BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF MECHANICAL ENGINEERING

FAKULTA STROJNÍHO INŽENÝRSTVÍ

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

ÚSTAV AUTOMATIZACE A INFORMATIKY

DESIGN OF ADVANCED METHODS IN THE FIELD OF INDUSTRIAL ROBOTICS FITTING INTO THE CONCEPT OF INDUSTRY 4.0.

NÁVRH POKROČILÝCH METOD V OBLASTI PRŮMYSLOVÉ ROBOTIKY ZAPADAJÍCÍCH DO KONCEPTU PRŮMYSLU 4.0.

DOCTORAL THESIS

DIZERTAČNÍ PRÁCE

AUTHOR — **Ing. Roman Parák**

AUTOR PRÁCE

SUPERVISOR — **prof. Ing. Radomil Matoušek, Ph.D.**

ŠKOLITEL

BRNO 2024

Abstract

The significance of robot manipulators in engineering applications and scientific research has increased substantially in recent years. This surge can be attributed to the increased use of technologies such as autonomous robotics, digital twins, complex system integration, etc., along with the utilization of artificial intelligence techniques. The limitation in the use of these technologies may be associated with a focus on a specific type of robotic manipulator and a particular solved task, hindering modularity and reproducibility in future expansions.

This dissertation focuses on the design and development of advanced methods in industrial robotics that are closely related to the principles of Industry 4.0. It encompasses the development of comprehensive kinematic solutions and motion planning strategies for various robotic manipulators, including under-articulated, articulated, and over-articulated structures. To ensure computational efficiency, a collision avoidance structure is designed to prevent collisions with external objects or self-collision. Within the area of motion planning, a key research objective involves exploring innovative methods using reinforcement learning. Another significant research objective relates to the design of a versatile robotic workstation in accordance with the Industry 4.0 concept. This entails incorporating vertical system integration for seamless interoperability and optimal interconnection of diverse systems within the workstation. The dissertation objective also includes the design of a sophisticated user interface and a unique simulation tool based on Unity3D. A critical aspect of this dissertation focuses on versatility, education, and future expansion. The purpose of this dissertation is to contribute not only to the advancement of industrial robotics but also to its application in educational settings.

Keywords

motion planning, kinematics, deep reinforcement learning, collision avoidance, industry 4.0, physics-based simulation, industrial robotics

Abstrakt

Význam robotických manipulátorů v inženýrských aplikacích a vědeckém výzkumu v posledních letech podstatně vzrostl. Tento nárůst lze přičíst zvýšenému využívání technologií, jako je autonomní robotika, digitální dvojčata, komplexní systémová integrace atd., spolu s využitím technik umělé inteligence. Omezení v použití těchto technologií může být spojeno se zaměřením na konkrétní typ robotického manipulátoru a konkrétní řešený úkol, což brání modularitě a reprodukovatelnosti v budoucím rozšiřování.

Tato disertační práce se zaměřuje na návrh a implementaci pokročilých metod v průmyslové robotice, které úzce souvisejí s principy Průmyslu 4.0. Zahrnuje vývoj komplexních kinematických řešení a strategií plánování pohybu pro různé robotické manipulátory, včetně redukovaných, standardních i redundantních struktur. Pro zajištění výpočetní efektivity je navržena kolizní struktura, která zabraňuje kolizím s vnějšími objekty nebo kolizím se sebou samým. V oblasti plánování pohybu je klíčovým výzkumným cílem zkoumání inovativních metod využívajících zpětnovazební učení. Další významný výzkumný cíl se týká návrhu všestranného robotického pracoviště v souladu s konceptem Průmysl 4.0. To znamená začlenění vertikální systémové integrace pro bezproblémovou interoperabilitu a optimální propojení různých systémů v rámci pracoviště. Cílem disertační práce je také návrh propracovaného uživatelského rozhraní a unikátního simulačního nástroje založeného na Unity3D. Kritický aspekt této disertační práce se zaměřuje na všestrannost, vzdělávání a budoucí expanzi. Smyslem disertační práce je přispět nejen k rozvoji průmyslové robotiky, ale také k jejímu uplatnění ve výuce.

Klíčová slova

plánování pohybu, kinematika, hluboké zpětnovazební učení, vyhýbání se kolizím, průmysl 4.0, fyzikální simulace, průmyslová robotika

Bibliographic Citation

PARÁK, Roman. Design of Advanced Methods in the Field of Industrial Robotics fitting into the Concept of Industry 4.0. Brno, 2024. Doctoral Thesis. Brno University of Technology, Faculty of Mechanical Engineering, Institute of Automation and Computer Science. Supervised by prof. Ing. Radomil Matoušek, Ph.D.

Declaration of Authorship

I declare that I have written my doctoral thesis on the theme of "Design of Advanced Methods in the Field of Industrial Robotics fitting into the Concept of Industry 4.0" independently, under the guidance of the doctoral thesis supervisor, and using the technical literature and other sources of information, which are all quoted in the thesis and detailed in the list of literature at the end of the thesis.

As the author of the doctoral thesis, I furthermore declare that, as regards the creation of this doctoral thesis, I have not infringed any copyright. In particular, I have not unlawfully encroached on anyone's personal and/or ownership rights, and I am fully aware of the consequences in the case of breaking Regulation S 11 and the following of the Copyright Act No. 121/2000 Sb., and of the rights related to intellectual property rights and changes in some Acts (Intellectual Property Act) and formulated in later regulations, inclusive of the possible consequences resulting from the provisions of Criminal Act No. 40/2009 Sb., Section 2, Head VI, Part 4.

February 29, 2024

....

Date Ing. Roman Parák

Acknowledgements

First and foremost, I would like to thank Prof. Ing. Radomil Matoušek, Ph.D., for supervising my dissertation thesis and providing invaluable advice, inspiration, and feedback. Prof. Matoušek's suggestions significantly improved my research experience, fostering productivity and stimulation. Additionally, I appreciate his unwavering trust, which allowed me to build a unique robotic cell, now recognized as one of the premier robotic laboratories at the Brno University of Technology. Without Prof. Matoušek's trust and tireless efforts in securing funding and industrial partnerships, this laboratory would not have become a reality. His guidance has not only shaped my academic journey but has also been instrumental in the success of our laboratory, for which I am truly grateful.

I would like to extend my sincere thanks to Ing. et Ing. Stanislav Lang, Ph.D., not only for his consistently helpful ideas and relevant feedback for my dissertation thesis, but also for his continuous support throughout my entire course of studies. The appreciation also goes to Assoc. Prof. Ing. Branislav Lacko, CSc., for his constant assistance in various areas related to the concept of Industry 4.0, and continuous support at various events, including conferences and workshops, in the Czech Republic and the surrounding area. Special gratitude goes to Ing. Martin Juříček for his interesting ideas, suggestions, and extremely valuable contributions to the practical realization of the robotic laboratory.

I would like to express my gratitude to Assoc. Univ.-Prof. DI Dr. Hubert Gattringer and Univ.-Prof. Dr.-Ing. habil. Andreas Müller from the Institute of Robotics at Johannes Kepler University (JKU) in Linz, where I spent six months during my doctoral studies. The feedback received during the research internship was extremely valuable and helped to improve the objectives of the dissertation thesis. I would also like to express my gratitude to B&R Automation, specifically to Dr. Roland Riepl, for providing me with this opportunity and facilitating my association with the JKU.

I would also like to express my sincere appreciation to the industrial partners who played a crucial role in the establishment of the robotics laboratory. These partners include SMC Industrial Automation, B&R Automation, ABB Robotics, and Universal Robots, all of whom were integral stakeholders from the beginning.

I would like to acknowledge the books that have significantly inspired me. These include the 'Springer Handbook of Robotics' [1] by Bruno Siciliano and Oussama Khatib, 'Modern Robotics' [2] by Kevin M. Lynch and Frank C. Park, and 'Deep Learning' [3] by Ian Goodfellow, Yoshua Bengio, and Aaron Courville.

Finally, I extend my deep gratitude to my family for their unwavering support throughout my academic and personal life. Their encouragement has been a constant source of strength. In addition to my family, I would like to extend my sincere thanks to MDDr. Lenka Mandráková. Her never-ending patience, support, and love have played a crucial role in maintaining my mental well-being. Without her, navigating the challenges would have been much more difficult, perhaps impossible.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

"An automated machine that does just one thing is not a robot. It is simply automation. A robot should have the capability of handling a range of jobs at a factory."

— Joseph Engelberger (1925 - 2015), *"The Father of Robotics"*

The significance of robot manipulators in engineering applications and scientific research has increased substantially in recent years. This surge can be attributed to the more frequent application of technologies such as autonomous robotics, digital twins, complex system integration, etc. [7, 8], within the Industry 4.0 concept [9], along with the utilization of artificial intelligence techniques. As technology continues to advance, the convergence of robotics and artificial intelligence within the concept of Industry 4.0 holds immense promise. This requires a comprehensive exploration and understanding of the transformative potential inherent in these advancements.

In the field of industrial robotics, motion planning poses a significant challenge, involving the generation of paths or trajectories for specific robotic structures to perform a wide range of tasks. This process includes addressing obstacle avoidance and solving complex kinematic problems [1, 2]. Traditionally, trajectory planning for robotic structures relied on sampling-based algorithms, exemplified by Rapidly Exploring Random Trees (RRTs) and Probabilistic Roadmaps (PRMs) [10, 11]. Despite their contributions, these traditional approaches face limitations in addressing the increasing complexity of robotic tasks. A paradigm shift in recent years involves the use of reinforcement learning (RL) algorithms [5], which enables autonomous agents to learn and adapt their strategies through experience, thus revolutionizing path planning.

This shift has seen a surge in the adoption of deep reinforcement learning (DRL), a combination of deep neural networks (DNN) and reinforcement learning (RL), particularly in robotics research. DRL offers a compelling solution for end-to-end control, allowing the learning of sequential actions directly from raw input observations. This is particularly evident in scenarios where the goal is to achieve tasks such as reaching a static [12] or random target [13, 14, 15]

The limitation of the aforementioned approaches is their narrow focus on a specific type of robotic manipulator and a particular solved task. This limitation hinders the ability to achieve modularity and reproducibility in future extensions. To address this issue, the objectives of this work are explicitly defined in the following section.

1.1 Research Objectives

The main research objective of this dissertation is the design and development of advanced methods in the field of industrial robotics in accordance with the concept of Industry 4.0. This involves developing a comprehensive approach to kinematic solutions and motion planning across a variety of robotic manipulators, including under-articulated, articulated, and over-articulated structures. It also encompasses the development of collision avoidance structures to prevent collisions with external objects or self-collision, all while ensuring computational efficiency. Within the motion planning, the research objective focuses on innovative and unexplored methods using reinforcement learning.

Another research objective of the dissertation is the design of a versatile robotic workstation in accordance with the Industry 4.0 concept. This involves incorporating vertical system integration to ensure seamless interoperability and optimal interconnection of various types of systems across the workstation. It also includes the development of a sophisticated user interface and a unique simulation tool based on Unity3D. A critical research objective is to consider the impact of versatility, education, and future expansion.

1.2 Structural Outline of Thesis

This section provides a structural outline of the individual parts of a dissertation that present the conducted research.

Chapter 2 provides a concise overview of the historical context, including the evolution of the Fourth Industrial Revolution. Following this, it delves into the fundamental characteristics of Industry 4.0 and explores the key pillars underpinning this industrial revolution. The emphasis is placed on the essential aspects of system integration, autonomous robotics, and simulation. While providing a brief overview of the remaining pillars, the chapter concludes with an analysis of future trends in the industrial field. This exploration considers the possibility of a new revolution or the evolution of the existing concept of Industry 4.0, with a focus on improving its key pillars.

Chapter 3 continues by delving into the fundamentals of coordinate transformations, clarifying their role in representing position and rotation within Euclidean space. This discussion leads to an exploration of homogeneous transformations, a pivotal aspect in the study of robotic manipulator kinematics. Following this introduction, the chapter provides a concise overview of the general representation of robotic manipulators. Within the context of kinematics, the geometric representation of robotic mechanisms is clarified, using the Denavit-Hartenberger convention. The final section of the chapter focuses on the computation of forward and inverse kinematic problems for serial-chain manipulators.

The discussion of the inverse kinematic problem encompasses both the closed-form solution and the numerical solution.

Chapter 4 discusses the basics of the motion planning problem for robotic manipulators, which includes the concept of configuration space and the obstacle avoidance problem. This leads to an examination of an efficient method for approximating individual obstacles using bounding boxes. After this introduction, the chapter presents a brief summary of the trajectory generation problem. This section explores parametric curves, specifically Bézier and B-Spline, the trapezoidal motion profile, and motion through a sequence of points known as via points. The final part of the chapter concentrates on reinforcement learning methods for motion planning problems in the field of robotic manipulators. The discussion encompasses a description of the fundamental concept of reinforcement learning and its extension called deep reinforcement learning , which incorporates deep learning techniques into traditional reinforcement learning methods.

Chapter 5 provides the design of a unique robotic workstation, which incorporates advanced vertical system integration. Following this introduction, the chapter delves into the implementation of a sophisticated and user-friendly Human-Machine Interface (HMI) customized for the presented workstation. Subsequently, a comprehensive approach to kinematics solutions is introduced, validated across a diverse range of robotic configurations. Within the area of digital twins, the chapter describes the development of physics-based simulators utilizing PyBullet and Unity3D. The concluding section of the chapter is dedicated to Deep Reinforcement Learning-Based planning algorithms designed for tasks focused on achieving randomly generated targets within the pre-defined configuration space.

Chapter 6 summarizes the key findings and insights presented throughout this dissertation. Additionally, it provides valuable reflections on the implications of the study and suggests areas for future research.

CHAPTER 2

Current State in the Field of Industry 4.0

The following chapter provides an overview of Industry 4.0, which was first introduced by Professor Wolfgang Wahlster at the Hannover Messe trade fair in Germany in 2011 [16]. Industry 4.0 represents the fourth industrial revolution, characterized by the integration of digital technologies into manufacturing processes. In 2013, Professor Wahlster elaborated on this vision, outlining key principles and components during the same trade fair [17]. This chapter explores the significance of Industry 4.0 in revolutionizing manufacturing and technology. It provides a foundation for understanding its key components and recent developments since its initiation in 2011.

The chapter begins with a concise overview of the historical context, including the evolution of the Fourth Industrial Revolution (Section 2.1). Following this, it delves into the fundamental characteristics of Industry 4.0 (Section 2.2) and explores the key pillars underpinning this industrial revolution (Section 2.3). The emphasis is placed on the essential aspects of system integration (Subsection 2.3.1), autonomous robotics (Subsection 2.3.2), and simulation (Subsection 2.3.3). While providing a brief overview of the remaining pillars (Subsection 2.3.4), the chapter concludes by analyzing future trends in the industrial field (Section 2.4). This exploration considers the possibility of a new revolution or the evolution of the existing concept of Industry 4.0, with a focus on improving its key pillars.

2.1 History of the Industrial Revolution

The following section presents a brief overview of the historical development that led to the emergence of the Fourth Industrial Revolution, commonly known as Industry 4.0 [18]. The historical process of industrial modernization, from the First to the Third Industrial Revolution, is thoroughly depicted in the book "The Industrial Revolution

in World History" [19], and a brief review of these three revolutions can be found in [20, 21, 22]. The Fourth Industrial Revolution is discussed in the book "The Fourth Industrial Revolution" [23], but as a still relatively new area of research, it is more widely described in scientific publications (see [24, 25, 26, 27, 28]).

The historical process of the sequence of industrial revolutions with key pillars is depicted in Figure 2.1.

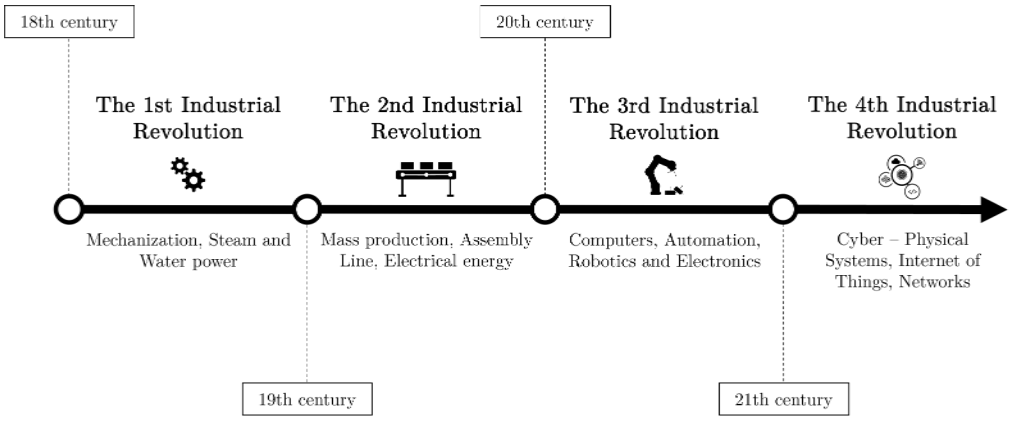


Figure 2.1: A visualization of the historical process of industrial revolutions that began in the 18th century and continues to the present day.

The First Industrial Revolution

The first phase of the Industrial Revolution began at the end of the 18th century, more precisely in 1760, and lasted until 1880.

The major milestones of the first industrial revolution include the invention of the steam engine and the development of steam power, as well as the use of turbine engines and water as power sources. The steam engine allowed the transition from agriculture to a new production process. This transition involved the use of coal as the main source of energy.

The combination of steam power and mechanized production brought about a significant change in performance, not only in terms of the growth of the regional and global market economy but also in education. This inspired the science and technology sector to restructure academic fields.

The Second Industrial Revolution

The second phase of the Industrial Revolution began in 1880 and lasted until 1950.

One of the major milestones of the second industrial revolution was the invention of the internal combustion engine. This invention facilitated technological advances in the industry, leading to rapid industrialization through the utilization of oil and electricity for mass production and assembly lines. A wave of systemic change has led to the

belief that science and technology are the means to a better life, and that progress is necessary in many ways. The revolution has brought significant changes in standardization and precision manufacturing, as well as large-scale technological infrastructure, such as electricity grids and new forms of public transportation based on the internal combustion engine.

Alongside innovations such as steamships, telephones, and gas turbines, there was a growing public demand for goods, travel, and information. These factors played a significant role in shaping future developments.

The Third Industrial Revolution

The third phase of the Industrial Revolution began in 1950 and lasted until 2010.

A characteristic feature of the Third Industrial Revolution was the implementation of electronics and information technology to automate production. The establishment of computer infrastructure brought about a significant change in information theory and data processing, and new channels were created to share information. The rapid advancement towards increased computing power has led to more interconnected and complex problems that require attention.

Innovations, such as programmable logic controllers and single/multiple purpose robotic systems, as well as the advancement of nuclear power, have opened the door to new areas of research, including space, robotics, and biotechnology.

The Fourth Industrial Revolution

The fourth phase, also known as Industry 4.0, was initially introduced in 2011 by Professor Wolfgang Wahlster at the Hannover Messe trade fair in Germany [16] and continues to the present day.

A characteristic feature of Industry 4.0 is the transformation of industrial production through the integration of digital and internet technologies, the utilization of cyber-physical systems, artificial intelligence techniques, augmented reality, physical simulation, additive technologies, and other key aspects to achieve the greatest possible flexibility in the production process.

A more detailed description of Industry 4.0, including the characteristics of the industrial concept and a brief introduction to the main pillars, is provided in Section 2.2.

2.2 Characteristics of the Fourth Industrial Revolution

The concept of Industry 4.0 involves the integration of intelligent machines and systems into the manufacturing processes of industrial enterprises [29, 30]. It is based on nine main pillars (see Fig. 2.2) [9, 7, 8], which together form the core idea of industrial digitalization. The concept aims to increase work efficiency and personalization, resulting in greater flexibility in the production of a given range of products. The Fourth Industrial Revolution focuses not only on technological advances, but also on the way people work and apply their creativity across industries. The integration of AI techniques increases the

level of information processing and evaluation, leading to improvements in areas such as safety, human-machine collaboration [31], predictive maintenance, and visual inspection. This concept affects not only the industrial sector, of which the Industry 4.0 initiative is an integral part, but also the development of education [32, 33], transportation, agriculture, and many other fields.

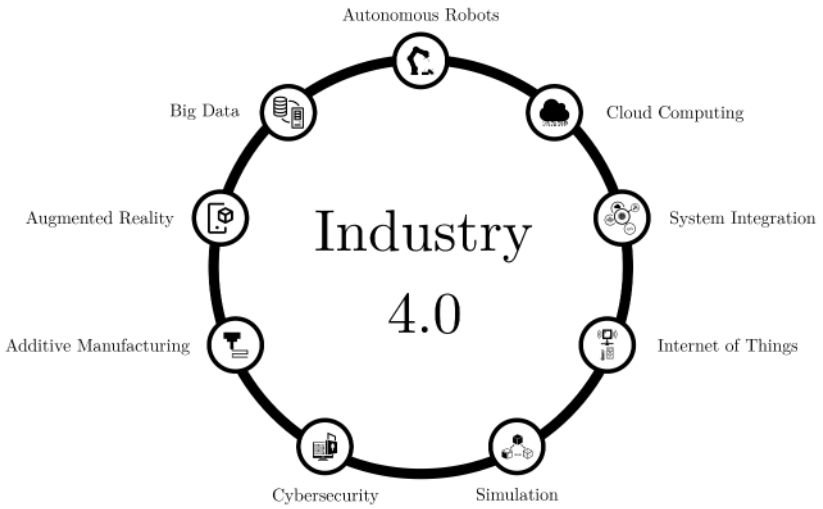


Figure 2.2: The nine main pillars of the concept of the Fourth Industrial Revolution.

Industry 4.0 encompasses six design principles [34, 35, 36], such as modularity, interoperability, etc. These principles contribute to the design or transition process from Industry 3.0 to Industry 4.0. They are referred to as design principles.

The main design principles of Industry 4.0

(a) Modularity

The principle of modularity refers to the ability to customize and adapt to different requirements, offering scalability, flexibility, and the ability to upgrade or replace specific components without affecting the entire system.

(b) Interoperability

The principle of interoperability relates to the fact that a cyber-physical system (CPS) consists of intelligent machines and storage systems that can exchange information, initiate actions, and control each other autonomously. This involves using standardized communication protocols and data formats to ensure compatibility between different components and technologies.

(c) Decentralization

The principle of decentralization refers to the ability of various components and machines to make independent decisions based on real-time data, thereby reducing the need for a central controller. Tasks are only delegated to a higher level in the event of a failure.

(d) Real-time capability

The principle of real-time capability refers to the ability of systems, manufacturing processes, and intelligent machines to operate and respond to events in real-time or near-real-time.

(e) Virtualization

The principle of virtualization involves creating virtual representations or simulations of physical entities, processes, or systems within the industrial environment. Sensor data is linked to virtual plant models and simulation models, allowing for the creation of a virtual copy of the physical world.

(f) Service orientation

The principle of service orientation emphasizes organizing and delivering functionality as services, which marks a shift from selling products to offering integrated products and services that provide more value to the customer. This involves the use of Service-Oriented Architectures (SOA) architecture.

2.3 Main Pillars of Industry 4.0

This section introduces the main pillars of the Fourth Industrial Revolution, as shown in Figure 2.2 from the previous section. It emphasizes key pillars, such as system integration, autonomous robots, and simulation, as they are crucial for practical implementation.

As see in [29, 30, 8, 37], the key pillars of Industry 4.0 do not include AI techniques such as machine learning, deep learning, etc. AI technologies are still a relatively new field in practical applications that have not been sufficiently tested to be incorporated into the pillars that form the Fourth Industrial Revolution. However, it is important to note that machine learning, deep learning, and other AI techniques are significant components in most of the key areas mentioned above.

2.3.1 System Integration

In the Industry 4.0 landscape, system integration plays a crucial role in the transformation of manufacturing processes into interconnected, intelligent, and efficient systems. The use of system integration is essential for the successful implementation of Industry 4.0. It helps incorporate various technologies, including the Internet of Things (IoT), artificial intelligence techniques, cloud computing, robotics, single/multiple purpose machines, and

cyber-physical systems. It ensures interoperability and real-time communication among different components within an intelligent manufacturing environment.

There are three dimensions of system integration within Industry 4.0 [37]: vertical integration, horizontal integration, and end-to-end integration.

(a) Vertical Integration

Vertical integration involves the seamless interconnection and collaboration between different levels of the production process hierarchy. The concept of vertical integration refers to the interconnectedness of the entire industrial enterprise, including all logical levels within the organization, from the production floor to the research department, product management, quality assurance, and sales department. This type of integration refers to flexible and reconfigurable systems within the factory that are fully integrated with each other.

(b) Horizontal Integration

Horizontal integration involves the sharing of data outside the organization, i.e., from suppliers through manufacturers to distribution, to the end customer, and subsequent service. This type of integration facilitates the exchange of information and resources between different entities within a specific phase of the production process.

(c) End-to-End Integration

The objective of end-to-end integration is to establish a smooth and interconnected flow of information and processes across the entire value chain, from the initial design phase to the delivery of the final product to the customer. This type of integration applies to all engineering processes throughout the product lifecycle.

Considering that vertical system integration enables seamless interoperability and optimal interconnection among different components within Industry 4.0, the following parts briefly describes the most commonly used Ethernet-based communication protocols.

Ethernet POWERLINK

Ethernet POWERLINK [38] is a real-time communication protocol used in industrial automation systems to ensure reliable and deterministic communication between devices. It was introduced in 2001 by Bernecker & Rainer (B&R) Industrie Elektronik GmbH and has since become a reliable solution for various applications. In 2002, the Ethernet Powerlink Standardization Group was formed, leading to the release of Ethernet Powerlink V2 in 2003. The revised version introduced an application layer, expanding the standard and enhancing its capabilities.

Ethernet POWERLINK achieves impressive real-time properties through a purely software-based solution, utilizing the Ethernet communication standard and eliminating the need for additional hardware. The POWERLINK standard is cyclical, with a user-settable cycle time. Communication parameters can be adjusted to meet specific user

requirements. Ethernet POWERLINK is a communication standard based on Fast Ethernet. It supports transmission speeds of up to 100 Mb/s and allows a maximum segment length of 100 m. Ethernet POWERLINK can be used with both star and tree topologies.

One of its key features is the ability to provide precise and synchronized communication, which is essential for applications where timing and accuracy are critical. This is accomplished through the use of a time-scheduled communication approach, which enables devices on the network to exchange data with predictable and low-latency performance. This deterministic communication is critical in industries such as manufacturing, where precise control and coordination of devices and processes are essential for optimal performance.

EtherNet/IP

EtherNet/IP (EtherNet/Industrial Protocol) [39] is one of the most widely used standards that provides users with the tools to deploy standard Ethernet technology (IEEE 802.3 combined with the TCP/IP Suite) in industrial automation applications. The protocol was developed through the collaborative efforts of organizations such as ODVA (Open DeviceNet Vendor Association), ControlNet International, and Rockwell Automation.

EtherNet/IP is a fully compatible industrial protocol according to the IEEE 802.3 standard, which utilizes a standard communication model with a solution at the application layer. Compliance with IEEE Ethernet standards provides a network interface with speeds ranging from 10 Megabits per second (Mbps) to 1 Gigabit per second (Gbps), which can be adjusted to meet user requirements. Additionally, it offers a flexible network architecture that is compatible with commercially available Ethernet installations. Within the EtherNet/IP network, individual nodes are assigned device types with specific characteristics and functions. The assigned device types and the communication network's application layer are created by the Common Industrial Protocol (CIP), which is used in industrial communications such as DeviceNet and ControlNet, employing a producer-consumer communication principle. The use of the CIP protocol achieves interoperability among all networks that support this protocol.

PROFINET

PROFINET (Process Field Network) [40, 41] is a widely used industrial communication protocol that plays a crucial role in real-time communication and data exchange within control systems. It was developed and standardized by PROFIBUS & PROFINET International (PI) to meet the demanding requirements of modern industrial environments. The PROFINET communication standard enables the control of manufacturing processes by linking the control of the production process to collect data from individual sensors. These sensors are capable of processing information at 100 Mb/s and are based on ISO/IEC 8802.3. It is an essential element of industrial automation.

PROFINET is divided into two main components: Profinet CBA (Component Based Automation) and Profinet IO. Profinet CBA is used for modular systems like Pro-

grammable Logic Controllers (PLCs) and sensors, while Profinet IO is used for distributed field devices. Both Profinet IO and Profinet CBA systems can operate simultaneously on the same network and can be implemented in the same communication station. The industrial PROFINET network can communicate in real-time (RT) at a speed defined by the cycle time.

OPC UA

OPC UA (Open Platform Communications Unified Architecture) [42] is an international standard for secure, reliable, interoperable, and platform-independent industrial communication in the context of the Fourth Industrial Revolution. It provides a solid foundation for modern industrial automation and control systems. The OPC UA industrial standard was published in 2008 and was standardized by the OPC Foundation [4] with the IEC 62541 norm.

The OPC UA supports various communication patterns, such as request/response and publish/subscribe, which can be adapted to different levels of real-time requirements. The real-time performance of the OPC UA depends on several factors, including the network infrastructure, system load, and the specific configuration of the OPC UA application. In applications where strict real-time requirements are critical, additional measures may need to be taken, such as implementing Quality of Service (QoS) mechanisms or optimizing network settings. It is important to note that although OPC UA supports real-time communication, it is not a real-time protocol. The speed at which OPC UA can communicate depends on several factors, including network conditions, system load, and specific implementation details of OPC UA server and client applications.

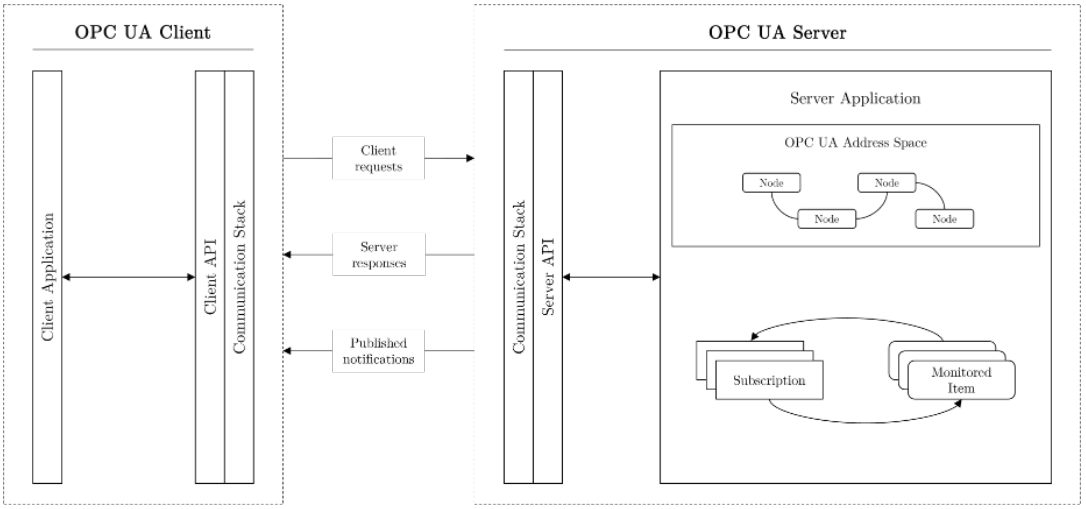


Figure 2.3: Typical client/server architecture of the OPC UA international standard [4].

The OPC UA standard is designed to seamlessly integrate automation systems vertically, allowing for any combination of client/server components at different levels of

the automation hierarchy for a specific application. It employs object-oriented techniques to model information, which are then represented in the OPC UA address space.

2.3.2 Simulation

Simulation tools can be used in production processes to capture real-time data and replicate the behavior of the physical world in a virtual model, commonly referred to as the digital twin. The use of simulation tools allows for the creation of a digital twin that can be used to optimize production processes and improve efficiency. The virtual model includes machines, sensors, products, and even people, resulting in increased productivity and production quality. By utilizing simulation tools, it is possible to optimize machine settings in a virtual environment before implementing them in the physical world. Simulations can be created for both 2D and 3D spaces to improve the production process, enhance product quality, and prevent collisions and unexpected situations. Advanced simulation, which incorporates a physical representation of individual parts within the production process, has the potential to improve efficiency, safety, and production quality while avoiding failures.

Digital Twin

The term "digital twin" has evolved over the years, but in the context of Industry 4.0, it has become more generalized. A digital twin is a virtual representation of a physical object, system, or process created through the integration of real-time data from sensors, devices, and other sources connected to the physical entity. The digital twin replicates the structure, behavior, and performance of the physical object, enabling simulation, analysis, and monitoring throughout its entire lifecycle, from design and manufacturing to operation and maintenance.

According to [43, 44], digital twins can be categorized into three subcategories (shown in Figure 2.4) based on the level of data integration. In the following text, all the major subcategories are briefly described: the digital model, the digital shadow, and the digital twin.

(a) Digital Model

A digital model is a digital representation of a physical object. There is no automated data flow between the physical and digital objects, meaning that a change in the state of the physical object does not directly affect the digital object, and vice versa.

(b) Digital Shadow

A digital shadow is a representation of an object in the digital world with unidirectional data flow between physical and digital entities. It reflects changes from the physical world to the digital world, but not vice versa.

(c) Digital Twin

A digital twin is a complete representation of a physical object that facilitates bidirectional data flow between the physical and digital counterparts. Any change in the state of the physical object results in a corresponding change in the state of the digital object, and vice versa.

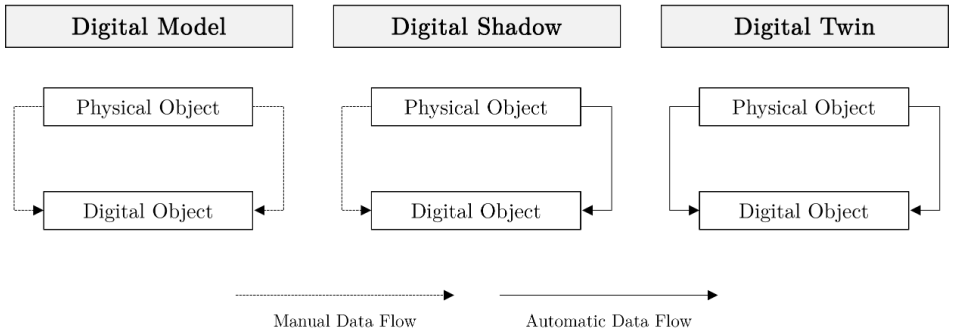


Figure 2.4: Three subcategories based on the data integration level of the digital twin model.

Deloitte's article [45] presents a conceptual architecture that explains the general workings of the digital twin (refer to Figure 2.5). The architecture comprises five enabling components: sensors, data, integration, analytics, and actuators. Additionally, it includes a six-step process that involves creating, communicating, aggregating, analyzing, gaining insights, and taking action. Sensors and actuators are located in a physical space, while data analysis is performed in a virtual space. Integration technologies should be utilized to facilitate seamless data communication between the physical and virtual worlds. The conceptual architecture comprises six operational steps based on digital twins: Create, Communicate, Aggregate, Analyze, Insight, and Act. More information about the conceptual architecture can be found in [46] and [47].

2.3.3 Autonomous Robots

Autonomous robotics is a crucial component in the context of Industry 4.0. It is primarily utilized in industries to address complex and hazardous tasks that are not suitable for humans. Furthermore, autonomous robotics replaces stereotypical tasks traditionally performed by humans, preventing the under-utilization of human potential. The use of robots in industrial enterprises is growing exponentially [48], as they significantly increase productivity, efficiency, repeatability, and human safety. Autonomous robots are used in various industries, including manufacturing, agriculture, and healthcare, to perform a wide range of tasks, such as intelligent sorting, safe interaction, and visual inspection.

A more detailed discussion of the principles of autonomous robotics, including the methods employed for motion planning and control, is provided in Section 4.

