trajectory planning, where it is assumed that the required modules for scanning the environment including obstacle detection and the tracking system of the moving truck are available. One of the limitations of the proposed trajectory (re)planning is that it works only within the predefined starting and target subspaces. However, arbitrary subspaces can be chosen according to application requirements. For example, in this work, the starting subspace covers the entire workspace, while the target subspace corresponds to the workspace of a moving truck in 2D.

In the next section, the core idea of the proposed two-step optimization-based trajectory planning is applied to another robotic system with a different task, i.e., performing the optimal swing-up of a of a spherical pendulum on the 7-axis KUKA LBR iiwa 14 R820 robot.

5

Swing-up trajectory optimization of a pendulum on a collaborative robot

In this section, the fast swing-up trajectory optimization of a custom-designed spherical pendulum on the robot KUKA LBR iiwa 14 R820 is presented. The complete mechanical system of nine degrees freedom (DoF) consists of the 7-axis KUKA LBR iiwa 14 R820 and the two unactuated DoF spherical pendulum. The primary focus of this work is the design of a fast trajectory planning for the swing-up by systematically incorporating the kinematic and dynamics constraints. Inspired by the two-step trajectory planning in the previous section, the proposed algorithm consists of the two following steps. First, an offline trajectory optimization is used to build a database of swing-up trajectories, with an average computing time of 10 s for one trajectory. Second, a fast trajectory replanner based on a constrained quadratic program is used to compute the swing-up trajectory for an arbitrary initial configuration of the system with an average computing time of 200 ms. To realize the proposed trajectory in experiments, a controller based on a discrete time-variant linear quadratic regulator is implemented. Simulations

and experimental results are presented to validate the feasibility of the proposed approach. Large parts of this section are identical to the author's publication [20].

5.1 Introduction

For many years, the inverted pendulum has been a popular benchmark for research in nonlinear control and trajectory optimization, see, e.g., [80]. Inverted pendulums with increasingly complex kinematics were published, such as the single pendulum [81], the double pendulum [82], the triple pendulum on a cart [83], and the spherical pendulum [84]. Despite its simple structure, the inverted pendulum represents an underactuated nonlinear mechanical system and exhibits an unstable equilibrium point and non-minimum phase behavior. Therefore, it offers exciting challenges for trajectory optimization and nonlinear control.

In the 1960s, several contributions on the stabilization of the single inverted pendulum were published, see, e.g., [85], [86]. Rather than only to achieve the stabilization task, the swing-up control of an inverted pendulum system was first presented using a simple feedforward control [87]. Later, more advanced control strategies were applied such as nonlinear model predictive control [88], reinforcement learning [89], and Lyapunov-based damping control [90]. As an alternative system to the pendulum on a cart, a rotary pendulum system was introduced in [91], where a bang-bang type feedback controller for the swing-up action together with a least-squares regulator for stabilization were employed. In 2013, the swing-up of the triple inverted pendulum on a cart was experimentally demonstrated in [83] by using an output-constrained feedforward controller [92] on a real-time hardware.

In most works from the literature, the swing-up and stabilization of a pendulum is performed using a custom-designed mechanical system which is capable of providing the required forces and torques. However, the complexity increases significantly for implementations on an industrial robot, where the robot's joint position, joint velocity, and input torque limits have to be considered. Winkler et al. [93] presented the swing-up and stabilizing of a single inverted pendulum on the KUKA KR3 by using an intuitive control scheme consisting of a feedforward and a balancing controller. Recently, experiments of balancing an inverted pendulum on a 7 degrees of freedom (DoFs) robotic arm are accomplished by using Bayesian

optimization [94] and model-based policy search [95] for automatic tuning of the controller. The stabilization of a spherical pendulum was first demonstrated on the robot "DLR LWR II" by Schreiber et al. in [96]. In [97], a first swing-up trajectory planning approach was proposed which only allows a restricted movement of the robot. Despite these restrictions, due to the complexity of the system with 9 DoFs and the associated kinematic and dynamic constraints, the calculation of feasible swing-up trajectories was very time consuming ($\approx$ several hours on a standard PC hardware). Additionally, the swing-up demonstration in [97] is limited to a fixed initial and target configuration.

Recently, in order to plan the desired motion of a system involving inverted pendulums, i.e., brachiation robots, desired trajectories are computed through a multiple-shooting optimization scheme [98]. Then, a feedback trajectory tracking controller is designed utilizing a sliding mode control [99] approach or a time-variant linear quadratic regulator (LQR) [100]. Another common approach is to combine the planning and controlling phase by using a library of trajectories in order to directly design the controllers for the nonlinear underactuated system, see, e.g., [101], [102]. In [101], the motion library is built on a sparse tree containing LQR-stabilized trajectories, its region of attraction is verified using a simulation-based falsification method. Although this approach shows promising results, a large computation time is required for generating and verifying a single tree policy.

In this work, the custom-built spherical pendulum tool from [97] is mounted on the 7-axis collaborative robot KUKA LBR iiwa 14 R820. The main contribution of this section is a fast algorithm for the computation of swing-up trajectories, which consists of two steps. In the *first* step, an offline trajectory planner computes a database of near time-optimal swing-up trajectories with initial states taken from equidistantly discretized joint configurations in the joint space. Note that the final state is fixed at the equilibrium state of the complete system. This planner systematically accounts for the dynamic limitation of the system states and control inputs. In the *second* step, a fast trajectory replanner calculates a new swing-up trajectory from an arbitrary initial state in the feasible range of the robot. To demonstrate the swing up, a discrete LQR is applied to stabilize the robot on this trajectory.

The remainder of this section is structured as follows. Section 5.2 deals with the offline trajectory planner and builds up a trajectory database for the complete

mechanical system. Section 5.3 is concerned with the design of a fast replanning algorithm, which chooses the closest swing-up trajectory from the database and adapts this trajectory via a constrained quadratic program. Simulation and experimental results are shown in Section 5.5. The last section gives some conclusions.

5.2 Offline trajectory optimization

The mathematical model of the complete system consisting of the robot KUKA LBR iiwa 14 R820 and the spherical pendulum was introduced in Section 2.2. In this section, the offline trajectory planner is derived to compute a database of time-optimal swing-up trajectories for the system (2.36) depicted in Fig. 2.5. These trajectories take into account the full system state $\mathbf{x}^{\mathrm{T}} = [\mathbf{q}^{\mathrm{T}}, \dot{\mathbf{q}}^{\mathrm{T}}]$ and the motor torques $\boldsymbol{\tau}_A$. In order to reduce the complexity of the system dynamics and thus enhance the computational speed, (2.36) is partitioned into an actuated subsystem $\mathbf{q}_A^{\mathrm{T}} = [q_1, q_2, \ldots, q_7]$ and an unactuated system $\mathbf{q}_U^{\mathrm{T}} = [q_8, q_9]$. Thus, (2.36) can be written as

$$\begin{bmatrix} \mathbf{M}_A & \mathbf{M}_{AU} \\ \mathbf{M}_{AU}^{\mathrm{T}} & \mathbf{M}_U \end{bmatrix} \begin{bmatrix} \ddot{\mathbf{q}}_A \\ \ddot{\mathbf{q}}_U \end{bmatrix} + \begin{bmatrix} \mathbf{C}_A & \mathbf{C}_{AU} \\ \mathbf{C}_{UA} & \mathbf{C}_U \end{bmatrix} \begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \end{bmatrix} + \begin{bmatrix} \mathbf{g}_A \\ \mathbf{g}_U \end{bmatrix} = \begin{bmatrix} \boldsymbol{\tau}_A \\ \mathbf{0} \end{bmatrix}, \tag{5.1}$$

where $\mathbf{q}^{\mathrm{T}} = [\mathbf{q}_A^{\mathrm{T}}, \mathbf{q}_U^{\mathrm{T}}]$. The acceleration of the unactuated joints $\ddot{\mathbf{q}}_U$ and the motor torques $\boldsymbol{\tau}_A$ can be calculated from (5.1), resulting in

$$\ddot{\mathbf{q}}_U = \mathbf{M}_U^{-1}(-\mathbf{M}_{AU}^{\mathrm{T}}\ddot{\mathbf{q}}_A - \mathbf{C}_{UA}\dot{\mathbf{q}}_A - \mathbf{C}_U\dot{\mathbf{q}}_U - \mathbf{g}_U) \tag{5.2a}$$

$$\boldsymbol{\tau}_A = \mathbf{M}_A\ddot{\mathbf{q}}_A + \mathbf{M}_{AU}\ddot{\mathbf{q}}_U + \mathbf{C}_A\dot{\mathbf{q}}_A + \mathbf{C}_{AU}\dot{\mathbf{q}}_U + \mathbf{g}_A\ . \tag{5.2b}$$

The acceleration of the actuated joints $\ddot{\mathbf{q}}_A$ directly acts on the unactuated system. Thus, by considering $\mathbf{u} = \ddot{\mathbf{q}}_A$ as the new control input to the system, the state-space representation of (2.36) reads as

$$\frac{\mathrm{d}}{\mathrm{dt}} \begin{bmatrix} \mathbf{q}_A \\ \mathbf{q}_U \\ \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \end{bmatrix} = \begin{bmatrix} \dot{\mathbf{q}}_A \\ \dot{\mathbf{q}}_U \\ \mathbf{u} \\ \mathbf{M}_U^{-1}(-\mathbf{M}_{AU}^{\mathrm{T}}\mathbf{u} - \mathbf{C}_{UA}\dot{\mathbf{q}}_A - \mathbf{C}_U\dot{\mathbf{q}}_U - \mathbf{g}_U) \end{bmatrix}. \tag{5.3}$$

In a more compact form, (5.3) is rewritten as

$$\frac{\mathrm{d}}{\mathrm{dt}}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}). \tag{5.4}$$

It is worth noting that the inversion of $\mathbf{M}_U \in \mathbb{R}^{2\times 2}$ in (5.3) is computationally much more efficient than an inversion of $\mathbf{M} \in \mathbb{R}^{9\times 9}$ in (2.36).