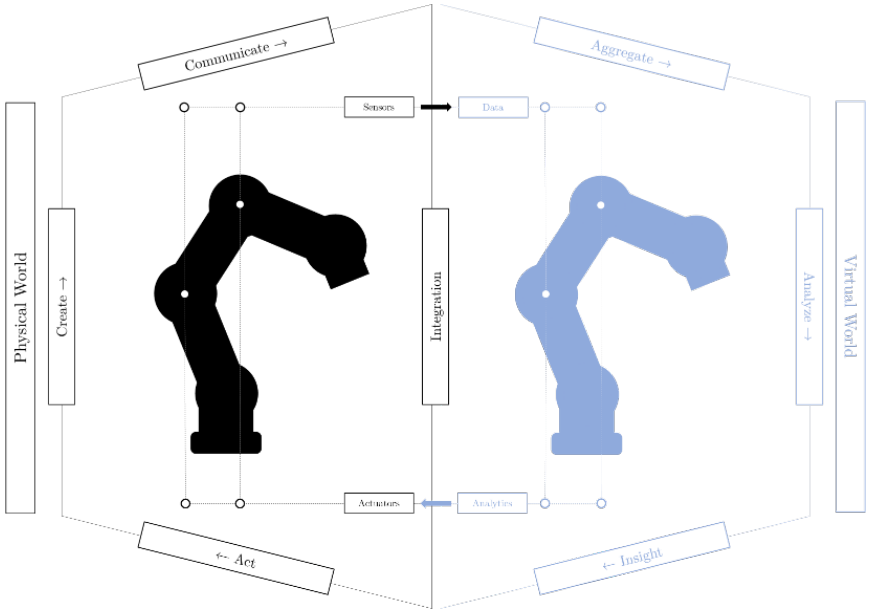


Figure 2.5: A conceptual architecture, consisting of five basic components and a six-step process, illustrates the general functioning of a digital twin.

2.3.4 Other Pillars

In addition to the technological pillars already described, there are other key aspects of the Fourth Industrial Revolution that are necessary to maintain the basic concept.

Cloud Computing

Cloud computing is a crucial component of Industry 4.0 because it enables industrial enterprises to manage and visualize data in real-time with minimal interaction with service providers. The industrial revolution promotes increased data sharing between workplaces, driven by the imperative to optimize production, thereby minimizing the restrictions on individual companies. Cloud computing enables user mobility, conserves resources, and facilitates distributed data analytics to address network-related issues. It also supports decentralization and intelligent processing of data generated by various Internet of Things (IoT) devices that integrate the physical world into cyberspace.

There are three main cloud service models [8, 49]: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

(a) Infrastructure as a Service (IaaS)

The Infrastructure as a Service (IaaS) cloud service model offers virtualized computing resources, such as virtual machines, storage, and networking, over the internet.

(b) Platform as a Service (PaaS)

The Platform as a Service (PaaS) cloud service model provides a platform that encompasses not only the underlying infrastructure but also the necessary development tools and services for application building, deployment, and management.

(c) Software as a Service (SaaS)

The Software as a Service (SaaS) cloud service model provides software applications over the internet on a subscription basis. Users can access these applications through a web browser.

Internet of Things

The Internet of Things (IoT) enables industrial communication among interconnected and uniformly addressed objects, such as industrial devices. These devices communicate through standard protocols. In a broader sense, IoT offers advanced interconnection capabilities for systems, services, and physical objects, enabling seamless communication between industrial devices and efficient sharing of substantial data. Implementing IoT in industrial enterprises can provide significant benefits. These benefits include improved integrability, which allows smoother coordination of various processes, improved agility to adapt to changing circumstances, and ultimately providing competitive advantages. For example, real-time monitoring of machinery or predictive maintenance through IoT can optimize production processes and minimize downtime, contributing to improved overall efficiency and competitiveness for the company.

The Internet of Things (IoT) consists of the Internet of Services (IoS), the Internet of Manufacturing Services (IoMs), the Internet of People (IoP), embedded systems, and the Integration of Information and Communication Technology (IICT) [37].

Cybersecurity

Cyber security is a critical aspect of Industry 4.0 as it has the potential to protect the business environment through various attacks. As connectivity increases and standard communication protocols are adopted (refer to Subsection 2.3.1) in accordance with the Fourth Industrial Revolution, protecting critical industrial systems and production lines from cybersecurity threats becomes increasingly important. The primary objective of cybersecurity is to ensure secure and reliable communication, along with sophisticated machine access management. Securely connecting both the physical and digital worlds can improve the quality of information needed for planning, optimization, and production.

Closely related to cybersecurity is the concept of the Cyber-Physical System (CPS), which plays a crucial role in Industry 4.0. CPS has been defined as a system in which the physical world is fully integrated with computing, communication, and control systems, collectively referred to as the digital world. The main characteristics of CPS are decentralization and automatic control of the production process. The interconnection between the physical and digital worlds can enhance the quality of information required for planning, optimizing, and operating production systems.

Additive Manufacturing

Additive manufacturing, also known as 3D printing, revolutionizes the production process by creating parts from computer-generated 3D models. It involves the production of small batches of personalized products based on customer specifications, providing construction benefits such as intricate and lightweight designs. This method streamlines production, bypassing the need for extensive technological preparation, reducing construction timescales, and utilizing prototypes instead of finished products. Consequently, time-to-market is significantly reduced. Accurate estimation of material quantities and simulation of the production process are crucial for optimizing order management. Gradually adding materials during production allows for precise determination of consumption and enhances overall efficiency.

Additive technologies allow for the production of structurally complex parts without the need for machine reconfiguration or complex software modifications. This production process can be customized to meet the specific needs of the customer.

Augmented Reality

Augmented reality is an interactive technology that connects the physical world with the virtual world. It is used to enhance human-machine interaction, machine and equipment maintenance, and visual product inspection in industrial enterprises. Spatial data are a key aspect of augmented reality, enabling the connection of various information to a specific location. The combination of a computer-generated environment and physical objects has numerous possibilities, as creativity knows no bounds.

From a technical standpoint, the augmented reality system must solve two spatial problems in real-time: localizing the user and providing spatial vision. To achieve this, augmented reality utilizes a combination of sensors, such as a gyroscope and accelerometer, along with computationally intensive machine vision algorithms that are based on artificial intelligence techniques.

Big Data

The concept of big data applies to the large, diverse, and complex amounts of data that impact the decision-making processes of industrial enterprises. According to Forrester's definition [50], Big Data comprises four dimensions: (1) data volume, (2) data variety, (3) data velocity, and (4) data value. The volume of data in the industry is growing exponentially, increasing the potential amount of usable information. However, it is necessary to collect and comprehensively evaluate the information for further development. Therefore, the increase in data level and technological capabilities accelerates productivity and innovation.

In the context of Industry 4.0, big data analytics is beneficial in several areas of an industrial enterprise. It helps in predictive maintenance, improves equipment service, optimizes production quality, and saves energy. Previously recorded data are analyzed to detect threats that have occurred in various production processes and to predict new problems that may arise in order to prevent them.

2.4 Forecasting the Future Trends of the Industry

Predicting the future of any industry involves uncertainty, as it depends on various factors such as technological advances, economic conditions, regulatory changes, consumer preferences, and possible global crises. This section presents the key trends and research areas that are likely to shape the industry's future.

Although Industry 4.0 is still in development, some scientific publications (see [51, 52, 53]) predict that the next step will lead to improvements in sustainability, modularity, efficiency, human-centered solutions, and universality. This could be the next industrial revolution, referred to as Industry 5.0, or the evolution of Industry 4.0 with enhanced pillars.

Artificial intelligence (AI) techniques are expected to play a crucial role in the future development of the industrial segment. These techniques will be integrated into the improvement of individual technologies within the overall concept. Autonomous robots will become increasingly important in manufacturing as they can increase productivity without replacing human workers [54, 55]. The integration of advanced physical simulations [56], representing both the physical and virtual worlds, will become crucial. Additionally, it will be important to evaluate and optimize supply chains in smart and sustainable manufacturing processes. The transformation of data driven by IoT, Big Data, and cloud computing will also be crucial.

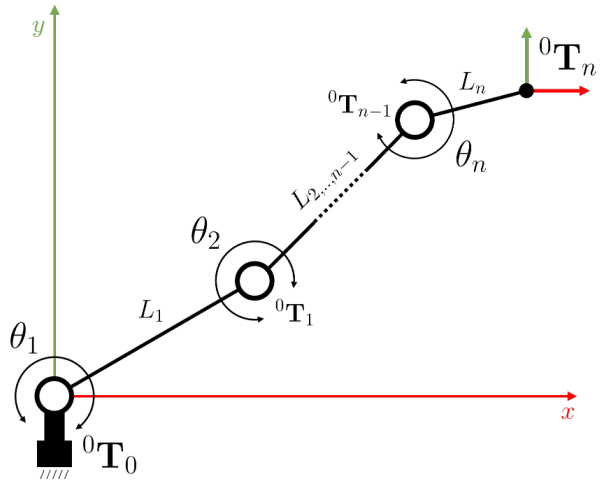
The industrial segment is experiencing a revolution to create a more flexible, scalable, and adaptable system. This will be achieved through improved technology, efficient and modular system integration, increased levels of robotic automation with a focus on autonomy, and harnessing the power of human creativity. These advancements are expected to increase productivity in the industry.

CHAPTER 3

Kinematics

Kinematics in robotic manipulators is a fundamental aspect that deals with the study of motion principles, without considering forces and torques [1, 2]. It involves the transformation of objects based on the position and orientation of different components within the mechanical structure of a robot. In general, the kinematic description is a geometric representation of the robot mechanism defined by specific conventions known as the Denavit-Hartenberg conventions.

The problem of manipulator kinematics is typically divided into two categories [57, 58]: forward kinematics and inverse kinematics. Forward kinematics establishes the relationship between joint variables and the homogeneous transformation matrix of the end-effector (Eq. 3.18), while inverse kinematics determines the joint configurations necessary to achieve a desired end-effector pose (Eq. 3.19).



Forward Kinematics

$$f(\theta_{1,\dots,n}) = {}^0\mathbf{T}_n$$

Forward Kinematics

$$\theta_{1,\dots,n} = f^{-1}({}^0\mathbf{T}_n)$$

Figure 3.1: A simple representation of an n-line robotic manipulator in two-dimensional space, illustrating the basic concept of kinematics computation.

The chapter begins by delving into the fundamentals of coordinate transformations (Section 3.1), clarifying their role in representing position (Subsection 3.1.1) and rotation (Subsection 3.1.2) within Euclidean space. This discussion leads to an exploration of homogeneous transformations (Subsection 3.1.3), a pivotal aspect in the study of robotic manipulator kinematics. Following this introduction, the chapter provides a concise overview of the general representation of robotic manipulators (Section 3.2). Within the context of kinematics, the geometric representation of robotic mechanisms is clarified, using the Denavit-Hartenberger convention (Section 3.3). The final section of the chapter focuses on the computation of forward (Section 3.4) and inverse (Section 3.5) kinematic problems for serial-chain manipulators. The discussion of the inverse kinematic problem encompasses both the closed-form solution and the numerical solution.

3.1 Coordinate Transformations

The study of kinematics for robotic manipulators or objects generally focuses on characterizing motions without considering forces and torques. In particular, the dynamics of the robotic system is neglected in this research area. In this context, coordinate transformations play a central role [1, 59, 60], referring to mathematical techniques used to describe the position and orientation of rigid bodies in different coordinate frames. These transformations are essential for accurate modeling and control of robot motion because they facilitate the conversion of coordinates from one reference frame to another. This feature provides a method to analyze and predict the spatial configuration of the robot.

In the context of Euclidean space, the pose of an object includes its position and orientation. For a rigid object, the required pose coordinates are defined by at least six parameters. Spatial pose can be represented using sets of redundant coordinates that have auxiliary relations between them. The difference between the number of coordinates in the set and six determines the number of independent constraints. The poses of an object are described with respect to the coordinate reference frame, which consists of an origin of the frame i, denoted O_i, and a set of three orthogonal basis vectors, denoted $(\vec{x}_i, \vec{y}_i, \vec{z}_i)$, all fixed within a particular body [1]. The pose of a body is consistently expressed relative to another body, conveyed as the pose of one coordinate frame relative to another.

This section is a compilation of content from various sources, including both scientific articles (see [61, 62, 63, 64]) and book publications (see [1, 59, 60, 65]).

3.1.1 Representation of Position

In a three-dimensional Euclidean space, the position, also called translation, of the origin of the coordinate frame i relative to the coordinate frame j, can be expressed by the vector ${}^{j}\mathbf{p}_i \in \mathbb{R}^3$.

$$ {}^{j}\mathbf{p}_i = \begin{bmatrix} {}^{j}p_i^x \\ {}^{j}p_i^y \\ {}^{j}p_i^z \end{bmatrix} . \tag{3.1} $$

The individual components of this vector are the Cartesian coordinates of the origin O_i in the coordinate frame j, which represent projections of the vector ${}^{j}\mathbf{p}_i$ onto the corresponding axes x, y, and z.

3.1.2 Representation of Rotation

The concept of expressing a rotation in three-dimensional space as a mathematical formulation is considerably broader than the straightforward representation of position. According to Euler's rotation theorem, any arbitrary rotation of a rigid body in a three-dimensional space can be represented as a single rotation about a fixed axis. It was concluded based on this statement that any rotation can be described by at least three parameters.

There are various ways of expressing rotation in three-dimensional space, all of which are described by three degrees of freedom. In the field of robotic manipulators, the most commonly used representations are rotation matrices, Euler angles, and quaternions. This section provides a brief description of each of these representations.

Rotation Matrices

In a three-dimensional Euclidean space, the orientation of the coordinate frame i relative to the coordinate frame j can be denoted by expressing the basis vectors $(\vec{x}_i, \vec{y}_i, \vec{z}_i)$ of the frame i in terms of the basis vectors $(\vec{x}_j, \vec{y}_j, \vec{z}_j)$ of the frame j. A rotation matrix ${}^{j}\mathbf{R}_i$ was obtained, in which each component is the scalar product of the basis vectors of both the i and j coordinate frames, denoted $({}^{j}\vec{x}_j, {}^{j}\vec{y}_j, {}^{j}\vec{z}_j)$.

$$ {}^{j}\mathbf{R}_i = \begin{bmatrix} \vec{x}_i \cdot \vec{x}_j & \vec{y}_i \cdot \vec{x}_j & \vec{z}_i \cdot \vec{x}_j \\ \vec{x}_i \cdot \vec{y}_j & \vec{y}_i \cdot \vec{y}_j & \vec{z}_i \cdot \vec{y}_j \\ \vec{x}_i \cdot \vec{z}_j & \vec{y}_i \cdot \vec{z}_j & \vec{z}_i \cdot \vec{z}_j \end{bmatrix} . \tag{3.2} $$

The basis vectors are equal to unit vectors, and the scalar product of any two unit vectors is the cosine of the angle between these vectors. The components of this angle are called directional cosines.

The rotation of frame i about the $\vec{x}_j$ axis through an angle θ can be expressed by the rotation matrix with the values

$$ \mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} , \tag{3.3} $$

while the same rotation about the $\vec{y}_j$ axis can be expressed by the rotation matrix with the values

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}, \tag{3.4}$$

lastly, the $\vec{z}_j$ axis can be expressed by the rotation matrix with the values.

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.5}$$

The general three-dimensional rotation is achieved by combining the above rotation matrices into one matrix through matrix multiplication. Since matrix multiplication is non-commutative, it is important to consider the order of the rotations, as follows

$$\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \neq \mathbf{R}_z\mathbf{R}_y\mathbf{R}_x. \tag{3.6}$$

This statement implies that rotation about the x, y, and z axes does not produce the same rotation as rotation about the z, y, and x axes. This property is consistent with the fact that rotations are also non-commutative.

The rotation matrix ${}^j\mathbf{R}_i$ contains nine elements (see Eq. 3.2), while only three parameters are needed to define the orientation of an object in a three-dimensional Euclidean space. Therefore, there are six additional auxiliary relations between the elements of the matrix. Since the basis vectors of both coordinate frames i and j are mutually orthonormal, the columns of ${}^j\mathbf{R}_i$ formed by the scalar products of these vectors are also mutually orthonormal. A matrix composed of mutually orthonormal vectors is called an orthogonal matrix and has the property that the inverse matrix of rotation is equal to the transpose matrix of rotation.

$${}^j\mathbf{R}_i^{-1} = {}^j\mathbf{R}_i^T. \tag{3.7}$$

This property provides the six auxiliary relations. Three of them require that the column vectors have unit length, and three require that the column vectors be mutually orthogonal.

Euler Angles

The orientation of the coordinate frame i relative to the coordinate frame j can be represented as a vector of three independent coordinates (α, β, γ), commonly known as Euler angles. These coordinates, employed in the Euler angle representation, describe three consecutive rotations about an axis of a moving coordinate frame. As the order of rotation about the axes is essential, the specification of rotations for Euler angles must always be accompanied by an appropriate convention, providing a detailed description of the axes.

Assuming that the moving frame i and the fixed frame j are initially coincident, the Euler angles of the Z-Y-X convention can be obtained by three successive rotations in

the order α, β, γ, corresponding to rotations about the z, y, and x axes of the frame i. In terms of the basic rotation matrices, the resulting rotational transformation of the Z-Y-X convention can be formed as the product of

$$\begin{aligned} {}^{j}\mathbf{R}_i(\alpha,\beta,\gamma) &= \mathbf{R}_z(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_x(\gamma) \\ &= \begin{bmatrix} c_\alpha c_\beta & c_\alpha s_\beta s_\gamma - s_\alpha c_\gamma & c_\alpha s_\beta c_\gamma + s_\alpha s_\gamma \\ s_\alpha c_\beta & s_\alpha s_\beta s_\gamma + c_\alpha c_\gamma & s_\alpha s_\beta c_\gamma - c_\alpha s_\gamma \\ -s_\beta & c_\beta s_\gamma & c_\beta c_\gamma \end{bmatrix}, \end{aligned} \tag{3.8}$$

where c and s are abbreviations of the cosine and sine functions.

However, Euler angles have significant disadvantages, such as susceptibility to gimbal lock and low accuracy in integrating incremental changes over time. Therefore, researchers have started to use alternative rotation representations, such as quaternions, to effectively represent rotations.

Quaternions

The quaternion representation of orientation, which was introduced by W. R. Hamilton in the 19th century, is a mathematical concept that is commonly used in three-dimensional geometry. Unlike traditional representations such as Euler angles, a quaternion can be expressed as a four-dimensional vector in the form

$$\mathbf{Q} = q_w + q_x i + q_y j + q_z k, \tag{3.9}$$

where q_w, q_x, q_y, $q_z \in \mathbb{R}$, are the four constituents of the quaternion $\mathbf{Q}$, sometimes referred to as Euler's parameters, denoting arbitrary real quantities: positive, negative, or zero. The symbols i, j, k are operators or simply symbols that represent three imaginary quantities.

A quaternion $\mathbf{Q}$, particularly a unit quaternion $\|\mathbf{Q}\| = 1$, is commonly expressed as a combination of a scalar and a vector. The scalar component contains information about the angle of rotation, while the vector component defines the axis of rotation. Quaternions offer a concise and effective method for performing three-dimensional rotations, avoiding certain mathematical ambiguities and gimbal lock issues associated with other rotation representations.

In terms of the basic rotation matrices, the resulting rotational transformation of a unit quaternion can be formulated as follows

$${}^{j}\mathbf{R}_i = \begin{bmatrix} q_w^2 + q_x^2 - q_y^2 - q_z^2 & 2(q_x q_y - q_w q_z) & 2(q_w q_y + q_x q_z) \\ 2(q_w q_z + q_x q_y) & q_w^2 - q_x^2 + q_y^2 - q_z^2 & 2(q_y q_z - q_w q_x) \\ 2(q_x q_z - q_w q_y) & 2(q_w q_x + q_y q_z) & q_w^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix}. \tag{3.10}$$

The corresponding rotation matrix ${}^{j}\mathbf{R}_i$ (see Eq. 3.10) is obtained as a rotation about the unit axis in the direction of the vector part of the unit quaternion by an angle of rotation $2\cos^{-1}(q_w)$.

3.1.3 Homogeneous Transformations

The previous sections dealt with the representation of position (Sect. 3.1.1) and orientation (Sect. 3.1.2) separately. However, when computing the pose representation in Euclidean space, it is often convenient to work with a combined representation known as the homogeneous transformation. In this representation, the position vector ${}^{j}\mathbf{p}_i$ and the rotation matrix ${}^{j}\mathbf{R}_i$ are combined in a compact notation. It follows from this statement that the homogeneous transformation allows the combination of both algebraic operations of translation and rotation within a fixed and consistent homogeneous transformation matrix ${}^{j}\mathbf{T}_i$.

$$ {}^{j}\mathbf{T}_i = \begin{bmatrix} {}^{j}\mathbf{R}_i & {}^{j}\mathbf{p}_i \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.11} $$

The matrix ${}^{j}\mathbf{T}_i$ transforms the vectors from the coordinate frame i to the coordinate frame j. On the other hand, the inverse ${}^{j}\mathbf{T}_i^{-1}$ transforms the vectors from the coordinate frame j to the coordinate frame i.

$$ {}^{j}\mathbf{T}_i^{-1} = \begin{bmatrix} {}^{j}\mathbf{R}_i^T & -{}^{j}\mathbf{R}_i^T {}^{j}\mathbf{p}_i \\ \mathbf{0}^T & 1 \end{bmatrix}. \tag{3.12} $$

The composition of homogeneous 4x4 transformation matrices is achieved by a simple matrix multiplication between the frames 0 and n, which is built from multiple piecewise transformations as described in

$$ {}^{0}\mathbf{T}_n = {}^{0}\mathbf{T}_1 {}^{1}\mathbf{T}_2 \dots {}^{n-1}\mathbf{T}_n. \tag{3.13} $$

This equation is suitable for describing the complex kinematic transformation along a serial chain of bodies connected by joints, which is a typical model for robotic manipulators.

The homogeneous transformation of a rotation about a particular axis can be denoted as **Rot**, so the rotation of α about the x axis can be expressed in the form

$$ \mathbf{Rot}_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.14} $$

while, the homogeneous transformation of a translation along a particular axis can be denoted as **Trans**, so the translation of t along the x axis can be expressed in the form

$$ \mathbf{Trans}_{x,t} = \begin{bmatrix} 1 & 0 & 0 & t \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.15} $$

Although homogeneous transformation matrices technically contain sixteen elements, four of them are defined as a vector of zeros or identity values to ensure the proper

composition and representation of transformations. The remaining elements consist of a rotation matrix and a position vector.

3.2 General Representation of Robotic Manipulators

Robotic manipulators generally consist of links (rigid bodies) connected by joints that allow controlled (linear) or free relative motion, forming a so-called kinematic chain [66] (see Fig. 3.2). The joints are usually represented as revolute (rotational motion) or prismatic (translational motion) and serve as a connection between two links. Equally important components in the representation of a robotic manipulator are the base, which is typically connected to the ground and directly linked to the world coordinate system, and the end-effector, which is the last link of the robotic structure and can be used to hold tools.

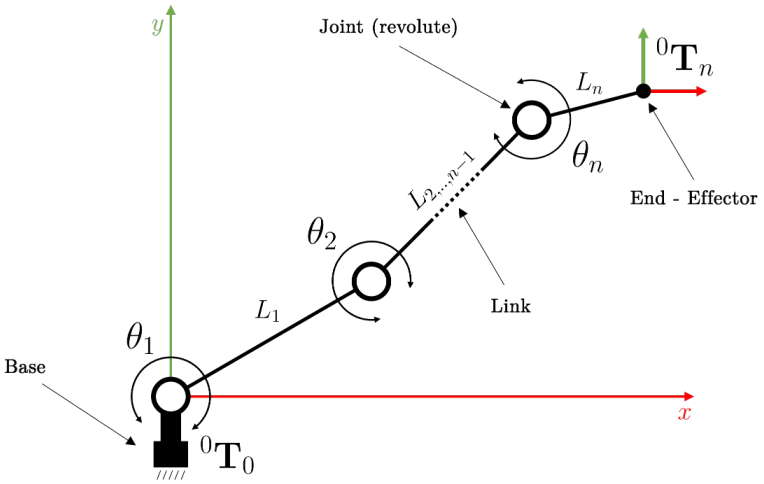


Figure 3.2: A simple representation of an n-link robotic manipulator in two-dimensional space, which describes the basic components of the robotic structure.

3.2.1 Degree of Freedom

The Degree of Freedom (DoF) refers to the number of independent motions or variables through which a rigid object can move within a defined space. The number of DoF is, therefore, equal to the dimension of the configuration space. A rigid object in two-dimensional space has 3 DoF (two for translation and one for rotation), while in three-dimensional space it has 6 DoF (three for translation and three for rotation), as see in Fig. 3.3.

In robotics, the concept of degrees of freedom (DoF) is crucial for understanding and designing robotic systems, especially robotic arms and manipulators. DoF refers to the number of independent motions a manipulator can perform, which is determined by the total number of joints and the constraints imposed by the robot's design [2].

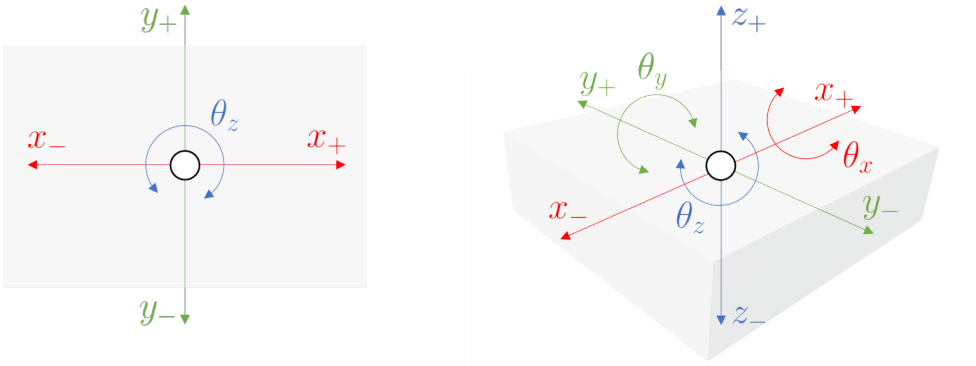


Figure 3.3: The number of degrees of freedom of the rigid body in two-dimensional (left) and three-dimensional (right) space.

According to this statement, the degrees of freedom in a robotic system can be categorized into two main parts: translational and rotational. The translational part involves linear motion along a specific axis, enabling the robot to move in a straight line. In contrast, the rotational part involves motion about a specific axis, enabling the robot to change its orientation.

A higher degree of freedom increases the redundancy of available variable joint configurations, leading to mathematical challenges such as singularities and suboptimal extrema [67]. Robotic manipulators with fewer DoFs are considered under-articulated, while those with more than six DoFs are described as over-articulated, or more precisely, kinematically redundant.

3.2.2 Joint Space and Operational Space

In the field of robotic manipulators, two configuration spaces are distinguished for defining the movement and control of robotic systems: joint space and operational space. These spaces represent different perspectives on the movement of robotic manipulators and provide distinct frameworks for analysis and control.

The joint space refers to the configuration space of a robot, where the state of the robot is defined by the positions, velocities, and accelerations of its individual joints. Each joint corresponds to a degree of freedom, and the combination of joint values determines the overall configuration of the robot. In joint space, the focus is on controlling and coordinating the movements of each joint independently. This space is often characterized by variables such as joint angles or joint displacements.

The operational space, also known as the task space or Cartesian space, represents the space in which the end-effector of the robot operates. The position and orientation of the end-effector define the operational space, and the focus is on achieving specific tasks or objectives in this space. Operational space allows for a more intuitive and task-oriented approach to control, as it directly deals with the position and orientation of the end-effector in the environment.

3.2.3 Types of Joints

Joints are essential components of the robotic structure that enable robots to achieve movement and flexibility. They are designed to mimic the range of motion found in natural limbs, providing the robot with the ability to perform various tasks. There are various types of robot joints available to define either translational or rotational motion of the individual segments. The most common types of robot joints include revolute, prismatic, helical, cylindrical, universal, and spherical [2].

The revolute joint, also known as a hinge joint, enables rotational motion about a specific axis. In contrast, the prismatic joint, or linear joint, facilitates translational motion along a designated axis. The helical joint, commonly referred to as a screw joint, allows simultaneous rotation and translation along a screw axis. All of these types of joints provide one degree of freedom.

Moving to joints with two degrees of freedom, the cylindrical joint permits independent translation and rotation about a single fixed axis. Similarly, the universal joint, composed of two revolute joints with orthogonal axes, also boasts two degrees of freedom. This joint is frequently employed to transmit motion between non-collinear axes, showcasing its versatility.

The spherical joint, also known as a ball-and-socket joint, has three degrees of freedom and functions similarly to the human shoulder joint. This design enables a wide range of motion and is utilized in various mechanical systems.

3.2.4 Workspace

The workspace of a general-purpose robotic manipulator is defined as the total volume covered by the end-effector during all possible motions. This space is determined by the manipulator's geometry and the movement limits of each joint. The workspace is commonly categorized into two types [1, 66]: reachable workspace, representing the total set of points accessible by the end-effector, and dexterous workspace, which is a subset of these points allowing arbitrary end-effector orientations.

Most serial-chain robotic manipulators exhibit regional and orientational structures [1]. The position of the end-effector in space is determined by the regional structure, which is formed by inner joints. On the other hand, the orientation of the end-effector is determined by the orientational structure, which is comprised of outer joints. The volume of the regional workspace can be calculated using the known geometry and joint motion limits of the serial-chain robot manipulator [1] or through a simplified version employing the Monte Carlo method [68].

3.3 Denavit-Hartenberg Convention

The geometric representation of a robotic mechanism is defined by attaching coordinate frames to a specific link within the structure. Although the coordinate frames can be positioned anywhere, it is advantageous for both clarity and computational efficiency to adhere to a mathematical convention for frame placement within the links. Jacques

Denavit and Richard Hartenberg introduced this fundamental convention in 1955 [69], known as the Denavit-Hartenberg (DH) standard method. This method was subsequently modified by Etienne Dombre and Wisama Khalil [70] and is referred to as the Denavit-Hartenberg (DH) modified method. While the standard method exhibits ambiguities when applied to closed or tree-structured robots, the modified method allows a unified description of complex and serial structures in articulated mechanical systems [71]. It is important to note that, despite the differences, the transformation matrices derived from both conventions will yield the same end-effector position and orientation.

In both forms, determining the relative location of one coordinate frame to another requires only four parameters, in contrast to the six parameters needed to describe any transformation in Euclidean space. The parameters, based on the Denavit-Hartenberg (DH) convention, include two link parameters: the link length a_i and the link twist α_i, along with two joint parameters: the joint offset d_i and the joint angle θ_i.

The reduction in the number of parameters is achieved by strategically placing the origins and axes of the coordinate frames. Specifically, the x axis of one frame is positioned to intersect both perpendicularly and tangentially with the z axis of the subsequent coordinate frame.

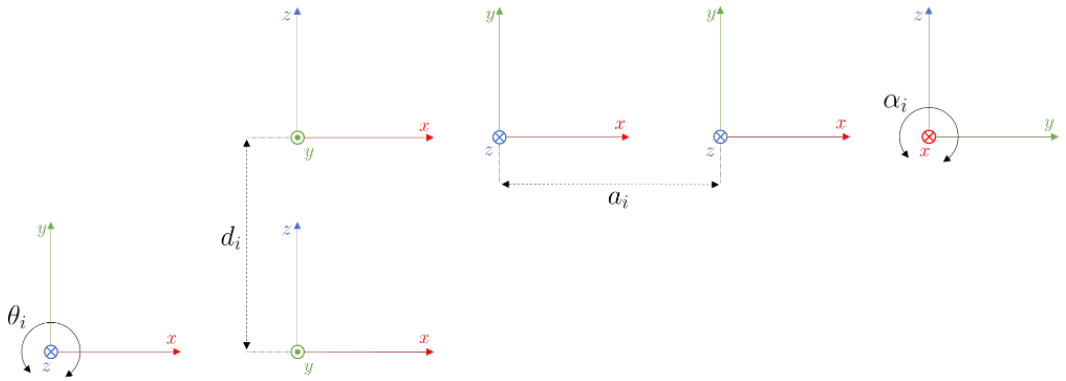


Figure 3.4: Illustration of the Denavit-Hartenberg convention defined by the four parameters a_i, α_i, d_i, and θ_i.

In the standard DH convention, each homogeneous transformation ${}^{i-1}\mathbf{T}_i$ is represented as a product of four basic transformations about particular axes

$$\begin{aligned} {}^{i-1}\mathbf{T}_i &= \mathbf{Rot}_{x,\alpha_i}\mathbf{Trans}_{x,a_i}\mathbf{Rot}_{z,\theta_i}\mathbf{Trans}_{z,d_i} \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{3.16}$$

and in the modified DH convention, similar to the previous case, but with the modified transformation rules

$$
\begin{aligned}
{}^{i-1}\mathbf{T}_i &= \mathbf{Rot}_{x,\alpha_{i-1}}\mathbf{Trans}_{x,a_{i-1}}\mathbf{Rot}_{z,\theta_i}\mathbf{Trans}_{z,d_i} \\
&= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & a_{i-1} \\ s_{\theta_i}c_{\alpha_{i-1}} & c_{\theta_i}c_{\alpha_{i-1}} & -s_{\alpha_{i-1}} & -d_i s_{\alpha_{i-1}} \\ s_{\theta_i}s_{\alpha_{i-1}} & c_{\theta_i}s_{\alpha_{i-1}} & c_{\alpha_{i-1}} & d_i c_{\alpha_{i-1}} \\ 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{3.17}
$$

The matrix ${}^{i-1}\mathbf{T}_i$ is a function of a single variable, with three of its four parameters determined by the joint type and geometry. In the case of a prismatic joint aligned along the z axis, only the joint offset d_i varies, while all other parameters remain constant. Conversely, for a revolute joint revolving around the z axis, only the joint angle θ_i is variable, and the remaining parameters are fixed.

3.4 Forward Kinematics

The computation of the forward kinematics problem involves determining the homogeneous transformation matrix ${}^0\mathbf{T}n$ for a serial-chain manipulator, represented by Equation 3.18. This matrix describes the end-effector's position and orientation relative to the base. The computation utilizes joint variables $\theta 1, \ldots, n$ in a recursive process, where coordinate transformations are performed along each link, leading from the base frame to the end-effector frame.

It is crucial to note that the frame attached to the end-effector is commonly known as the tool frame. This frame, which maintains a constant displacement in both position and orientation, plays a significant role in the kinematic analysis. Importantly, the tool transformation matrix must be added as the last element in the chain.

$$
f(\theta_{1,\ldots,n}) = {}^0\mathbf{T}_n. \tag{3.18}
$$

The forward kinematics problem is critical to the development of robot coordination algorithms because the orientation of joints is typically measured by sensors integrated into the design of a particular joint. In practice, solving the forward kinematics problem involves computing the transformation between the tool and the station frames [1]. This process is straightforward for a serial chain because the transformation describing the pose of the end-effector relative to the base is obtained by simply concatenating transformations between frames fixed in adjacent links of the chain, as explained using the Denavit-Hartenberger convention in Sect. 3.3. It is important to note that the calculation of forward kinematics always provides a unique solution.

3.5 Inverse Kinematics

The computation of the inverse kinematics problem, expressed in Eq. (3.19), for a serial-chain manipulator involves identifying joint variables denoted as $\theta_{1,\ldots,n}$. These variables

correspond to a given homogeneous transformation matrix ${}^0\mathbf{T}_n$, representing the end-effector's position relative to the base. Unlike the forward kinematics problem, determining this set of joint parameters is not straightforward. Equation (3.13) underscores the necessity of solving systems of non-linear equations to address the challenges posed by the inverse kinematics problem.

$$\theta_{1,\dots,n} = f^{-1}({}^0\mathbf{T}_n). \tag{3.19}$$

Addressing the characterization of an identical Cartesian target with different joint configurations poses a challenging issue, potentially resulting in zero to infinite solutions. While algebraic solutions can be obtained for simpler geometries, their applicability diminishes for more complex cases. Consequently, numerous methods are based on numerical solutions [72, 73, 74, 75]. Although analytical solutions may prove useful in specific cases [76, 77], a numerical approach generally offers greater universality. However, it is crucial to recognize that numerical solutions come with challenges, including suboptimal extremes, computational cost, and the potential to encounter singularities.

It is important to highlight that singularities present a significant challenge in the field of robot kinematics. Singular points lead to an immediate reduction in degrees of freedom during the resolution of motion postures. To address this challenge, a strategic approach involves transforming the task into a workspace region where configurations with singularities are avoided [78, 79]. This transformation helps maintain a higher level of flexibility in the robot's movements. For instance, this may involve adjusting the angles of the joints. Such strategies can effectively mitigate the effects of singularities on robot kinematics.