5.2.1 Near time-optimal swing-up trajectory optimization

Similar to (4.1) in Section 4.2.1, the offline optimization scheme, adapted for (5.4), is formulated using the direct collocation method, see, e.g., [69], by discretizing the swing-up trajectory into $N+1$ grid points and solving the resulting static optimization problem

$$\min_{\boldsymbol{\xi}} J(\boldsymbol{\xi}) = t_F + \frac{1}{2}h\sum_{k=0}^{N}\mathbf{u}_k^{\mathrm{T}}\mathbf{R}\mathbf{u}_k \tag{5.5a}$$

$$\text{s.t. } \mathbf{x}_{k+1} - \mathbf{x}_k = \frac{1}{2}h(\mathbf{f}_k + \mathbf{f}_{k+1}) \tag{5.5b}$$

$$\mathbf{x}_0 = \mathbf{x}_S,\ \ \mathbf{x}_N = \mathbf{x}_T,\ \ g(\mathbf{x}_T) = 0 \tag{5.5c}$$

$$\underline{\mathbf{x}} \leq \mathbf{x}_k \leq \overline{\mathbf{x}},\ k = 0, \ldots, N\,, \tag{5.5d}$$

where the index k indicates the discrete time step $t = kh$, $\mathbf{f}_k = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k)$ is the right-hand side of (5.4), and the vector of optimization variables

$$\boldsymbol{\xi}^{\mathrm{T}} = [t_F, \mathbf{x}_0^{\mathrm{T}}, \ldots, \mathbf{x}_N^{\mathrm{T}}, \mathbf{u}_0^{\mathrm{T}}, \ldots, \mathbf{u}_N^{\mathrm{T}}] \tag{5.6}$$

is introduced. Note that the final time t_F in (5.5a) denotes the time it takes for the system (5.4) to transition from the starting state $\mathbf{x}_S$ to the target state $\mathbf{x}_T$. Additionally, $h = t_F/N$ is the time step and $\mathbf{R}$ is the positive definite weighting matrix for the control input $\mathbf{u}$. The target state $\mathbf{x}_T$ for the spherical pendulum is the upright position, i.e., the unstable equilibrium with $[q_8, q_9] = [180°, 0]$. To avoid collisions with the surroundings, the target state $\mathbf{x}_T$ is partly constrained using

$$\mathbf{x}_T = [0, q_{2F}, 0, q_{4F}, 0, q_{6F}, 0, 180°, 0]^{\mathrm{T}},$$

with

$$g(\mathbf{x}_T) = q_{2F} - q_{4F} + q_{6F} - 90° = 0, \tag{5.7}$$

as shown in Fig. 5.1. Note that the target configuration $\mathbf{x}_T$ is systematically optimized by the equality condition (5.7). This could lead to a better and faster solution than using a fixed target configuration, since the initial condition is chosen randomly.

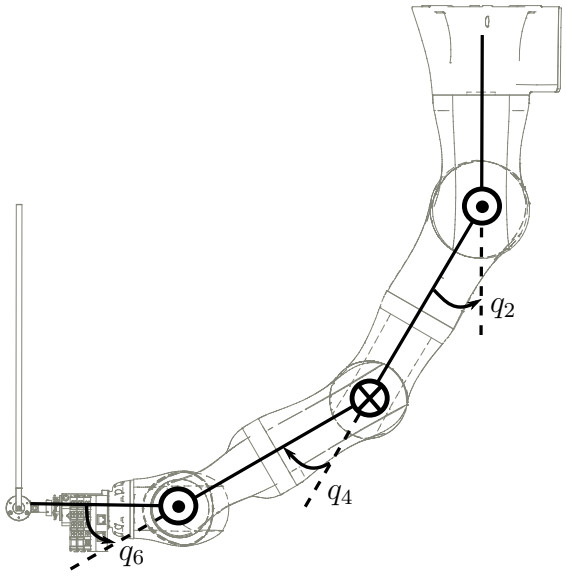


Figure 5.1: Schematic drawing of an example target configuration. The black circle and the cross indicate the local z-axis pointing out of and into the plane of the paper, respectively.

Furthermore, the system dynamics (5.4) are approximated using the trapezoidal rule in (5.5b). Moreover, $\underline{\mathbf{x}}$ and $\overline{\mathbf{x}}$ denote the symmetric lower and upper bounds of the state, respectively. Since the accelerations of the actuated joints are considered as control inputs, i.e. $\mathbf{u} = \ddot{\mathbf{q}}_A$, the motor torque limits have to be incorporated as additional constraints. By substituting (5.2a) into (5.2b), the motor torque limits are reformulated as

$$\underline{\boldsymbol{\tau}_A} \leq \widetilde{\mathbf{M}}(\mathbf{q}_k)\mathbf{u}_k + \widetilde{\mathbf{C}}(\mathbf{q}_k, \dot{\mathbf{q}}_k)\dot{\mathbf{q}}_k + \widetilde{\mathbf{g}}(\mathbf{q}_k) \leq \overline{\boldsymbol{\tau}_A}\,, \tag{5.8}$$

where

$$\begin{aligned} \widetilde{\mathbf{M}} &= \mathbf{M}_A - \mathbf{M}_{AU}\mathbf{M}_U^{-1}\mathbf{M}_{AU}^{\mathrm{T}} \\ \widetilde{\mathbf{C}} &= \begin{bmatrix} \mathbf{C}_A - \mathbf{M}_{AU}\mathbf{M}_U^{-1}\mathbf{C}_{UA} & \mathbf{C}_{AU} - \mathbf{M}_{AU}\mathbf{M}_U^{-1}\mathbf{C}_U \end{bmatrix} \\ \widetilde{\mathbf{g}} &= \mathbf{g}_A - \mathbf{M}_{AU}\mathbf{M}_U^{-1}\mathbf{g}_U\,. \end{aligned} \tag{5.9}$$

Note that (5.8) is a computationally costly inequality constraint, mainly due to the large expressions in the Coriolis matrix $\widetilde{\mathbf{C}}$. In fact, the Coriolis matrix is often neglected in industrial applications [103]. To account for the effect of the Coriolis matrix $\widetilde{\mathbf{C}}$, the value range of $\widetilde{\mathbf{C}}\dot{\mathbf{q}}$ for the complete system is examined using a Monte-Carlo simulation. In this simulation, 10^8 uniformly distributed random state vectors $\mathbf{x}$ are selected from the admissible range, see Tab. 5.1. This simulation reveals that the values of $\widetilde{\mathbf{C}}\dot{\mathbf{q}}$ are bounded between $\overline{\mathbf{c}}^{\mathrm{T}} = [6, 8, 3, 4, 1, 1, 0.1]$ N m and $\underline{\mathbf{c}}^{\mathrm{T}} = -[6, 7, 3, 4, 1, 1, 0.1]$ N m, which is significantly smaller than the motor torque limits. Although the effect of the Coriolis matrix on the overall system dynamics is not significant, it is still worth considering the determined bounds in the optimization scheme to avoid exceeding the motor torque limits. To this end, the costly inequality constraint (5.8) is replaced by

$$\underline{\boldsymbol{\tau}_A} - \underline{\mathbf{c}} \leq \widetilde{\mathbf{M}}(\mathbf{q}_k)\mathbf{u}_k + \widetilde{\mathbf{g}}(\mathbf{q}_k) \leq \overline{\boldsymbol{\tau}_A} - \overline{\mathbf{c}}\,. \tag{5.10}$$

The swing-up trajectory is found by solving the static optimization problem (5.5), (5.7), and (5.10) using the Interior Point OPTimize (IPOPT), which is an open-source package based on the interior point method (IPM) for large scale nonlinear programming, see, e.g., [71]. In order to speed up the computation, the gradient of (5.5a), the Jacobian of the constraints (5.5b-d), (5.7), and (5.10) are computed analytically. After the optimal values are found at the collocation points, the trajectory of the state $\mathbf{x}$ and the input $\mathbf{u}$ need to be interpolated. Between two adjacent collocation points k and $k+1$, the input $\mathbf{u}(t)$ and the state $\mathbf{x}(t)$ for $t \in [kh, (k+1)h]$ are a simple linear and quadratic spline similar to (4.12).

5.2.2 Trajectory database

After deriving all ingredients for solving a single swing-up trajectory optimization, an offline database is built from a discrete joint space. Assuming that the swing-up trajectory starts from an equilibrium point of the complete system, the two angles q_8 and q_9 and the joint velocities $\dot{\mathbf{q}}$ of $\mathbf{x}_S$ are zero. Thereby, a time-optimal swing-up trajectory is planned offline from every grid point in this discrete joint space using (5.5), (5.7), and (5.10). Then, these offline trajectories are stored in a database. Note that each trajectory in the database is represented by a label which contains the actuated joint angles $\mathbf{q}_A$ of the starting state $\mathbf{x}_S$. In order to efficiently search for a suitable trajectory in the database, this labeled data is preprocessed with the

k-d tree algorithm [74], which is a well-known space-partitioning data structure for partitioning and organizing points. An example of a 2-D data tree is shown in Fig. 5.2, where the 2-D data points (blue dots) are partitioned into different regions. In Fig. 5.2, it is obvious when finding the nearest point in this data tree to an arbitrary 2-D data point, such as $(-35, -25)$, only data points in region ① are taken into account for the search. A k-d tree data structure is considered to be efficient with $k \leq 10$ because of the so-called "curse of dimensionality" [104].