3.5.1 Closed-Form Solutions

The methods for obtaining closed-form solutions in inverse kinematics problems strongly depend on the geometry of robotic mechanisms and are, therefore, not sufficiently general. However, the solutions obtained are exact and can provide all existing joint configurations for a particular reachable Cartesian pose. Although these methods have a lower computational cost, they are typically only available for specific geometries, primarily for non-redundant manipulators with fewer degrees of freedom. The complexity rapidly increases with each additional degree of freedom. The necessary conditions for a closed-form solution of a non-redundant six-degree of freedom robotic mechanism are as follows [80]:

(a) Three consecutive revolute joint axes intersect at a common point, as in a spherical wrist.

(b) Three consecutive revolute joint axes are parallel.

The most effective methods for obtaining closed-form solutions involve analytical techniques that utilize specific geometric features of particular mechanisms. Generally, closed-form solutions can be obtained for systems that have six or fewer degrees of

freedom, characterized by geometric parameters using the Denavit-Hartenberg convention, as described in Section 3.3.

3.5.2 Numerical Solutions

The numerical methods for solving inverse kinematics problems are not dependent on the geometry of the robotic mechanisms, unlike the closed-form solutions discussed in the previous section (see Sect. 3.5.1). Therefore, they can be applied to any kinematic structure, regardless of the number of degrees of freedom. Since numerical methods require the approximation of derivatives, they are computationally more expensive than methods based on closed-form solutions. However, it is possible for them to operate directly in a joint space. This simplifies the use of joint limits and allows for greater flexibility in including additional objectives.

In general, the numerical solution of the discussed problem provides the relationship between the joint velocities and the corresponding end-effector linear and angular velocity. In other words, it is desirable to express the linear velocity of the end-effector $\dot{\mathbf{p}}_e$ and the angular velocity of the end-effector $\dot{\omega}_e$ as functions of the joint velocities $\dot{\theta}$.

$$\dot{\mathbf{p}}_e = \mathbf{J}_P(\theta)\dot{\theta}, \tag{3.20}$$

$$\dot{\omega}_e = \mathbf{J}_O(\theta)\dot{\theta}, \tag{3.21}$$

where $\mathbf{J}_P \in \mathbb{R}^{3\times n}$ represents the translational part of the manipulator that relates the contribution of the joint velocities to the linear velocity of the end-effector, while $\mathbf{J}_O \in \mathbb{R}^{3\times n}$ represents the rotational part of the manipulator that relates the contribution of the joint velocities to the angular velocity of the end-effector.

In compact form, equations (3.20) and (3.21) can be written as

$$\mathbf{v}_e = \begin{bmatrix} \dot{\mathbf{p}}_e \\ \dot{\omega}_e \end{bmatrix} = J(\theta)\dot{\theta}, \tag{3.22}$$

which represents the general numerical kinematics equation. The vector $\mathbf{v}_e$ is the spatial velocity of the end-effector expressed in an arbitrary frame, $\dot{\theta}$ is an n-dimensional vector composed of the joint velocities, and $J(\theta) \in \mathbb{R}^{6\times n}$ is the so-called Jacobian matrix of non-linear functions of θ, expressed relative to the same coordinate frame as the spatial velocity $\mathbf{v}_e$. The Jacobian matrix is a matrix of first-order partial derivatives of each joint variable, providing a linear approximation of the resulting end-effector velocities in Cartesian space.

$$J(\theta) = \begin{bmatrix} \mathbf{J}_P \\ \mathbf{J}_O \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{P_1} & \dots & \mathbf{J}_{P_n} \\ \vdots & \ddots & \vdots \\ \mathbf{J}_{O_1} & \dots & \mathbf{J}_{O_n} \end{bmatrix} = \begin{bmatrix} \frac{\partial p_e^x}{\partial \theta_1} & \dots & \frac{\partial p_e^x}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \omega_e^z}{\partial \theta_1} & \dots & \frac{\partial \omega_e^z}{\partial \theta_n} \end{bmatrix}. \tag{3.23}$$

It can be observed from Equation (3.23) that the i-th column of the Jacobian matrix corresponds to the i-th joint of the robot manipulator. Depending on whether the i-th joint is prismatic or revolute, it takes one of two forms [66, 57], as follows

$$J_i(\theta) = \begin{bmatrix} \mathbf{J}_{P_i} \\ \mathbf{J}_{O_i} \end{bmatrix} = \begin{cases} \begin{bmatrix} p^z_{e_{i-1}} \\ 0 \end{bmatrix} \rightarrow \text{prismatic joint} \\ \begin{bmatrix} p^z_{e_{i-1}} \times (\mathbf{p}_{e_i} - \mathbf{p}_{e_{i-1}}) \\ p^z_{e_{i-1}} \end{bmatrix} \rightarrow \text{revolute joint} \end{cases} \tag{3.24}$$

To solve the linear system of equations in the joint velocities obtained by the decomposition, expressed in Eq. (3.22), into its component equations when $\mathbf{v}_e$ is known, it is necessary to invert the Jacobian matrix as follows

$$\dot{\theta} = J(\theta)^{-1}\mathbf{v}_e. \tag{3.25}$$

However, the numerical approach is only feasible if the Jacobian matrix J is invertible. This requires the matrix to be square-shaped and have a non-zero determinant. It is clear from that statement that redundant manipulators will not be invertible because the Jacobian matrix will not be square. Although the robotic mechanism is not in a singularity configuration, but only in a near-singular configuration, the values of J^{-1} become very large, leading to very high joint space velocities and instability in the numerical solution approach. Various methods are employed to address the aforementioned shortcomings, specifically to reduce instability and enhance the usability of redundant robotic structures. These methods will be discussed further.

Jacobian Transpose Method

The numerical computation of the inverse kinematics problem can be achieved using the Jacobian transpose method, which involves using the transpose of the Jacobian matrix instead of its inverse [81, 82]. This method modifies the general equation (3.25) of numerical inverse kinematics solutions as follows

$$\dot{\theta} = \alpha J(\theta)^T \mathbf{v}_e, \tag{3.26}$$

for some appropriate scalar α, to minimize the new value of the vector after the update. The method of expressing the parameter α was described in [83], where the relationship between the desired change $\mathbf{v}_e$ and an approximated change of the end-effector, defined as $J(\theta)J(\theta)^T\mathbf{v}_e$, was used. Assuming that $J(\theta)J(\theta)^T\mathbf{v}_e$ is the real change, α is chosen to be as close as possible to $\mathbf{v}_e$ as follows

$$\alpha = \frac{\left\langle \mathbf{v}_e, J(\theta)J(\theta)^T\mathbf{v}_e \right\rangle}{\langle J(\theta)J(\theta)^T\mathbf{v}_e, J(\theta)J(\theta)^T\mathbf{v}_e \rangle}. \tag{3.27}$$

It is important to note that the transpose method generates smooth motion but often suffers from slower convergence.

Newton-Raphson Method

The numerical computation of the inverse kinematic problem is often solved using the Newton-Raphson method [2]. A common implementation involves taking advantage of the Moore-Penrose pseudoinverse [84], leading to the so-called pseudoinverse method.. This method is defined by the equation as follows

$$\dot{\theta} = J(\theta)^{\dagger}\mathbf{v}_e, \tag{3.28}$$

where $J(\theta)^{\dagger} = J(\theta)^T(J(\theta)J(\theta)^T)^{-1}$ represents the pseudoinverse of $J(\theta)$ with the same dimensions. This definition holds for all matrices $J(\theta)$, even those that are not square or do not have full row rank.

The pseudoinverse method is a technique used to enhance the stability and usability of redundant robotic structures where ordinary inversion is not feasible [83]. In simple terms, it helps address the challenges posed by having more degrees of freedom than necessary. However, it is important to note that the pseudoinverse method can exhibit instability in nearly singular configurations. In these situations, the system may experience jerky movements, lacking the desired smoothness and stability. Additionally, it may encounter difficulties in finding feasible solutions due to the peculiarities of these configurations.

Gauss-Newton Method

The Gauss-Newton method is a numerical approach employed to enhance the solvability and compatibility of the inverse kinematics problem, particularly in the context of redundant robotic manipulators [85, 74]. This method utilizes the Jacobian matrix $J(\theta)$ to approximate the Hessian matrix $\mathbf{H}$. The approximation is necessary because computing the exact Hessian matrix can be computationally expensive, and often, it is not necessary to obtain the exact Hessian.

$$\dot{\theta} = (\mathbf{H})^{-1}J(\theta)^T\mathbf{W}_e\mathbf{v}_e, \tag{3.29}$$

where $\mathbf{W}_e = \text{diag}(\mathbf{w_e})(\mathbf{w_e} \in (\mathbb{R}^+)^n)$ is a diagonal weighted matrix that prefers the corresponding error term, and $\mathbf{H} = J(\theta)^T\mathbf{W}_eJ(\theta)$ describes the approximate Hessian matrix.

However, equation (3.29) is only valid if $\mathbf{H}$ is invertible, as described in [86]. This issue can be resolved, as mentioned earlier, by using the Moore-Penrose pseudoinverse applied to the aforementioned part ($\mathbf{H}^{-1} \rightarrow \mathbf{H}^{\dagger}$).

Levenberg-Marquardt Method

The Levenberg-Marquardt method, also known as the Damped Least Squares (DLS) method, is designed to address issues associated with the pseudoinverse method, particularly when dealing with near singularities in the Jacobian matrix, denoted as $J(\theta)$. It provides a numerically stable approach for selecting the parameter $\dot{\theta}$.

$$\dot{\theta} = (\mathbf{A})^{-1}J(\theta)^T\mathbf{W}_e\mathbf{v}_e, \tag{3.30}$$

where $\mathbf{W}_e = \text{diag}(\mathbf{w}_e)$ ($\mathbf{w}_e \in (\mathbb{R}^+)^n$) is a diagonal weighted matrix that prefers the corresponding error term, and $\mathbf{A}$ is defined as follows

$$\mathbf{A} = J(\theta)^T \mathbf{W}_e J(\theta) + \mathbf{W}_n, \tag{3.31}$$

where $\mathbf{W}_n = \text{diag}(\mathbf{w_n})(\mathbf{w_n} \in (\mathbb{R}^+)^n)$ is a diagonal damping matrix. The damping matrix guarantees that $\mathbf{A}$ is both non-singular and positive definite. As is evident, the performance of the method is highly dependent on the selection of $\mathbf{W}_n$.

The diagonal damping matrix w_n was first described as a constant by Wampler [87]. A little later, Chan and Lawrence [88] proposed a damped least-squares method where w_n directly depends on the error value $E \in \mathbb{R}$. Last but not least, Sugihara [74] proposed a robust method for obtaining the damping matrix $\mathbf{W}_n$, as follows

$$\mathbf{W}_n = E\mathbf{1}_n + \text{diag}(\tilde{\mathbf{w}}_n), \tag{3.32}$$

where the value of $\tilde{\mathbf{w}}_n$, which causes the bias, provides sufficiently robust convergence of the calculation, with $\tilde{\mathbf{w}}_n = 1.0 \times 10^{-4}$.

CHAPTER 4

Motion Planning

The motion planning problem, shown schematically in Figure 4.1, is a fundamental and challenging research topic in industrial robotics. It involves generating a collision-free path or trajectory for various robotic structures to perform specific tasks while adhering to constraints and avoiding obstacles. The main objective is to determine an efficient path for the robotic manipulator $\mathcal{R}$ to move from its initial configuration θ_s to the desired final configuration θ_f and generate reference inputs for the motion control system. Traditionally, formulations of the motion planning problem for robotic manipulators rely on the concept of configuration space ($\mathcal{C}$ or C-space) [1, 57], representing all possible transformations based on the robot's kinematics model.

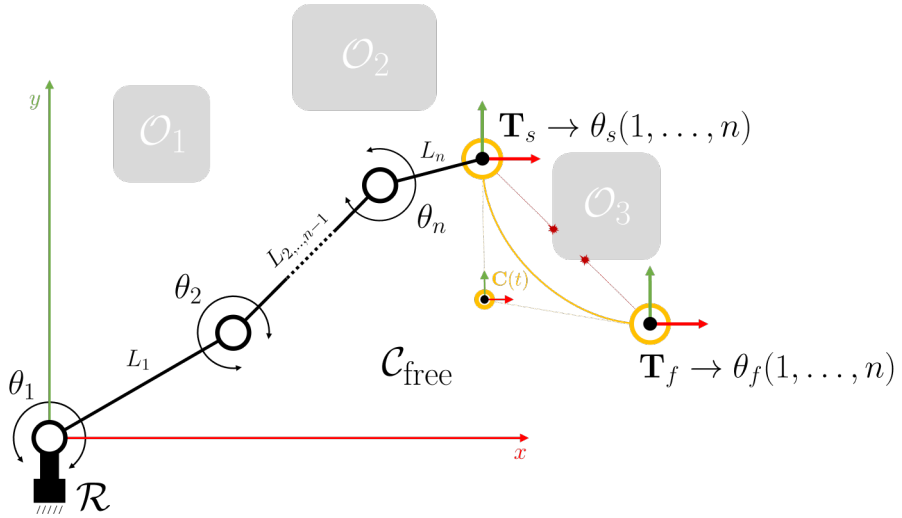


Figure 4.1: A schematic representation of a motion planning problem in two-dimensional space, illustrating an n-link robotic manipulator $\mathcal{R}$ within a collision-free configuration space $\mathcal{C}_{\text{free}}$ with obstacles denoted as $\mathcal{O}$.

The chapter begins by discussing the basics of the motion planning problem for robotic manipulators, which includes the concept of configuration space (Section 4.1) and the obstacle avoidance problem (Subsection 4.1.1). This leads to an examination of an efficient method for approximating individual obstacles using bounding boxes. After this introduction, the chapter presents a brief summary of the trajectory generation problem (Section 4.2). This section explores parametric curves (Subsection 4.2.1), specifically Bézier and B-Spline, the trapezoidal motion profile (Subsection 4.2.2), and motion through a sequence of points known as via points (Subsection 4.2.3). The final part of the chapter concentrates on reinforcement learning methods for motion planning problems in the field of robotic manipulators (Section 4.3). The discussion encompasses a description of the fundamental concept of reinforcement learning (Subsection 4.3.1) and its extension called deep reinforcement learning (Subsection 4.3.2), which incorporates deep learning techniques into traditional reinforcement learning methods.

4.1 Configuration Space

In the context of general motion planning, it is essential to describe the geometry of the robotic manipulator $\mathcal{R}$ and its workspace, denoted as $\mathcal{W} \in \mathbb{R}^{2\vee 3}$ [1, 57]. The workspace $\mathcal{W}$ is defined in a two-dimensional (2D) or three-dimensional (3D) space and is considered a static environment populated with obstacles, denoted as $\mathcal{O}$. This workspace serves as the operational domain for the robotic manipulator $\mathcal{R}$, and understanding its structure is fundamental to addressing challenges in motion planning. The main objective is to find a collision-free path for $\mathcal{R}$ to move from an initial (or starting) configuration θ_s to the final configuration θ_f.

In the initial phases of motion planning research, it was recognized that the use of configuration space proves to be a valuable abstraction to address planning problems [89]. The physical space in which robots and obstacles exist can also be considered as the world space. The configuration space $\mathcal{C}$, also known as the C-space, represents the set of all transformations that can be applied to a robot based on its kinematics ($\theta \in \mathcal{C}$), as explained in Chapter 3. In the context of robotic manipulators, a configuration refers to the set of absolute joint positions that completely describe the robot's state. The C-space represents all possible joint configurations, creating a multidimensional space where each dimension corresponds to a specific joint. The collision-free C-space, denoted as $\mathcal{C}_{\text{free}}$, is composed of configurations in which the robot does not collide with any obstacles or violate any joint limits.

To ensure a collision-free path, it is necessary to prevent the robot from reaching configurations with absolute joint positions θ that could potentially lead to contact with obstacles. The region of configurations in which the robot collides with an obstacle is known as the C-space obstacle region [1, 66], defined as

$$\mathcal{C}_{\mathcal{O}} = \{\theta \in \mathcal{C} \mid \mathcal{R}(\theta) \cap \mathcal{O} \neq \emptyset\}, \tag{4.1}$$

where $\mathcal{O} = \bigcup \mathcal{O}_i$ is expressed as a subset of the workspace occupied by the i-th obstacle, denoted as $\mathcal{O}_i$. The set of collision-free configurations, known as the free configuration

space, can be described as

$$\mathcal{C}_{\text{free}} = \mathcal{C}/\mathcal{C}_{\mathcal{O}}. \tag{4.2}$$

4.1.1 Obstacle Avoidance

The obstacle avoidance problem involves controlling the robotic manipulator to follow the specified path of the end-effector while ensuring the avoidance of collisions between the robot and the workspace obstacles, as well as self-collisions of the individual parts of the robotic structure. To prevent collisions with potential obstacles, the manipulator must reposition itself and adjust its configuration to increase the distance between the manipulator and the obstacles. Additionally, to avoid self-collisions, it is necessary to rotate the joint where the collision occurs to a new, collision-free position.

An integral part of determining the distance between two independent objects is the so-called Euclidean distance, defined as

$$d(\mathbf{p}_1, \mathbf{p}_2) = \sqrt{\sum_{i=1}^{n} (p_1^i - p_2^i)^2}, \tag{4.3}$$

where $\mathbf{p}_1$, $\mathbf{p}_2$ represents two closest points of the independent obstacles in Euclidean n-dimensional space.

To address the challenge of collision avoidance in robotics, encompassing both the detection of collisions between the robot and obstacles within the workspace and the subsequent self-collision, an effective approach is to approximate individual objects using bounding boxes [90, 91, 92].

There are two types of bounding boxes that can represent objects in two-dimensional or three-dimensional space (see Fig. 4.2), namely AABB (Axis-Aligned Bounding Box) and OBB (Oriented Bounding Box).

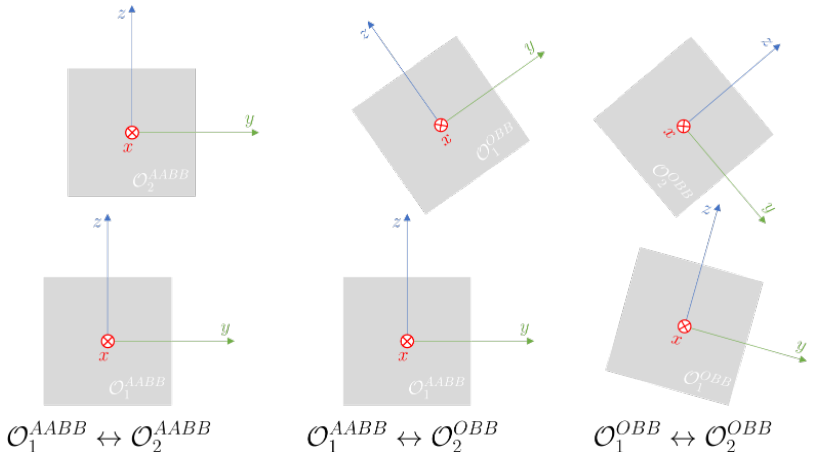


Figure 4.2: An illustration of different approaches to object detection using bounding boxes.

Axis-Aligned Bounding Box (AABB)

The Axis-Aligned Bounding Box (AABB) is a bounding box with edges parallel to the axes of the coordinate system [93, 94]. It is defined by a centroid and a size, taking the form of a rectangle in two-dimensional space and a cuboid in three-dimensional space. The centroid and size parameters characterize the position and dimensions of the AABB. One of the key advantages of the AABB is its fast overlap check, achieved through a straightforward comparison of individual coordinate values. This makes the AABB a particularly efficient choice for certain applications.

Oriented Bounding Box (OBB)

The Oriented Bounding Box (OBB) is a bounding box whose edges are not parallel to the axes of the coordinate system [93, 94]. It is similar to an Axis-Aligned Bounding Box (AABB) but allows for arbitrary orientation. In general, an OBB is defined by its position, size, and orientation in a two-dimensional or three-dimensional space. The Separating Axis Theorem (SAT) [95] is commonly used to determine whether two OBBs intersect. According to this theorem, two convex shapes do not overlap if there exists a separating axis along which the projections of the shapes do not intersect.

4.2 Trajectory Generation

In the context of robotic manipulators, achieving smooth motion from the initial position to the desired final position involves organizing the movement of each joint through a coordinated and time-dependent approach [66, 96]. This process, known as trajectory planning, aims to generate a time sequence of values based on an interpolating function, typically a polynomial [97], that defines the desired trajectory. Therefore, generating trajectories becomes the computational challenge of determining these functions.

In order to avoid confusion between terms often used interchangeably, let's delve into the distinct meanings of 'path' and 'trajectory.' Initially, both terms might be casually considered synonymous. However, a path refers to the set of points in the joint space or operational space, that describe the geometric trajectory that the manipulator must follow during the assigned motion. In contrast, a trajectory is a specific path with a defined timing law, that incorporates details such as velocities and/or accelerations at each point.

It is important to note that a trajectory is defined not only by its initial and final positions, but also by intermediate points known as via points [2]. These via points are critical because they determine the trajectory that the manipulator must follow en route to the desired position. Trajectories can be defined by a geometric curve or a sequence of points, either in an operational space or in joint space. In operational space, the trajectory represents the movement of the end-effector, while in joint space, it defines the motion of individual joints.

4.2.1 Parametric Curves

Path planning is an essential component of robotic control and plays a pivotal role in ensuring the efficient and smooth movement of various types of robotic manipulators. The most effective solution to this problem involves using parametric curves [98, 99, 100], which are mathematical representations that describe the robot's motion as a function of time. Parametric curves are widely used mathematical tools in robotics and related fields. The two most commonly used representations of parametric curves in the path planning of robotic manipulators are Bézier and B-Spline curves [101, 102]. Both describe the motion of a robot as a polynomial function that depends on time. Each has unique features, and the main difference between them lies in the complexity of their mathematical definitions.

Polynomial interpolation is the process of fitting a polynomial function to a set of given data points, resulting in a continuous trajectory that passes precisely through these waypoints. The most common approach to constructing a curve from a set of points is to interpolate the points or require the curve to pass through them.

The basis of parametric curves involves a linear interpolation (LERP) technique utilized in straight-line spaces [103, 104], described by a geometric formula as follows

$$\mathrm{LERP}(\mathbf{p}_0, \mathbf{p}_1; t) = (1 - t)\mathbf{p}_0 + t\mathbf{p}_1, \tag{4.4}$$

where $\mathbf{p}_0$ and $\mathbf{p}_1$ represent the initial and desired final points in in straight-line space, and $t \in [0, 1]$ is the interpolation parameter. The interpolation in rotational spaces [103], known as spherical linear interpolation (SLERP), is described by the geometric dot product of two quaternions as follows

$$\mathrm{SLERP}(\mathbf{Q}_0, \mathbf{Q}_1; t) = (\mathbf{Q}_1\mathbf{Q}_0^{-1})^t\mathbf{Q}_0, \tag{4.5}$$

where $\mathbf{Q}_0$ and $\mathbf{Q}_1$ represent the initial and desired final points in rotational space, and $t \in [0, 1]$ is the interpolation parameter.

An integral aspect of efficient path planning, particularly in the context of parametric curves with a focus on minimizing path length, involves a mathematical concept known as arc length. Generally, arc length, denoted as L, quantifies the distance along a curve. For parametric curves, where positions are defined as functions of points depending on time, arc length can be expressed as an integral that involves the derivatives of the parametric functions [105], as described in the following equation

$$L(t) = \int_0^t \sqrt{(\dot{x}(t)^2 + \dot{y}(t)^2)}dt. \tag{4.6}$$

Bézier Curves

A Bézier curve, developed in the 1960s by Pierre Bézier [106], is a parametric curve used for various applications, including robotic path planning. It is defined by a set of control points $\mathbf{p}_0$ to $\mathbf{p}_n$, where n is the degree of the curve. It is important to note that a Bézier curve of degree n requires $n + 1$ control points. The static points of the curve include the

initial control point $\mathbf{p}_0$ and the final control point $\mathbf{p}_n$. Intermediate control points, if present, generally do not lie on the curve. Instead, the curve smoothly interpolates the shape defined by these control points, with $\mathbf{p}_0$ and $\mathbf{p}_n$ serving as the anchor points.

The general form of the parametric n-degree Bézier curve for $n+1$ control points can be defined as follows

$$\mathbf{C}(t) = \sum_{i=0}^{n} B_{i,n}(t)\mathbf{p}_i, \tag{4.7}$$

where $B_{(i,n)}(t)$ is a Bernstein polynomial, defined as

$$\mathbf{B}_{i,n}(t) = \binom{n}{i}(1-t)^{n-i}t^i, \tag{4.8}$$

where $\binom{n}{i}$ is the binomial coefficient, and $t \in [0,1]$ is the interpolation parameter that can be represented as a normalized time.

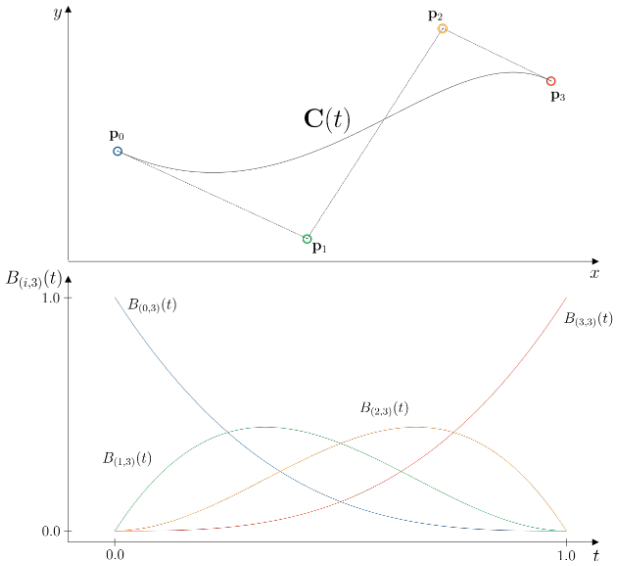


Figure 4.3: A parametric Bézier curve of degree $n = 3$ (top), denoted as a cubic Bézier curve, with the corresponding representation of Bernstein basis polynomials (bottom).

B-Spline Curves

The term B-spline was introduced by Curry and Schoenberg in [107, 108]. It is short for a 'basis spline'. The recursive definition, commonly used to introduce B-splines, was presented by Carl R. de Boor in [109]. B-splines, similar to Bézier curves, use polynomials to create a parametric curve. However, unlike Bézier curves, B-splines use a sequence of control points to define the local geometry of the curve. This characteristic guarantees that only a small section of the curve is affected when a control point is adjusted.

As mentioned above, a Bézier curve of degree n requires $n+1$ control points. Therefore, fitting a Bezier curve to a large number of control points can be computationally expensive due to the large number of coefficients (i.e. Bernstien polynomials) that need to be determined. In contrast, B-splines have the advantage of their degrees being completely independent of the number of control points. B-splines offer the advantages of Bezier curves without the problems of continuity and coefficient computation. This is because they interpolate multiple segments of the Bezier curve instead of multiple points.

The general form of the parametric n-degree B-Spline curve for $n+1$ control points can be defined as follows

$$\mathbf{C}(t) = \sum_{i=0}^{n} N_{i,n}(t)\mathbf{p}_i, \tag{4.9}$$

where $N_{i,n}(t)$ is a B-spline basic function defined as

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{4.10}$$

$$N_{i,n}(t) = \frac{t - t_i}{t_{i+n} - t_i} N_{i,n-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,n-1}(t), \tag{4.11}$$

where t_i is the so-called knot vector, represented as non decreasing sequence of real numbers that can be obtained through various methods [110], including Uniformly-Spaced, Centripetal, or Chord Length, and $t \in [0, 1]$ is the interpolation parameter that can be represented as a normalized time.

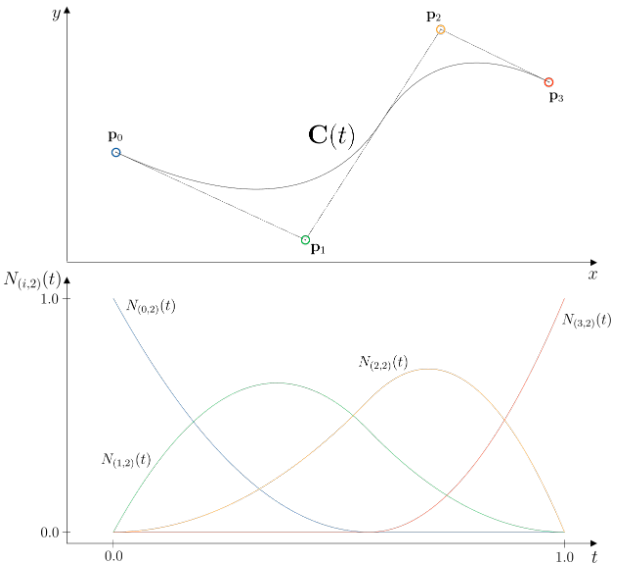


Figure 4.4: A parametric B-spline curve of degree $n = 2$ (top), with the corresponding representation of B-Spline basis functions (bottom).

4.2.2 Trapezoidal Motion Profiles

One of the most fundamental ways to generate trajectories in joint space is by using so-called linear segments with parabolic blends (LSPB). This type of trajectory is characterized by a trapezoidal motion profile [111, 112], which is divided into three phases: acceleration, constant velocity, and deceleration. The resulting trajectory is formed by a linear segment connected by two parabolic segments to the initial and final positions. The velocity increases to the desired value and then decreases as it approaches the final position.

To generate a desired motion profile, briefly specify the three phases mentioned above (see Fig. 4.5). The first phase, referred to as the acceleration phase, spans from the initial time, denoted as t_s, to the mixture time, denoted as t_b. During this phase, a linear velocity, often termed a ramp, is represented by a quadratic polynomial. The second phase, known as the constant velocity phase, extends throughout the blend time. This phase is characterized by a linear function of time, which corresponds to a constant velocity. The third and final phase, termed the deceleration phase, transitions the trajectory back to a quadratic polynomial $t_f - t_b$, where t_f is the desired finite time. This phase completes the motion profile, providing a comprehensive description of the desired trajectory.

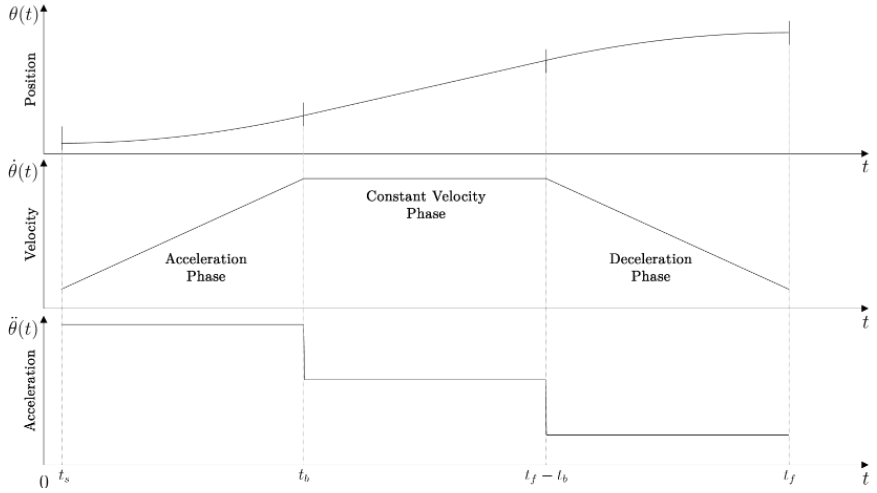


Figure 4.5: Position (θ), velocity ($\dot{\theta}$) and acceleration ($\ddot{\theta}$) of a linear trajectory with a parabola (LSPB), characterized by a trapezoidal motion profile.

Assuming the general case where $t_s \neq 0$, the trapezoidal motion profile for the position part can be defined as follows

$$\theta(t) = \begin{cases} \theta_s + \frac{v}{2t_b}(t - t_s)^2, & t_s \leq t < t_s + t_b \\ \theta_s + v(t - t_s - \frac{t_b}{2}), & t_s + t_b \leq t < t_f - t_b \\ \theta_f - \frac{v}{2t_b}(t_f - t)^2, & t_f - t_b \leq t \leq t_f \end{cases}, \tag{4.12}$$

where θ_s and θ_f are the initial and desired final positions of the joint, and v is the constant velocity.

4.2.3 Motion Through a Sequence of Points

In robotics applications, achieving smooth and effective motion along a pre-defined path is often a critical requirement [113]. This involves motion planning through a sequence of points, commonly known as via points, without interruptions (see Fig. 4.6). Such seamless motion is essential for tasks such as avoiding obstacles in the workspace or executing actions involving intermediate points between the initial and final positions, as exemplified by the widely addressed 'pick and place' task.

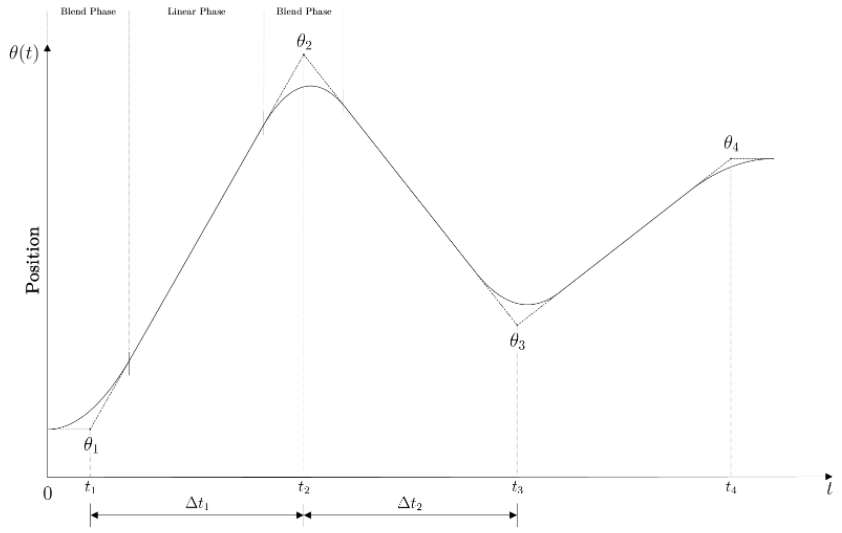


Figure 4.6: Characterization of motion through a sequence of points using a linear function with parabolic blends (LSPB).

One of the most fundamental approaches to generate motion through a sequence of points is the utilization of linear paths with parabolic blends, as mentioned above. This concept can be extended to scenarios that involve an arbitrary number of specified via points, as described in [114, 65]. In this method, linear functions are employed to connect the via points, with parabolic blend regions seamlessly integrated around each via point.

4.3 Reinforcement Learning for Motion Planning Problems in Robotic Manipulators

Motion planning is a fundamental problem in industrial robotics because it involves generating a path or trajectory for various types of robotic structures to follow, accomplishing a specific task while avoiding obstacles, and complying with various constraints. The traditional task of planning trajectories for robots with high degrees of freedom,

such as industrial robots (typically four to seven DOFs), has undergone a significant revolution in motion planning with the development of sampling-based algorithms [10, 11]. Representative examples of these algorithms include Rapidly Exploring Random Trees (RRTs) and Probabilistic Roadmaps (PRMs). The motion planning problem has also been addressed using evolutionary computation techniques, as discussed in the comprehensive review [115]. Another approach to solving this problem involves using reinforcement learning (RL) algorithms [5] to complete partial tasks within a specific area. In contrast to traditional methods, which often rely on pre-defined maps and algorithms, RL has emerged as a groundbreaking paradigm, revolutionizing path planning by allowing autonomous agents to learn and adapt their strategies through experience.

In recent years, the combination of deep neural networks (DNN) [3] and reinforcement learning (RL), termed deep reinforcement learning (DRL) [6], has become a popular choice for end-to-end control in robotics research, where sequential actions are learned directly from raw input observations, especially in the problem of reaching a static [12] or random target [13, 14, 15]. Among the most commonly used algorithms for the presented problem are the actor-critic model-free algorithms such as the Deep Deterministic Policy Gradient (DDPG) [116], the Twin Delayed Deep Deterministic Policy Gradient (TD3) [117], and the Soft Actor-Critic (SAC) [118], often used with an experience replay buffer [119].

4.3.1 Fundamental Concept of Reinforcement Learning

Reinforcement learning (RL) is an area of machine learning in which the intelligent agent learns how to achieve specific behavior by maximizing rewards from the environment through the actions it chooses to execute, much like a human being, through the trial and error method. A higher reward encourages the agent to choose the selected action in the future, while a lower reward, also known as a penalty, has the opposite effect. The idea is that the agent will learn how to achieve its behavior within the specific environment in an effective way that converges to an optimal solution.