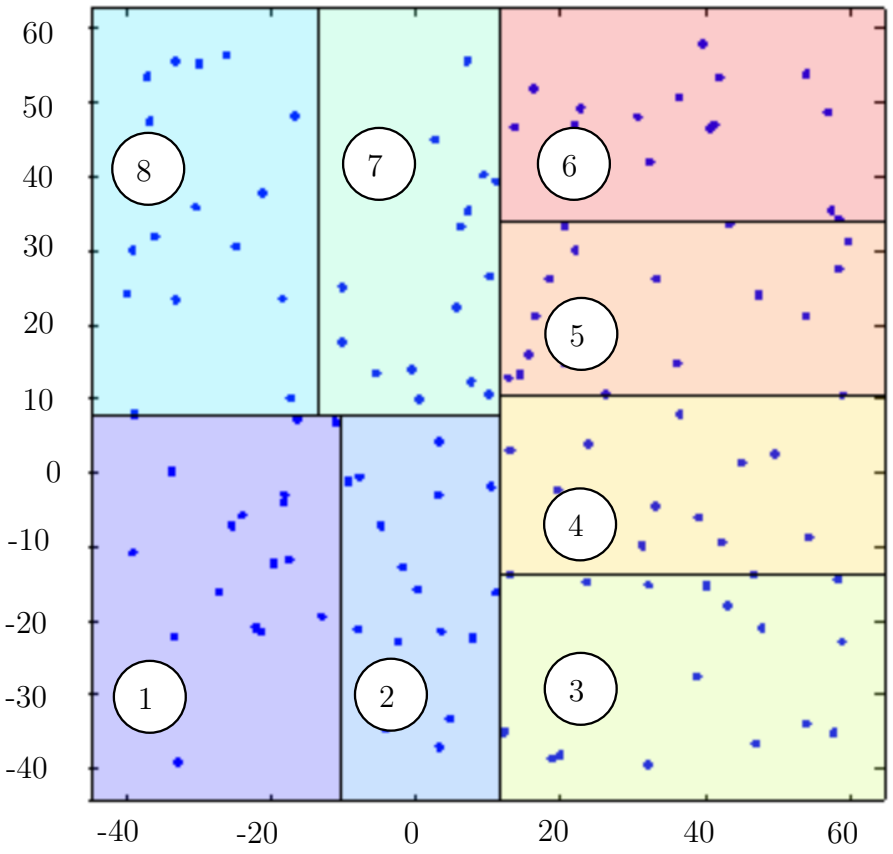


Figure 5.2: Example of a k-d tree with $k = 2$. The leafs of the tree are illustrated in different color regions, and separate data points (blue dots).

5.3 Fast trajectory replanner

In the previous section, an offline trajectory optimization is used to build a database of swing-up trajectories for a discrete set of starting states $\mathbf{x}_S$ of the system (5.4).

However, the actual starting state of the system $\tilde{\mathbf{x}}_S$ is almost certainly not contained in the database. Therefore, a fast trajectory replanner is proposed in this section to adapt the precomputed trajectories.

First, the nearest-neighbor search algorithm [76] is applied to find the grid point in the database closest to $\tilde{\mathbf{x}}_S$. Using the pre-built k-d tree, the search for the closest grid point quickly eliminates large portions of the search space. The near time-optimal swing-up trajectory from the database corresponding to the grid point closest to $\tilde{\mathbf{x}}_S$ is denoted as

$$(\boldsymbol{\xi}^*)^{\mathrm{T}} = [(t_F^*)^{\mathrm{T}}, (\mathbf{x}_0^*)^{\mathrm{T}}, \dots, (\mathbf{x}_N^*)^{\mathrm{T}}, (\mathbf{u}_0^*)^{\mathrm{T}}, \dots, (\mathbf{u}_N^*)^{\mathrm{T}}] \,. \tag{5.11}$$

In the second step, $\boldsymbol{\xi}^*$ is exploited to calculate the swing-up trajectory for the actual starting state $\bar{\mathbf{x}}_S$ which deviates from $\mathbf{x}_0^*$. It can be considered that only small deviations $\delta\boldsymbol{\xi} = [\delta t_F, \delta\mathbf{x}_0, ..., \delta\mathbf{x}_N, \delta\mathbf{u}_0, ..., \delta\mathbf{u}_N]^{\mathrm{T}}$ are required to obtain the adapted swing-up trajectory. Thus, the discrete-time system dynamics (5.5b) are linearized around $\boldsymbol{\xi}^*$ in the form

$$\delta\mathbf{x}_{k+1} = \delta\mathbf{x}_k + \frac{t_F^*}{2N}\left(\boldsymbol{\Gamma}_k^{\mathbf{x}}\delta\mathbf{x}_k + \boldsymbol{\Gamma}_k^{\mathbf{u}}\delta\mathbf{u}_k + \boldsymbol{\Gamma}_{k+1}^{\mathbf{x}}\delta\mathbf{x}_{k+1} + \boldsymbol{\Gamma}_{k+1}^{\mathbf{u}}\delta\mathbf{u}_{k+1}\right) + \frac{\delta t_F}{2N}(\mathbf{f}_k^* + \mathbf{f}_{k+1}^*) \,, \tag{5.12}$$

with

$$\delta\mathbf{x}_k = \mathbf{x}_k - \mathbf{x}_k^* \tag{5.13a}$$
$$\delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_k^* \tag{5.13b}$$
$$\delta t_F = t_F - t_F^* \tag{5.13c}$$
$$\mathbf{f}_k^* = \mathbf{f}(\mathbf{x}_k^*, \mathbf{u}_k^*) \,, \tag{5.13d}$$

and

$$\boldsymbol{\Gamma}_k^{\mathbf{x}} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{x}}\right|_{\mathbf{x}_k^*, \mathbf{u}_k^*} \tag{5.14a}$$
$$\boldsymbol{\Gamma}_k^{\mathbf{u}} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{\mathbf{x}_k^*, \mathbf{u}_k^*} \tag{5.14b}$$

for $k = 0, \ldots, N-1$. In addition, the deviations

$$\delta \mathbf{x}_0 = \tilde{\mathbf{x}}_S - \mathbf{x}_0^* \tag{5.15a}$$

$$\delta \mathbf{x}_N = \tilde{\mathbf{x}}_T - \mathbf{x}_N^* = \mathbf{0} \tag{5.15b}$$

are fixed due to the given actual system state $\tilde{\mathbf{x}}_S$ and the equilibrium state $\mathbf{x}_N^*$ from the trajectory $\boldsymbol{\xi}^*$ found in the database. The deviation $\delta\boldsymbol{\xi}$ is obtained as the solution of a constrained quadratic program (QP) [105]

$$\min_{\delta\boldsymbol{\xi}} J_{\boldsymbol{\xi}} = \frac{1}{2}\delta\boldsymbol{\xi}^{\mathrm{T}}\mathbf{Q}\delta\boldsymbol{\xi} \tag{5.16a}$$

$$\text{s.t. } \mathbf{A}\delta\boldsymbol{\xi} = \mathbf{b} \tag{5.16b}$$

$$\underline{\delta\boldsymbol{\xi}} \leq \delta\boldsymbol{\xi} \leq \overline{\delta\boldsymbol{\xi}}\,, \tag{5.16c}$$

with the positive definite weighting matrix

$$\mathbf{Q} = \mathrm{diag}(Q_{t_F}, \mathbf{Q}_{\mathbf{x}_0}, \ldots, \mathbf{Q}_{\mathbf{x}_N}, \mathbf{Q}_{\mathbf{u}_0}, \ldots, \mathbf{Q}_{\mathbf{u}_N})\,. \tag{5.17}$$

The equality constraints (5.16b) are constructed by combining the equations of the linearized system dynamics (5.12) together with (5.15). The inequality constraints (5.16c) correspond to (5.5d) and (5.10), where $\underline{\delta\boldsymbol{\xi}}^{\mathrm{T}} = [0, \underline{\delta\mathbf{x}}^{\mathrm{T}}, \underline{\delta\mathbf{u}}^{\mathrm{T}}]$, $\overline{\delta\boldsymbol{\xi}}^{\mathrm{T}} = [\overline{\delta t}_F, \overline{\delta\mathbf{x}}^{\mathrm{T}}, \overline{\delta\mathbf{u}}^{\mathrm{T}}]$, with $\underline{\delta\mathbf{x}} = \underline{\mathbf{x}} - \mathbf{x}^*$, $\overline{\delta\mathbf{x}} = \overline{\mathbf{x}} - \mathbf{x}^*$, $\underline{\delta\mathbf{u}} = \underline{\mathbf{u}} - \mathbf{u}^*$, $\overline{\delta\mathbf{u}} = \overline{\mathbf{u}} - \mathbf{u}^*$, and the sufficiently large upper bound $\overline{\delta t}_F$. The computational efficiency is further improved by implementing a receding horizon scheme which splits the full horizon of N grid points into s sub-horizons consisting of N/s grid points. Subsequently, the constrained QP (5.16) is applied to each sub-horizon individually while considering that the endpoint of one sub-horizon corresponds to the starting point of the next one.

5.4 Trajectory tracking controller

In order to perform the swing-up trajectory in the experiment, a closed-loop controller is necessary to handle the neglected friction and model uncertainties. Thus, a discrete time-variant LQR is employed to stabilize the system (5.4) around the swing-up trajectory from Section 5.3.