The problem of reinforcement learning is formally defined in terms of optimal control of a Markov Decision Process (MDP). Figure 4.8 illustrates its basic structure in an iteration loop. A characteristic property of MDPs is that each state is only dependent on the previous state, i.e., a memoryless property where each state contains all the information necessary to predict the next state.

A Markov Decision Process is a discrete-time stochastic control process and can be represented as the tuple $(S,\ A,\ R,\ p)$, where S is the continuous multidimensional state space, and A is the continuous multidimensional action space. The state transition probabilities are defined by the function $p : S \times S \times A \to [0, 1]$, which represents the probability density of the next state $s' \in S$ based on the current state $s \in S$ and action $a \in A$.

$$p(s' \mid s, a) = \mathbb{P}\left\{S_{t+1} = s' \mid S_t = s, A_t = a\right\} \tag{4.13}$$

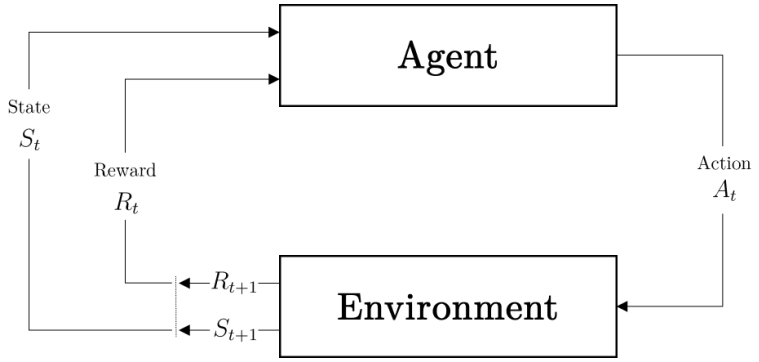


Figure 4.7: The interaction between an agent and the environment in a Markov Decision Process (MDP) [5].

The reward function, defined by $R : S \times S \times A \to \mathbb{R}$, represents the immediate reward that the agent receives after the transition from $s \in S$ to $s' \in S$ using an action $a \in A$.

$$R(s, a, s') = \mathbb{E}\left\{R_t \mid S_t = s, A_t = a, S_{t+1} = s'\right\} \tag{4.14}$$

The objective of the process is for the agent to learn optimal behavior, also known as a policy denoted as $\pi = \pi(a \mid s)$, which maximizes its discounted expected return. A discount factor γ in the range $(0, 1]$ is usually introduced when calculating the long-term reward, resulting in the following expression for the continuous ongoing process

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k}, \tag{4.15}$$

where γ determines the priority of long-term future rewards.

In the field of reinforcement learning, the value function is a crucial concept that guides decision-making processes. It serves as a quantitative measure of the expected cumulative reward that an agent can anticipate while occupying a specific state or following a particular action. This metric helps the agent assess the desirability of states and actions, facilitating the learning of an optimal policy.

More formally, a state value function can be defined to determine the expected return when following a policy π for a particular state s, with the value function $V^\pi(s)$.

$$V^\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s] \tag{4.16}$$

The action value function, which represents the expected discounted return when starting in state s and initially taking action a but then following policy π, can be defined as follows

$$Q^\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]. \tag{4.17}$$

The primary goal of the agent in reinforcement learning is to find the optimal policy π^*, which is better than or equal to all other policies. This can be achieved by estimating the corresponding optimal functions for the action value function, defined as follows

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a). \tag{4.18}$$

The optimal action value function is satisfied by the Bellman equation, a fundamental concept in reinforcement learning. The Bellman optimality equation explains that the value of a state is equal to the expected return when the best action a' is taken in that state. The equation for Q^* is defined as follows

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(s',a') \mid S_t = s, A_t = a]. \tag{4.19}$$

The Bellman optimality equation is a fundamental formula in reinforcement learning theory. After finding the optimal action value function $Q^*(s,a)$, the optimal policy π^* can be followed by selecting the optimal action a^* in each state s, defined as

$$a^*(s) = \operatorname*{argmax}_{a'} Q^*(s,a'). \tag{4.20}$$

It is important to note that the optimization of the value function is often performed off-policy and can, therefore, utilize experience replay.

The well-known and straightforward algorithm for this type of problem is Q-learning, as described in [120]. The Q-learning algorithm belongs to the category of Temporal Difference (TD) learning [121]. TD learning is a form of a value-based approach in which the value function is optimized by minimizing the TD error, denoted as δ.

$$\delta_t = R_t + \gamma \max_{a'} Q^{\pi}(s',a') - Q^{\pi}(s_t,a_t) \tag{4.21}$$

The Q-learning update rule for the estimation of $Q^*(s_t,a_t)$ becomes an optimization problem described as follows

$$Q^*(s_t,a_t) \leftarrow Q^{\pi}(s_t,a_t) + \alpha\delta_t = Q^{\pi}(s_t,a_t) + \alpha[R_t + \gamma \max_{a'} Q^{\pi}(s',a') - Q^{\pi}(s_t,a_t)], \tag{4.22}$$

where $\alpha \in [0,1)$ is the learning rate.

It is important to note that the standard form of Q-learning is inefficient for large environments because it must consider every possible state-action pair to determine the optimal $Q^*(s,a)$, for example, by using a tabular approach. To address this inefficiency, a general approximator based on deep neural networks can be used instead of large tables.

4.3.2 Deep Reinforcement Learning

Deep reinforcement learning (DRL) [6] is an extension of reinforcement learning (RL) that uses deep neural networks [3] as function approximators for value functions or policies. RL, at its core, involves training agents to make sequential decisions in environments, and

DRL enhances this process by employing deep neural networks to handle high-dimensional data.

DRL has significant applications in robotics, especially in addressing challenges related to path planning. In this context, several DRL algorithms, also known as actor-critic model-free RL algorithms, such as the Deep Deterministic Policy Gradient (DDPG) [116], the Twin Delayed Deep Deterministic Policy Gradient (TD3) [117], and the Soft Actor-Critic (SAC) [118], often used with an experience replay buffer [119], have proven effective. These algorithms optimize both the policy and the value functions with the overarching goal of maximizing cumulative rewards, enabling them to converge to optimal results. To enhance the learning process, an experience replay buffer is commonly employed to store transitions during interactions with the environment.

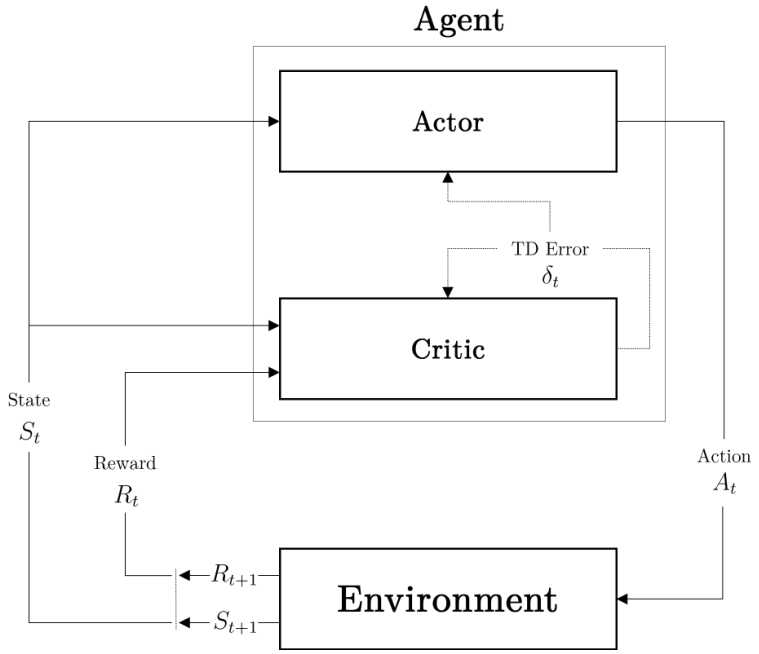


Figure 4.8: An overview of the actor-critic architecture. The Temporal Difference (TD) error δ_t is utilized to adjust both the critic's action value function $Q(s, a)$ and the actor's policy $\pi(a \mid s, \theta)$, which is parameterized by θ [5, 6].

The integration of Deep Reinforcement Learning (DRL) into robotics and path planning demonstrates its capability to handle complex and high-dimensional data, facilitating the effective learning of various tasks [13, 14, 15]. In the following sections, the details of the DDPG, TD3 and SAC algorithms are briefly described.

Deep Deterministic Policy Gradient (DDPG)

The Deep Deterministic Policy Gradient (DDPG) [116] is an off-policy reinforcement learning algorithm designed for continuous action spaces that uses a deterministic policy.

It utilizes two neural networks: the actor and the critic. The actor network predicts the optimal action to take in a given state, while the critic network evaluates the quality of the actor's chosen action.

The DDPG algorithm learns the optimal policy and Q-function simultaneously. Initially, it uses the Bellman equation and off-policy memory to learn the Q-function. Subsequently, the algorithm utilizes the acquired Q-function to learn the policy. The primary objective is to determine the optimal action value, denoted as a^*, for each state in a continuous control environment, similar to the Q-learning method (see Eq. 4.19).

The Bellman equation is utilized to estimate the optimal Q-function through a set of artificial neural networks, denoted as $Q_\theta(s, a)$, where θ represents the network's parameter. The principle of the DDPG algorithm is to minimize the Mean Squared Bellman Error (MSBE) function defined for the set D of transitions (s, a, R, s', d). The equation for the MSBE function can be described as follows

$$L(\theta, D) = \mathbb{E}_D[(Q_\theta(s, a) - (R_t + \gamma(1 - d)\max_{a'} Q_\theta(s', a'))^2], \tag{4.23}$$

where d indicates whether the terminal condition is satisfied or not, and the term $(R_t + \gamma(1 - d)\max_{a'} Q_\theta(s', a'))$ is called the target.

The problem is that both Q_θ and the target depend on the same parameter θ. Due to this dependency, the minimization of the MSBE function becomes unstable. Therefore, another neural network, $Q_{\theta_{\text{targ}}}$, referred to as the target network, is used to compute the error function. Since the action space is infinite, it is possible to use another neural network, μ_{targ}, known as the target policy network, to approximate the function $\max Q_\theta(s', a')$. The target networks are then updated using the parameters of the main networks using the Polyak averaging formula [122].

Based on the above information, the Eq. 4.23 will be rewritten as follows

$$L(\theta, D) = \mathbb{E}_D[(Q_\theta(s, a) - (R_t + \gamma(1 - d)\max_{a'} Q_{\theta_{\text{targ}}}(s', \mu_{\text{targ}}(s')))^2]. \tag{4.24}$$

Twin Delayed Deep Deterministic Policy Gradient (TD3)

The Twin Delayed Deep Deterministic Policy Gradient (TD3) [117] is an off-policy reinforcement learning algorithm designed for a continuous action space. It is an extension of the DDPG algorithm that aims to overcome some of its limitations by implementing several significant improvements.

Firstly, the action is selected using the so-called target policy smoothing, as follows

$$a'(s') = \text{clip}(\mu_{\text{targ}}(s') + \text{clip}(\epsilon, -c, c), a_{\text{Low}}, a_{\text{High}}), \tag{4.25}$$

where ϵ is a random variable with a normal distribution, and c is a hyperparameter.

Secondly, the algorithm uses clipped double Q-learning [123], which involves learning two Q-functions concurrently instead of one and selecting the smaller one to calculate the target term in the Bellman equation. This approach enables TD3 to more effectively

capture the uncertainty in the environment and reduce the overestimation of value estimates that may occur in DDPG.

The TD3 algorithm includes two primary networks, denoted as Q_{θ_0} and Q_{θ_1}, and two target networks, denoted as $Q_{\theta_{0,\text{targ}}}$ and $Q_{\theta_{1,\text{targ}}}$. The general form of the Mean Squared Bellman Error (MSBE) function for the TD3 algorithm is described in Eq. 4.26.

$$L(\theta_i, D) = \mathbb{E}_D[(Q_{\theta_i}(s,a) - (R_t + \gamma(1-d)\min_{i\in 0,1} Q_{\theta_{i,\text{targ}}}(s', \mu_{\text{targ}}(s')))^2] \tag{4.26}$$

Soft Actor-Critic (SAC)

The Soft Actor-Critic (SAC) [118] is an off-policy reinforcement learning algorithm designed for a continuous action space. It differs from DDPG and TD3 in its approach, although it incorporates some techniques from these algorithms, such as the clipped double Q-trick. The most significant aspect of the SAC algorithm is the entropy regularization feature. This algorithm maximizes the expected return and entropy during policy training, eliminating improper convergence.

The concept of entropy, which evaluates the randomness of a variable x using its density function P, can be defined as follows

$$H(P) = \mathbb{E}_P[-\log P(x)]. \tag{4.27}$$

Taking into account the effect of entropy, where the agent receives a bonus reward proportional to the entropy of the policy at the time step, the optimal policy can be defined as follows

$$\pi^* = \underset{\pi}{\text{argmax}} \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha H(\pi(a_t \mid s_t)))], \tag{4.28}$$

where $\alpha > 0$ is a hyperparameter called the trade-off coefficient.

In addition to the optimal policy, two Q-functions, Q_{θ_0} and Q_{θ_1}, are optimized by the SAC algorithm. It uses a clipped double Q-learning trick, and updates the target networks using Polyak averaging. The Mean Squared Bellman Error (MSBE) function for the SAC algorithm is described in Eq. 4.29.

$$L(\theta_i, D) = \mathbb{E}_D[(Q_{\theta_i}(s,a) - y(r, s', d))^2], \tag{4.29}$$

where the target term, denoted $y(r, s', d)$, is expressed as

$$y(r, s', d) = R_t + \gamma(1-d)(\min_{i\in 0,1} Q_{\theta_{i,\text{targ}}}(s', a') - \alpha \log \pi(a' \mid s')), \tag{4.30}$$

where π is the current learned policy and $a' \sim \pi$. To optimize the policy, the SAC algorithm uses a reparameterization trick to determine the action as follows

$$a' = f_\theta(\epsilon, s), \tag{4.31}$$

where f_θ is a deterministic function and ϵ is random noise with normal distribution.

CHAPTER 5

Versatile Intelligent Robotic Workstation in the Context of Industry 4.0

The following chapter presents the design of advanced methods in the field of industrial robotics, practically demonstrated on a unique robotic workstation in accordance with the Industry 4.0 concept. The presented workstation, known as the Versatile Intelligent Robotic Workstation (VInRoS), is a significant part of the so-called Industry 4.0 Cell (I4C), which was first introduced in 2021 [124]. It is an advanced robotic laboratory located at the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology. The laboratory contains a variety of robotic structures, as shown in Figure 5.1, which represents a set of the most common geometric representations used for experiments in robotic research. Namely, an industrial robot ABB IRB 120 with six degrees of freedom (DoF), the same robot extended by a linear axis providing seven DoF, a SCARA robot Epson LS3-B401S with four DoF, a dual-arm collaborative robot ABB IRB14000 with seven DoF on each arm, and finally, a collaborative robot Universal Robots UR3 with six DoF.

The construction design, inspired by the main pillars of the Industry 4.0 concept and realized for the VInRoS robotic workstation presented in this chapter, is driven by the theoretical foundations discussed in the previous chapters. Furthermore, the implementation of advanced methods across all robotic structures in the I4C laboratory serves the purpose of validating modularity and versatility. This strategic approach is a direct result of the insights gained from the theoretical background discussed earlier.

The chapter begins with the design of a unique robotic workstation (Section 5.1), which incorporates advanced vertical system integration (Subsection 5.2). Following this introduction, the chapter delves into the implementation of a sophisticated and user-friendly Human-Machine Interface (HMI) customized for the presented worksta-

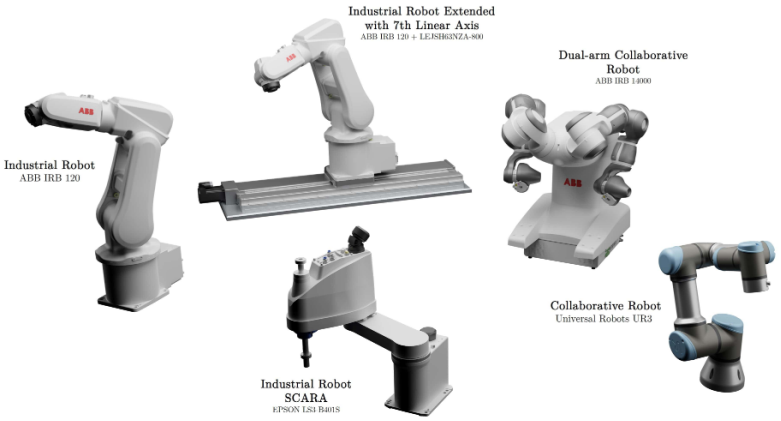


Figure 5.1: An illustration of the robotic structures that are part of the Industry 4.0 Cell (I4C).

tion (Section 5.2). Subsequently, a comprehensive approach to kinematics solutions is introduced, validated across a diverse range of robotic configurations (Section 5.3). Within the area of digital twins, the chapter describes the development of physics-based simulators utilizing PyBullet (Subsection 5.4.1) and Unity3D (Subsection 5.4.2). The concluding section of the chapter is dedicated to Deep Reinforcement Learning-Based planning algorithms designed for tasks focused on achieving randomly generated targets within the pre-defined configuration space (Section 5.5).

5.1 Construction Design of a Robotic Workstation

The design of the presented robotic workstation, known as the Versatile Intelligent Robotic Workstation (VInRoS), as shown in Figure 5.2, was inspired by the main pillars of the Industry 4.0 concept described in Chapter 2. The VInRoS is a significant part of the so-called Industry 4.0 Cell (I4C) [124], an advanced robotic laboratory located at the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology.

The name of the presented workstation was derived from its main purpose, which is to demonstrate versatility, including modularity, efficiency, and reproducibility. The key idea in constructing the workstation was its adaptability to a wide range of automation technologies in the field of Industry 4.0, as well as improving the quality of practical teaching at universities by introducing advanced technologies to courses related to robotics and automation.

The design of the robotic workstation involved a meticulous selection process for its main components, which included both hardware and software aspects. This crucial phase, which lasted approximately two years, included a thorough evaluation and consideration of various criteria to ensure optimal functionality and efficiency. During this period, key factors such as functionality, research potential, and education were taken into account,

Figure 5.2: The fundamental concept of the Versatile Intelligent Robotic Workstation (VInRoS), which illustrates its main components.

both in terms of the presented solution and with an eye toward future expansions.

The real-world realization of the robotic workstation, as depicted in Figure 5.3, comprises two articulated robotic arms: an ABB IRB 120 extended by a linear axis providing seven degrees of freedom (DoF) and a collaborative dual-arm robot, ABB IRB 14000, with seven DoF on each arm. The connection between these robotic arms is provided by two linear conveyors. In addition, the workstation includes other hardware components, such as a 3D scanner, an air compressor, and an electrical switchboard. The design includes additional parts made using both traditional manufacturing technologies and additive technologies.

Figure 5.3: The real-world realization of the Versatile Intelligent Robotic Workstation (VInRoS) located at the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology.

5.1.1 System Integration

The vertical system integration within the VInRoS workstation was designed to ensure seamless interoperability and optimal interconnection of various types of systems. This involved understanding the most commonly used Ethernet-based communication protocols described in Subsection 2.3. The aim was to highlight one of the main aspects of the Industry 4.0 concept, which does not differentiate between manufacturers, communication protocols, etc., in the design of a robotic system. Within the design of the system integration structure, emphasis was placed on important factors such as clarity, efficiency, modularity, and, finally, reproducibility.

The Industrial PC (IPC) by B&R Automation was chosen as the primary control system for the entire robotic workstation. A set of unique and modular packages was designed within this system to control each component of the workstation using a variety of Ethernet-based communication protocols. Specifically, the ABB robots were controlled through PROFINET, the B&R components through Ethernet POWERLINK, and the SMC wireless module through Ethernet/IP. External devices such as the Human-Machine Interface (HMI) or Unity3D digital twin were controlled through OPC UA.

The resulting structure of the advanced vertical system integration of the presented robotic workplace is illustrated in Figure 5.4. The objective was to highlight the potential of diverse integration, which involves multiple communication protocols that meet standardization, safety, and speed requirements for real-world operations.

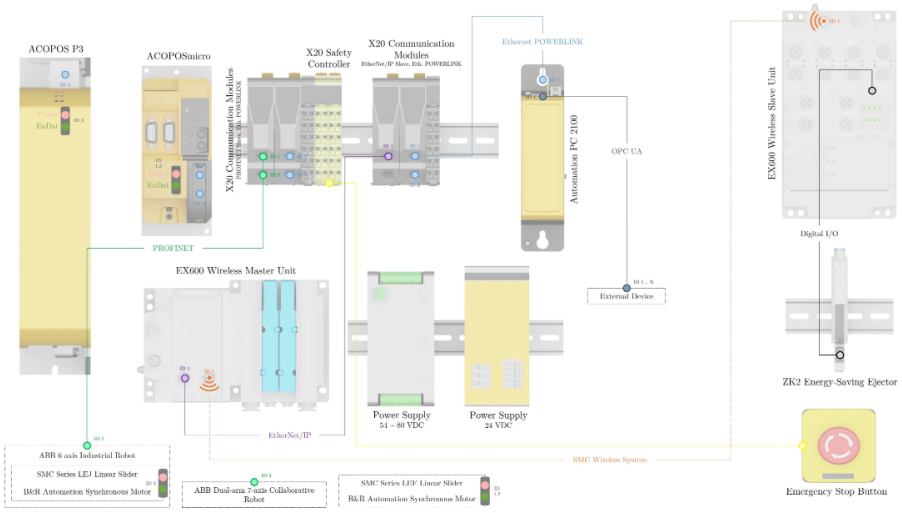


Figure 5.4: The structure of the advanced vertical system integration of the presented robotic workplace.

The only limitation of the presented design is the inability to simulate individual devices within the B&R Automation Studio environment. This absence restricts the ability to verify developed methods and troubleshoot. To address this limitation, a custom simulation tool was developed using Unity3D to integrate the entire workstation, as detailed in Subsection 5.4.2.

5.2 Human-Machine Interface

The Human-Machine Interface (HMI) is a user interface essential for the functionality and efficiency of a general robotic system in manufacturing processes. The HMI takes the form of multi-touch-enabled industrial control panels or devices that can display a web browser, such as smartphones and tablets. The user interface serves as the bridge between human operators and the complex machinery, facilitating seamless communication, control, and monitoring of different parts of the robotic system.

In the scope of our research, a sophisticated and modular user interface was developed for the presented Versatile Intelligent Robotic Workstation (VInRoS). The structure of the visualization was designed within B&R's development environment known as Automation Studio. Specifically, web-based technology intended to create user interfaces within the given software, referred to as mapp View, was utilized. The process of selecting the technology to create the user interface was straightforward due to the main control system of the robotic station, as described in Section 5.1.

The emphasis of the HMI design was placed on creating an efficient and intuitive user interface with a pleasant visual appearance. The resulting effect of the created user interface was focused on the simplicity of control, management of errors, and seamless system monitoring, eliminating the need for extensive operator training. Figures 5.5 and 5.6 provide an overview of the user interface designed using mapp View technology. The entire structure of the visualization interface was designed using unique widgets built on web standards, including HTML5, CSS3, and JavaScript. The data management of the user interface was based on the OPC UA architecture, allowing up to one hundred simultaneously connected devices, depending on the type of Industrial PC (IPC) or Programmable Logic Controller (PLC) used.

Figure 5.5: An overview of the home screen structure of the web-based Human-Machine Interface (HMI) created using B&R Automation technology, known as mapp View.

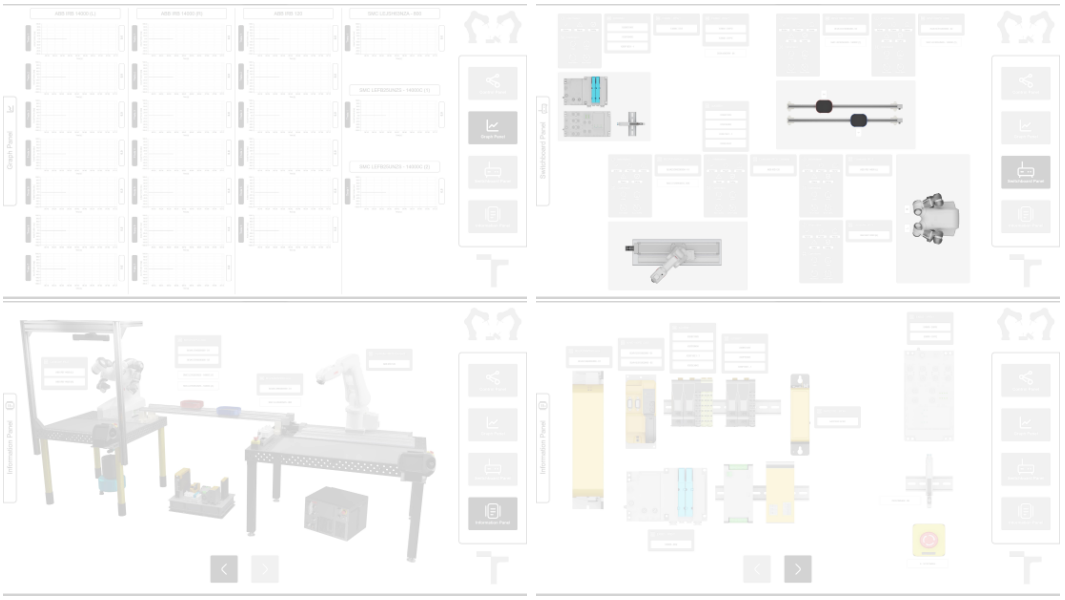


Figure 5.6: An overview of the remaining screens of the web-based Human-Machine Interface (HMI).

In conclusion, the user interface is a critical element of the robotic workstation design. It serves to enable efficient control, error handling, adaptability, and monitoring of the entire system. As technology advances, the development of user-friendly and sophisticated HMIs will continue to be a focal point in ensuring advanced system integration and intuitive control of robotic systems in various industries. For this reason, it is important to focus on both visual impact and functionality.

5.3 Comprehensive Approach to Kinematics Solutions

Kinematics is a fundamental research area in robotic manipulators that deals with the principles of motion, without considering forces and torques. The problem of kinematics is typically divided into two categories, as described in Chapter 2: forward kinematics and inverse kinematics. Forward kinematics establishes the relationship between joint variables and the homogeneous transformation matrix of the end-effector (Eq. 3.18), while inverse kinematics determines the joint configurations necessary to achieve a desired end-effector pose (Eq. 3.19).

In consideration of the research objectives, which emphasize the implementation of a comprehensive approach to kinematic solution across a variety of robotic structures, a numerical method was chosen. This method offers greater universality compared to analytical solutions, as it is not dependent on the specific geometry of robotic mechanisms. Moreover, this method allows for solving kinematics for a wide range of robotic structures, including those with kinematic redundancy, based only on the geometric parameters defined by the Denavit-Hartenberg convention. Based on the variety of robotic structures

selected for the experiment, as shown in Figure 5.1, and the extensive research described in Chapter 2, the method was unequivocally chosen.

A complete summary of the results, including a Denavit-Hartenberg table for the standard and modified forms of the parameters for each robotic structure, is described in the Appendix A.

5.3.1 Geometric Representation

The geometric representation of robotic mechanisms is defined by specific conventions known as the Denavit-Hartenberg conventions, both in the standard and modified forms. In both forms, determining the relative location of one coordinate frame to another requires only four parameters, represented as a product of four basic transformations about particular axes. The result is a so-called homogeneous transformation matrix that represents this product of transformations. The product of homogeneous transformation matrices during the Forward Kinematics calculation provides a compact notation, but is inefficient with regard to computational demands considering the frequent use of this method in robotic applications. As the number of joints increases, the calculation time also increases, leading to the computational complexity of the inverse kinematic solution, since it directly depends on forward kinematics.

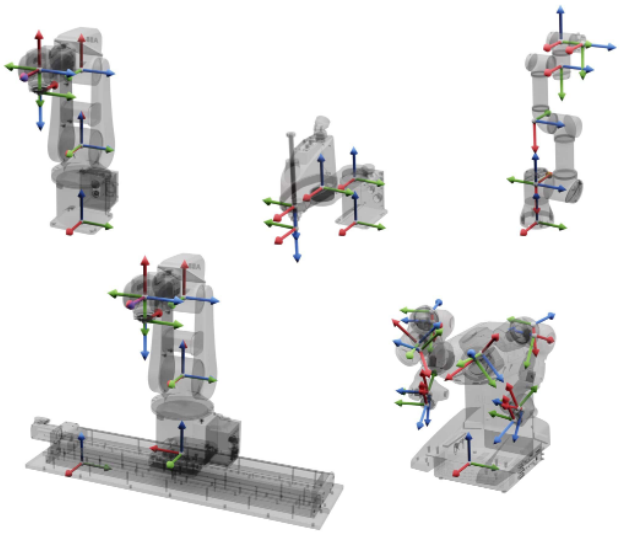


Figure 5.7: An illustration of the geometric representations of the robotic arms portfolio, calculated using the modified Denavit-Hartenberg method.

In the scope of the research, an effective computation method for forward kinematics was developed. The method involves multiplying transformation matrices that contain symbolic variables, such as joint angles and displacements, using forward kinematics to obtain the overall transformation matrix that describes the position and orientation of the end-effector with respect to the base frame. The focus was on simplifying the matrix,

extracting position and orientation, and eliminating redundant calculations. The results of these computations are presented in Table 5.1, providing a clear summary.

Table 5.1: The table provides the total result time in seconds obtained by the test, which consists of 100,000 random targets, for Forward Kinematics calculation using the specified method.

Robot Type	Standard Denavit–Hartenberg Parameters	Modified Denavit–Hartenberg Parameters	Simplified Denavit–Hartenberg Parameters
Universal Robots UR3	13.32	13.55	5.66
ABB IRB 120	13.88	14.39	5.69
ABB IRB 120 Ext.	16.34	16.80	5.98
ABB IRB 14000 (L)	17.33	17.78	7.59
ABB IRB 14000 (R)	17.32	17.75	7.57
Epson LS3-B401S	10.17	10.45	3.38

The table demonstrates that the time required to compute forward kinematics using the simplified method was significantly less than that of the other methods. To validate the effectiveness and correctness of the approach for forward kinematics calculation, the method was applied to a variety of robotic structures within the Industry 4.0 Cell (I4C), as illustrated in Figure 5.7.

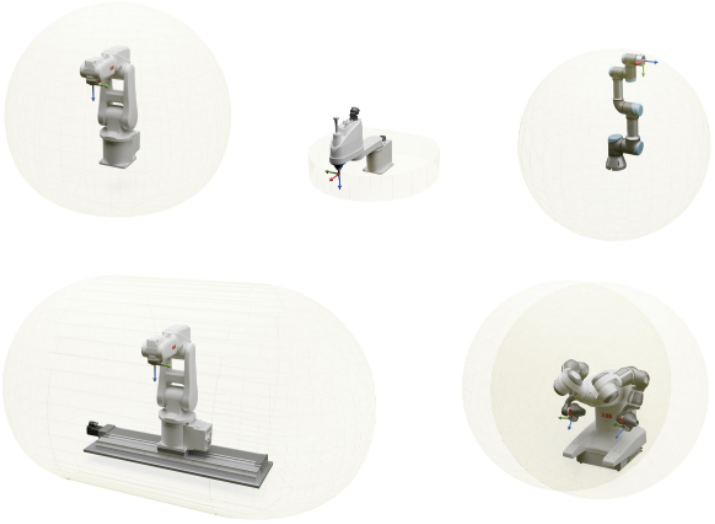


Figure 5.8: An illustration of a simplified version of workspace for the presented robotic arms, determined using the Monte Carlo method.

Given that the workspace of a general-purpose robotic manipulator is defined by the geometry of the manipulator and the movement limits of each joint, a reachable version of the workspace can be determined based on the presented forward kinematics method. This was achieved using the Monte Carlo method, where a set of joint positions was randomly generated, and with the help of forward kinematics, the positions were found

in a three-dimensional Euclidean space. The resulting structure of the workspace for each robotic manipulator is illustrated in Figure 5.8.

5.3.2 Collision Structure

One of the challenges in the field of robotic research is ensuring the avoidance of collisions between the robot and the obstacles in the workspace, as well as self-collisions among the individual parts of the robotic structure. Collisions within the robot's workspace dramatically constrain the working range and affect the continuous motion of each arm.

The design of a collision structure for a wide range of robotic manipulators, considering universality, was a critical aspect of this challenge. An effective approach was to approximate the individual parts of the robotic structure using bounding boxes. The static objects forming the base of the robot were approximated with Axis-Aligned Bounding Boxes (AABBs), and dynamic objects such as joints with Oriented Bounding Boxes (OBBs). The individual components of the structure were automatically approximated and aligned using homogeneous transformation matrices, ensuring correspondence with their respective origins. As a result, the transformation matrix derived for the collision object mirrors that of the real object. Figure 5.9 illustrates the resulting collision structure for each robotic manipulator.

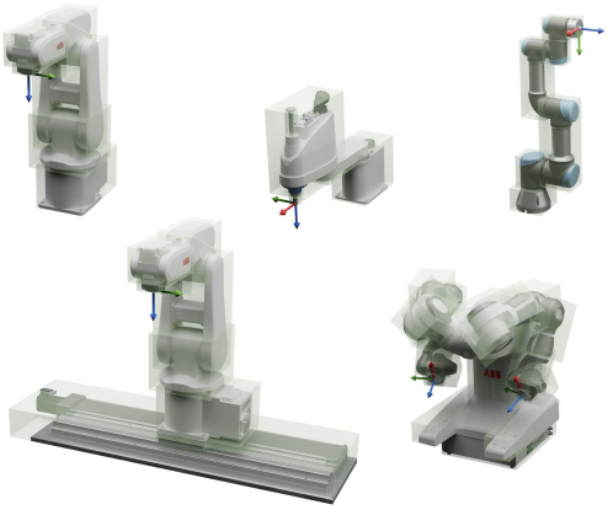


Figure 5.9: An illustration of the collision structures of the robotic arm portfolio, represented using Oriented Bounding Boxes (OBBs).

In the context of robotic structures, the conventional collision detection method faced a notable challenge in terms of computational efficiency. The real-time nature of the method requires optimal efficiency in its calculations. Specifically, the method must check all possible collision pairs to determine the presence of any collisions. To address this challenge, a strategy was implemented to enhance computational efficiency by reducing

the number of collision pairs. This optimization was achieved through focused efforts on a set of randomly generated targets designed to check for potential collision pairs. The results of these optimizations are presented in Table 5.2.

Table 5.2: The table provides the results of reducing the number of collision pairs through optimization performed on 100,000 random targets.

Robot Type	Number of Base Colliders $\mathcal{O}_{\text{base}}$	Number of Joint Colliders $\mathcal{O}_{\theta}$	Initial Number of Collision Pairs	Optimized Number of Collision Pairs
Universal Robots UR3	1	6	21	8
ABB IRB 120	1	6	21	7
ABB IRB 120 Ext.	2	7	36	14
ABB IRB 14000 (L)	6	7	78	26
ABB IRB 14000 (R)	6	7	78	26
Epson LS3-B401S	1	4	10	2

5.3.3 Numerical Solution of Inverse Kinematics

Inverse kinematics is a fundamental research area in robotic manipulators. The main objective of the problem is to find an initial configuration of the robotic structure that satisfies specified constraints regarding the position and orientation of an end-effector.

As mentioned earlier, considering the research objectives that emphasize the implementation of a comprehensive approach to kinematic solutions across a variety of robotic structures, a numerical method was chosen. More precisely, the Levenberg-Marquardt method, also known as the Damped Least Squares (DLS) method, as described in Section 3.5. This method, highlighted in the literature [74], has shown better outcomes in obtaining kinematic solutions compared to other numerical methods. However, it faces challenges related to the reachability of more distant points, independence from the type of robotic structure, and the potential occurrence of self-collision.