The system (5.4) is linearized around the desired trajectory

$$(\mathbf{x}^d, \mathbf{u}^d) = (\mathbf{x}^* + \delta\mathbf{x}, \mathbf{u}^* + \delta\mathbf{u}) \,, \tag{5.18}$$

with $(\mathbf{x}^d)^\mathrm{T} = [(\mathbf{q}_A^d)^\mathrm{T}, (\mathbf{q}_U^d)^\mathrm{T}, (\dot{\mathbf{q}}_A^d)^\mathrm{T}, (\dot{\mathbf{q}}_U^d)^\mathrm{T}]$ and $(\delta\mathbf{x}, \delta\mathbf{u})$ as the solution of the constrained QP (5.16). Assuming that the system matrices remain constant for each discrete time step k, the discrete time-variant system dynamics read as

$$\boldsymbol{\Delta}\mathbf{x}_{k+1} = \boldsymbol{\Phi}_k \boldsymbol{\Delta}\mathbf{x}_k + \boldsymbol{\Gamma}_k \boldsymbol{\Delta}\mathbf{u}_k \,, \tag{5.19}$$

with

$$\begin{aligned} \boldsymbol{\Delta}\mathbf{x}_k &= \mathbf{x}_k - \mathbf{x}_k^d & \boldsymbol{\Delta}\mathbf{u}_k &= \mathbf{u}_k - \mathbf{u}_k^d \\ \boldsymbol{\Phi}_k &= \mathbf{I}_{18} + T_s \mathbf{A}_k & \boldsymbol{\Gamma}_k &= T_s \mathbf{B}_k \\ \mathbf{A}_k &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}_k^d, \mathbf{u}_k^d} & \mathbf{B}_k &= \left.\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right|_{\mathbf{x}_k^d, \mathbf{u}_k^d} \,, \end{aligned} \tag{5.20}$$

and the sampling time T_s. The discrete time-variant LQR is applied to stabilize this error system by minimizing the objective function

$$J = \sum_{k=0}^{N_c-1} (\boldsymbol{\Delta}\mathbf{x}_k^\mathrm{T} \mathbf{Q}_c \boldsymbol{\Delta}\mathbf{x}_k + \boldsymbol{\Delta}\mathbf{u}_k^\mathrm{T} \mathbf{R}_c \boldsymbol{\Delta}\mathbf{u}_k) + \boldsymbol{\Delta}\mathbf{x}_N^\mathrm{T} \mathbf{X} \boldsymbol{\Delta}\mathbf{x}_N \,, \tag{5.21}$$

where $\mathbf{Q}_c$ and $\mathbf{R}_c$ are the positive definite weighting matrices. Moreover, the matrix $\mathbf{X}$ is chosen as the unique solution of the algebraic Riccati equation, see [106],

$$\mathbf{X} = \boldsymbol{\Phi}_{N_c}^\mathrm{T} \mathbf{X} \boldsymbol{\Phi}_{N_c} - (\boldsymbol{\Phi}_{N_c}^\mathrm{T} \mathbf{X} \boldsymbol{\Gamma}_{N_c})(\mathbf{R}_c + \boldsymbol{\Gamma}_{N_c}^\mathrm{T} \mathbf{X} \boldsymbol{\Gamma}_{N_c})^{-1} (\boldsymbol{\Gamma}_{N_c}^\mathrm{T} \mathbf{X} \boldsymbol{\Phi}_{N_c}) + \mathbf{Q}_c \,. \tag{5.22}$$

The LQR control law reads as, see, e.g., [107],

$$\boldsymbol{\Delta}\mathbf{u}_k = \mathbf{K}_k \boldsymbol{\Delta}\mathbf{x}_k \,, \tag{5.23}$$

with the controller feedback gain matrix $\mathbf{K}_k$. The gain matrix is computed by iterating the discrete-time Riccati-equation

$$\mathbf{K}_k = -(\mathbf{R}_c + \boldsymbol{\Gamma}_k^\mathrm{T} \mathbf{P}_{k+1} \boldsymbol{\Gamma}_k)^{-1} (\boldsymbol{\Gamma}_k^\mathrm{T} \mathbf{P}_{k+1} \boldsymbol{\Phi}_k) \tag{5.24a}$$

$$\mathbf{P}_k = (\mathbf{Q}_c + \boldsymbol{\Phi}_k^\mathrm{T} \mathbf{P}_{k+1} \boldsymbol{\Phi}_k) - \boldsymbol{\Phi}_k^\mathrm{T} \mathbf{P}_{k+1} \boldsymbol{\Gamma}_k (\mathbf{R}_c + \boldsymbol{\Gamma}_k^\mathrm{T} \mathbf{P}_{k+1} \boldsymbol{\Gamma}_k)^{-1} \boldsymbol{\Gamma}_k^\mathrm{T} \mathbf{P}_{k+1} \boldsymbol{\Phi}_k$$

backwards from $k = N_c - 1, \ldots, 1$ with $\mathbf{P}_{N_c} = \mathbf{X}$. With this approach, no switching

of the controller is required. Finally, by substituting

$$\ddot{\mathbf{q}}_A = \mathbf{u} = \mathbf{u}^d + \Delta\mathbf{u}_k \ ,$$

with $\Delta\mathbf{u}_k$ from (5.23), into (5.2b), an expression for the motor torque $\boldsymbol{\tau}_A$ is found, reading as, see also (5.8) and (5.9)

$$\boldsymbol{\tau}_{A,k} = \widetilde{\mathbf{M}}(\mathbf{q}_k)(\mathbf{u}^d + \mathbf{K}_k\boldsymbol{\Delta}\mathbf{x}_k) + \widetilde{\mathbf{C}}(\mathbf{q}_k, \dot{\mathbf{q}}_k)\dot{\mathbf{q}}_k + \widetilde{\mathbf{g}}(\mathbf{q}_k) \ . \tag{5.25}$$

5.5 Simulation and experimental results

The offline and online trajectory optimization is implemented in MATLAB/ SIMULINK R2019b using a standard computer equipped with a 1.8 GHz Intel Core i7-8550K and 16 GB RAM. The nonlinear optimization problem (5.5) is solved using the state-of-the-art interior-point solver IPOPT together with the linear solver MA57, see, e.g., [71], [108]. In order to enhance the computational speed, automatic differentiation (AD) of the cost function and the constraint functions is implemented with the CasADi package [77].

The experimental setup, shown in Fig. 5.3, comprises three main components, i.e. the ceiling-mounted robot KUKA LBR iiwa 14 R820, the PC running MATLAB/SIMULINK modules inside the real-time automation software BECKHOFF TwinCAT, and the spherical pendulum tool. The PC communicates with the robot and the spherical pendulum tool via two network interface cards (NIC) using the EtherCAT protocol.

Each trajectory is discretized into $N = 100$ collocation points, yielding a total of 2501 optimization variables. For safety reasons, the constraints on the joint velocities and motor torques according to Tab. 5.1 are reduced by 50%. Without loss of generality and to avoid collisions with the surroundings, the angles of four joints q_1, q_2, q_4, and q_6 of the starting state $\mathbf{x}_S$ are discretized equidistantly in the ranges $q_1 \in [-60°, 45°]$, $q_2 \in [-30°, 30°]$, $q_4 \in [-120°, -60°]$, and $q_6 \in [-60°, 60°]$. The remaining joint angles are fixed at 0. In each admissible range, four grid points are used. Thus, the number of offline planned swing-up trajectories in the database is 256. The total computing time for the offline database is about 43 min, which results in an average computing time for one trajectory of 10 s. In order to verify the efficiency of the fast trajectory replanner, a Monte Carlo simulation

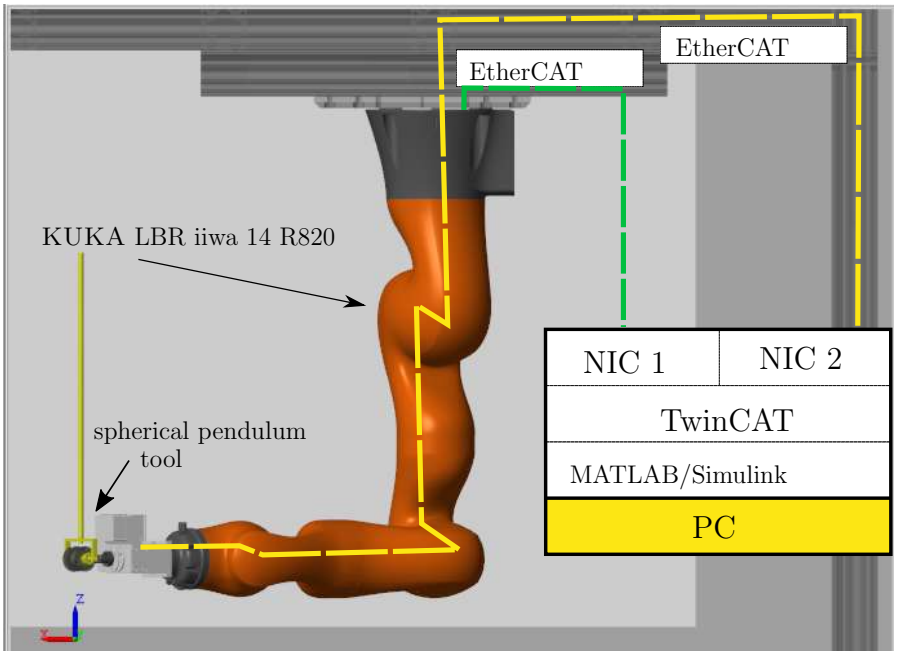


Figure 5.3: Overview of the experimental setup.

is performed by randomly selecting 10^4 initial states $\tilde{\mathbf{x}}_S$ from a uniform random distribution in the admissible ranges. Even though the offline database is built using a very coarse grid, all 10^4 test cases in the Monte Carlo simulation provide feasible solutions and adhere to the input and state constraints. The computing time for a fast trajectory replanning is approximate 200 ms on average, which is 50 times faster than the offline trajectory optimization.

Figure 5.4 illustrates the simulation results of the fast trajectory replanner from the starting state $\tilde{\mathbf{x}}_S^{\mathrm{T}} = [-15°, -15°, 0, -95°, 0, 10°, 0, 0, 0]$ in solid lines, which deviates from the closest grid point in the database, i.e. the starting state $\mathbf{x}_S^{\mathrm{T}} = [0, 0, 0, -90°, 0, 0, 0, 0, 0]$ shown as dashed lines. The trajectories in Fig. 5.4 are demonstrated in simulation without the closed-loop control from Section 5.4. Note that for both, the offline trajectory optimization and the fast trajectory replanner, a straight line from the starting state to the target state is used as an initial guess. Both trajectories are feasible with respect to the dynamical constraints (5.5b) as well as the limits of the robot (5.5d), (5.10), and (5.7).