The research aimed to improve the LM method by incorporating a self-collision function and improving the reachability of distant points. These modifications were implemented with a focus on enhancing the universality of robotic structures. The desired change, denoted as $\mathbf{v}_e$, from Eq. 3.30, was determined as described in Eq. 5.1. This expression represents the translation and rotation from the end-effector's current pose to the desired pose.

$$\mathbf{v}_{e,i}(\theta) = \begin{bmatrix} \mathbf{p}_d - \mathbf{p}_i \\ \alpha(\mathbf{R}_d \mathbf{R}_i{}^T) \end{bmatrix}, \tag{5.1}$$

where $\mathbf{p}_i$ and $\mathbf{p}_d$ are the current and desired position of the end-effector, and $\mathbf{R}_i$ and $\mathbf{R}_d$ are the current and desired orientation of the end-effector. Additionally, $\alpha(\mathbf{R})$ represents the Euler vector in angle-axis form, specifying the axis of rotation and the angle of rotation about that axis as described in [74]. Subsequently, the quadratic error, which describes the difference between the actual end-effector position and the desired end-effector position, is expressed as follows

$$E = \frac{1}{2}\mathbf{v}_e{}^T\mathbf{W}_e\mathbf{v}_e, \tag{5.2}$$

where $\mathbf{W}_e$ is a diagonal weighted matrix.

The primary optimization criterion for minimizing the inverse kinematics function was the quadratic error of Eq. 5.2. To demonstrate the universality and accuracy of the proposed method across a diverse range of robotic structures, a comprehensive test was conducted. The test involved generating one hundred passing points between the initial and final positions of the end effector. The initial and final poses of the end-effector were carefully selected to ensure variations in both position and orientation across each axis. The sequence of points was generated systematically using the trapezoidal motion profile. The purpose of this experimental setup was to validate the effectiveness and reliability of the method in handling a wide range of robotic configurations. The results of the complex test are presented in Table 5.3.

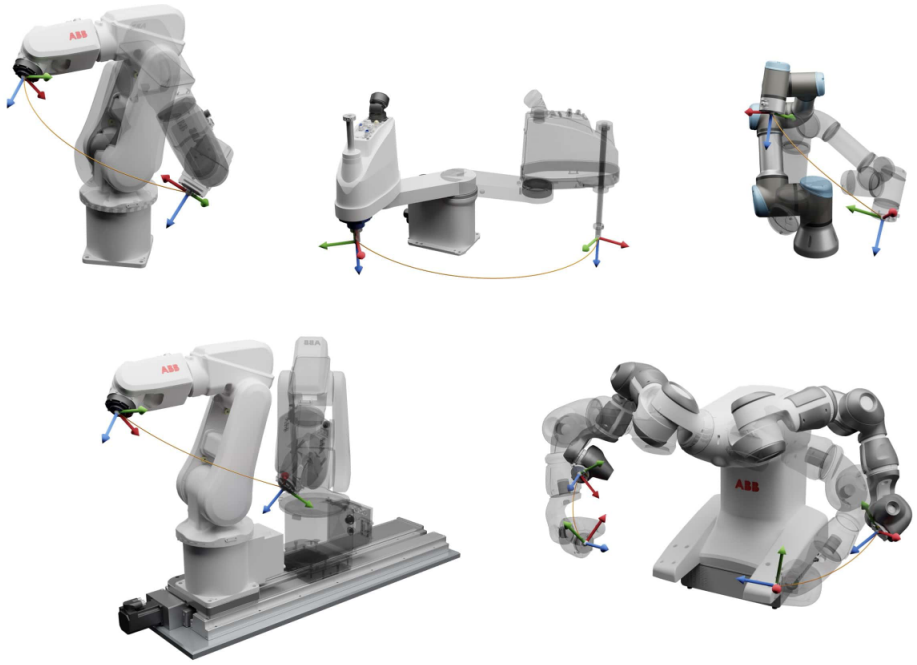


Figure 5.10: An illustration of the kinematic solution trajectory obtained through the informed Levenberg-Marquardt method for an individual robotic structure.

Table 5.3: The table presents the results of the kinematic solution obtained through the informed Levenberg-Marquardt method, based on one hundred generated points. The tolerance E was set to 1e-30. The position error is given in metres.

Robot Type	Mean Position Error $\bar{e}_p$	Mean Orientation Error $\bar{e}_Q$	Mean Quadratic Error $\bar{E}$	Mean Number of Iterations $\bar{N}$
Universal Robots UR3	4.012e-16	1.060e-14	2.608e-31	7.633
ABB IRB 120	1.227e-16	1.230e-14	4.602e-32	6.861
ABB IRB 120 Ext.	2.460e-16	1.601e-14	1.168e-31	6.514
ABB IRB 14000 (L)	2.466e-16	2.077e-05	1.281e-31	6.683
ABB IRB 14000 (R)	1.879e-16	1.538e-05	8.873e-32	6.792
Epson LS3-B401S	4.386e-16	2.880e-14	1.673e-31	10.702

The table provides a comprehensive overview of the performance of the informed LM method, which incorporates crucial features such as the structure of the self-collision, the singularity control, and the ability to reach more distant points. The results clearly demonstrate the high efficiency of the method across a diverse range of robotic structures. Figure 5.10 visually depicts the trajectory generated using a trapezoidal motion profile for a variety of robotic arms in our portfolio. This illustration serves to underscore the adaptability and effectiveness of the method in real-world scenarios.

5.4 Physics-Based Simulators for Industrial Robotics

An indispensable tool in the advancement of robotics research is a physics-based simulation that serves to validate the functionality of developed methods, improve efficiency, avoid collisions, and accelerate development. In short, simulation tools can be used in robotics research to replicate the behavior of the physical world in a virtual model, commonly referred to as the digital twin. Simulation tools can be used to optimize robotic behavior in a virtual environment before implementing it in the physical world.

The article [125] provides a comprehensive overview of the simulators available to researchers in various domains of robotics, particularly those related to robotic arms and single/multiple purpose machines. Notable simulation tools include PyBullet [126] and MuJoCo (Multi-Joint Dynamics with Contact) [127], which allow the creation of models for robot arms and environments to test robot behavior. The Robot Operating System (ROS) [128] is known for its openness and extensive developer base [**?**, 129]. Commercial software options, such as RoboDK and Visual Components, are also valuable for developing robotics applications. It should be noted that conventional simulation tools are available from industrial manufacturers, such as B&R Automation's Scene Viewer, ABB's RobotStudio, and Universal Robot's PolyScope. However, these tools have the disadvantage of focusing exclusively on a specific type of robot or machine, which is not ideal when designing a modular solution. Unity3D [130] has become a popular multi-platform development tool for robotic applications due to its integration with NVIDIA's physics engine, allowing for dynamic simulation of robots.

To accomplish the research objectives, the focus was exclusively on open-source, multi-purpose simulation tools. After evaluating various robotic simulators, PyBullet

and Unity3D were selected for their unique purposes in achieving our research objectives. PyBullet is an open-source simulation tool, unlike others that require a license, and Unity3D has a free student/personal license available, which significantly encourages reproducibility. PyBullet is an open source option highly regarded due to its maturity, the use of the Python programming language, and extensive research in robotics [125, 131, 132]. It simplifies the implementation of the OpenAI Gym interface [133], which is used to develop and test learning agents. However, it does not possess realistic rendering capabilities, which affects only the visual appearance and not the functionality. The Unity3D development platform provides realistic rendering and could be a promising option for future robotics research. However, it is still an unexplored area in this field.

In the scope of our research, the PyBullet simulator was selected as the main tool to validate our developed methods, with a specific focus on kinematics and reinforcement learning techniques. Furthermore, Unity3D was utilized as the digital twin for simulation and real-world applications, which differs from conventional simulation tools.

5.4.1 Bullet Real-Time Physics Simulation

PyBullet simulator [126] is an open-source tool for physics simulations, particularly in the context of robotics and machine perception, including support for artificial intelligence research. It is commonly used to test and validate algorithms before deploying them on physical robots. PyBullet simulator is built on top of the Bullet physics engine and has experimental support for NVIDIA PhysX. It uses the Universal Robot Description Format (URDF) as a standardized representation for robotic models, enabling interoperability and ease of integration across various simulation platforms.

URDF is an XML-based file that enables robotics developers to describe a robot using a universal format. This format can be imported and exported by different tools to visualize or simulate the robot. However, the process of creating the provided format is problematic, especially when it is generated automatically. Standard software such as SolidWorks can be used to convert the software assembly into an URDF file using a plugin. However, creating new configurations every time in computer-aided design (CAD) software is not time-efficient or suitable for reproducibility, as it requires a license. Some scientific publications use geometric representations of the robot to automatically generate URDF files [134], or vice versa [135]. However, these research papers lack universality and are not open-source.

One of the challenges faced in this study was the automatic generation of URDF files for various configurations of robotic arms. In response to this challenge, a universal and automatic URDF file generator capable of accommodating a wide range of robotic structures was designed. The underlying principle of this generator is rooted in the kinematic structure of the robot, specifically relying on the Denavit-Hartenberg (DH) convention. The generation process incorporates additional information, including joint types and the presence of an external axis.

To validate the effectiveness of our approach, the method developed was applied to the generation and subsequent control of various robotic structures within the Industry 4.0 Cell (I4C), as illustrated in Figure 5.11. This validation encompasses under-articulated,

articulated, and over-articulated structures, including those with more than six degrees of freedom, commonly referred to as kinematically redundant.

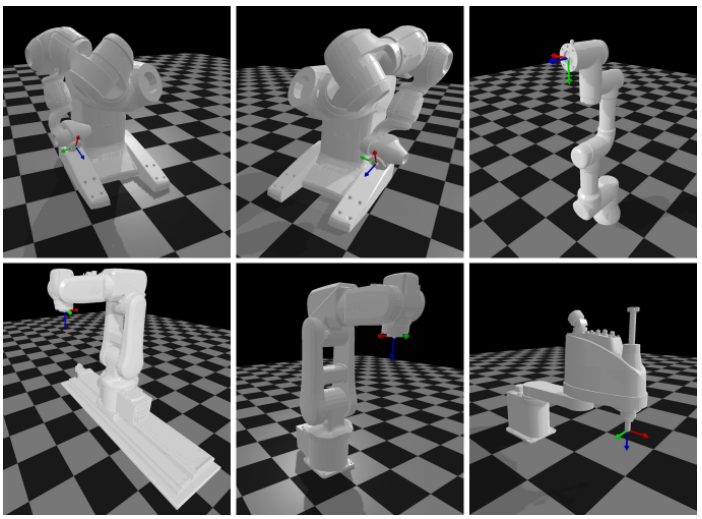


Figure 5.11: An illustration of the robotic structures that are part of the Industry 4.0 Cell (I4C) within the PyBullet physical simulation.

5.4.2 Unity Real-Time Development Platform

Unity3D software [130] is a multi-platform game development tool developed by Unity Technologies, providing the ability to develop two or three-dimensional applications for a wide range of devices, including smartphones and desktops. It also enables the import of objects and rigid-body structures from third-party applications, such as Blender. Unity includes an editor, physics engine, and supports rendering through libraries like DirectX, OpenGL, and WebGL. The primary programming languages used for program creation, commonly referred to as scripts, are C# and JavaScript. Unity3D utilizes the NVIDIA PhysX physics engine to simulate objects based on their physical properties. In recent times, Unity3D has expanded its use beyond the gaming industry, particularly in industrial robotics and automation, primarily for the development of augmented [136] and virtual reality [137, 138] applications.

As a part of our research, a custom simulation tool based on Unity3D was developed to integrate the entire Versatile Intelligent Robotic Workstation (VInRoS), which consists of hardware components from different manufacturers. This is because conventional tools focus only on specific types of robots or machines, which is not ideal for designing a modular solution. Although Unity3D is increasingly being used in industrial robotics and automation, it is still an unexplored area in this field. This presents an opportunity to introduce a new approach to robotic simulation.

As the primary objective of our research in the given area, a modular digital twin application was developed for the presented robotic workstation, as illustrated in Figure

5.12. The application's data management is entirely based on the OPC UA architecture. In this architecture, the Unity3D application functions as the client, while B&R's Automation Studio, specifically the Industrial PC (IPC) or Programmable Logic Controller (PLC), functions as the server. Although the application can be installed on smartphones and tablets, it is primarily designed for use on a standard computer to validate the developed methods within B&R's Automation Studio before implementing them in the physical world. The application uses performance optimization through multi-threaded programming. The High Definition Render Pipeline (HDRP) technology has been employed for realistic rendering. The emphasis of the application design was also placed on creating an intuitive user interface with a pleasant visual appearance.

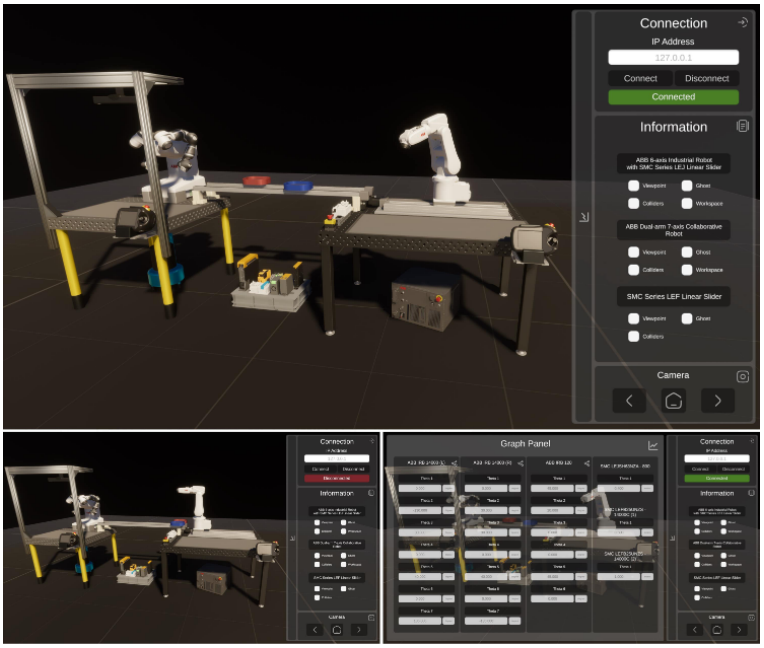


Figure 5.12: An illustration of the Versatile Intelligent Robotic Workstation (VInRoS) within the Unity3D development platform.

To demonstrate the modularity and reproducibility of the principles designed to create robotic simulations, various Unity3D applications in the field of industrial robotics were developed. These applications include single/multiple purpose robotic systems, as well as an augmented reality (AR) application. Most of them are shown in Figure 5.13.

In addition to the already mentioned OPC UA protocol, other protocols within Unity3D were implemented, such as TCP/IP (Transmission Control Protocol/Internet Protocol) and UDP (User Datagram Protocol) in the Unity3D framework. This enables the simulation and control of robotic arms sourced from prominent companies such as Universal Robots and ABB. In the context of our simulation tool, several key features were incorporated into specific applications to improve the overall user experience. These include collision detection mechanisms, accurate representation of physical objects, and virtual camera control.

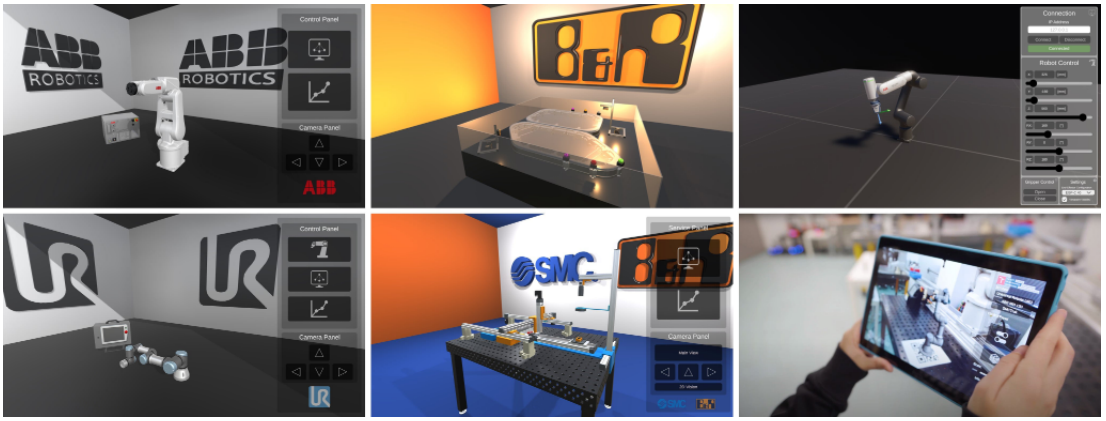


Figure 5.13: An illustration of the wide range of Unity3D applications in the context of industrial robotics, including single/multiple purpose robotic systems, as well as an augmented reality (AR) application.

5.5 Deep Reinforcement Learning-Based Motion Planning

The combination of deep neural networks (DNN) and reinforcement learning (RL), known as deep reinforcement learning (DRL), represents a relatively unexplored area of research in robotics and motion planning. The complex approach of learning sequential actions directly from raw input observations, especially in the problem of reaching a static [12] or random target [13, 14, 15] within the robot configuration space, shows potential in this area. Among the most commonly used algorithms for the presented problem, as described in Chapter 3, are the actor-critic model-free algorithms such as Deep Deterministic Policy Gradient (DDPG), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC), often used with experience replay.

In consideration of the research objectives, which prioritize the implementation of a modular approach to DRL-based motion planning solutions across a variety of robotic structures, each algorithm was compared in a specific experiment. The methods highlighted in the above-mentioned literature face challenges related to accuracy, independence from the type of robotic structure, potential occurrence of self-collision, or collision avoidance in general. The research aimed to address these challenges, eliminate shortcomings, and achieve greater versatility.

A complete summary of the results, including the hyperparameter structure used for experiments in both types of environments, is described in Appendix B.

5.5.1 Comparison of Actor-Critic Algorithms

The comparison of deep reinforcement learning (DRL) algorithms was focused mainly on the specific robotic arm Universal Robots UR3, considering that it was part of previous research [12]. The solved problem was divided into two parts (see Fig. 5.14), with both parts focusing on reaching the target in a pre-defined configuration space. The

first part, defined by the environment $\mathcal{E}_1$, was focused on reaching the target within the configuration space without any external collision. The second part, defined by the environment $\mathcal{E}_2$, focused on the same problem, but with the presence of a statically positioned external collision object.

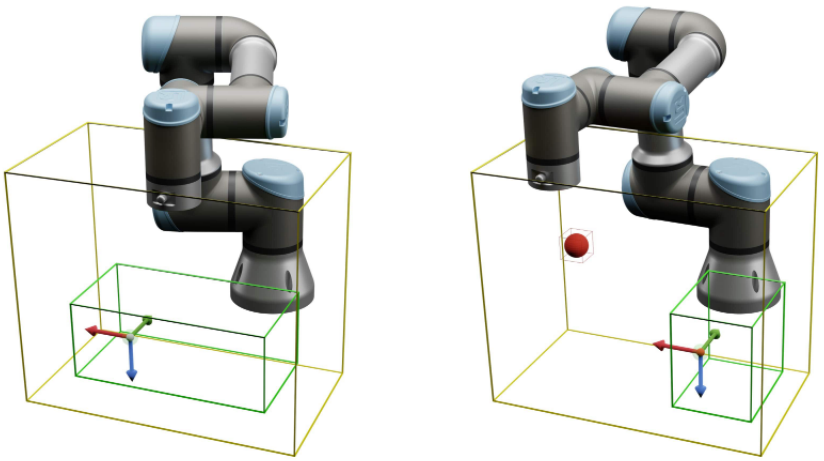


Figure 5.14: An illustration of both types of environments, $\mathcal{E}_1$ (left) and $\mathcal{E}_2$ (right), that were used in an experiment focused on reaching the target in a pre-defined configuration space $\mathcal{C}_{\text{free}}$. The yellow wireframe determines a pre-defined configuration space $\mathcal{C}_{\text{free}}$, while the green wireframe delineates the area where the target was randomly generated. The red sphere within the environment $\mathcal{E}_2$ represents an obstacle approximated with AABB, denoted as $\mathcal{O}_{AABB}$.

In both cases, the task was to move the robotic arm from the initial position to the desired final position within the pre-defined configuration space without collisions, including self-collisions. The movement was executed by generating the trajectory using linear segments with parabolic blends (LSPB). The inverse kinematics approach described in the previous section was used to define the motion, and PyBullet was used to simulate the learning process along with an additional package known as OpenAI Gym.

The observation space consists of the position of the end-effector in the current iteration and the desired end-effector position. The desired position, known as the target, was randomly generated in each episode to enhance the learning model. The action space consists of the end-effector movement command in three-dimensional Euclidean space. The reward function, as a critical part of the learning process, was defined for the $\mathcal{E}_1$ environment as follows

$$R_i = -(\|\mathbf{p}_d - \mathbf{p}_i\|), \tag{5.3}$$

and for the $\mathcal{E}_2$ environment as

$$R_i = -(\|\mathbf{p}_d - \mathbf{p}_i\| + \frac{\gamma_{\mathcal{O}}}{1 + \|\mathbf{p}_{\mathcal{O}} - \mathbf{p}_i\|}). \tag{5.4}$$

In Eq. 5.3, the $\mathcal{E}_1$ environment was determined by the simple Euclidean distance between the current position of the end-effector, denoted as $\mathbf{p}_i$, and the desired position of the end-effector, denoted as $\mathbf{p}_d$. Eq. 5.4 for the $\mathcal{E}_2$ environment incorporates a penalty term related to a collision object. This penalty is defined by the threshold $\gamma_{\mathcal{O}}$, which affects the significance of the penalty.

The learning process is interrupted when the robot encounters an external object or self-collision. Conditions for process interruption include the singularity check, along with evaluating the feasibility of the inverse kinematics solution. Successful completion of the learning episode is achieved when the current end-effector position of the robotic arm aligns closely with the randomly generated target, defined by the desired end-effector position.

The training process using the DDPG, SAC and TD3 algorithms, both with and without Hindsight Experience Replay (HER), is illustrated in Figure 5.15 for environment $\mathcal{E}_1$ and Figure 5.16 for environment $\mathcal{E}_2$. The comprehensive results of the evaluation process are described in Table 5.4 for environment $\mathcal{E}_1$ and Table 5.5 for environment $\mathcal{E}_2$. The results underscore the importance of selecting appropriate algorithms based on the characteristics of the environment. Specifically, the TD3 algorithm proved to be effective in environments without external collisions, while the DDPG algorithm was well-suited for scenarios featuring statically positioned external collision objects. This strategic approach improved the adaptability of the chosen methods for a diverse range of robotic arms in future applications.

Table 5.4: The table presents experimental results comparing various DRL algorithms within the $\mathcal{E}_2$ environment. The required minimum success rate to meet the specified criteria was set at 0.98.

Algorithm	Success Rate	Percentage of Successful Targets	Mean Reward per Episode	Mean Episode Length
DDPG	0.98 − 1.0	95.86%	-0.388	5.299
DDPG + HER	0.98 − 1.0	89.83%	-0.387	5.431
SAC	0.98 − 1.0	96.52%	-0.408	5.682
SAC + HER	0.98 − 1.0	94.85%	-0.407	5.701
TD3	0.98 − 1.0	96.61%	-0.386	5.298
TD3 + HER	0.98 − 1.0	78.73%	-0.395	5.720

Table 5.5: The table presents experimental results comparing various DRL algorithms within the $\mathcal{E}_2$ environment. The required minimum success rate to meet the specified criteria was set at 0.8.

Algorithm	Success Rate	Percentage of Successful Targets	Mean Reward per Episode	Mean Episode Length
DDPG	0.8 − 0.97	95.01%	-0.710	5.866
DDPG + HER	0.8 − 0.96	86.72%	-0.711	5.944
SAC	0.8 − 0.85	5.048%	-0.709	5.684
SAC + HER	0.8 − 0.88	19.07%	-0.713	5.844
TD3	0.8 − 0.96	89.59%	-0.711	5.901
TD3 + HER	0.8 − 0.94	81.99%	-0.718	6.146

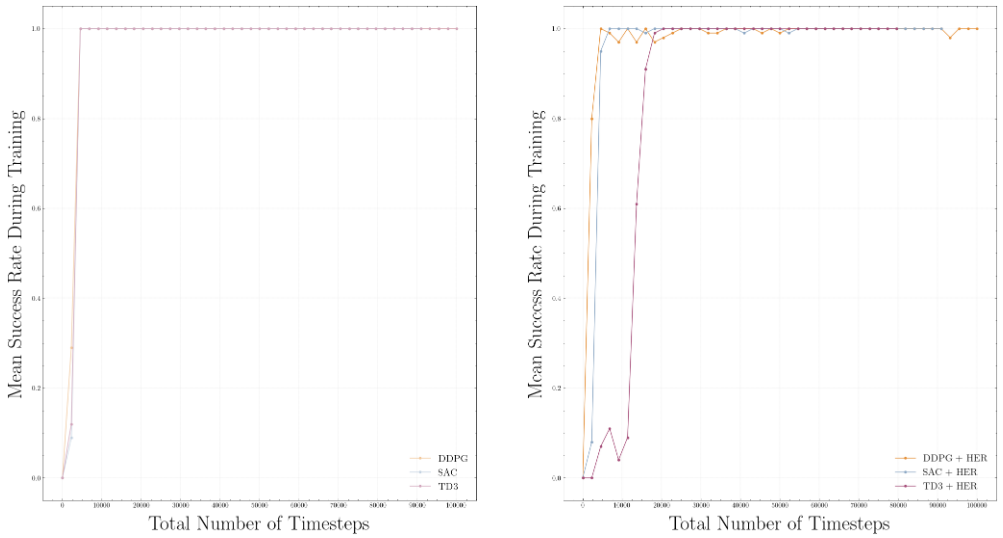


Figure 5.15: The training process shows success rates within the environment type $\mathcal{E}_1$ for the DDPG, SAC, and TD3 algorithms (right), and an extension with HER (left).

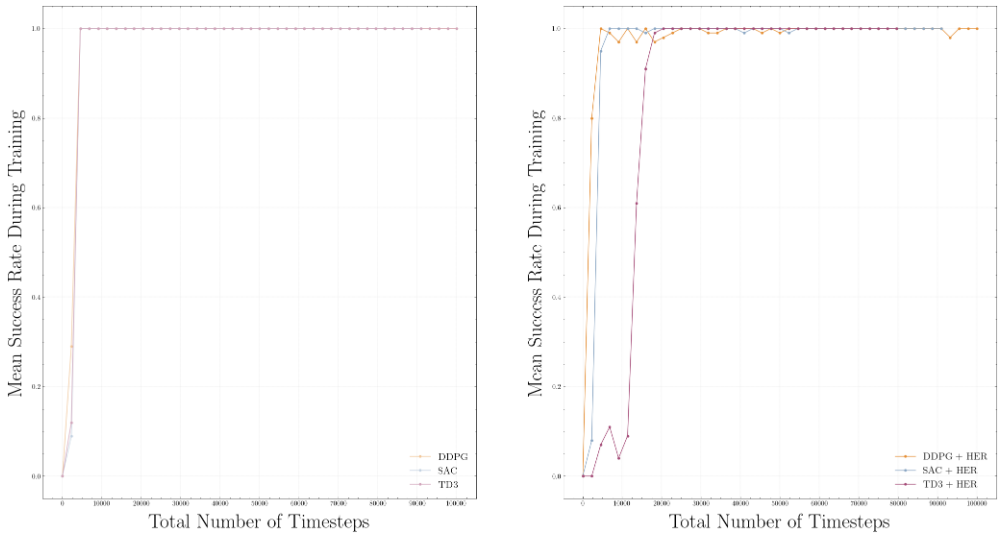


Figure 5.16: The training process shows success rates within the environment type $\mathcal{E}_2$ for the DDPG, SAC, and TD3 algorithms (right), and an extension with HER (right).

5.5.2 Application of Selected Algorithms to a Wide Range of Robotic Structures

The application of suitable algorithms for a specific case, obtained from the comprehensive experiments described in the previous section, was used for a wide range of robotic arms in both the $\mathcal{E}_1$ environment, as described in Figure 5.17, and the $\mathcal{E}_2$ environment, as

described in Figure 5.18. Specifically, the TD3 algorithm was employed in environments without external collisions, while the DDPG algorithm was utilized for scenarios featuring statically positioned external collision objects.

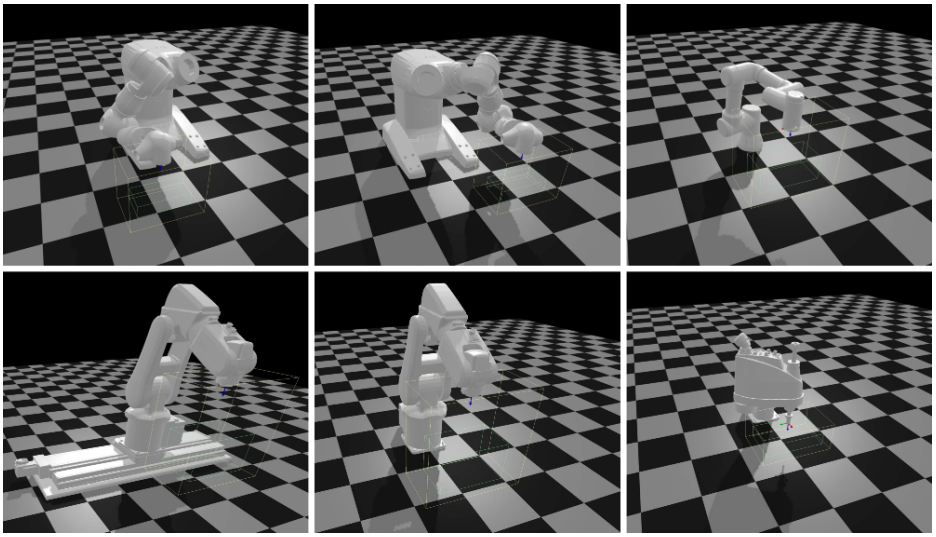


Figure 5.17: An illustration of a wide range of robotic structures within the $\mathcal{E}_1$ environment. These structures were used in an experiment that focused on reaching the target in a pre-defined configuration space.

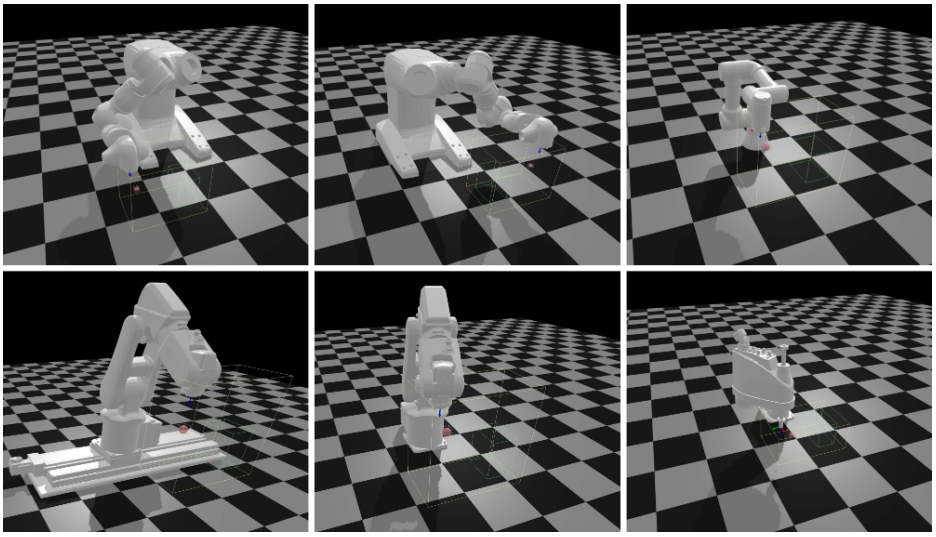


Figure 5.18: An illustration of a wide range of robotic structures within the $\mathcal{E}_2$ environment. These structures were used in an experiment that focused on reaching the target in a pre-defined configuration space.

The results of the training process, that was evaluated for specific robot arms, namely, ABB IRB 120 with and without an extended linear axis, Epson LS3-B401S, dual-arm collaborative robot ABB IRB14000, and Universal Robots UR3, are presented in Table 5.6 for environment $\mathcal{E}_1$ and Table 5.7 for environment $\mathcal{E}_2$.

Table 5.6: The table presents experimental results comparing a wide range of robotic structures using the TD3 algorithm within the $\mathcal{E}_2$ environment. The minimum success rate required to meet the specifications was set at 0.98.

Robot Type	Success Rate	Percentage of Successful Targets	Mean Reward per Episode	Mean Episode Length
Universal Robots UR3	0.98 − 1.0	96.61%	-0.386	5.298
ABB IRB 120	0.98 − 1.0	84.32%	-0.602	6.073
ABB IRB 120 Ext.	0.98 − 1.0	87.65%	-0.637	6.647
ABB IRB 14000 (L)	0.98 − 1.0	94.51%	-0.256	4.593
ABB IRB 14000 (R)	0.98 − 1.0	91.69%	-0.256	4.608
Epson LS3-B401S	0.98 − 1.0	94.39%	-0.064	2.474

Table 5.7: The table presents experimental results comparing a wide range of robotic structures using the DDPG algorithm within the $\mathcal{E}_2$ environment. The minimum success rate required to meet the specifications was set at 0.8.

Robot Type	Success Rate	Percentage of Successful Targets	Mean Reward per Episode	Mean Episode Length
Universal Robots UR3	0.8 − 0.97	95.01%	-0.710	5.866
ABB IRB 120	0.8 − 0.98	95.26%	-0.874	6.609
ABB IRB 120 Ext.	0.8 − 0.98	96.83%	-1.031	8.041
ABB IRB 14000 (L)	0.8 − 0.97	94.85%	-0.503	5.356
ABB IRB 14000 (R)	0.8 − 0.96	92.14%	-0.499	5.227
Epson LS3-B401S	0.8 − 0.99	98.19%	-0.396	5.040

To underscore the universality of the proposed method, an additional prediction test was conducted for a portfolio of robotic arms. The results are presented in Table 5.8 for environment $\mathcal{E}_1$ and Table 5.9 for environment $\mathcal{E}_2$. The results clearly demonstrate the high efficiency and universality of the algorithms in each environment across a diverse range of robotic structures.

Table 5.8: The table presents experimental results comparing a wide range of robotic structures using the TD3 algorithm within the $\mathcal{E}_1$ environment for one hundred randomly generated targets. The position error is given in metres.

Robot Type	Success Rate	Mean Reward per Episode	Mean Episode Length	Mean Absolute Position Error
Universal Robots UR3	1.0	-0.365	4.94	0.0024
ABB IRB 120	1.0	-0.501	5.26	0.0055
ABB IRB 120 Ext.	1.0	-0.502	5.16	0.0061
ABB IRB 14000 (L)	1.0	-0.233	4.21	0.0024
ABB IRB 14000 (R)	1.0	-0.228	4.13	0.0028
Epson LS3-B401S	1.0	-0.059	2.32	0.0030

Table 5.9: The table presents experimental results comparing a wide range of robotic structures using the DDPG algorithm within the $\mathcal{E}_2$ environment for one hundred randomly generated targets. The position error is given in metres.

Robot Type	**Success Rate**	**Mean Reward per Episode**	**Mean Episode Length**	**Mean Absolute Position Error**
Universal Robots UR3	1.0	-0.711	6.04	0.0058
ABB IRB 120	1.0	-0.859	6.45	0.0023
ABB IRB 120 Ext.	1.0	-0.945	6.98	0.0065
ABB IRB 14000 (L)	1.0	-0.471	5.15	0.0040
ABB IRB 14000 (R)	1.0	-0.470	5.12	0.0032
Epson LS3-B401S	1.0	-0.374	4.79	0.0027

CHAPTER 6

Conclusion

"If you want to improve something, you must first understand it. The combination of theoretical and practical knowledge is not an option, it is a must."

— Roman Parak

This dissertation presents the design and development of advanced methods in the field of industrial robotics. These methods were verified and practically demonstrated on a unique robotic workstation developed based on the fundamental idea of the Industry 4.0 concept. The proposed construction design of the workstation was grounded in the theoretical foundations of the dissertation, which take into account the main pillars of the Industry 4.0 concept. It demonstrates versatility, including modularity, efficiency, and reproducibility, and incorporates vertical system integration to ensure seamless interoperability and optimal interconnection of various types of systems across the workstation. This robotic workstation, also known as the Versatile Intelligent Robotic Workstation (VInRoS), is located at the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, and is a significant part of the so-called Industry 4.0 Cell (I4C), which was first introduced in 2021.