In the experiment, the joint velocities of the complete system cannot be measured directly. Therefore, differential filters with a time constant of $T_1 = 12$ ms are used

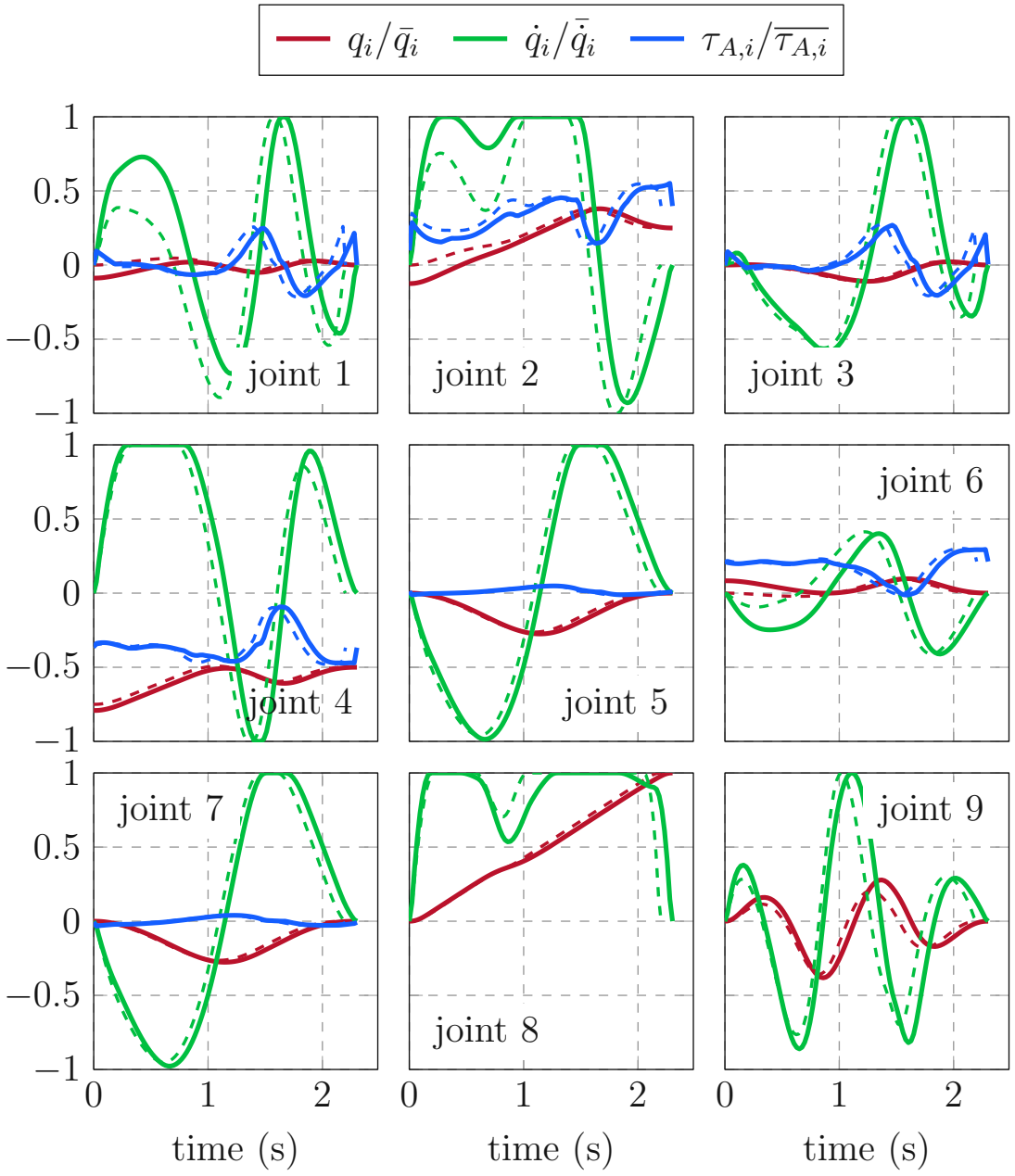


Figure 5.4: Time evolution of the offline and the replanned trajectory optimization. The dashed lines are the scaled offline trajectory from the database in Section 5.2, and the solid lines represent the scaled replanned trajectory from Section 5.3. For safety reasons, the limits for the joint velocities and the motor torques are 50% lower than the limits in Tab. 5.1.

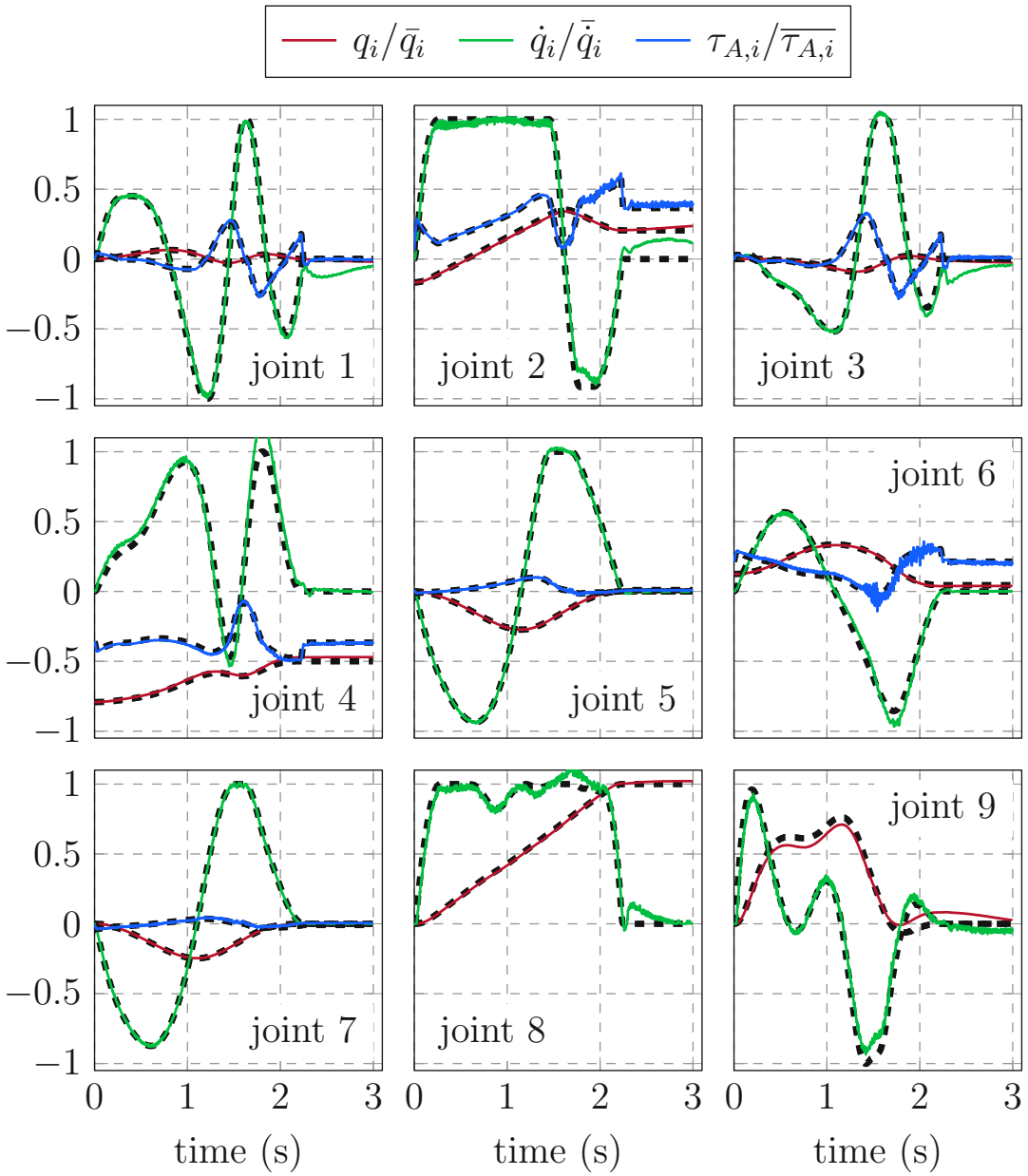


Figure 5.5: Experimental results for the swing-up and stabilization of the spherical pendulum with the fast trajectory replanner. The colored solid lines present the scaled measurement data, and the corresponding black dashed lines show the desired trajectory from Section 5.3.

Table 5.1: Kinematic and dynamic limits of the compete system consisting of the robot KUKA LBR iiwa 14 R820 and the spherical pendulum in Fig. 2.5.

Joint i	**Angle limits** $\overline{q_i}$ (°)	**Velocity limits** $\overline{\dot{q}_i}$ (°/s)	**Torque limits** $\overline{\tau_{A,i}}$ (N)
1	170	85	320
2	120	85	320
3	170	100	176
4	120	75	176
5	170	130	110
6	120	135	40
7	175	135	40
8	180	180	–
9	45	180	–

for each joint. The weighting matrices in (5.21) are chosen as

$$\mathbf{Q}_c^{\mathrm{T}} = \mathrm{diag}([\mathbf{Q}_{c,\mathbf{q}}^{\mathrm{T}}, \mathbf{Q}_{c,\dot{\mathbf{q}}}^{\mathrm{T}}]) \ ,$$

with

$$\mathbf{Q}_{c,\mathbf{q}}^{\mathrm{T}} = [1\,1\,1\,1\,1\,1\,1\,10^{-3}\,10^{-2}] \ ,$$
$$\mathbf{Q}_{c,\dot{\mathbf{q}}}^{\mathrm{T}} = 10^{-1}[1\,1\,1\,1\,1\,1\,1\,10^{-2}\,10^{-1}] \ ,$$

and $\mathbf{R}_c = 10^{-1}\mathbf{I}_7$. The weights of the pendulum velocities $\dot{q}_8$ and $\dot{q}_9$ in $\mathbf{Q}_c$ are reduced to avoid oscillations during the swing-up. Note that the sampling time for the controller is $T_s = 125$ µs, while the time step $h = t_F/N \approx 300$ ms of the trajectory is significantly larger. Therefore, the desired trajectory from the fast replanner has to be interpolated using (4.12). The torque input $\boldsymbol{\tau}_A$ is calculated from (5.25) with the time-variant feedback matrix $\mathbf{K}_k$ from (5.24). The experimental results of the swing-up of the spherical pendulum from the initial state $\tilde{\mathbf{x}}_S^{\mathrm{T}} = [0, -15°, 0, -95°, 0, 10°, 0, 0, 0]$ are presented in Fig. 5.5. The measured signals are shown as colored solid lines, and the desired trajectories are drawn as black dashed lines. Note that the discrete time-variant LQR is able to stabilize the swing-up trajectory and the upright position of the spherical pendulum using the KUKA LBR iiwa 14 R820. The small oscillations around the trajectory in Fig. 5.5 can be attributed to the parameter and model uncertainties and to the measurement noise present in the velocity signals. In addition, a video of several demonstrations of the swing-up and stabilization from random starting states $\tilde{\mathbf{x}}_S$

is provided at `www.acin.tuwien.ac.at/8c60`.

5.6 Conclusions

In this section, the fast trajectory optimization for the swing-up of a spherical pendulum on a 7-axis robot is presented and experimentally validated. First, the swing-up trajectories are computed offline using the direct collocation trajectory optimization approach and are stored in a k-d tree database. The offline trajectory optimization employed the state-of-the-art nonlinear solver IPOPT and takes into account the system constraints of the complete system to conclude the swing-up motion in approximately 10 s on average. For an arbitrary initial configuration of the system, a nearest neighbor search algorithm is applied to find the nearest configuration in the k-d tree database. This trajectory is adapted by solving a constrained quadratic optimization, which takes on average 200 ms even if the grid in the database is relatively coarse. Finally, the experimental demonstration shows the successful swing-up and stabilization of a spherical pendulum tool mounted on the robot KUKA LBR iiwa 14 R820.

6

Conclusions and Outlook

The main focus of this thesis is the development of a fast trajectory planning framework that can be used for repetitive motion planning tasks of robotic systems, e.g., moving goods in factories, warehouses, and ports. In particular, the following scenario is considered. A lab-scale 3D gantry crane has to move goods or material from one specific place (starting space) to another place (target space) in a static environment with known obstacles. The task is to plan a time-optimal trajectory from a given starting point to a given target point in real time, which systematically accounts for both the obstacles and the dynamic constraints on the state variables and control inputs. In these tasks, the starting and target states are changed only slightly, however, state-of-the-art trajectory planning approaches require long computation times to calculate the entire trajectory from scratch. This results in a waste of computational resources and makes it impossible to handle targets that are moving. The lack of a viable approach in the literature provides the motivation for this work: *Is there a trajectory planning framework that can generate collision-free trajectories in real time, taking into account system constraints and a dynamically moving target?*.

In Section 3, a novel sampling-based trajectory planning method, named flat informed-optimal rapid exploring random tree (RRT*), was proposed. With the

given information, such as the dimension of the workspace and the location of the obstacles, the proposed flat informed RRT* creates a trajectory tree that serves as a basis for the replanning process when the target state changes. Different from other motion planning approaches in the literature, the proposed flat informed RRT* incorporates a local planner based on a linear quadratic minimum-time (LQMT) solver. The smoothness of the computed trajectories is ensured by the local planner. Additionally, the informed set is used to eliminate points in the tree. This helps to reduce the memory consumption and computational complexity of the proposed algorithm. Simulation studies for different scenarios show the feasibility of the proposed sampling-based algorithm. The computing time, statically measured via Monte Carlo simulations, of the online replanning that reuses the computed trajectory tree, is less than 75 ms. Although the proposed algorithm is applicable to other robotic systems, such as unmanned aerial vehicles, the computation time is not yet sufficient for real-time applications.

To address this deficiency, a novel two-step optimization-based trajectory framework was proposed in Section 4. This proposed algorithm computes a collision-free, dynamically feasible trajectory that directs the 3D gantry crane from an initial state to a target state (a moving truck) in an environment with static obstacles. Moreover, this algorithm enables online replanning when the target state moves. A model predictive controller (MPC)-based trajectory tracking controller was further implemented to account for model uncertainties and disturbances. In the first step of the proposed algorithm, a collision-free offline trajectory database was built using the direct optimization method. The computation time for one offline trajectory in the offline database is about 50 ms on average on a standard PC. Using the offline database, the online trajectory planner minimizes the deviation from a suitably selected trajectory in the database considering the current speed of the gantry crane and the moving target. The simulation results show that the online trajectory replanner provides a feasible trajectory within an average computation time of 2.5 ms, which is 25 faster than the offline trajectory optimization. Moreover, Monte Carlo simulations demonstrate the effectiveness of the proposed approach with a success rate of over 97% in two scenarios, i.e., the stationary target scenario and the moving target scenario. In the experiments, the proposed trajectory planning framework and the MPC-based tracking controller were implemented on the dSPACE MicroLabBox 1202 with a sampling time of 15 ms and 1 ms, respectively. The online replanner successfully generates collision-free

and dynamically feasible trajectories for the moving truck. The trajectory tracking controller exhibits a good tracking performance. While the deviations of the sway angles from the desired trajectory are visible, they are limited in the range $[-4^\circ\,, 4^\circ]$. To reduce the complexity of the lab experiment, an OptiTrack system was used to detect the truck position. For real-world applications in a larger or more complex environment, we need to use other methods, such as GPS and/or visual methods for obstacle detection and tracking.

In Section 5, the two-step trajectory planning framework was applied to swing up a spherical pendulum with a 7-axis robot. Again, in the first step, an offline trajectory database for swinging up the spherical pendulum was built. The computation time of an offline trajectory in the database is approximately 10 s, which outperforms recent works in the literature. Next, an online trajectory replanner was presented which allows to quickly compute a dynamically feasible trajectory from an arbitrary initial configuration to an upswing position. The computation time of one trajectory with the online trajectory replanner is approximately 200 ms. Using a discrete time-variant linear quadratic regulator (LQR), experimental demonstrations show the successful swing-up and stabilization of the spherical pendulum on the 7-DoF KUKA LBR iiwa 14 R820.

The proposed two-stage trajectory planning was also successfully applied to a task for dynamically grasping a 3D object with a collaborative robot [109]. Here, the proposed algorithm plays an important role in iteratively computing online trajectories of the robot while approaching the object. In future work, extensions of the proposed two-step trajectory planning will be investigated for dynamically changing obstacles. Furthermore, in a mid-term perspective, the real-time trajectory planning framework will be combined with environment detection and vision-based object tracking.

A

Parameters

A.1 Parameters of the lab-scale 3D gantry crane

Name	Value	Unit	Name	Value	Unit
I_x	39.99	$\mathrm{kg\,cm^2}$	R_x	38	mm
I_y	32.89	$\mathrm{kg\,cm^2}$	R_y	38	mm
I_z	41.71	$\mathrm{kg\,cm^2}$	R_z	13.25	mm
I_α	86.52	$\mathrm{kg\,cm^2}$	b_1	43.5	mm
I_β	71.72	$\mathrm{kg\,cm^2}$	h_1	61	mm
$s_{x,0}$	215	mm	m_x	4.43	kg
$s_{y,0}$	275	mm	m_y	1.62	kg
$s_{z,0}$	95	mm	m_z	2.16	kg
$s_{z,max}$	1000	mm	g	9.81	$\mathrm{m\,s^{-2}}$

Table A.1: System parameters of the lab-scale 3D gantry crane.

A.2 Parameters of the 7-axis KUKA LBR iiwa 14 R820 and the spherical pendulum.

Name	Value	Unit	Name	Value	Unit
d_1	0.1525	m	d_5	0.2175	m
d_2	0.2025	m	d_6	0.1825	m
d_3	0.2325	m	d_7	0.081	m
d_4	0.1825	m	d_8	0.155	m
d_9	0.073	m	m_1	6.495	kg
m_2	8.807	kg	m_5	1.889	kg
m_3	2.8	kg	m_6	2.32	kg
m_4	5.283	kg	m_7	1.56	kg
m_8	0.23	kg	m_9	0.186	kg
$I_{1,xx}$	0.069	$\mathrm{kg\,m^2}$	$I_{1,yy}$	0.071	$\mathrm{kg\,m^2}$
$I_{1,zz}$	0.02	$\mathrm{kg\,m^2}$	$I_{2,xx}$	0.082	$\mathrm{kg\,m^2}$
$I_{2,yy}$	0.016	$\mathrm{kg\,m^2}$	$I_{2,zz}$	0.087	$\mathrm{kg\,m^2}$
$I_{3,xx}$	0.023	$\mathrm{kg\,m^2}$	$I_{3,yy}$	0.022	$\mathrm{kg\,m^2}$
$I_{3,zz}$	0.055	$\mathrm{kg\,m^2}$	$I_{4,xx}$	0.047	$\mathrm{kg\,m^2}$
$I_{4,yy}$	0.009	$\mathrm{kg\,m^2}$	$I_{4,zz}$	0.046	$\mathrm{kg\,m^2}$
$I_{5,xx}$	0.014	$\mathrm{kg\,m^2}$	$I_{5,yy}$	0.011	$\mathrm{kg\,m^2}$
$I_{5,zz}$	0.0057	$\mathrm{kg\,m^2}$	$I_{6,xx}$	0.007	$\mathrm{kg\,m^2}$
$I_{6,yy}$	0.004	$\mathrm{kg\,m^2}$	$I_{6,zz}$	0.006	$\mathrm{kg\,m^2}$
$I_{7,xx}$	0.003	$\mathrm{kg\,m^2}$	$I_{7,yy}$	0.002	$\mathrm{kg\,m^2}$
$I_{7,zz}$	0.0021	$\mathrm{kg\,m^2}$	$I_{8,xx}$	0.003	$\mathrm{kg\,m^2}$
$I_{8,yy}$	0.000027	$\mathrm{kg\,m^2}$	$I_{8,zz}$	0.0035	$\mathrm{kg\,m^2}$
$I_{9,xx}$	0.013	$\mathrm{kg\,m^2}$	$I_{9,yy}$	0.012	$\mathrm{kg\,m^2}$
$I_{9,zz}$	0.008	$\mathrm{kg\,m^2}$	g	-9.81	$\mathrm{m\,s^{-2}}$

Table A.2: Parameters of the 7-axis KUKA LBR iiwa 14 R820 and the spherical pendulum.