As the laboratory contains a variety of robotic structures, represented by a set of the most common geometric representations used for experiments in robotic research, the dissertation proposes modular methods applicable to a wide range of robotic manipulators, demonstrating the universality of the proposed solution. The validation of these methods was conducted on various robotic platforms, including an industrial robot ABB IRB 120 with six degrees of freedom (DoF), the same robot extended by a linear axis providing seven DoF, a SCARA robot Epson LS3-B401S with four DoF, a dual-arm collaborative robot ABB IRB 14000 with seven DoF on each arm, and finally, a collaborative robot Universal Robots UR3 with six DoF. This strategic approach is directly aligned with the insights gained from the theoretical background.

This dissertation also introduces a sophisticated and modular user interface for a robotic workstation, designed using B&R's development environment and the web-based technology mapp View. The data management of the user interface was built upon

the OPC UA architecture, leading to an efficient system for control, error handling, adaptability, and monitoring.

As one of the crucial components of the dissertation, which emphasizes the implementation of a comprehensive approach to kinematic solutions across a variety of robotic structures, an informed numerical solution based on the Levenberg-Marquardt method was developed. Compared to the standard methods, it incorporates crucial features such as the avoidance of self-collision, singularity control, and the ability to reach more distant points with reliable accuracy. The designed method presents an effective computational approach for forward kinematics, solving the simplification of Denavit-Hartenberg (DH) notation and also the universal creation of robotic collision structures approximated with a combination of Axis-Aligned Bounding Boxes (AABBs) and Oriented Bounding Boxes (OBBs). To demonstrate the computational efficiency of the self-collision method, a set of randomly generated targets was optimized to check for potential collision pairs. The results, demonstrated on the trajectory generated using a trapezoidal motion profile, clearly highlight the high efficiency of the proposed method across a diverse range of robotic structures. A complete summary of the results, including a Denavit-Hartenberg table for the standard and modified forms of the parameters for each robotic structure, is described in Appendix A.

The dissertation further presents a reliable, multi-purpose, physics-based simulation tool selection, which was crucial for the robotic research discussed in this thesis. First, it presents a custom library with a variety of algorithms for the PyBullet simulator. This library serves as the primary tool for validating the developed methods, including kinematics and motion planning. In addition, it involves the automatic generation of the Universal Robot Description Format (URDF) to provide a standardized representation of robotic structures capable of accommodating a wide range of robotic configurations. The design of this generator is based on the underlying principle rooted in the kinematic structure of the robot, specifically relying on the Denavit-Hartenberg (DH) convention. Subsequently, a modular simulation tool based on Unity3D was developed to integrate the entire robotic workstation, demonstrating modularity and reproducibility. Various Unity3D applications in the field of industrial robotics were developed to demonstrate the proposed design, including single/multiple-purpose robotic systems and an augmented reality (AR) application. These applications implement protocols such as OPC UA, TCP/IP, and UDP, along with key features such as collision detection mechanisms, accurate representation of physical objects, and virtual camera control.

The final part of the practical realization of the dissertation presents the development of a method for planning motion across a diverse range of robotic structures based on innovative algorithms. The combination of deep neural networks (DNN) and reinforcement learning (RL), known as deep reinforcement learning (DRL), was represented as a relatively unexplored area of research in robotics and motion planning. Especially in the problem of reaching a static or random target within the robot configuration space. The methods highlighted in the above-mentioned literature faced challenges related to accuracy, independence from the type of robotic structure, potential occurrences of self-collision, or collision avoidance, in general. The proposed method of the dissertation

effectively addresses the aforementioned shortcomings and showcases the high efficiency and universality of the algorithms across a diverse range of robotic structures. To substantiate this claim, a comprehensive comparison of actor-critic model-free algorithms was conducted, including Deep Deterministic Policy Gradient (DDPG), Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC), both with and without Hindsight Experience Replay (HER). The problem was divided into two parts, with both parts focusing on reaching the target in a pre-defined configuration space. The first part, defined by the environment E1, was focused on reaching the target within the configuration space without any external collision. The second part, defined by the environment E2, focused on the same problem, but with the presence of a statically positioned external collision object. The movement was executed by generating the trajectory using linear segments with parabolic blends (LSPB). The results underscored the importance of selecting appropriate algorithms based on the characteristics of the environment. Specifically, the TD3 algorithm proved to be effective in environments without external collisions, while the DDPG algorithm was well-suited for scenarios featuring statically positioned external collision objects. This strategic approach improved the adaptability of the methods chosen for a diverse range of robotic arms, as applied to the portfolio of presented robotic arms. A complete summary of the results, including the hyperparameter structure used for experiments in both types of environments, is described in Appendix B.

The design of the robotic workstation was subjected to a meticulous selection process for its primary components, encompassing both the hardware and the software aspects. This critical phase involved a comprehensive evaluation of various criteria to ensure optimal functionality and efficiency. Key considerations, such as research potential and educational impact, were carefully weighed, not only for the presented solution, but also with an eye toward future expansions. The experiences and knowledge gained during the development of this dissertation were applied to diverse projects in the field of robotics and educational activities. These activities included creating a new course related to the field of robotics, presenting at various conferences, and establishing valuable collaborations with industrial partners (see Appendix C). A notable contribution to the successful realization of this research was a six-month research traineeship at Johannes Kepler University in Linz, specifically at the Institute of Robotics. In adherence to the principles of open-source development and to foster continued progress, all code pertinent to the implementation of this dissertation has been disclosed. Further details are described in Appendix E. Finally, the list of publications is described in Appendix D.

6.1 Outlook for Future Research

The presented robotic workstation, as well as the entire robotic cell, has been designed with future expansion in mind, allowing seamless integration of new software and hardware aspects. In the future, investigations into DRL-based motion planning could extend to solving more complex problems, such as bin-picking, multi-robotics dynamics motion planning, and intelligent grasping of diverse objects. To address these challenges, one

possible solution is to enhance DRL algorithms by incorporating a multi-agent structure, such as the Multi-Agent Deep Deterministic Policy Gradient (MADDPG). Another approach to solving this type of problem can be to use evolutionary algorithms. These methods have been applied to efficiently solve kinematics problems [139] and motion planning problems, as described in a comprehensive survey [115]. A research objective could be to enhance the Unity3D application by extending deep reinforcement learning agents. This improvement is particularly relevant for consolidating the physics simulation tools into a single, unified framework, streamlining the methodology outlined in this dissertation.

Bibliography

[1] B. Siciliano and O. Khatib, *ed. Springer Handbook of Robotics*. Berlin, Heidelberg: Springer-Verlag, 2016.

[2] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[4] OPC Foundation, "Opc unified architecture." `www.opcfoundation.org/`. Online. Accessed on 7 October 2023.

[5] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[6] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, J. Pineau, *et al.*, "An introduction to deep reinforcement learning," *Foundations and Trends® in Machine Learning*, vol. 11, no. 3-4, pp. 219–354, 2018.

[7] Y. Uygun, "The fourth industrial revolution-industry 4.0," *Available at SSRN 3909340*, 2021.

[8] G. Erboz, "How to define industry 4.0: Main pillars of industry 4.0," *Managerial trends in the development of enterprises in globalization era*, vol. 761, pp. 761–767, 2017.

[9] D. Palka and J. Ciukaj, "Prospects for development movement in the industry concept 4.0," *Multidisciplinary Aspects of Production Engineering*, vol. 2, pp. 315–326, 09 2019.

[10] S. Liu and P. Liu, "Benchmarking and optimization of robot motion planning with motion planning pipeline," *The International Journal of Advanced Manufacturing Technology*, pp. 1–13, 2022.

[11] M. P. Xanthidis, J. M. Esposito, I. Rekleitis, and J. M. O'Kane, "Analysis of motion planning by sampling in subspaces of progressively increasing dimension," *arXiv preprint arXiv:1802.00328*, 2018.

[12] R. Parák and R. Matoušek, "Comparison of multiple reinforcement learning and deep reinforcement learning methods for the task aimed at achieving the goal," *Mendel Journal series*, vol. 27, no. 1, pp. 1–8, 2021.

[13] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, *et al.*, "Multi-goal reinforcement learning: Challenging robotics environments and request for research," *arXiv preprint arXiv:1802.09464*, 2018.

[14] Q. Gallouédec, N. Cazin, E. Dellandréa, and L. Chen, "panda-gym: Open-source goal-conditioned environments for robotic learning," *arXiv preprint arXiv:2106.13687*, 2021.

[15] A. Rzayev and V. T. Aghaei, "Off-policy deep reinforcement learning algorithms for handling various robotic manipulator tasks," *arXiv preprint arXiv:2212.05572*, 2022.

[16] H. Kagermann, W. Wahlster, and J. Helbig, *Recommendations For Implementing The Strategic Initiative Industrie 4.0: Final Report of the Industrie 4.0 Working Group.* Ulrike Findeklee: Acatech – National Academy of Science and Engineering, 2013.

[17] H. Kagermann, W.-D. Lukas, and W. Wahlster, "Industrie 4.0: Mit dem internet der dinge auf dem weg zur 4. industriellen revolution," *VDI nachrichten*, vol. 13, 2011.

[18] A. Sharma and B. J. Singh, "Evolution of industrial revolutions: A review," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 11, pp. 66–73, 2020.

[19] P. N. Stearns, *The industrial revolution in world history.* Routledge, 2020.

[20] H. Mohajan, "The first industrial revolution: Creation of a new global human era," 2019.

[21] H. Mohajan, "The second industrial revolution has brought modern social and economic developments," 2019.

[22] J. Mokyr and R. H. Strotz, "The second industrial revolution, 1870-1914," *Storia dell'economia Mondiale*, vol. 21945, no. 1, 1998.

[23] K. Schwab, *The fourth industrial revolution.* Currency, 2017.

[24] M. Xu, J. M. David, S. H. Kim, *et al.*, "The fourth industrial revolution: Opportunities and challenges," *International journal of financial research*, vol. 9, no. 2, pp. 90–95, 2018.

[25] P. Thomas and D. Nicholas, "The fourth industrial revolution: Shaping new era," *Journal of International Affairs*, vol. 72, no. 1, pp. 17–22, 2018.

[26] J. Rymarczyk *et al.*, "Technologies, opportunities and challenges of the industrial revolution 4.0: theoretical considerations," *Entrepreneurial business and economics review*, vol. 8, no. 1, pp. 185–198, 2020.

[27] C. O. Klingenberg, M. A. V. Borges, and J. A. do Vale Antunes Jr, "Industry 4.0: What makes it a revolution? a historical framework to understand the phenomenon," *Technology in Society*, vol. 70, p. 102009, 2022.

[28] A. Rojko, "Industry 4.0 concept: Background and overview.," *International journal of interactive mobile technologies*, vol. 11, no. 5, 2017.

[29] S. I. Tay, T. Lee, N. Hamid, and A. N. A. Ahmad, "An overview of industry 4.0: Definition, components, and government initiatives," *Journal of Advanced Research in Dynamical and Control Systems*, vol. 10, no. 14, pp. 1379–1387, 2018.

[30] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch, "Industry 4.0: The future of productivity and growth in manufacturing industries," *Boston consulting group*, vol. 9, no. 1, pp. 54–89, 2015.

[31] F. Sherwani, M. M. Asad, and B. S. K. K. Ibrahim, "Collaborative robots and industrial revolution 4.0 (ir 4.0)," in *2020 International Conference on Emerging Trends in Smart Technologies (ICETST)*, pp. 1–5, IEEE, 2020.

[32] A. Oke and F. A. P. Fernandes, "Innovations in teaching and learning: Exploring the perceptions of the education sector on the 4th industrial revolution (4ir)," *Journal of Open Innovation: Technology, Market, and Complexity*, vol. 6, no. 2, p. 31, 2020.

[33] A. A. Shahroom and N. Hussin, "Industrial revolution 4.0 and education," *International Journal of Academic Research in Business and Social Sciences*, vol. 8, no. 9, pp. 314–319, 2018.

[34] M. Hermann, T. Pentek, and B. Otto, "Design principles for industrie 4.0 scenarios: A literature review," 01 2015.

[35] J. Ortiz, W. Marroquin, and L. Cifuentes, *Industry 4.0: Current Status and Future Trends*. 03 2020.

[36] R. Hall, S. Schumacher, and A. Bildstein, "Systematic analysis of industrie 4.0 design principles," *Procedia CIRP*, vol. 107, pp. 440–445, 2022. Leading manufacturing systems transformation – Proceedings of the 55th CIRP Conference on Manufacturing Systems 2022.

[37] S. Vaidya, P. Ambad, and S. Bhosle, "Industry 4.0 – a glimpse," *Procedia Manufacturing*, vol. 20, pp. 233–238, 2018. 2nd International Conference on Materials, Manufacturing and Design Engineering (iCMMD2017), 11-12 December 2017, MIT Aurangabad, Maharashtra, INDIA.

[38] Ethernet POWERLINK Standardization Group (EPSG), "System overview: Ethernet powerlink basics." `www.ethernet-powerlink.org/uploads/media/POWERLINKBasics_brochure_e.pdf`. Online. Accessed on 1 October 2023.

[39] P. Brooks, "Ethernet/ip-industrial protocol," in *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 01TH8597)*, vol. 2, pp. 505–514, IEEE, 2001.

[40] J. Feld, "Profinet-scalable factory communication for all applications," in *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings.*, pp. 33–38, IEEE, 2004.

[41] A. L. Dias, G. S. Sestito, A. C. Turcato, and D. Brandão, "Panorama, challenges and opportunities in profinet protocol research," in *2018 13th IEEE International Conference on Industry Applications (INDUSCON)*, pp. 186–193, IEEE, 2018.

[42] W. Mahnke, S.-H. Leitner, and M. Damm, *OPC unified architecture*. Springer Science & Business Media, 2009.

[43] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *Ifac-PapersOnline*, vol. 51, no. 11, pp. 1016–1022, 2018.

[44] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE access*, vol. 8, pp. 108952–108971, 2020.

[45] A. Parrott and L. Warshaw, "Industry 4.0 and the digital twin," *Deloitte Insights*, 2017.

[46] F. Pires, A. Cachada, J. Barbosa, A. P. Moreira, and P. Leitão, "Digital twin in industry 4.0: Technologies, applications and challenges," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1, pp. 721–726, IEEE, 2019.

[47] T. R. Wanasinghe, L. Wroblewski, B. K. Petersen, R. G. Gosine, L. A. James, O. De Silva, G. K. Mann, and P. J. Warrian, "Digital twin for the oil and gas industry: Overview, research trends, opportunities, and challenges," *IEEE access*, vol. 8, pp. 104175–104197, 2020.

[48] C. Weidemann, N. Mandischer, F. Kerkom, B. Corves, M. Hüsing, T. Kraus, and C. Garus, "Literature review on recent trends and perspectives of collaborative robotics in work 4.0," *Robotics*, vol. 12, p. 84, 06 2023.

[49] K. C. Haug, T. Kretschmer, and T. Strobel, "Cloud adaptiveness within industry sectors–measurement and observations," *Telecommunications policy*, vol. 40, no. 4, pp. 291–306, 2016.

[50] K. Witkowski, "Internet of things, big data, industry 4.0–innovative solutions in logistics and supply chains management," *Procedia engineering*, vol. 182, pp. 763–769, 2017.

[51] P. K. R. Maddikunta, Q.-V. Pham, B. Prabadevi, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[52] S. Huang, B. Wang, X. Li, P. Zheng, D. Mourtzis, and L. Wang, "Industry 5.0 and society 5.0—comparison, complementation and co-evolution," *Journal of manufacturing systems*, vol. 64, pp. 424–428, 2022.

[53] A. Akundi, D. Euresti, S. Luna, W. Ankobiah, A. Lopes, and I. Edinbarough, "State of industry 5.0—analysis and identification of current research trends," *Applied System Innovation*, vol. 5, no. 1, p. 27, 2022.

[54] K. A. Demir, G. Döven, and B. Sezen, "Industry 5.0 and human-robot co-working," *Procedia computer science*, vol. 158, pp. 688–695, 2019.

[55] S. Nahavandi, "Industry 5.0—a human-centric solution," *Sustainability*, vol. 11, no. 16, p. 4371, 2019.

[56] Z. Lv, "Digital twins in industry 5.0," *Research*, vol. 6, p. 0071, 2023.

[57] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*. Springer Publishing Company, Incorporated, 1st ed., 2008.

[58] Z. Kolíbal *et al.*, "Roboty a robotizované vỳrobní technologie," *Brno: VUTIUM*, 2016.

[59] S. LaValle, "Planning algorithms," *Cambridge University Press google schola*, vol. 2, pp. 3671–3678, 2006.

[60] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in Python*, vol. 146. Springer Nature, 2023.

[61] J. Diebel *et al.*, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," *Matrix*, vol. 58, no. 15-16, pp. 1–35, 2006.

[62] P. R. Evans, "Rotations and rotation matrices," *Acta Crystallographica Section D: Biological Crystallography*, vol. 57, no. 10, pp. 1355–1359, 2001.

[63] A. Lerios, "Rotations and quaternions," *Standford University,[online],[accessed Oct. 23, 2012], available from World Wide Web:< http://citeseerx. ist. psu. edu/viewdoc/summary*, 1995.

[64] S. W. Shepperd, "Quaternion from rotation matrix," *Journal of guidance and control*, vol. 1, no. 3, pp. 223–224, 1978.

[65] A. Renfrew, "Introduction to robotics: Mechanics and control," *International Journal of Electrical Engineering & Education*, vol. 41, no. 4, p. 388, 2004.

[66] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot modeling and control*. John Wiley & Sons, 2020.

[67] S. Starke, N. Hendrich, and J. Zhang, "Memetic evolution for generic full-body inverse kinematics in robotics and animation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 406–420, 2018.

[68] A. Peidró, Ó. Reinoso, A. Gil, J. M. Marín, and L. Payá, "An improved monte carlo method based on gaussian growth to calculate the workspace of robots," *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 197–207, 2017.

[69] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," 1955.

[70] W. Khalil and E. Dombre, *Modeling identification and control of robots*. CRC Press, 2002.

[71] E. Dombre and W. Khalil, *Robot manipulators: modeling, performance analysis and control*. John Wiley & Sons, 2013.

[72] S. Chiaverini, B. Siciliano, and O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on control systems technology*, vol. 2, no. 2, pp. 123–134, 1994.

[73] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose," *Pseudoinverse and Damped Least Squares methods*, p. 19, 2004.

[74] T. Sugihara, "Solvability-unconcerned inverse kinematics by the levenberg–marquardt method," *IEEE transactions on robotics*, vol. 27, no. 5, pp. 984–991, 2011.

[75] W. Suleiman, F. Kanehiro, and E. Yoshida, "Infeasibility-free inverse kinematics method," in *2015 IEEE/SICE International Symposium on System Integration (SII)*, pp. 307–312, IEEE, 2015.

[76] S. KuCuk and Z. Bingul, "The inverse kinematics solutions of industrial robot manipulators," in *Proceedings of the IEEE International Conference on Mechatronics, 2004. ICM'04.*, pp. 274–279, IEEE, 2004.

[77] R. Konietschke and G. Hirzinger, "Inverse kinematics with closed form solutions for highly redundant robotic systems," in *2009 IEEE International Conference on Robotics and Automation*, pp. 2945–2950, IEEE, 2009.

[78] P. Donelan, *Kinematic singularities of robot manipulators*. INTECH Open Access Publisher, 2010.

[79] B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of intelligent and robotic systems*, vol. 3, pp. 201–212, 1990.

[80] M. T. Mason, *Mechanics of robotic manipulation*. MIT press, 2001.

[81] A. Balestrino, G. De Maria, and L. Sciavicco, "Robust control of robotic manipulators," *IFAC Proceedings Volumes*, vol. 17, no. 2, pp. 2435–2440, 1984.

[82] W. A. Wolovich and H. Elliott, "A computational technique for inverse kinematics," in *The 23rd IEEE Conference on Decision and Control*, pp. 1359–1363, IEEE, 1984.

[83] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, vol. 17, no. 1-19, p. 16, 2004.

[84] M. Meredith and S. Maddock, "Real-time inverse kinematics: The return of the jacobian," 2004.

[85] J. E. Dennis Jr and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*. SIAM, 1996.

[86] K. Anderson and J. Angeles, "Kinematic inversion of robotic manipulators in the presence of redundancies," *The International Journal of Robotics Research*, vol. 8, no. 6, pp. 80–97, 1989.

[87] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 1, pp. 93–101, 1986.

[88] S. K. Chan and P. D. Lawrence, "General inverse kinematics with the error damped pseudoinverse," in *Proceedings. 1988 IEEE international conference on robotics and automation*, pp. 834–839, IEEE, 1988.

[89] T. Lozano-Perez, *Spatial planning: A configuration space approach*. Springer, 1990.

[90] T. Reichenbach and Z. Kovačić, "Derivation of kinematic parameters from a 3d robot model used for collision-free path planning," in *Proceedings of the 11th Mediterranean Conference on Control and Automation, MED*, vol. 3, 2003.

[91] V. Román-Ibáñez, F. A. Pujol-López, H. Mora-Mora, M. L. Pertegal-Felices, and A. Jimeno-Morenilla, "A low-cost immersive virtual reality system for teaching robotic manipulators programming," *Sustainability*, vol. 10, no. 4, p. 1102, 2018.

[92] J. S. Park, C. Park, and D. Manocha, "Efficient probabilistic collision detection for non-convex shapes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1944–1951, IEEE, 2017.

[93] C. Ericson, *Real-time collision detection*. Crc Press, 2004.

[94] G. Van Den Bergen, *Collision detection in interactive 3D environments*. CRC Press, 2003.

[95] S. Gottschalk, "Separating axis theorem," 1996.

[96] J. J. Craig, *Introduction to robotics*. Pearson Educacion, 2006.

[97] L. Biagiotti and C. Melchiorri, *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[98] L. Biagiotti, L. Moriello, and C. Melchiorri, "A repetitive control scheme for industrial robots based on b-spline trajectories," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5417–5422, IEEE, 2015.

[99] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.

[100] J. Peng, P. Huang, Y. Ding, and H. Ding, "An analytical method for decoupled local smoothing of linear paths in industrial robots," *Robotics and Computer-Integrated Manufacturing*, vol. 72, p. 102193, 2021.

[101] L. Piegl and W. Tiller, *The NURBS book*. Springer Science & Business Media, 2012.

[102] G. Dahlquist and Å. Björck, *Numerical methods in scientific computing, volume i*. SIAM, 2008.

[103] V. E. Kremer, "Quaternions and slerp," in *Embots. dfki. de/doc/seminar ca/Kremer Quaternions. pdf*, 2008.

[104] J. Vince and J. A. Vince, *Mathematics for computer graphics*, vol. 251. Springer, 2006.

[105] H. Henkel, "Calculating the cubic bezier arc length by elliptic integrals," *Oftersheim, Alemanha*, 2014.

[106] P. Bezier, "Mathematical and practical possibilities of unisurf," in *Computer aided geometric design*, pp. 127–152, Elsevier, 1974.

[107] H. B. Curry and I. J. Schoenberg, "On pólya frequency functions iv: the fundamental spline functions and their limits," *J. Analyse math*, vol. 17, no. 71, p. 107, 1966.

[108] H. B. Curry and I. J. Schoenberg, "On spline distributions and their limits-the polya distribution functions," in *Bulletin of the American Mathematical Society*, vol. 53, pp. 1114–1114, AMER MATHEMATICAL SOC 201 CHARLES ST, PROVIDENCE, RI 02940-2213, 1947.

[109] C. De Boor, "On calculating with b-splines," *Journal of Approximation theory*, vol. 6, no. 1, pp. 50–62, 1972.

[110] J.-J. Fang and C.-L. Hung, "An improved parameterization method for b-spline curve and surface interpolation," *Computer-Aided Design*, vol. 45, no. 6, pp. 1005–1028, 2013.

[111] X. Li, S. Tan, X. Feng, and H. Rong, "Lspb trajectory planning: Design for the modular robot arm applications," in *2009 International Conference on Information Engineering and Computer Science*, pp. 1–4, IEEE, 2009.

[112] L. Fu and J. Zhao, "Robot compliant catching by maxwell model based cartesian admittance control," *Assembly Automation*, vol. 41, no. 2, pp. 133–143, 2021.

[113] A. Walch, C. Eitzinger, S. Zambal, and W. Palfinger, "Lspb trajectory planning using quadratic splines," in *Proceedings of the 3rd International Conference on Mechatronics and Robotics Engineering*, pp. 81–87, 2017.

[114] T. Kunz and M. Stilman, "Turning paths into trajectories using parabolic blends," tech. rep., Georgia Institute of Technology, 2011.

[115] M. Juříček, R. Parák, and J. Kůdela, "Evolutionary computation techniques for path planning problems in industrial robotics: A state-of-the-art review," *Computation*, vol. 11, no. 12, p. 23, 2023.

[116] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[117] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*, pp. 1587–1596, PMLR, 2018.

[118] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.

[119] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[120] P. Dayan and C. Watkins, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[121] G. Tesauro *et al.*, "Temporal difference learning and td-gammon," *Communications of the ACM*, vol. 38, no. 3, pp. 58–68, 1995.

[122] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM journal on control and optimization*, vol. 30, no. 4, pp. 838–855, 1992.

[123] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[124] R. Parák, R. Matoušek, and B. Lacko, "I4C - Robotic cell according to the Industry 4.0 concept," *Automa*, vol. 27, no. 1, pp. 10–12, 2021.

[125] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51416–51431, 2021.

[126] E. Coumans, Y. P. Bai, and A. PyBullet, "Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016."

[127] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033, IEEE, 2012.

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

[129] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Sendai, Japan), pp. 2149–2154, Sep 2004.

[130] Unity Technologies, "Unity real-time development platform." `www.unity.com`. Online. Accessed on 5 August 2023.

[131] X. Yang, Z. Ji, J. Wu, and Y.-K. Lai, "An open-source multi-goal reinforcement learning environment for robotic manipulation with pybullet," in *Annual Conference Towards Autonomous Robotic Systems*, pp. 14–24, Springer, 2021.

[132] C. E. Mower, T. Stouraitis, J. Moura, C. Rauch, L. Yan, N. Z. Behabadi, M. Gienger, T. K. M. Vercauteren, C. Bergeles, and S. Vijayakumar, "Ros-pybullet interface: A framework for reliable contact simulation and human-robot interaction," in *Conference on Robot Learning*, 2022.

[133] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[134] Y. Kang, D. Kim, and K. Kim, "Urdf generator for manipulator robot," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*, pp. 483–487, IEEE, 2019.

[135] J. He, M. Zhang, F. Xu, Q. Liu, and H. Li, "A model conversion algorithm from urdf to dh for camera robot," in *2022 International Conference on Culture-Oriented Science and Technology (CoST)*, pp. 134–139, IEEE, 2022.

[136] D. Krupke, S. Starke, L. Einig, J. Zhang, and F. Steinicke, "Prototyping of immersive hri scenarios," in *Human-Centric Robotics: Proceedings of CLAWAR 2017: 20th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pp. 537–544, World Scientific, 2018.

[137] E. Sita, C. M. Horváth, T. Thomessen, P. Korondi, and A. G. Pipe, "Ros-unity3d based system for monitoring of an industrial robotic process," in *2017 IEEE/SICE International Symposium on System Integration (SII)*, pp. 1047–1052, IEEE, 2017.

[138] S. Bin, W. Yanwu, Z. Xiyong, and C. Huabin, "Virtual reality design of industrial robot teaching based on unity3d," in *2021 7th International Symposium on Mechatronics and Industrial Informatics (ISMII)*, pp. 1–4, IEEE, 2021.

[139] J. Kůdela, M. Juříček, and R. Parák, "A collection of robotics problems for benchmarking evolutionary computation methods," in *26th International Conference on Applications of Evolutionary Computation, EvoApplications 2023, held as part of EvoStar 2023*, vol. 13989, (Cham), pp. 364 –379, Springer, 2023.

[140] D. Kinga, J. B. Adam, *et al.*, "A method for stochastic optimization," in *International conference on learning representations (ICLR)*, vol. 5, p. 6, San Diego, California;, 2015.

[141] R. Parák and M. Juříček, "Intelligent sampling of anterior human nasal swabs using a collaborative robotic arm," *Mendel Journal series*, vol. 28, no. 1, pp. 32–40, 2022.

Appendix A: Kinematics

The appendix contains a complete summary of the results from Section 5.3, which focused on the comprehensive approach to kinematics solutions. The objective was to validate the modularity of the designed solutions across a variety of robotic structures, as shown in Figure A.1.

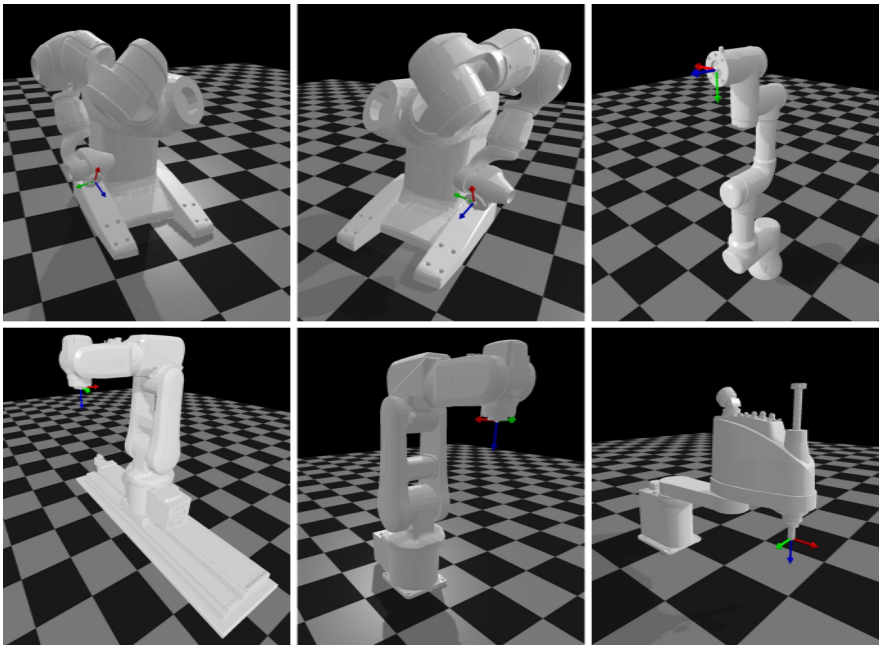


Figure A.1: An illustration of the various robotic structures used to test the kinematic solution. In the top row, there are the right and left arms of the ABB IRB 14000 and the Universal Robots UR3. In the bottom row, there is an ABB IRB 120 extended by a linear axis, the same robot without an extension, and lastly, the Epson LS3-B401S.

Firstly, this section presents a Denavit-Hartenberg table for the standard and modified forms of the parameters for each robotic structure. The Denavit-Hartenberg parameters provide a foundation for understanding the robotic structures, which is necessary for exploring kinematic solutions.

Table A.1: Denavit-Hartenberg in standard and modified form for Universal Robots UR3 robotic structure.

i	**Standard Denavit–Hartenberg Parameters**				**Modified Denavit–Hartenberg Parameters**			
	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]
1	0.0	0.0	0.1519	1.5708	0.0	0.0	0.1519	0.0
2	0.0	-0.24365	0.0	0.0	0.0	0.0	0.0	1.5708
3	0.0	-0.21325	0.0	0.0	0.0	-0.24365	0.0	0.0
4	0.0	0.0	0.11235	1.5708	0.0	-0.21325	0.11235	0.0
5	0.0	0.0	0.08535	-1.5708	0.0	0.0	0.08535	1.5708
6	0.0	0.0	0.0819	0.0	0.0	0.0	0.0819	-1.5708

Table A.2: Denavit-Hartenberg in standard and modified form for Epson LS3-B401S robotic structure.

i	**Standard Denavit–Hartenberg Parameters**				**Modified Denavit–Hartenberg Parameters**			
	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]
1	0.0	0.225	0.1731	0.0	0.0	0.0	0.1731	0.0
2	0.0	0.175	0.0499	3.1415	0.0	0.225	0.0499	0.0
3	0.0	0.0	0.0	0.0	0.0	0.175	0.0	3.1415
4	0.0	0.0	0.0785	0.0	0.0	0.0	0.0785	0.0

Table A.3: Denavit-Hartenberg in standard and modified form for ABB IRB 120 robotic structure.

i	**Standard Denavit–Hartenberg Parameters**				**Modified Denavit–Hartenberg Parameters**			
	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]
1	0.0	0.0	0.290	-1.5708	0.0	0.0	0.290	0.0
2	-1.5708	-0.270	0.0	0.0	-1.5708	0.0	0.0	-1.5708
3	0.0	0.07	0.0	-1.5708	0.0	-0.270	0.0	0.0
4	0.0	0.0	0.302	1.5708	0.0	0.07	0.302	-1.5708
5	0.0	0.0	0.0	-1.5708	0.0	0.0	0.0	1.5708
6	0.0	0.0	0.072	0.0	0.0	0.0	0.072	-1.5708

Table A.4: Denavit-Hartenberg in standard and modified form for ABB IRB 120 robotic structure extended by a linear axis.

i	**Standard Denavit–Hartenberg Parameters**				**Modified Denavit–Hartenberg Parameters**			
	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]
1	0.0	0.0	0.113	0.0	0.0	0.0	0.113	0.0
2	0.0	0.0	0.290	-1.5708	0.0	0.0	0.290	0.0
3	-1.5708	-0.270	0.0	0.0	-1.5708	0.0	0.0	-1.5708
4	0.0	0.07	0.0	-1.5708	0.0	-0.270	0.0	0.0
5	0.0	0.0	0.302	1.5708	0.0	0.07	0.302	-1.5708
6	0.0	0.0	0.0	-1.5708	0.0	0.0	0.0	1.5708
7	0.0	0.0	0.072	0.0	0.0	0.0	0.072	-1.5708

Table A.5: Denavit-Hartenberg in standard and modified form for both the left and right arm of the ABB IRB 14000 robotic structure.

i	**Standard Denavit–Hartenberg Parameters**				**Modified Denavit–Hartenberg Parameters**			
	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]	θ_i [rad]	a_i [m]	d_i [m]	α_i [rad]
1	0.0	-0.030	0.1	-1.5708	0.0	0.0	0.1	0.0
2	0.0	0.030	0.0	1.5708	0.0	-0.030	0.0	-1.5708
3	0.0	0.0405	0.2515	-1.5708	0.0	0.030	0.2515	1.5708
4	-1.5708	0.0405	0.0	-1.5708	-1.5708	0.0405	0.0	-1.5708
5	3.1415	0.027	0.265	-1.5708	3.1415	0.0405	0.265	-1.5708
6	0.0	-0.027	0.0	1.5708	0.0	0.027	0.0	-1.5708
7	0.0	0.0	0.036	0.0	0.0	-0.027	0.036	1.5708

Subsequently, the results of the kinematic solution obtained by the Levenberg-Marquardt method are illustrated. These results are based on one hundred via points generated between the initial and final poses of the end-effector. The sequence of points was generated using the trapezoidal motion profile.

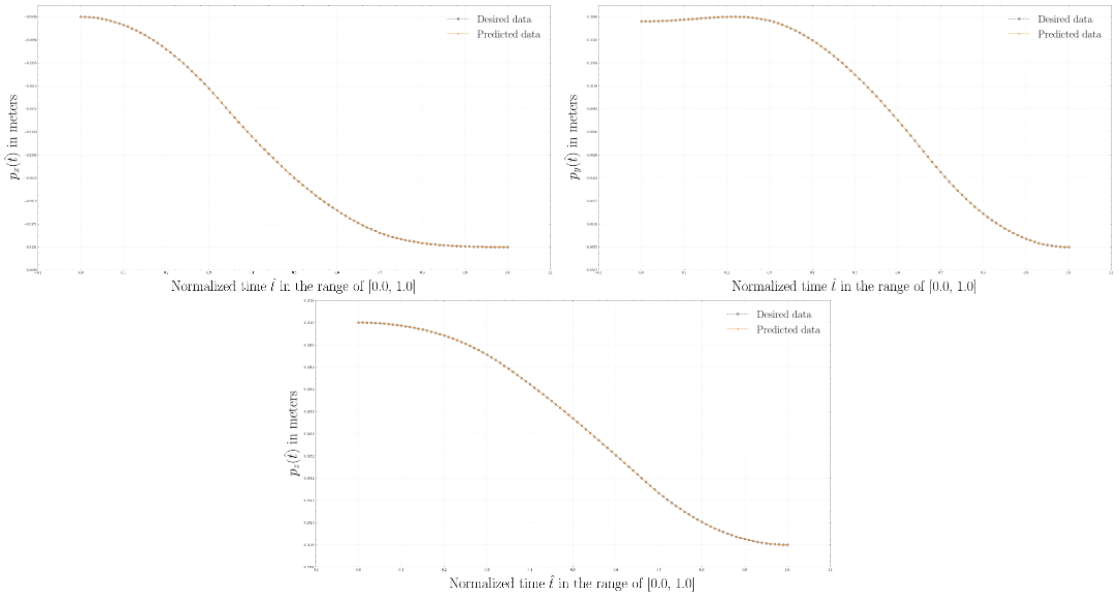


Figure A.2: The results of the positional aspect $\mathbf{p} = [p_x, p_y, p_z]$ of the kinematic solution regarding the Universal Robots UR3 robotic structure.