Bibliography

[1] M. Shariatee, A. Akbarzadeh, A. Mousavi, and S. Alimardani, "Design of an economical scara robot for industrial applications," *Proceedings of the RSI/ISM International Conference on Robotics and Mechatronics*, pp. 534–539, 2014.

[2] L. Rey and R. Clavel, *The delta parallel robot.* Springer: London, United Kingdom, 1999.

[3] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, 2019.

[4] P. Aivaliotis, S. Aivaliotis, C. Gkournelos, K. Kokkalis, G. Michalos, and S. Makris, "Power and force limiting on industrial robots for human-robot collaboration," *Robotics and Computer-Integrated Manufacturing*, vol. 59, pp. 346–360, 2019.

[5] O. Sawodny, H. Aschemann, and S. Lahres, "An automated gantry crane as a large workspace robot," *Control Engineering Practice*, vol. 10, no. 12, pp. 1323–1338, 2002.

[6] M. Al-Hussein, M. A. Niaz, H. Yu, and H. Kim, "Integrating 3d visualization and simulation for tower crane operations on construction sites," *Automation in construction*, vol. 15, no. 5, pp. 554–562, 2006.

[7] J. Kalmari, J. Backman, and A. Visala, "Nonlinear model predictive control of hydraulic forestry crane with automatic sway damping," *Computers and*

Electronics in Agriculture, vol. 109, pp. 36–45, 2014.

[8] R. Mueller, M. Vette, and A. Geenen, "Skill-based dynamic task allocation in human-robot-cooperation with the example of welding application," *Procedia Manufacturing*, vol. 11, pp. 13–21, 2017.

[9] B. Wang, S. J. Hu, L. Sun, and T. Freiheit, "Intelligent welding system technologies: State-of-the-art review and perspectives," *Journal of Manufacturing Systems*, vol. 56, pp. 373–391, 2020.

[10] S. Zhang, H. Huang, D. Huang, L. Yao, J. Wei, and Q. Fan, "Subtask-learning based for robot self-assembly in flexible collaborative assembly in manufacturing," *The International Journal of Advanced Manufacturing Technology*, vol. 120, no. 9-10, pp. 6807–6819, 2022.

[11] T. Weingartshofer, B. Bischof, M. Meiringer, C. Hartl-Nesic, and A. Kugi, "Optimization-based path planning framework for industrial manufacturing processes with complex continuous paths," *Robotics and Computer-Integrated Manufacturing*, vol. 82, p. 102516, 2023.

[12] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," *Proceedings of the International Conference on Robotics and Automation*, pp. 1478–1483, 2011.

[13] M. P. Strub and J. D. Gammell, "Adaptively Informed Trees (AIT): Fast asymptotically optimal path planning through adaptive heuristics," *Proceedings of the International Conference on Robotics and Automation*, pp. 3191–3198, 2020.

[14] S. M. La Valle, "Motion planning," *IEEE Robotics & Automation Magazine*, vol. 18, no. 2, pp. 108–118, 2011.

[15] A. V. Rao, *Trajectory optimization: a survey*. Springer: Cham, Switzerland, 2014.

[16] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," *Proceedings of the International Conference on Robotics and Automation*, pp. 3067–3074, 2015.

[17] M. Böck and A. Kugi, "Real-time nonlinear model predictive path-following control of a laboratory tower crane," *IEEE Transactions on Control Systems*

Technology, vol. 22, no. 4, pp. 1461–1473, 2013.

[18] Y. R. Stürz, L. M. Affolter, and R. S. Smith, "Parameter identification of the kuka lbr iiwa robot including constraints on physical feasibility," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6863–6868, 2017.

[19] M. N. Vu, P. Zips, A. Lobe, F. Beck, W. Kemmetmüller, and A. Kugi, "Fast motion planning for a laboratory 3d gantry crane in the presence of obstacles," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 9508–9514, 2020.

[20] M. N. Vu, C. Hartl-Nesic, and A. Kugi, "Fast swing-up trajectory optimization for a spherical pendulum on a 7-dof collaborative robot," *Proceedings of the International Conference on Robotics and Automation*, pp. 10 114–10 120, 2021.

[21] M. N. Vu, A. Lobe, F. Beck, T. Weingartshofer, C. Hartl-Nesic, and A. Kugi, "Fast trajectory planning and control of a lab-scale 3d gantry crane for a moving target in an environment with obstacles," *Control Engineering Practice*, vol. 126, p. 105255, 2022.

[22] M. N. Vu, M. Schwegel, C. Hartl-Nesic, and A. Kugi, "Sampling-based trajectory (re)planning for differentially flat systems: Application to a 3d gantry crane," *IFAC-PapersOnLine*, vol. 55, no. 38, pp. 33–40, 2022.

[23] F. Beck, M. N. Vu, C. Hartl-Nesic, and A. Kugi, "Singlularity avoidance with application to online trajectory optimization for serial manipulators," *IFAC-PapersOnLine*, [Accepted for IFAC World Congress], 2023.

[24] M. Zimmermann, M. N. Vu, F. Beck, C. Hartl-Nesic, and A. Kugi, "Two-step online trajectory planning of a quadcopter in indoor environments with obstacles," *IFAC-PapersOnLine*, [Accepted for IFAC World Congress], 2023.

[25] M. Vu, F. Beck, M. Schwegel, C. Hartl-Nesic, A. Nguyen, and A. Kugi, "Machine learning-based framework for optimally solving the analytical inverse kinematics for redundant manipulators," *Mechatronics*, vol. 91, p. 102970, 2023.

[26] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley: New Jersey, USA, 2005.

[27] A. Lobe, A. Ettl, A. Steinböck, and A. Kugi, "Flatness-based nonlinear control of a three-dimensional gantry crane," *IFAC-PapersOnLine*, vol. 51,

no. 22, pp. 331–336, 2018.

[28] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[29] B. Kolar, H. Rams, and K. Schlacher, "Time-optimal flatness based control of a gantry crane," *Control Engineering Practice*, vol. 60, pp. 18–27, 2017.

[30] J. Kretschmer, "Swing-up and stabilization of a spherical inverted pendulum on a robot," Master's thesis, Automation and Control Institute (ACIN), TU Wien, 2021.

[31] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, and W. Burgard, *Principles of robot motion: theory, algorithms, and implementations*. MIT press: Cambridge, USA, 2005.

[32] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[33] D. Ferguson, M. Likhachev, and A. Stentz, "A guide to heuristic-based path planning," *Proceedings of the International Workshop on Planning under Uncertainty for Autonomous Systems*, pp. 9–18, 2005.

[34] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1-2, pp. 93–146, 2004.

[35] D. Ferguson and A. Stentz, "The delayed D* algorithm for efficient path replanning," *Proceedings of the International Conference on Robotics and Automation*, pp. 2045–2050, 2005.

[36] S. Koenig and M. Likhachev, "Improved fast replanning for robot navigation in unknown terrain," *Proceedings of the International Conference on Robotics and Automation*, pp. 968–975, 2002.

[37] M. Likhachev, G. J. Gordon, and S. Thrun, "ARA*: Anytime A* with provable bounds on sub-optimality," *Advances in Neural Information Processing Systems*, vol. 16, pp. 767–774, 2003.

[38] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artificial Intelligence*, vol. 31, no. 3, pp. 295–353, 1987.

[39] K. Kondo, "Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 267–277, 1991.

[40] M. Cherif, "Kinodynamic motion planning for all-terrain wheeled vehicles," *Proceedings of the International Conference on Robotics and Automation*, pp. 317–322, 1999.

[41] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

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

[43] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," *Proceedings of the International Conference on Robotics and Automation*, pp. 521–528, 2000.

[44] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-SMART: A rapid convergence implementation of RRT," *International Journal of Advanced Robotic Systems*, vol. 10, no. 7, p. 299, 2013.

[45] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 2997–3004, 2014.

[46] M. P. Strub and J. D. Gammell, "Advanced BIT*(ABIT*): Sampling-based planning with advanced graph-search techniques," *Proceedings of the International Conference on Robotics and Automation*, pp. 130–136, 2020.

[47] C. Xie, J. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," *Proceedings of the International Conference on Robotics and Automation*, pp. 4187–4194, 2015.

[48] H. B. Keller, *Numerical methods for two-point boundary-value problems*. Courier Dover Publications: New York, USA, 2018.

[49] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically

optimal motion planning for robots with linear dynamics," *Proceedings of the International Conference on Robotics and Automation*, pp. 5054–5061, 2013.