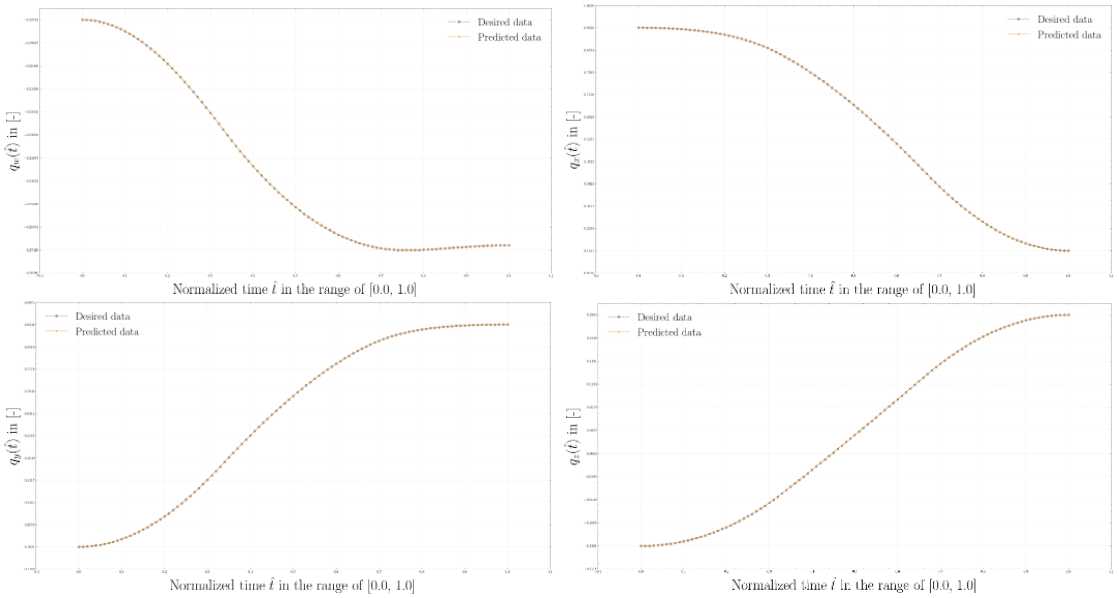


Figure A.3: The results of the orientational aspect $\mathbf{Q} = [q_w, q_x, q_y, q_z]$ of the kinematic solution regarding the Universal Robots UR3 robotic structure.

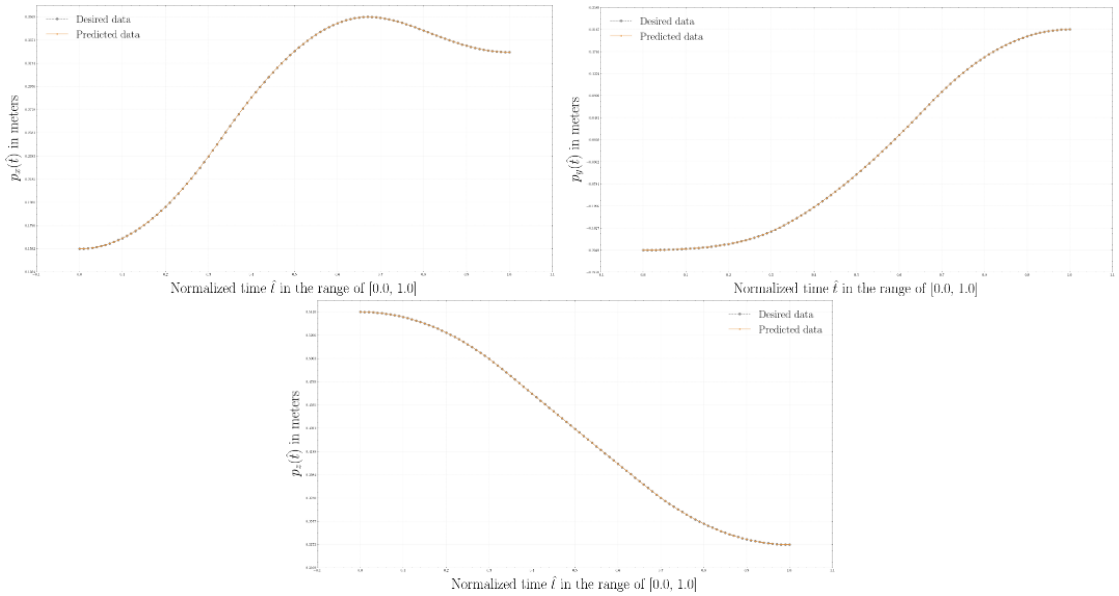


Figure A.4: The results of the positional aspect $\mathbf{p} = [p_x, p_y, p_z]$ of the kinematic solution regarding the ABB IRB 120 robotic structure.

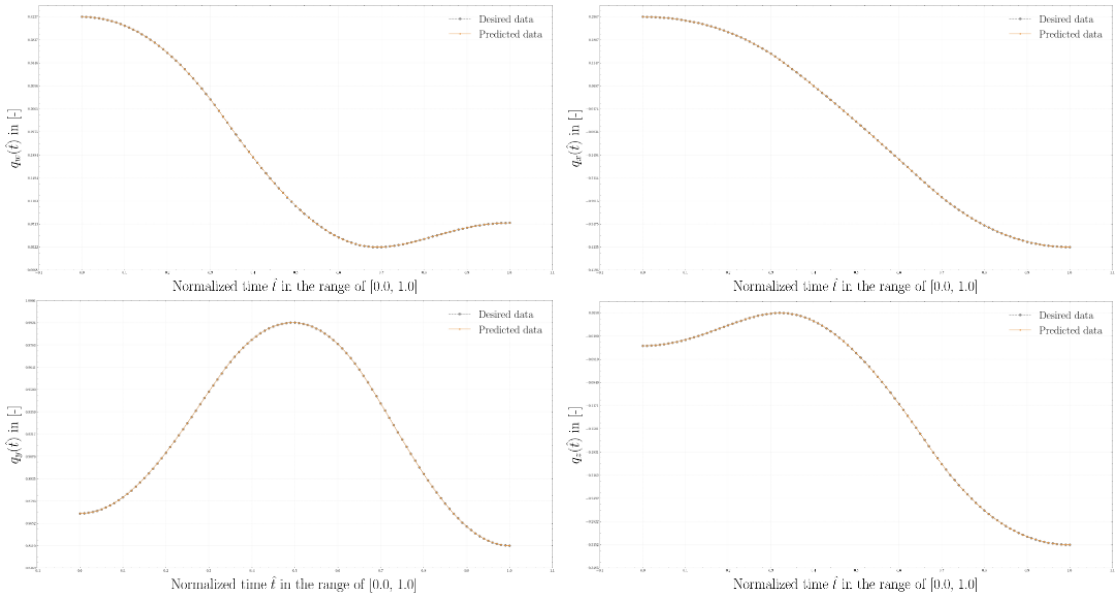


Figure A.5: The results of the orientational aspect $\mathbf{Q} = [q_w, q_x, q_y, q_z]$ of the kinematic solution regarding the ABB IRB 120 robotic structure.

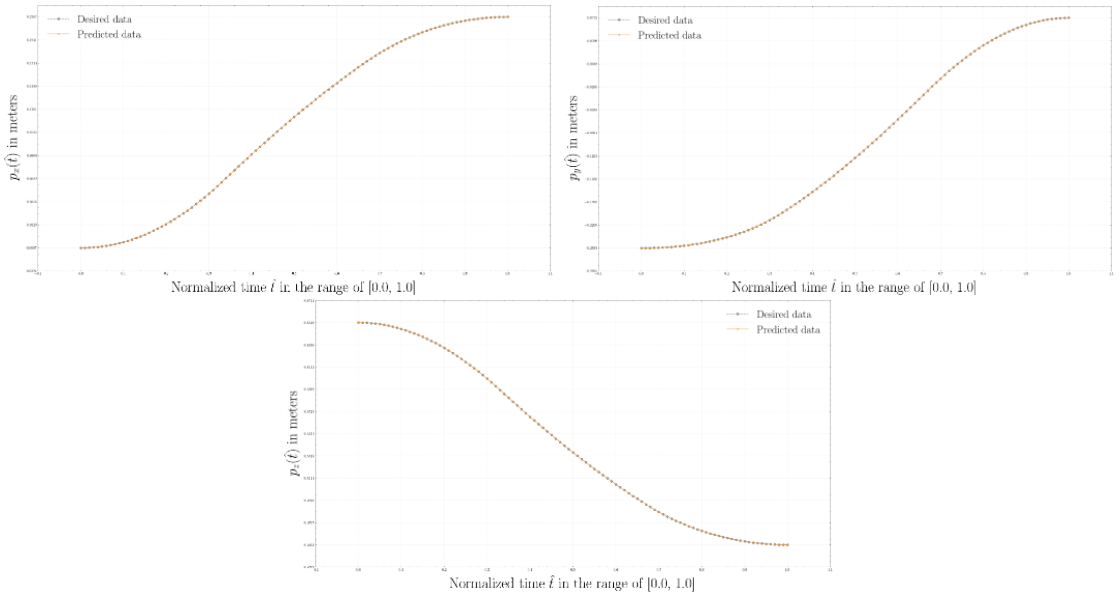


Figure A.6: The results of the positional aspect $\mathbf{p} = [p_x, p_y, p_z]$ of the kinematic solution regarding the ABB IRB 120 robotic structure extended by a linear axis.

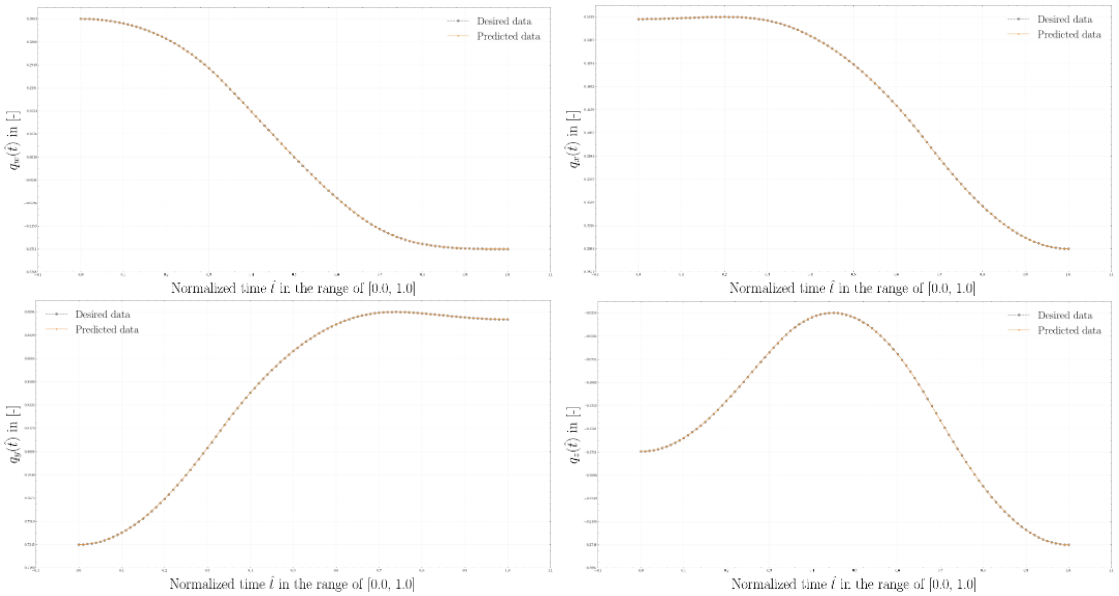


Figure A.7: The results of the orientational aspect $\mathbf{Q} = [q_w, q_x, q_y, q_z]$ of the kinematic solution regarding the ABB IRB 120 robotic structure extended by a linear axis.

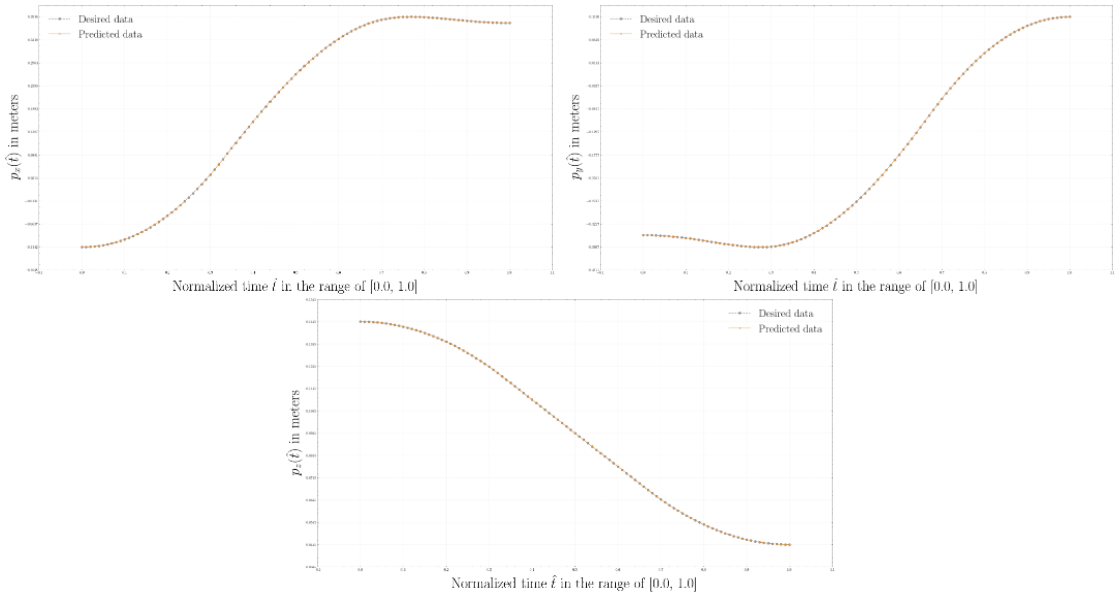


Figure A.8: The results of the positional aspect $\mathbf{p} = [p_x, p_y, p_z]$ of the kinematic solution regarding the Epson LS3-B401S robotic structure.

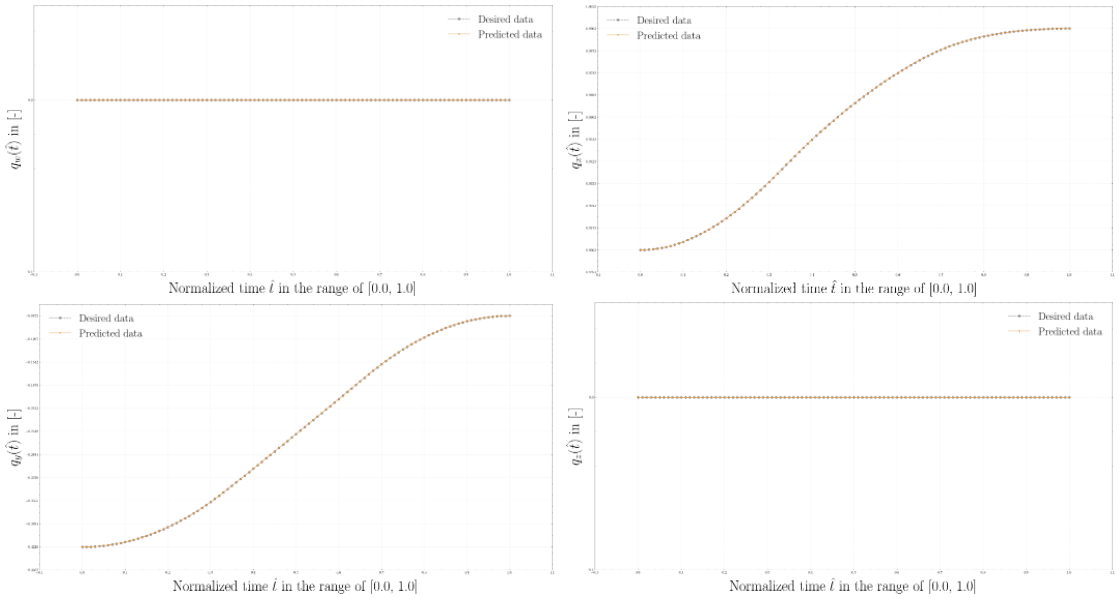


Figure A.9: The results of the orientational aspect $\mathbf{Q} = [q_w, q_x, q_y, q_z]$ of the kinematic solution regarding the Epson LS3-B401S robotic structure.

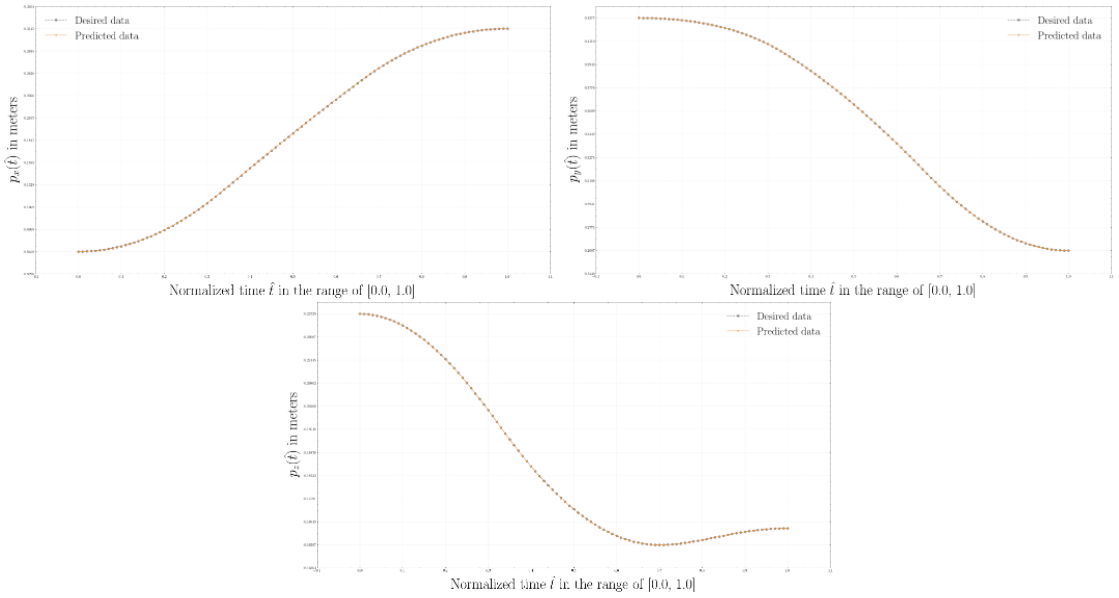


Figure A.10: The results of the positional aspect $\mathbf{p} = [p_x, p_y, p_z]$ of the kinematic solution regarding the ABB IRB 14000 (left) robotic structure.

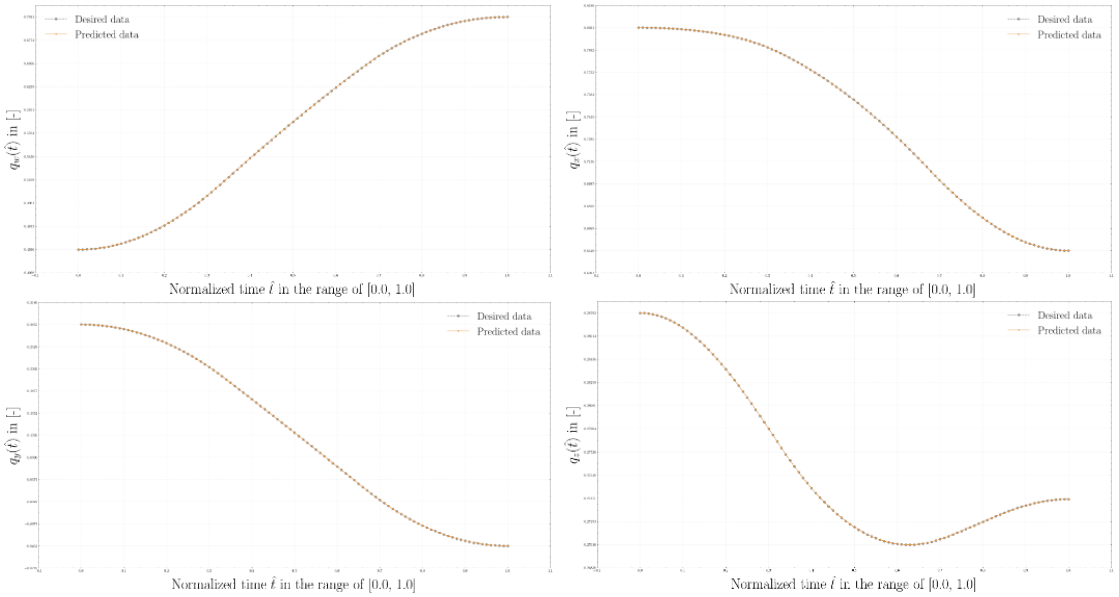


Figure A.11: The results of the orientational aspect $\mathbf{Q} = [q_w, q_x, q_y, q_z]$ of the kinematic solution regarding the ABB IRB 14000 (left) robotic structure.

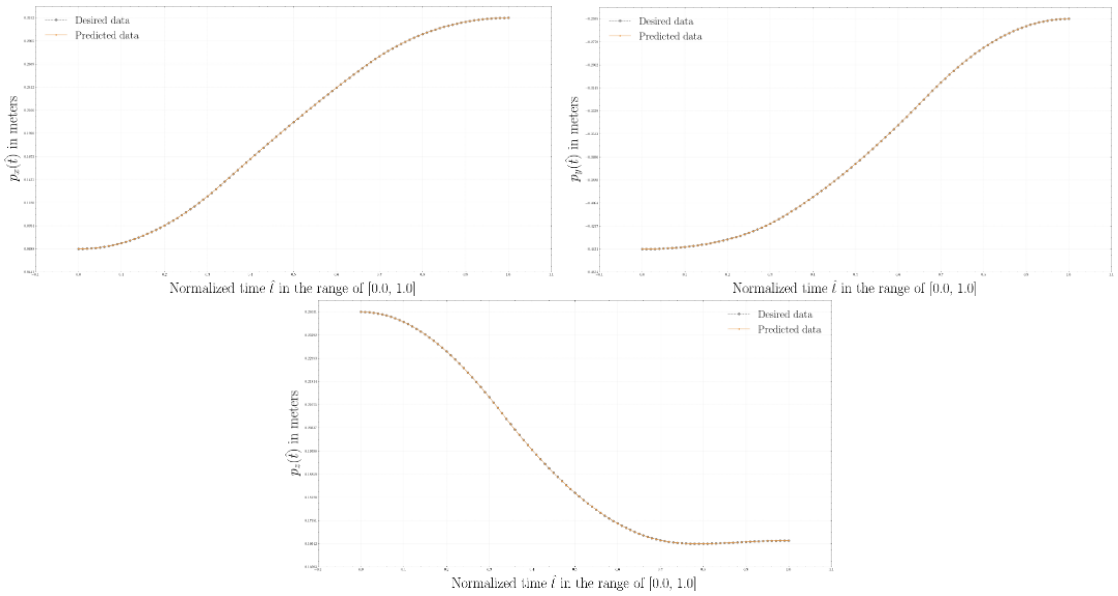


Figure A.12: The results of the positional aspect $\mathbf{p} = [p_x, p_y, p_z]$ of the kinematic solution regarding the ABB IRB 14000 (right) robotic structure.

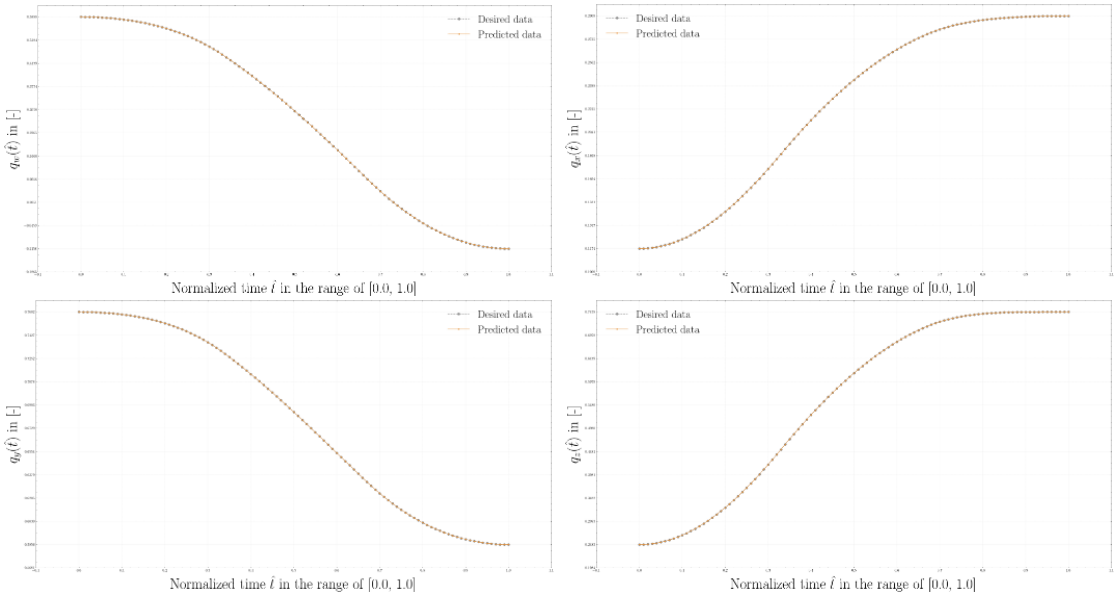


Figure A.13: The results of the orientational aspect $\mathbf{Q} = [q_w, q_x, q_y, q_z]$ of the kinematic solution regarding the ABB IRB 14000 (right) robotic structure.

Appendix B: DRL-Based Motion Planning

The appendix contains a complete summary of the results from Section 5.5, which focused on DRL-based motion planning. The objective was to validate the modularity of the designed solutions across a variety of robotic structures, as shown in Figure 5.1.

Firstly, this section presents the hyperparameter structure that was used in both the $\mathcal{E}_1$ and $\mathcal{E}_2$ environments. The selection of these hyperparameters was inspired by a comprehensive review detailed in [13].

Table B.1: Hyperparameter structure used for experiments in both types of environments.

Hyperparameters	**Description**
Network type	Multi-layer Perceptron (MLP)
Network size	3 layers with 256 units each and ReLU non-linearities
Optimizer	Adam optimizer [140] with 1.10^{-3} to train both actor and critic
Learning rate	0.001
Polyak-averaging coefficient [122]	0.95
Action L2 norm coefficient	1.0
Observation clipping	$[-1, 1]$
Action clipping	$[-1, 1]$
Probability of random actions	0.3
Scale of additive Gaussian noise	0.2
Replay Buffer size	10^6 transitions
Batch size	256
Episode length	100
Total timesteps	10^5

Subsequently, the results of the Deep Reinforcement Learning (DRL) motion planning solutions obtained by the Twin Delayed DDPG (TD3) algorithm for an environment type $\mathcal{E}_1$ and by the Deep Deterministic Policy Gradients (DDPG) algorithm for an environment type $\mathcal{E}_2$. The results are based on the prediction of the motion from the initial position to the desired final position, defined by a static target within the observed area. The trajectory is interpolated using a parametric B-spline curve.

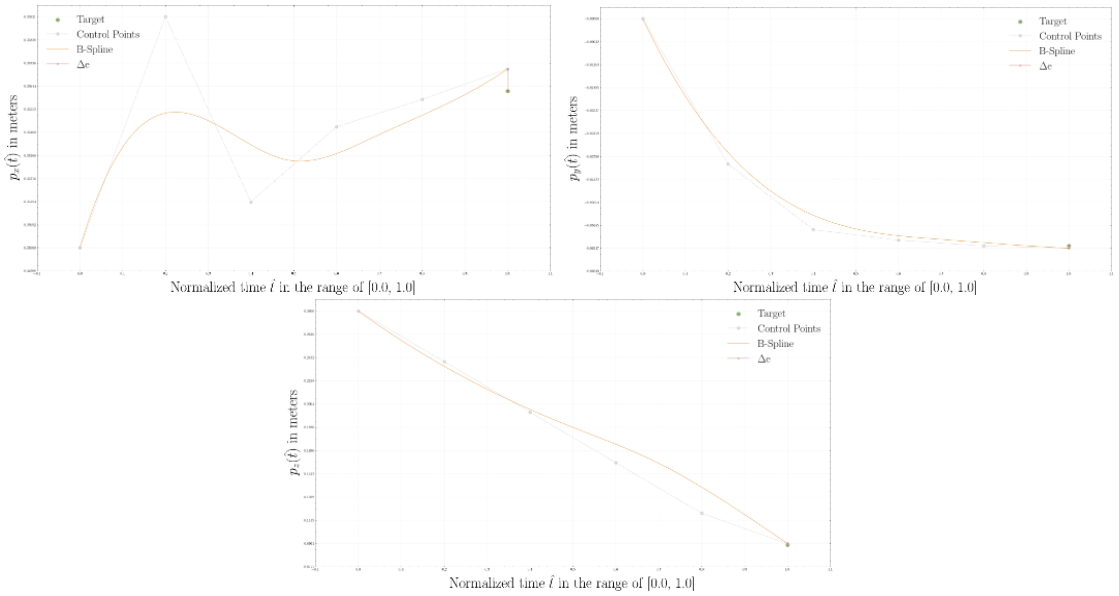


Figure B.1: The results of the positional aspect of the TD3 solution for an environment type $\mathcal{E}_1$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the Universal Robots UR3 robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00153$.

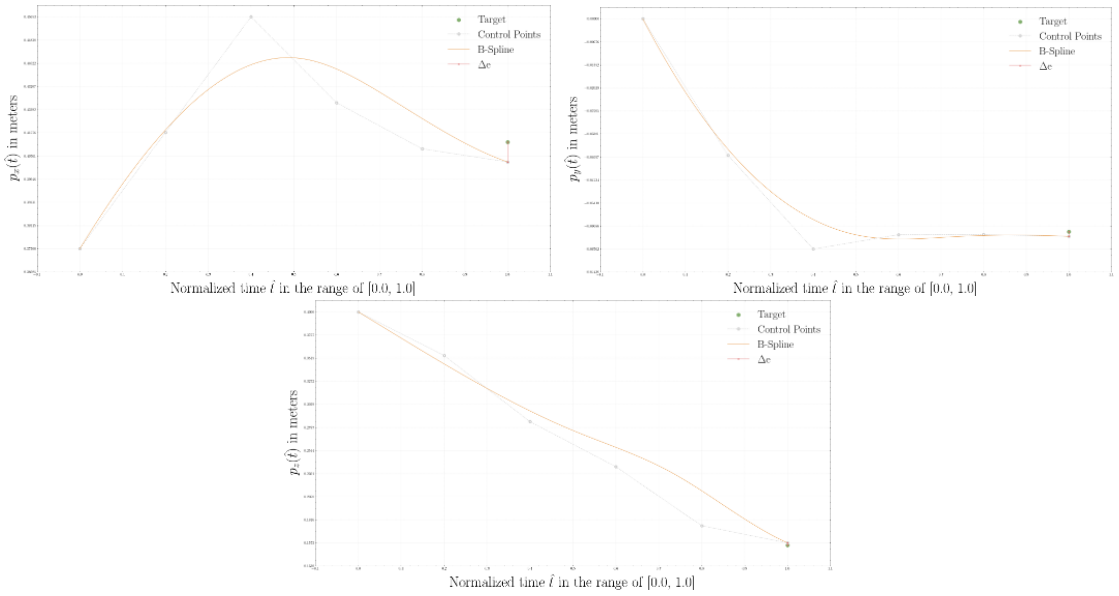


Figure B.2: The results of the positional aspect of the TD3 solution for an environment type $\mathcal{E}_1$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 120 robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00750$.

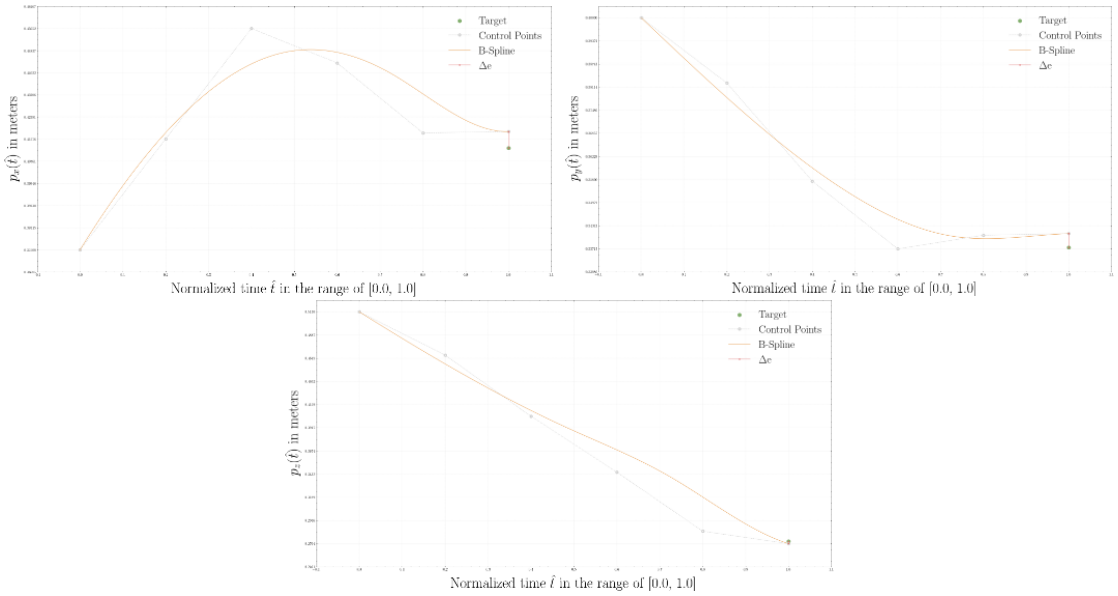


Figure B.3: The results of the positional aspect of the TD3 solution for an environment type $\mathcal{E}_1$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 120 robotic structure extended by a linear axis. The mean absolute error of the predicted solution was equal to $e_p = 0.00764$.

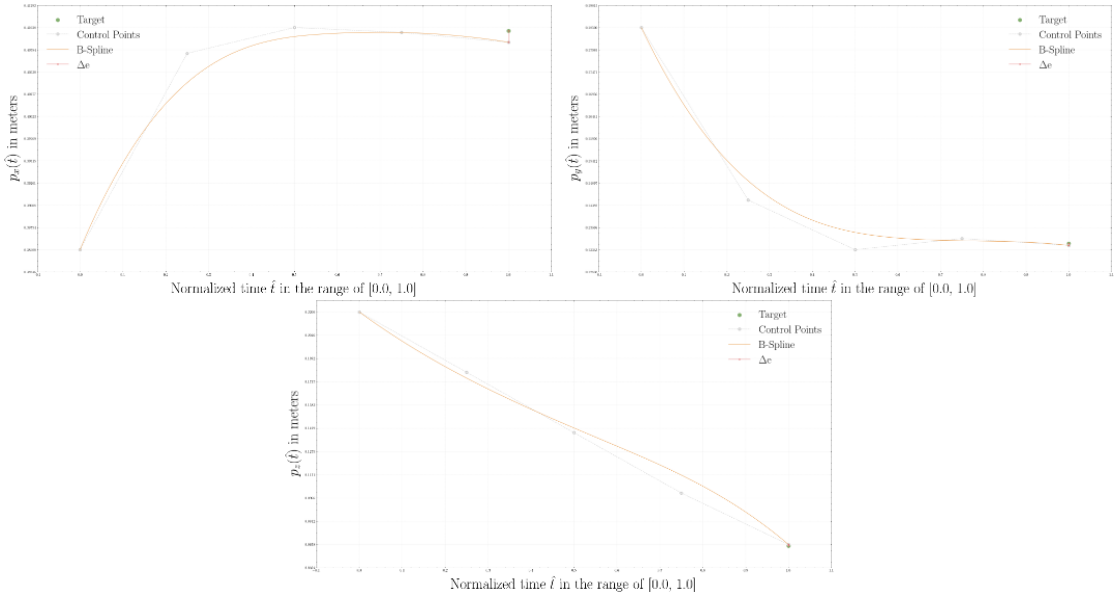


Figure B.4: The results of the positional aspect of the TD3 solution for an environment type $\mathcal{E}_1$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 14000 (left) robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00160$.

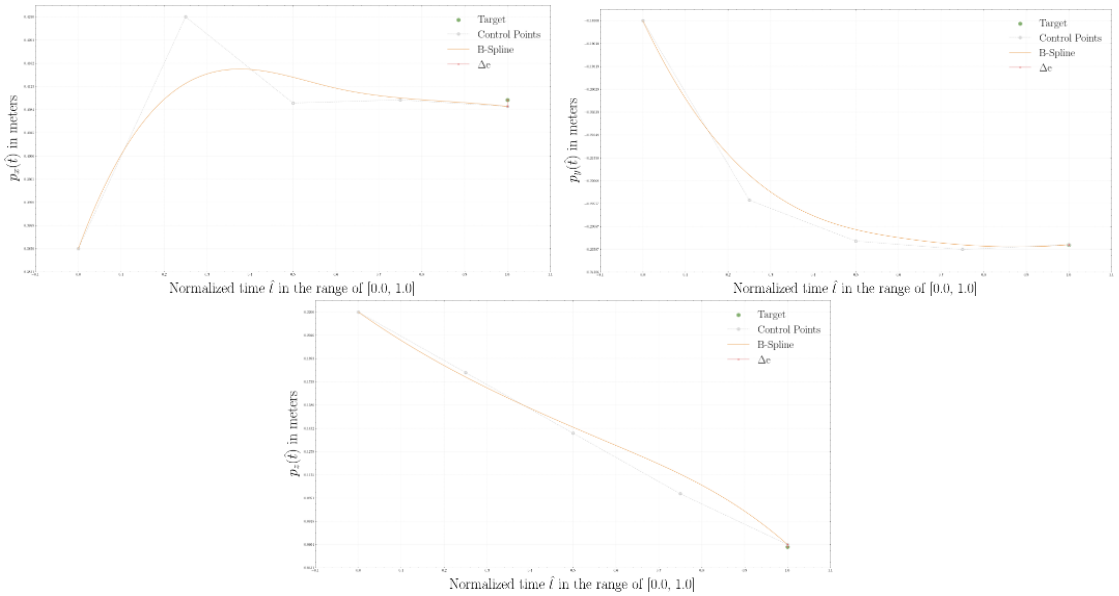


Figure B.5: The results of the positional aspect of the TD3 solution for an environment type $\mathcal{E}_1$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 14000 (right) robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00179$.

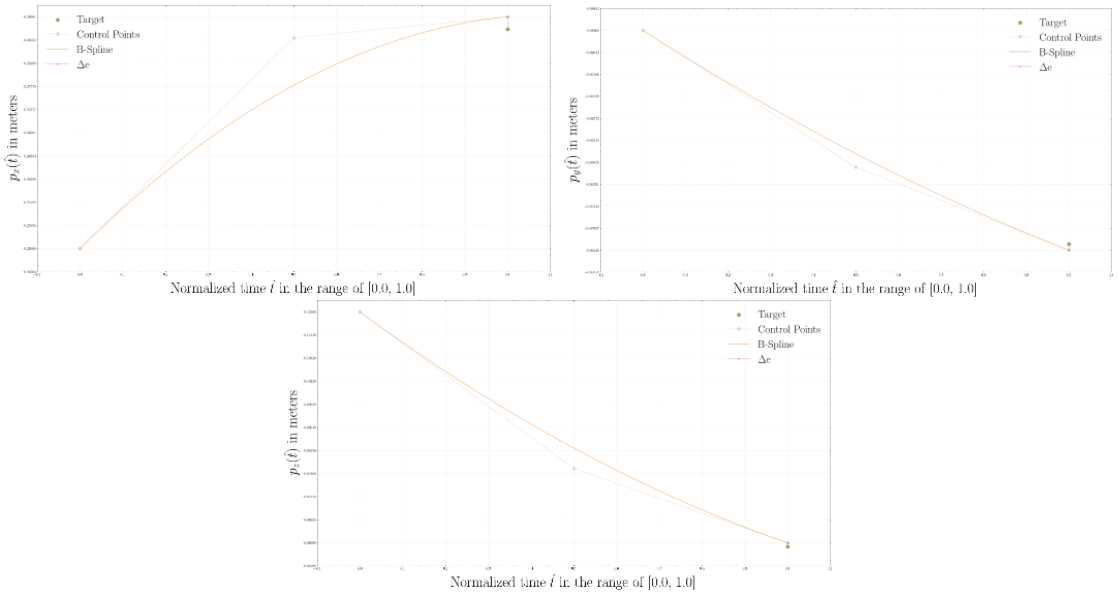


Figure B.6: The results of the positional aspect of the TD3 solution for an environment type $\mathcal{E}_1$, interpolated by the parametric B-spline curve of degree $n = 2$, regarding the Epson LS3-B401S robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00294$.

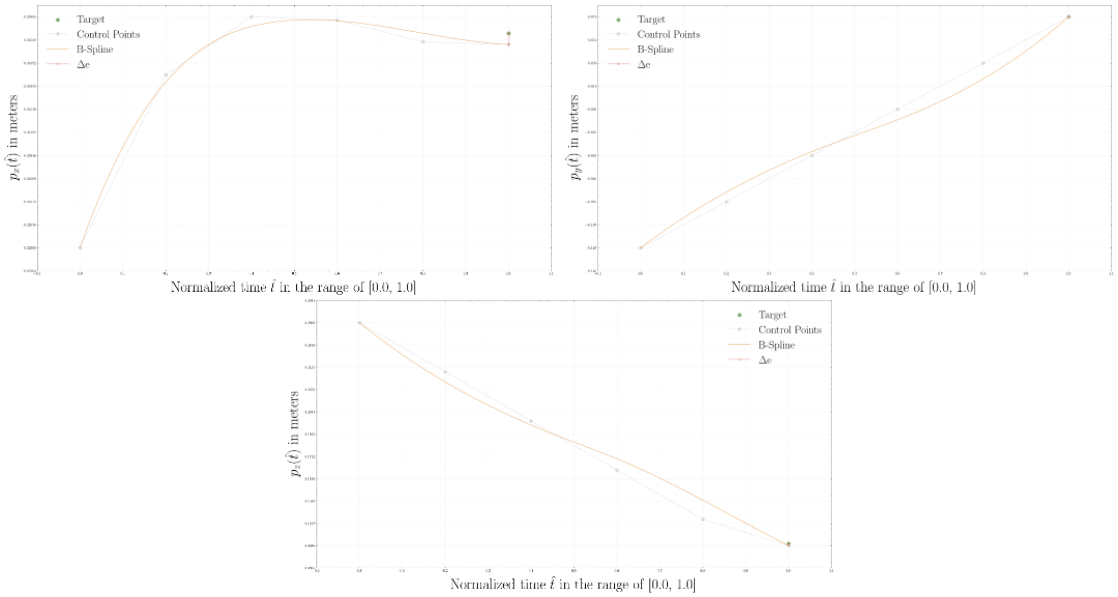


Figure B.7: The results of the positional aspect of the DDPG solution for an environment type $\mathcal{E}_2$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the Universal Robots UR3 robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00215$.

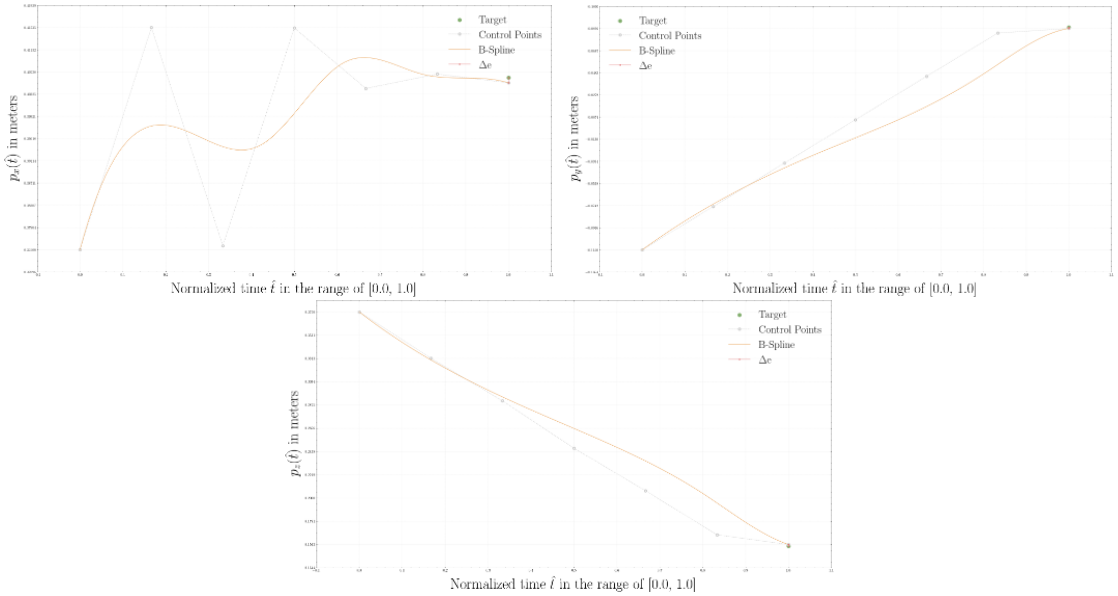


Figure B.8: The results of the positional aspect of the DDPG solution for an environment type $\mathcal{E}_2$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 120 robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00184$.

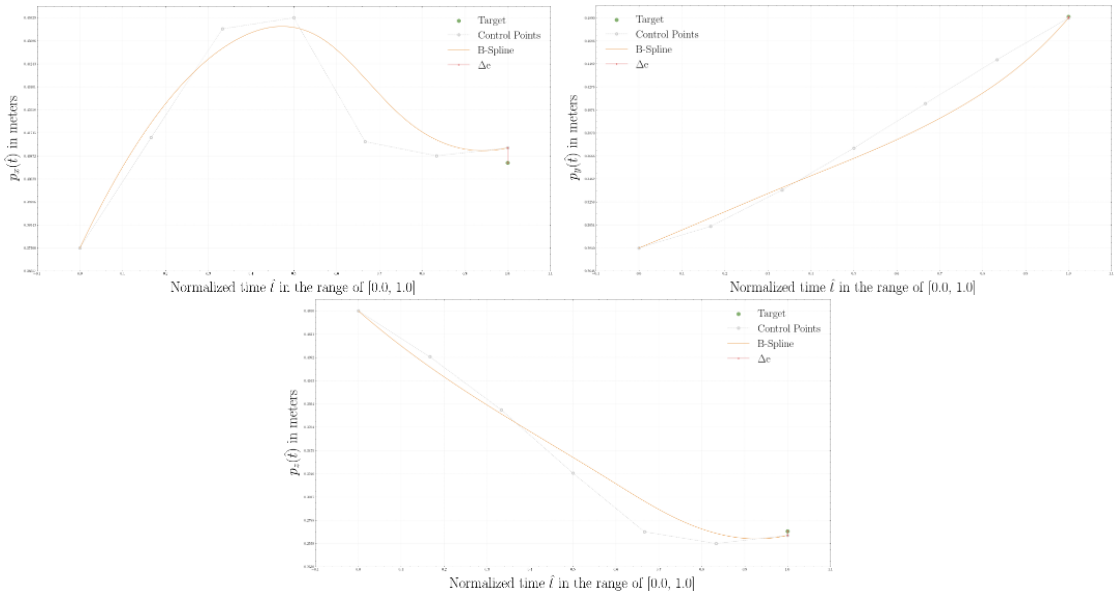


Figure B.9: The results of the positional aspect of the DDPG solution for an environment type $\mathcal{E}_2$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 120 robotic structure extended by a linear axis. The mean absolute error of the predicted solution was equal to $e_p = 0.00684$.

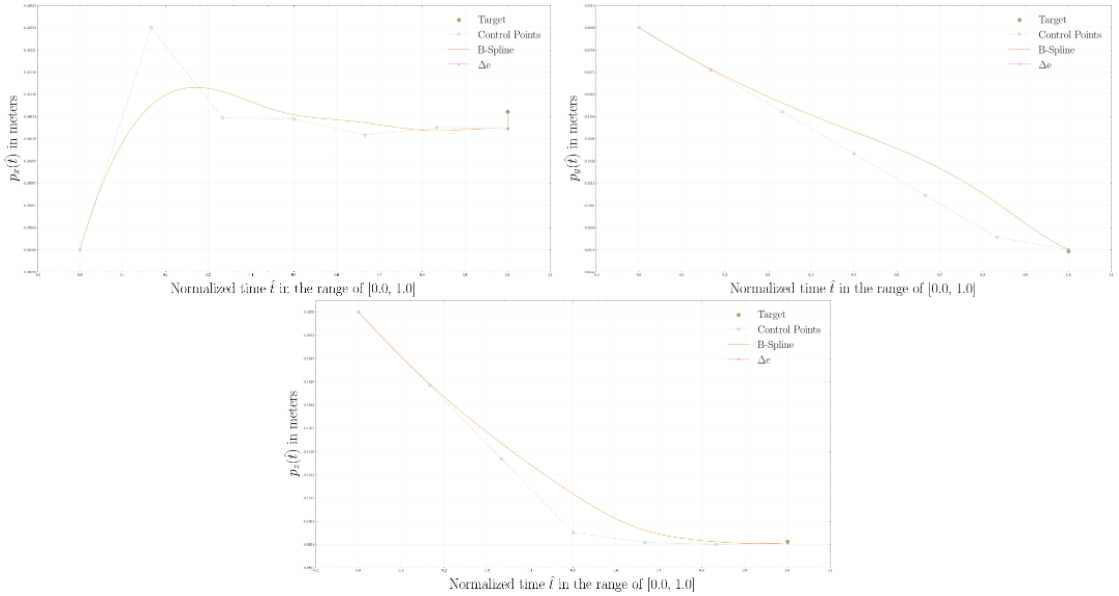


Figure B.10: The results of the positional aspect of the DDPG solution for an environment type $\mathcal{E}_2$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 14000 (left) robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00342$.

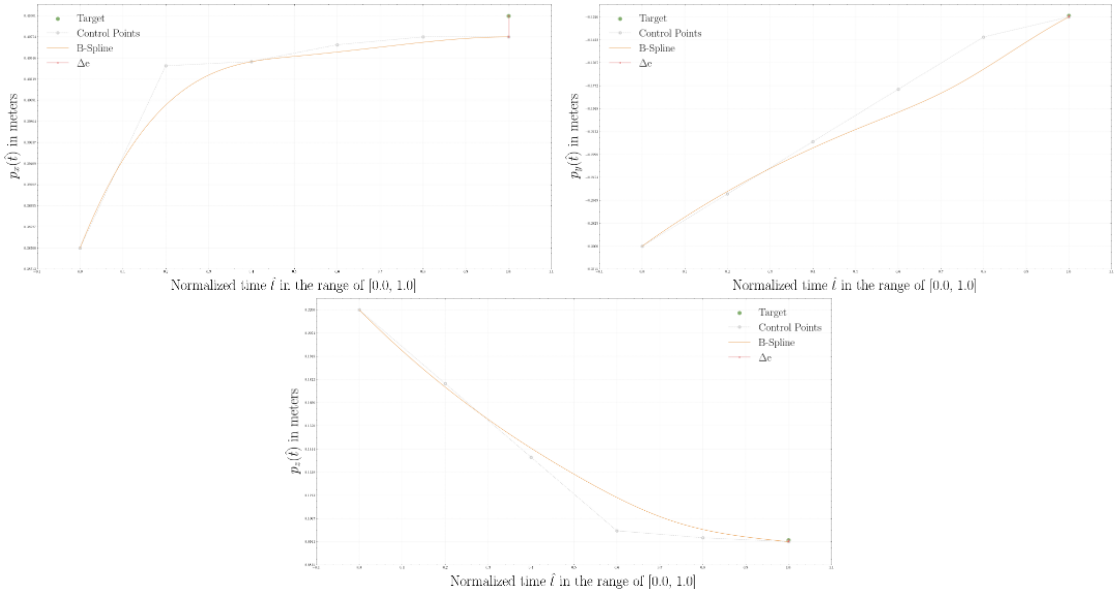


Figure B.11: The results of the positional aspect of the DDPG solution for an environment type $\mathcal{E}_2$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the ABB IRB 14000 (right) robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00260$.

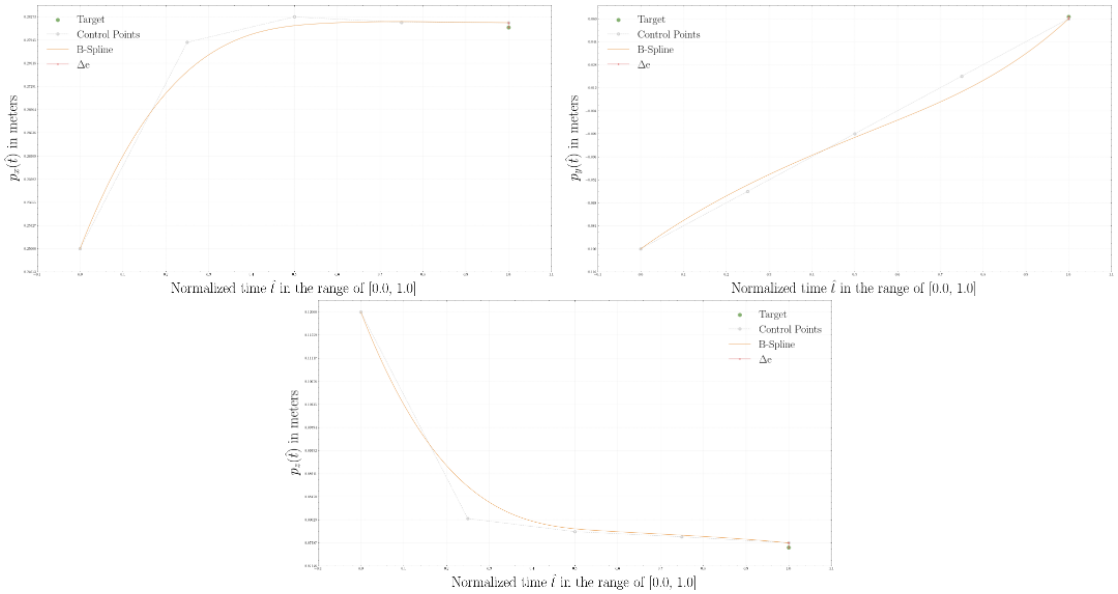


Figure B.12: The results of the positional aspect of the DDPG solution for an environment type $\mathcal{E}_2$, interpolated by the parametric B-spline curve of degree $n = 3$, regarding the Epson LS3-B401S robotic structure. The mean absolute error of the predicted solution was equal to $e_p = 0.00168$.

Appendix C: Activities Related to Doctoral Studies

The appendix offers a comprehensive summary of the main activities conducted throughout the doctoral studies. This section provides a brief overview of various experiences, including pedagogical practice, research projects, and supervision of both Bachelor's and Master's theses, which have defined the academic journey. The objective is to demonstrate active participation in a diverse range of activities that shows a commitment to doctoral studies.

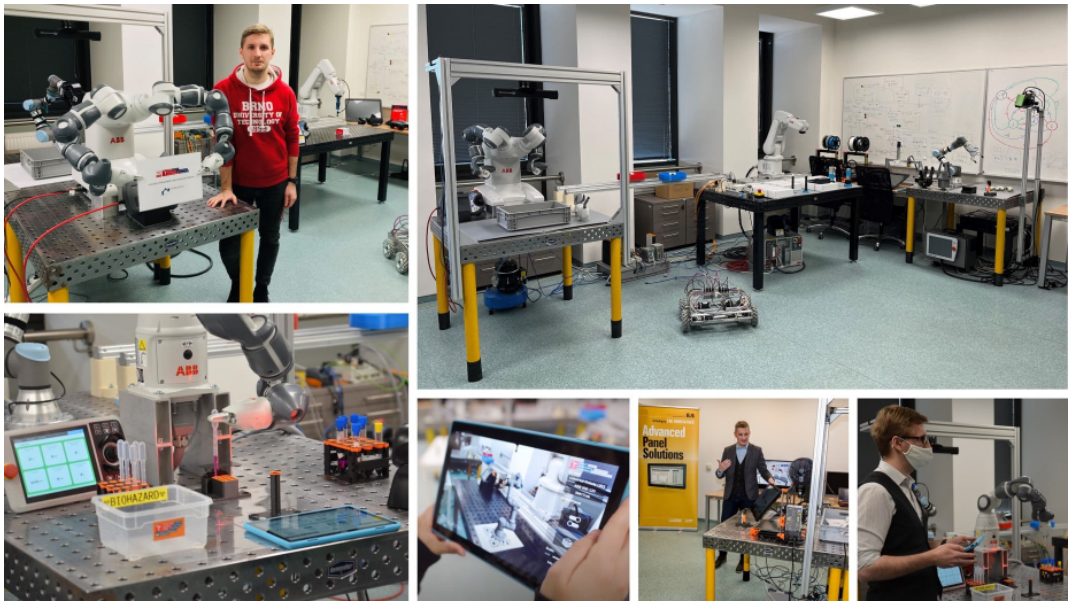


Figure C.1: Various events, such as open days, project presentations, and workshops within the Robotics Lab Industry 4.0 Cell (I4C) at the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology.

General Activities

Traineeship

A six-month Erasmus+ research traineeship in the academic year 2021/22 at the Institute of Robotics, Johannes Kepler University in the field of advanced robotics.
Supervisor: Assoc. Univ.-Prof. DI Dr. Hubert Gattringer
Co-Supervisor: Univ.-Prof. Dr.-Ing. habil. Andreas Müller

Completed Courses

Optimization - Mathematical Programming (9OMP), Computational Intelligence (9VIN), Basic of Measurement Theory (9ZTM), Methods and Algorithms for System Simulation and Optimization (9MAS), English for Doctoral Degree Study (9AJ)

Cooperation with Industrial Partners

B&R Automation, SMC Industrial Automation, ABB Robotics, Industry Cluster 4.0, Intemac Solutions

Other

(a) Co-organization of conferences such as the International Conference on Soft Computing MENDEL and Principia Cybernetica.

(b) Development of laboratories in the fields of general robotics, augmented reality, machine vision, and programmable logic controllers.

(c) Collaboration on creating a new course for the Master's degree in 'Programming of Robots and Manipulators' and the Bachelor's degree in 'Industry 4.0'. Creation of teaching materials for Bachelor's and Master's studies in English and Czech.

(d) Supervisor (13 students) and co-supervisor (33 students) for the Bachelor's and Master's study programs in the field of robotics, machine vision, and artificial intelligence techniques. Reviewer of 21 student theses.

Pedagogical Practice

Academic Year: 2017/18
Winter semester: Computer Science (1IN), Control Theory I (VA1)
Summer semester: Automation (6AA), Control Theory II (VA2)

Academic Year: 2018/19
Winter semester: Control Theory I (VA1), Control Theory I in English (VA1-A)
Summer semester: Control Theory II (VA2)

Academic Year: 2019/20
Summer semester: Industry 4.0 (0P4)

Academic Year: 2020/21
Summer semester: Programming for Robots and Manipulators (VRM & VRM-K)

Academic Year: 2021/22
Summer semester: Programming for Robots and Manipulators (VRM & VRM-K), Industry 4.0 in English (VI4-A)

Academic Year: 2022/23
Winter semester: Programmable Controller Systems (VPL & VPL-K)
Summer semester: Programming for Robots and Manipulators (VRM & VRM-K), Industry 4.0 in English (VI4-A)

Academic Year: 2023/24
Winter semester: Programmable Controller Systems (VPL & VPL-K), Virtual Reality (V0R)
Summer semester: Programming for Robots and Manipulators (VRM & VRM-K)

Projects

Research in the field of digital twins for the production of electrical switchboards. In cooperation with ABB Group.
Duration: 01.12.2018 — 31.05.2020

Use of augmented reality for product presentation. In cooperation with SMC Industrial Automation.
Duration: 01.12.2019 — 29.02.2020

Industry 4.0 and Artificial Intelligence methods.
Duration: 01.03.2020 — 28.02.2023

OpenTube: Robotic workplace for analysis test samples of Covid-19. In cooperation with the University Hospital Brno.
Duration: 18.01.2021 — 30.04.2022

ATCZ281 - TESTBED EXCHANGE: Networking of Industry 4.0 testbeds in Czech-Austrian cooperation from the Operational Programme Interreg V-A Austria - Czech Republic.
Duration: 01.10.2021 — 31.12.2022

Artificial intelligence methods in engineering applications.
Duration: 01.03.2023 — 28.02.2026

Conference and Workshop Presentations

R. Parák and B. Lacko, "Introduction to Robotics for Young Scientists," Science enjoys us: Interactive and fun camps for children, Brno, Czech republic, 2018.

R. Parák and B. Lacko, "Introduction to Robotics for Young Scientists," Science enjoys us: Interactive and fun camps for children, Brno, Czech republic, 2019.

R. Parák and B. Lacko, "Workshop: Industry 4.0 Cell (I4C), " Trade Media International: Conference on Robotics, Brno, Czech republic, 2020.

R. Parák, "Industry 4.0 Cell (I4C): A Robotic Cell Based on the Industry 4.0 Concept," Trade Media International: Conference on Robotics, Brno, Czech republic, 2021.

R. Parák, "Industry 4.0 Cell (I4C): A Robotic Cell Based on the Industry 4.0 Concept," Industry 4.0 Cluster: Industry 4.0 Conference, Brno, Czech republic, 2021.

R. Parák, R. Matoušek, and B. Lacko, "Industry 4.0 Cell (I4C): A Brief Overview," Networking Czech and Austrian Testbeds for Industry 4.0, Vienna, Austria, 2022.

R. Parák and R. Matoušek, "Industry 4.0 Cell (I4C): A Brief Overview," B&R Headquarters: Networking Czech and Slovak Robotic Laboratories, Eggelsberg, Austria, 2023.

Honors & Awards

Silver medal (Team Award), Brno University of Technology, 2020.
Rector's Award for Ph.D. students, Brno University of Technology, 2020.
Rector's Award for Teachers, Brno University of Technology, 2023.

Overview of Supervised Bachelor's and Master's Theses

The table below presents an overview of the bachelor's and master's theses supervised during the course of doctoral studies. It is worth noting that all theses received above-average evaluations, and some were even honored with significant prizes.

Table C.1: List of all theses (bachelor's, marked as BT, and master's, marked as MT) supervised during the doctoral studies.

Student	Type	Acad. year	Title	Awarded
VAVERKA, Pavel	BT	2017/18	Robotic table football - game strategy	
SLÁMA, Ondřej	BT	2017/18	Robotic table football - control of game axes	BUT, FME
NEVŘIVA, Václav	BT	2018/19	Design of the control program for industrial robot IRB 120 using RobotStudio software	
SOBOTKA, Pavel	BT	2018/19	Human-machine collaboration – using computer vision	BUT, FME
POLLACH, Ondřej	BT	2018/19	Mecanum wheel mobile platform – detection of obstacles	
GAŠKO, Viktor	BT	2018/19	Using machine learning for quality control in industrial applications	
HUBER, Michal	BT	2018/19	Detection of objects for industrial robots using computer vision	
MARŠALA, Štěpán	BT	2019/20	Detection of multiple objects using computer vision	
FILIP, Jakub	BT	2019/20	Design and implementation of control program for mobile robot platform Turtlebot3 Burger	
JURÍČEK, Martin	BT	2019/20	Utilization of Robotic Operating System (ROS) for control of collaborative robot UR3	BUT, FME
Bc. VAVRÍK, Michal	MT	2019/20	Design and implementation of control program for CNC machine via B&R Automation	
KOŘENEK, Miroslav	BT	2020/21	Design and implementation of a control program for a robotic cell	BUT, FME
Bc. JURÍČEK, Martin	MT	2021/22	Design and implementation of the robotic platform for an experimental laboratory task	Top 4 Industry 4.0 - Werner von Siemens Award

Appendix D: List of Scientific Publications

I4C - Robotic cell according to the Industry 4.0 concept

R. Parák, R. Matoušek, and B. Lacko, "I4C - Robotic cell according to the Industry 4.0 concept," Automa, vol. 27, no. 1, pp. 10–12, 2021. [124]

Description

The article discusses the design of a robotic cell in the context of Industry 4.0 and its implementation at the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology. The main focus is on the construction of the cell, which includes advanced system integration, human-machine interface, tools for simulating robots as dynamic systems (Digital-Twin), and the use of augmented/virtual reality. The conclusion presents the future development direction of the Industry 4.0 cell.

Comparison of Multiple Reinforcement Learning and Deep Reinforcement Learning Methods for the Task Aimed at Achieving the Goal

R. Parák and R. Matoušek, "Comparison of multiple reinforcement learning and deep reinforcement learning methods for the task aimed at achieving the goal," Mendel Journal series, vol. 27, no. 1, pp. 1–8, 2021. [12]

Description

Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL) methods are a promising approach to solving complex tasks in the real world with physical robots. In this paper, we compare several reinforcement learning (Q-Learning, SARSA) and deep reinforcement learning (Deep Q-Network, Deep Sarsa) methods for a task aimed at achieving a specific goal using robotics arm UR3. The main optimization problem of this experiment is to find the best solution for each RL/DRL scenario and minimize

the Euclidean distance accuracy error and smooth the resulting path by the Bézier spline method. The simulation and real word applications are controlled by the Robot Operating System (ROS). The learning environment is implemented using the OpenAI Gym library which uses the RVIZ simulation tool and the Gazebo 3D modeling tool for dynamics and kinematics.

Intelligent Sampling of Anterior Human Nasal Swabs using a Collaborative Robotic Arm

R. Parák and M. Juříček, "Intelligent sampling of anterior human nasal swabs using a collaborative robotic arm," Mendel Journal series, vol. 28, no. 1, pp. 32–40, 2022. [141]

Description

Advanced robotics does not always have to be associated with Industry 4.0, but can also be applied, for example, in the Smart Hospital concept. Developments in this field have been driven by the coronavirus disease (COVID-19), and any improvement in the work of medical staff is welcome. In this paper, an experimental robotic platform was designed and implemented whose main function is the swabbing samples from the nasal vestibule. The robotic platform represents a complete integration of software and hardware, where the operator has access to a web-based application and can control a number of functions. The increased safety and collaborative approach cannot be overlooked. The result of this work is a functional prototype of the robotic platform that can be further extended, for example, by using alternative technologies, extending patient safety, or clinical tests and studies.

A collection of robotics problems for benchmarking evolutionary computation methods

J. Kůdela, M. Juříček, and **R. Parák**, "A collection of robotics problems for benchmarking evolutionary computation methods," in 26th International Conference on Applications of Evolutionary Computation, EvoApplications 2023, held as part of EvoStar 2023, vol. 13989, (Cham), pp. 364 –379, Springer, 2023. [139]

Description

The utilization of benchmarking techniques has a crucial role in the development of novel optimization algorithms, and also in performing comparisons between already existing methods. This is especially true in the field of evolutionary computation, where the theoretical performance of the method is difficult to analyze. For these benchmarking purposes, artificial (or synthetic) functions are currently the most widely used ones. In this paper, we present a collection of real-world robotics problems that can be used for benchmarking evolutionary computation methods. The proposed benchmark problems

are a combination of inverse kinematics and path planning in robotics that can be parameterized. We conducted an extensive numerical investigation that encompassed solving 200 benchmark problems by seven selected metaheuristic algorithms. The results of this investigation showed that the proposed benchmark problems are quite difficult (multimodal and non-separable) and that they can be successfully used for differentiating and ranking various metaheuristics.

Evolutionary Computation Techniques for Path Planning Problems in Industrial Robotics: A State-of-the-Art Review

M. Juříček, **R. Parák**, and J. Kůdela, "Evolutionary computation techniques for path planning problems in industrial robotics: A state-of-the-art review," Computation, vol. 11, no. 12, p. 23, 2023. [115]

Description

The significance of robot manipulators in engineering applications and scientific research has increased substantially in recent years. The utilization of robot manipulators to save labor and increase production accuracy is becoming a common practice in industry. Evolutionary computation (EC) techniques are optimization methods that have found their use in diverse engineering fields. This state-of-the-art review focuses on recent developments and progress in their applications for industrial robotics, especially for path planning problems that need to satisfy various constraints that are implied by both the geometry of the robot and its surroundings. We discuss the most-used EC method and the modifications that suit this particular purpose, as well as the different simulation environments that are used for their development. Lastly, we outline the possible research gaps and the expected directions future research in this area will entail.

Appendix E: Source Codes

"Active participation within the open-source community, not only as a user but also as a contributor, is essential to ensuring continued growth."

— Roman Parak

The appendix contains supplementary information related to the source codes used in both the development of the dissertation and the instruction of university courses. The inclusion of these repositories aims to improve the transparency and reproducibility of the research findings. The codes hosted on GitHub serve as valuable resources for readers interested in exploring technical details, replicating experiments, or expanding the scope of the dissertation thesis. Each repository includes detailed descriptions and instructions, making it easier for users to start individual projects.

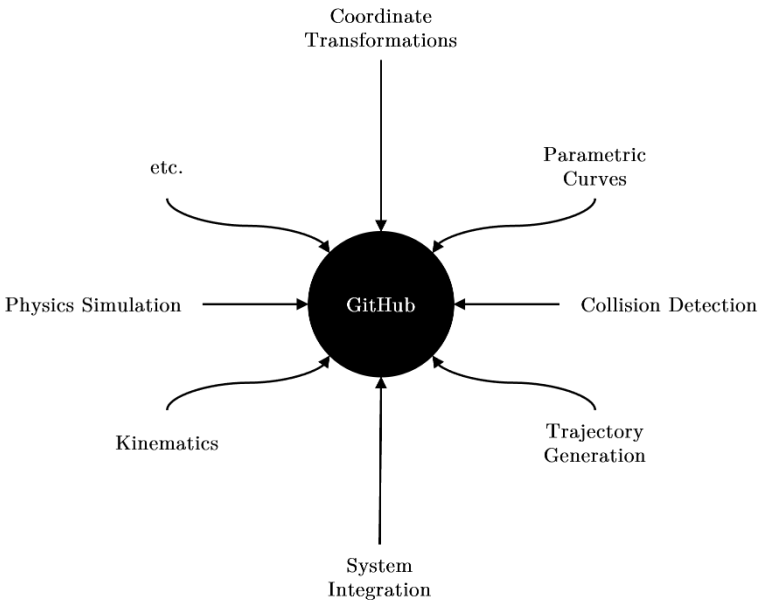


Figure E.1: A brief diagram illustrating the areas of projects available online on GitHub.

Dissertation Repositories

R. Parák, "An Open-Source Transformation Library Useful for Robotics Applications." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "An Open-Source Trajectory Generation Library Useful for Robotics Applications." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "An open-source parametric curves library useful for robotics applications." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "An Open-Source Parametric Curves Library Useful for Robotics Applications." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "An Open-Source Python Library to Control ABB Robot Arm via Externally Guided Motion (EGM)." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "OPC UA Communication between B&R Automation PLC (Server) and Client (C#, Python)." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "A Digital Twins in the field of Industrial Robotics integrated into the Unity3D Development Platform." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "SMC Industrial Automation: End-Effector Prototypes." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "Robotics Library for Everyone (RoLE)." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "An Industrial Robotics Gym in PyBullet." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "Versatile Intelligent Robotic Workstation (VInRoS)." Available online on GitHub.
Link to the GitHub repository: Click here.

University Courses and Seminars Repositories

R. Parák, "University Course: Programming for Robots and Manipulators." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "University Course: Control Theory I & II." Available online on GitHub.
Link to the GitHub repository: Click here.

R. Parák, "University Workshop: Motion Control with MappTechnology and PLCOpen." Available online on GitHub.
Link to the GitHub repository: Click here.