[50] E. Verriest and F. Lewis, "On the linear quadratic minimum-time problem," *IEEE Transactions on Automatic Control*, vol. 36, no. 7, pp. 859–863, 1991.

[51] D. J. Evans, "Iterative methods for solving non-linear two point boundary value problems," *International Journal of Computer Mathematics*, vol. 72, no. 3, pp. 395–401, 1999.

[52] Q. Chen, Y. Zhang, S. Liao, and F. Wan, "Newton–kantorovich/pseudospectral solution to perturbed astrodynamic two-point boundary-value problems," *Journal of Guidance, Control, and Dynamics*, vol. 36, no. 2, pp. 485–498, 2013.

[53] E. Delaleau and J. Rudolph, "Decoupling and linearization by quasi-static feedback of generalized states," *Proceedings of the European Control Conference*, pp. 1069–1074, 1995.

[54] B. Kolar and K. Schlacher, "Flatness based control of a gantry crane," *Proceedings of the 9th IFAC Symposium on Nonlinear Control Systems*, pp. 487–492, 2013.

[55] D. C. Kozen, *The Design and Analysis of Algorithms*. Springer: New York, USA, 1992.

[56] C. R. Maurer, R. Qi, and V. Raghavan, "A linear time algorithm for computing exact euclidean distance transforms of binary images in arbitrary dimensions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[57] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[58] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[59] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics*

Research, vol. 32, no. 9-10, pp. 1164–1193, 2013.

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

[61] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[62] S. Iftikhar, O. J. Faqir, and E. C. Kemgan, "Nonlinear model predictive control of an overhead laboratory-scale gantry crane with obstacle avoidance," *Proceedings of the Conference on Control Technology and Applications*, pp. 382–387, 2019.

[63] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2020.

[64] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots," *Proceedings of the IEEE Conference on Robotics and Automation*, pp. 1398–1403, 2002.

[65] S. M. Khansari-Zadeh and A. Billard, "Learning Stable Nonlinear Dynamical Systems with Gaussian Mixture Models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[66] A. Abdullahi, Z. Mohamed, H. Selamat, H. Pota, M. Z. Abidin, and S. Fasih, "Efficient control of a 3D overhead crane with simultaneous payload hoisting and wind disturbance: design, simulation and experiment," *Mechanical Systems and Signal Processing*, vol. 145, p. 106893, 2020.

[67] X. Wang, J. Liu, Y. Zhang, B. Shi, D. Jiang, and H. Peng, "A unified symplectic pseudospectral method for motion planning and tracking control of 3d underactuated overhead cranes," *International Journal of Robust and Nonlinear Control*, vol. 29, no. 7, pp. 2236–2253, 2019.

[68] W. Zhang, H. Chen, H. Chen, and W. Liu, "A time optimal trajectory planning method for double-pendulum crane systems with obstacle avoidance," *IEEE Access*, vol. 9, pp. 13 022–13 030, 2021.

[69] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Siam: Philadelphia, USA, 2010.

[70] M. Kelly, "An introduction to trajectory optimization: how to do your own direct collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[71] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[72] N. T. An, D. Giles, N. M. Nam, and R. B. Rector, "The log-exponential smoothing technique and nesterov's accelerated gradient method for generalized sylvester problems," *Journal of Optimization Theory and Applications*, vol. 168, no. 2, pp. 559–583, 2016.

[73] F. Nielsen and K. Sun, "Guaranteed bounds on information-theoretic measures of univariate mixtures using piecewise log-sum-exp inequalities," *Entropy*, vol. 18, no. 12, p. 442, 2016.

[74] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[75] R. Pinkham, S. Zeng, and Z. Zhang, "Quicknn: Memory and performance optimization of kd tree based nearest neighbor search for 3d point clouds," *Proceedings of the International Symposium on High Performance Computer Architecture*, pp. 180–192, 2020.

[76] M. Soleymani and S. Morgera, "An efficient nearest neighbor search method," *IEEE Transactions on Communications*, vol. 35, no. 6, pp. 677–679, 1987.

[77] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[78] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[79] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[80] K. H. Lundberg and T. W. Barton, "History of Inverted-pendulum Systems," *IFAC Proceedings Volumes*, vol. 42, no. 24, pp. 131–135, 2010.

[81] Y. Xu, M. Iwase, and K. Furuta, "Time Optimal Swing-up Control of Single Pendulum," *Journal of Dynamic Systems, Measurement, and Control*, vol. 123, no. 3, pp. 518–527, 2001.

[82] M. Yamakita, M. Iwashiro, Y. Sugahara, and K. Furuta, "Robust Swing Up Control of Double Pendulum," *Proceedings of the American Control Conference*, vol. 1, pp. 290–295, 1995.

[83] T. Glück, A. Eder, and A. Kugi, "Swing-up control of a triple pendulum on a cart with experimental validation," *Automatica*, vol. 49, no. 3, pp. 801–808, 2013.

[84] J. Shen, A. K. Sanyal, N. A. Chaturvedi, D. Bernstein, and H. McClamroch, "Dynamics and control of a 3D pendulum," *Proceedings of the International Conference on Decision and Control*, vol. 1, pp. 323–328, 2004.

[85] R. H. Cannon, *Dynamics of physical systems.* McGraw-Hill: California, USA, 1967.

[86] K. Ogata and Y. Yang, *Modern control engineering.* Prentice Hall: New Jersey, USA, 1970.

[87] S. Mori, H. Nishihara, and K. Furuta, "Control of unstable mechanical system control of pendulum," *International Journal of Control*, vol. 23, no. 5, pp. 673–692, 1976.

[88] A. Mills, A. Wills, and B. Ninness, "Nonlinear model predictive control of an inverted pendulum," *Proceedings of the American Control Conference*, pp. 2335–2340, 2009.

[89] E. Derner, J. Kubalík, and R. Babuška, "Reinforcement learning with symbolic input-output models," *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 3004–3009, 2018.

[90] T. Chen and B. Goodwinel, "A simple approach on global control of a class of underactuated mechanical robotic systems," *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 5139–5145, 2019.

[91] K. Furuta, M. Yamakita, and S. Kobayashi, "Swing-up control of inverted

pendulum using pseudo-state feedback," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 206, no. 4, pp. 263–269, 1992.

[92] K. Graichen and M. Zeitz, "Feedforward control design for finite-time transition problems of nonlinear systems with input and output constraints," *IEEE Transactions on Automatic Control*, vol. 53, no. 5, pp. 1273–1278, 2008.

[93] A. Winkler and J. Suchỳ, "Erecting and balancing of the inverted pendulum by an industrial robot," *IFAC Proceedings Volumes*, vol. 42, no. 16, pp. 323–328, 2009.

[94] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, "Automatic LQR tuning based on gaussian process global optimization," *Proceedings of the International Conference on Robotics and Automation*, pp. 270–277, 2016.

[95] A. Doerr, D. Nguyen-Tuong, A. Marco, S. Schaal, and S. Trimpe, "Model-based policy search for automatic tuning of multivariate PID controllers," *Proceedings of the International Conference on Robotics and Automation*, pp. 5295–5301, 2017.

[96] G. Schreiber, C. Ott, and G. Hirzinger, "Interactive redundant robotics: Control of the inverted pendulum with nullspace motion," *Proceedings of the International Conference on Intelligent Robots and Systems*, vol. 1, pp. 158–164, 2001.

[97] C. Hartl-Nesic, J. Kretschmer, M. Schwegel, T. Glück, and A. Kugi, "Swing-up of a spherical pendulum on a 7-axis industrial robot," *IFAC-PapersOnLine*, vol. 52, no. 15, pp. 346–351, 2019.

[98] S. Farzan, A.-P. Hu, E. Davies, and J. Rogers, "Modeling and control of brachiating robots traversing flexible cables," *Proceedings of the International Conference on Robotics and Automation*, pp. 1645–1652, 2018.

[99] K.-D. Nguyen and D. Liu, "Robust control of a brachiating robot," *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 6555–6560, 2017.

[100] S. Farzan, A.-P. Hu, E. Davies, and J. Rogers, "Feedback motion planning and control of brachiating robots traversing flexible cables," *Proceedings of the American Control Conference*, pp. 1323–1329, 2019.

[101] P. Reist, P. Preiswerk, and R. Tedrake, "Feedback-motion-planning with simulation-based LQR-trees," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1393–1416, 2016.

[102] C. Liu and C. G. Atkeson, "Standing balance control using a trajectory library," *Proceedings of the International Conference on Intelligent Robots and Systems*, pp. 3031–3036, 2009.

[103] E. E. Binder and J. H. Herzog, "Distributed computer architecture and fast parallel algorithms in real-time robot control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 4, pp. 543–549, 1986.

[104] R. Marimont and M. Shapiro, "Nearest neighbour searches and the curse of dimensionality," *IMA Journal of Applied Mathematics*, vol. 24, no. 1, pp. 59–70, 1979.

[105] J. J. Moré and G. Toraldo, "Algorithms for bound constrained quadratic programming problems," *Numerische Mathematik*, vol. 55, no. 4, pp. 377–400, 1989.

[106] S. Bittanti, A. J. Laub, and J. C. Willems, *The Riccati Equation*. Springer: Berlin, Heidelberg, 2012.

[107] G. F. Franklin, J. D. Powell, A. Emami-Naeini, and J. D. Powell, *Feedback control of dynamic systems*. Prentice Hall, 2002.

[108] I. S. Duff, "MA57—A Code for the Solution of Sparse Symmetric Definite and Indefinite Systems," *ACM Transactions on Mathematical Software*, vol. 30, no. 2, pp. 118–144, 2004.

[109] F. Grander, "Dynamisches Greifen von 3D-Objekten mittels robotischem System," Master's thesis, in German, Automation and Control Institute (ACIN), TU Wien, 2021.